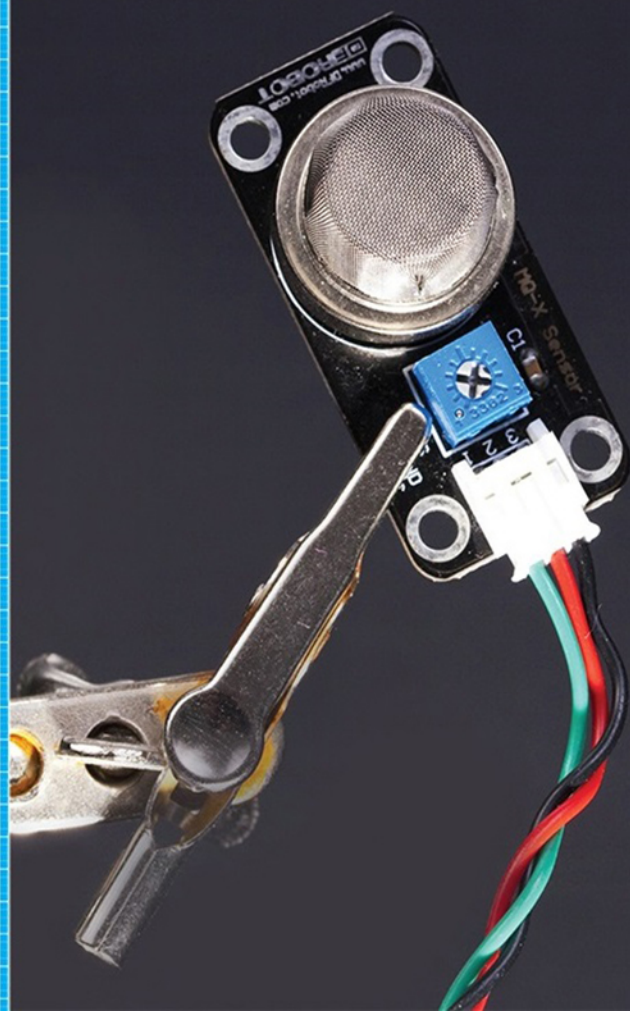


Делаем сенсоры

Проекты сенсорных
устройств на базе Arduino
и Raspberry Pi



Теро Карвинен
Киммо Карвинен
Вилле Валтокари

Make: Sensors

**Tero Karvinen
Kimmo Karvinen
and Ville Valtokari**



Делаем сенсоры

ПРОЕКТЫ СЕНСОРНЫХ УСТРОЙСТВ НА БАЗЕ
ARDUINO И RASPBERRY PI

Теро Карвинен
Киммо Карвинен
Вилле Валтокари



Москва • Санкт-Петербург • Киев
2015

ББК 32.85
K21
УДК 621.38

Издательский дом “Вильямс”
Главный редактор *С.Н. Тригуб*
Зав. редакцией *В.Р. Гинзбург*
Перевод с английского и редакция *И.В. Василенко*

По общим вопросам обращайтесь в Издательский дом “Вильямс” по адресу:
info@williamspublishing.com, <http://www.williamspublishing.com>

Карвинен, Теро, Карвинен, Киммо, Вальтокари, Вилле.

K21 Делаем сенсоры: проекты сенсорных устройств на базе Arduino и Raspberry Pi. :
Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2015. — 432 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1954-0 (рус.)

ББК 32.85

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства O'Reilly Media, Inc.

Authorized Russian translation of the English edition of *Make: Sensors* (ISBN 978-1-449-36810-4) © 2014 Tero Karvinen, Kimmo Karvinen, and Ville Valtokari, published by Maker Media Inc.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the Publisher.

Научно-популярное издание

Теро Карвинен, Киммо Карвинен, Вилле Вальтокари
Делаем сенсоры: проекты сенсорных устройств
на базе Arduino и Raspberry Pi

Литературный редактор	<i>И.А. Попова</i>
Верстка	<i>Л.В. Чернокозинская</i>
Художественный редактор	<i>В.Г. Павлютин</i>
Корректор	<i>Л.А. Гордиенко</i>

Подписано в печать 31.07.2015. Формат 70х100/16

Гарнитура Times.

Усл. печ. л. 34,83. Уч.-изд. л. 23,2.

Тираж 500 экз. Заказ № 4252

Отпечатано способом ролевой струйной печати

в АО «Первая Образцовая типография»

Филиал «Чеховский Печатный Двор»

142300, Московская область, г. Чехов, ул. Полиграфистов, д.1

ООО “И. Д. Вильямс”, 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-1954-0 (рус.)

ISBN 978-1-449-36810-4 (англ.)

© 2015, Издательский дом “Вильямс”

© 2014 Tero Karvinen, Kimmo Karvinen, and
Ville Valtokari

Оглавление

Введение	17
Глава 1. Знакомство с Raspberry Pi	29
Глава 2. Знакомство с Arduino	57
Глава 3. Расстояние	65
Глава 4. Дым и газ	103
Глава 5. Прикосновение	125
Глава 6. Движение	163
Глава 7. Свет	201
Глава 8. Ускорение	243
Глава 9. Идентификация	285
Глава 10. Электричество и магнетизм	325
Глава 11. Звук	363
Глава 12. Погода и климат	377
Приложение А. Краткий справочник по командам Linux в Raspberry Pi	421
Предметный указатель	423

Содержание

Об авторах	15
Введение	17
Сначала была идея...	17
Как читать эту книгу	19
Ввод, обработка и вывод данных	20
Интерфейсы	21
Самостоятельная работа	23
Приобретение электронных компонентов	26
Соглашения, принятые в этой книге	26
Программные коды примеров	27
Ждем ваших отзывов!	28
Глава 1. Знакомство с Raspberry Pi	29
Raspberry Pi — с нуля и до первой загрузки ОС	31
Распаковка файла NOOBS*.zip	31
Подключение оборудования ввода-вывода	32
Загрузка и установка Raspbian	32
Устранение неполадок запуска Raspberry Pi	35
Старая добрая Linux	37
Командная оболочка (как же без нее)	37
Быстрое знакомство	38
Настройки, хранимые в текстовых файлах	39
sudo — это наше все	39
Подключение оборудования к Raspberry Pi	42
Привет всем! Мигание светодиодом	42
Построение электрической цепи	44
Две системы обозначения: функциональная и последовательная	45
Управление GPIO из командной оболочки	46
Запись в файл, минуя редактор	46
Управление светодиодом	47
Устранение неполадок	48

Управление портом GPIO без прав суперпользователя	50
Устранение неполадок при работе с GPIO	52
Управление GPIO из Python	53
Старый служака Python	53
Что дальше?	56
Глава 2. Знакомство с Arduino	57
Базовая настройка Arduino	58
Ubuntu Linux	58
Windows 7 и Windows 8	59
OS X	60
Приветствие светодиодом	60
Структура программы Arduino	61
Дополнительные модули: простота и удобство использования	62
Глава 3. Расстояние	65
Эксперимент: измерение расстояния ультразвуковым датчиком Ping	66
Подключение к Arduino и программа управления датчиком Ping	68
Подключение к Raspberry Pi и программа управления датчиком Ping	70
Эксперимент: измерение расстояния ультразвуковым датчиком HC-SR04	73
Подключение к Arduino и программа управления датчиком HC-SR04	73
Подключение к Raspberry Pi и программа управления датчиком HC-SR04	75
Расчет времени возвращения эхо-сигнала	77
Эксперимент в окружающей среде: невидимые объекты	79
Эксперимент в окружающей среде: обнаружение преград датчиком инфракрасного излучения	80
Подключение к Arduino и программа управления датчиком инфракрасного излучения	81
Подключение к Raspberry Pi и программа управления датчиком инфракрасного излучения	82
Эксперимент: инфракрасное зрение	83
Эксперимент в окружающей среде: слежение за перемещением объекта (составной датчик инфракрасного излучения)	85
Подключение к Arduino и программа управления составным датчиком инфракрасного излучения	87
Подключение к Raspberry Pi и программа управления составным датчиком инфракрасного излучения	89
Подключение библиотеки SpiDev	92
Другие варианты подключения датчика инфракрасного излучения к Raspberry Pi	93
Пилотный проект: контроль осанки (Arduino)	94
Получаемые навыки	95
Пьезоэлектрический зуммер	95

Сирена	97
Сочетание зуммера с инфракрасным датчиком	98
Заключение сигнализации в корпус	100
Глава 4. Дым и газ	103
Эксперимент: выявление дыма (аналоговый газовый датчик)	104
Подключение к Arduino и программа управления датчиком MQ-2	105
Подключение к Raspberry Pi и программа управления датчиком MQ-2	107
Эксперимент в окружающей среде: задымление помещения	108
Эксперимент: алкотестер (датчик уровня алкоголя MQ-303A)	110
Эксперимент в окружающей среде: жизнь без алкоголя	113
Пилотный проект: отправка извещения о задымленности по электронной почте	114
Получаемые навыки	114
Отправка электронных писем и извещений с помощью Python	115
Тестирование оборудования	115
Почтовый клиент	116
Трудности отправки извещений из Arduino	117
Программа отправки извещений из Raspberry Pi	117
Корпус для дымовой сигнализации	120
Глава 5. Прикосновение	125
Эксперимент: нажатие кнопки	126
Подтягивающий (нагрузочный) резистор	127
Подключение к Arduino и программа управления кнопкой	127
Подключение к Raspberry Pi и программа управления кнопкой	129
Эксперимент: микропереключатель	131
Подключение к Arduino и программа управления микропереключателем	131
Подключение к Raspberry Pi и программа управления микропереключателем	133
Эксперимент: потенциометр (переменный резистор)	135
Подключение к Arduino и программа управления потенциометром	136
Подключение к Raspberry Pi и программа управления потенциометром	137
Эксперимент: касание без прикосновения (емкостный датчик прикосновения QT113)	140
Подключение к Arduino и программа управления датчиком прикосновения QT113	141
Подключение к Raspberry Pi и программа управления датчиком прикосновения QT113	142
Эксперимент в окружающей среде: распознавание прикосновения через дерево	143
Эксперимент: почувствуй нажим (датчик FlexiForce)	144
Подключение к Arduino и программа управления датчиком FlexiForce	145

Подключение к Raspberry Pi и программа управления датчиком FlexiForce	146
Эксперимент: создание собственного датчика прикосновения	148
Подключение к Raspberry Pi и программа управления собственным датчиком прикосновения	150
Пилотный проект: сенсорный звонок	151
Получаемые навыки	152
Сервоприводы	152
Подключение к Arduino и программа управления сенсорным звонком	158
Подключение сервопривода к звонку	160
Глава 6. Движение	163
Эксперимент: где верх, а где низ (датчик наклона)?	163
Подключение к Arduino и программа управления датчиком наклона	164
Подключение к Raspberry Pi и программа управления датчиком наклона	165
Эксперимент: вибродатчик, или цифровой датчик вибрации	166
Подключение к Arduino и программа управления датчиком вибрации	167
Подключение к Raspberry Pi и программа управления датчиком вибрации	168
Эксперимент: поверни до упора (датчик угла поворота)	170
Подключение к Arduino и программа управления кодовым датчиком угла поворота	171
Подключение к Raspberry Pi и программа управления кодовым датчиком угла поворота	172
Эксперимент: джойстик под большой палец (аналоговый двухкоординатный резистивный джойстик)	174
Подключение к Arduino и программа управления двухкоординатным джойстиком	176
Подключение к Raspberry Pi и программа управления джойстиком	177
Эксперимент в окружающей среде: вторая жизнь старого игрового контроллера	179
Эксперимент: охранная сигнализация (пассивный инфракрасный датчик движения)	181
Подключение к Arduino и программа управления охранной сигнализацией	181
Подключение к Raspberry Pi и программа управления охранной сигнализацией	183
Эксперимент в окружающей среде: взлом охранной сигнализации	185
Пилотный проект: электронная игра	188
Получаемые навыки	189
Подключение контроллеров	190
Корпус игрового контроллера	194
Автоматический запуск игры при загрузке Raspberry Pi	197
Запуск игры при регистрации	197
Автоматический вход	198

Глава 7. Свет	201
Эксперимент: обнаружение пламени (датчик пламени)	201
Подключение к Arduino и программа управления датчиком пламени	203
Подключение к Raspberry Pi и программа управления датчиком пламени	204
Эксперимент в окружающей среде: ярче пламя!	205
Эксперимент: увидеть свет (фоторезистор)	206
Подключение к Arduino и программа управления фоторезистором	208
Подключение к Raspberry Pi и программа управления фоторезистором	209
Эксперимент в окружающей среде: направленный свет	210
Эксперимент: следи за линией (детектор линий)	212
Подключение к Arduino и программа управления детектором линий	213
Подключение к Raspberry Pi и программа управления детектором линий	214
Эксперимент в окружающей среде: черное или белое?	215
Эксперимент в окружающей среде: все цвета радуги	218
Подключение к Arduino и программа управления датчиком цвета	219
Подключение к Raspberry Pi и программа управления датчиком цвета	221
Пилотный проект: цветовой купол	224
Получаемые навыки	224
RGB-светодиод	224
Масштабирование входных и выходных значений	231
Объединение программного кода	232
Корпус в виде полусферы	237
Глава 8. Ускорение	243
Ускорение и угловая скорость	244
Эксперимент: определение ускорения датчиком MX2125	244
Определение длительности импульса датчика MX2125	245
Подключение к Arduino и программа управления акселерометром	248
Подключение к Raspberry Pi и программа управления акселерометром	249
Эксперимент: совмещение акселерометра и гироскопа	250
Подключение к Arduino и программа управления устройством MPU 6050	252
Подключение к Raspberry Pi и программа управления устройством MPU 6050	258
Шестнадцатеричная, двоичная и другие системы счисления	262
Побитовые операции	266
Эксперимент в окружающей среде: подключение контроллера Wii Nunchuk к порту I ² C	269
Подключение к Arduino и программа управления контроллером Wii Nunchuk	271
Подключение к Raspberry Pi и программа управления контроллером Wii Nunchuk	275
Пилотный проект: управление механическим манипулятором с помощью Wii Nunchuk	277

Получаемые навыки	277
Устройство механического манипулятора	282
Глава 9. Идентификация	285
Цифровая клавиатура	286
Подключение к Arduino и программа управления цифровой клавиатурой	287
Подключение к Raspberry Pi и программа управления цифровой клавиатурой	290
Эксперимент в окружающей среде: снимаем отпечатки пальцев	292
Дактилоскопический сканер GT-511C3	294
Подключение к Arduino Mega и программа управления дактилоскопическим сканером	296
Подключение к Raspberry Pi и программа управления дактилоскопическим сканером	302
Модуль радиочастотной идентификации ELB149C5M	308
Подключение к Arduino Mega и программа управления модулем радиочастотной идентификации	310
Подключение к Raspberry Pi и программа управления модулем радиочастотной идентификации	312
Пилотный проект: старинный сундук с современным замком	315
Получаемые навыки	316
Управление сундуком с сокровищами	316
Старинный сундук	316
Подключение к Arduino и программа управления сундуком с сокровищами	319
Кто или что ты?	324
Глава 10. Электричество и магнетизм	325
Эксперимент: определение напряжения и тока	325
Подключение к Arduino и программа управления датчиком тока/напряжения AttoPilot	327
Подключение к Raspberry Pi и программа управления датчиком тока/напряжения AttoPilot	329
Эксперимент: определение напряженности магнитного поля	330
Подключение к Arduino и программа управления датчиком Холла	331
Подключение к Raspberry Pi и программа управления датчиком Холла	333
Эксперимент: определение северного магнитного полюса компасом-акселерометром LSM303	334
Калибровка компаса	336
Подключение к Arduino и программа управления компасом LSM303	336
Подключение к Raspberry Pi и программа управления компасом LSM303	341

Рабочий протокол модуля LSM303	346
Вычисление направления по компасу	346
Эксперимент: переключатель на эффекте Холла	348
Подключение к Arduino и программа управления переключателем на эффекте Холла	349
Подключение к Raspberry Pi и программа управления переключателем на эффекте Холла	350
Пилотный проект: интернет-мониторинг рабочего напряжения фотоэлемента	352
Получаемые навыки	352
Подключение фотоэлемента	353
Создание веб-сервера на базе Raspberry Pi	356
Определение своего IP-адреса	356
Создание в Raspberry Pi домашней страницы	357
Подключение к Raspberry Pi и программа мониторинга рабочих параметров фотоэлемента	357
Отсроченные задания и планировщик cron	360
Что дальше?	361
Глава 11. Звук	363
Эксперимент: запись звука и настройка уровня громкости	363
Подключение к Arduino и программа управления микрофоном	364
Подключение к Raspberry Pi и программа управления микрофоном	365
Эксперимент в окружающей среде: услышать падение булавки	367
Пилотный проект: визуализация звука через HDMI-порт	368
Получаемые навыки	368
Включение последовательного порта в Raspberry Pi	368
Подключение к Raspberry Pi и программа визуализации звука	369
Быстрое преобразование Фурье	372
Что дальше?	374
Глава 12. Погода и климат	377
Эксперимент: насколько жарко в помещении?	377
Подключение к Arduino и программа управления температурным датчиком LM35	378
Подключение к Raspberry Pi и программа управления температурным датчиком LM35	380
Эксперимент в окружающей среде: изменение температуры	381
Эксперимент: определение влажности	382
Влажность выдыхаемого воздуха	383
Подключение к Arduino и программа управления датчиком DHT11	384
Подключение к Raspberry Pi и программа управления датчиком DHT11	386
Доступ к Arduino из Raspberry Pi	388

Датчик атмосферного давления GY65	389
Подключение к Arduino и программа управления датчиком GY65	391
Библиотеки Arduino	392
Описание библиотеки Arduino gy_65	393
Подключение к Raspberry Pi и программа управления датчиком GY65	398
Эксперимент в окружающей среде: автоматический полив (датчик влажности почвы)	401
Подключение к Arduino и программа управления датчиком влажности почвы	402
Подключение к Raspberry Pi и программа управления датчиком влажности почвы	403
Пилотный проект: прогноз погоды с выводом на электронную бумагу	404
Получаемые навыки	404
Подключение датчиков к Arduino и программа получения прогноза погоды	406
Эксперимент в окружающей среде: без источника питания	413
Хранение изображений в заголовочных файлах	414
Преобразование растровых файлов в программный код C	415
Корпус для метеостанции	417
Приложение А. Краткий справочник по командам Linux в Raspberry Pi	421
Предметный указатель	423

Об авторах

Теро Карвинен изучал Linux и встроенные микроконтроллерные системы в Университете прикладных наук Хаага-Хелия, где он сейчас занимается разработкой образовательных программ и учебных курсов по беспроводным технологиям. До этого ему посчастливилось поработать на должности руководителя небольшого рекламного агентства. К его научным достижениям также можно приобщить диплом магистра экономических наук.

Киммо Карвинен в настоящее время занимает должность генерального директора ведущей финской компании, специализирующейся на производстве систем автоматической обработки аудио- и видеоинформации. До этого он занимался решением логистических задач в строительной компании, специализирующейся на возведении “умных” домов, руководил отделом по налаживанию деловых связей, а также был креативным директором рекламного агентства. Карвинен имеет степень магистра искусств и в настоящее время готовится к получению степени доктора наук в Хельсинкском технологическом университете.

Вилле Валтокари является ведущим программистом в компании, занимающейся производством всевозможных систем автоматизации. До этого он успешно разрабатывал и создавал программные решения для передовых систем обработки аудио- и видеоинформации. В его послужном списке многочисленные проекты по разработке и производству видеоигр, созданию робототехнических устройств и исследованию технических возможностей всевозможного оборудования.

Введение

Мы очень рады, что ваш выбор пал на нашу книгу. С ее помощью вы совсем скоро научитесь конструировать устройства, умеющие анализировать окружающую их действительность, представленную реальной средой и обладающую четкими физическими характеристиками. К ним относятся, например, газовые анализаторы и охранные сигнализации. Они не только представляют физические параметры в виде точных числовых значений, но и умеют выполнять различные действия, основанные на результатах анализа окружающей среды.

Современные датчики умеют много чего: измерять температуру, давление, световые потоки, ускорение и выводить полученные результаты в виде числовых данных, например 22 °C, 1015 мБар, 2,3 g. Обратите внимание на то, что в случае светового анализатора возвращается не числовое значение, а значение булева типа — истина или ложь, — и с этим вы также столкнетесь в одном из проектов, описанных в этой книге.

Центром управления всех рассматриваемых в книге проектов является микроконтроллерная система, для эффективного использования которой вам придется научиться создавать программы управления оборудованием, подключаемым к микроконтроллерной плате. В книге вы познакомитесь с двумя популярнейшими на сегодняшний день платформами: Arduino и Raspberry Pi. Каждая из них не требует глубоких познаний в области программирования для управления электронным оборудованием, поддерживаемым микроконтроллерной системой.

Сначала была идея...

Если вы не хотите тратить половину жизни на изучение основ электротехники, а мечтаете как можно скорее приступить к созданию многофункциональных гаджетов, роботов и других полезных устройств, то вы выбрали правильную книгу. В ней мы покажем, как практически и в кратчайшие сроки реализовать все возникшие в голове идеи.

Стоит заметить, что любые знания становятся полезными лишь тогда, когда к ним проявляется постоянный интерес. Не имея выхода в виде практической задачи, приобретенные ранее умения и навыки становятся бессмысленными и быстро забываются. Поэтому не бойтесь проявлять творческую инициативу, смело экспериментируйте, почаще воплощайте свои идеи и не забывайте делиться своими изысканиями с интернет-сообществом.

В каждой части вы найдете описание отдельного пилотного проекта, в котором для получения необходимого результата применяется несколько несвязанных между собой технологий. К примеру, в одной из глав описан проект создания деревянного сундука, открывающегося только после сканирования отпечатков пальцев владельца, а в другой главе рассматривается проект произвольно изменяющего свой оттенок сферического купола. Эти забавные устройства при первом знакомстве сложно воспринимать серьезно, но полученные при их реализации знания станут вашей отправной точкой в мире инновационных решений.

Навыки, полученные при работе с Arduino, вы легко сможете использовать при решении многих серьезных задач, имеющих практическое применение. Так, например, на рис. 1 показан прототип датчика солнечного излучения, сконструированный на базе Arduino, который планируется использовать в первом финском спутнике.



Рис. 1. Финляндия планировала запустить свой первый спутник в 2014 году. В этом же году мы сконструировали прототип анализатора солнечного излучения на базе Arduino

У вас непреодолимое влечение к конструированию новых вещей? Замечательно! С помощью этой книги вы сможете реализовать себя, построив работающий прототип желаемого устройства. Вместо того чтобы часами изучать технические характеристики электронных компонентов, воспользуйтесь описанием уже работающей электрической цепи и простой программы управления микроконтроллерной платформой. В своих проектах вы будете использовать датчики для изучения окружающей среды и определения способов взаимодействия ваших прототипов с объектами в ней. Каждый из проектов будет создаваться из отдельных элементов, зачастую повторяющихся, но, в отличие от детских конструкторов, таких как “Мессано” и “Lego”, раскрывающиеся перед вами возможности будут неисчерпаемыми благодаря Arduino и Raspberry Pi.

Чтобы подобрать для проекта правильный датчик, нужно однозначно определить-ся с измеряемой характеристикой. Данная книга структурирована так, что в каждом проекте рассматривается отдельный тип датчиков:

- датчик расстояния (глава 3);
- анализаторы дыма и газа (глава 4);
- датчик прикосновения (глава 5);
- датчик движения (глава 6);
- фоторезистор (глава 7);
- акселерометр и гироскоп (глава 8);
- дактилоскопический сканер (глава 9);
- датчик электрического и магнитного полей (глава 10);
- микрофон (глава 11);
- климатические датчики (глава 12).

Вы также можете использовать книгу как кладезь новых идей для будущих проектов, поэтому после прочтения не прячьте ее далеко.

Детально принципы взаимодействия датчиков с микроконтроллерными платформами Arduino и Raspberry Pi рассматриваются при описании кодов соответствующего программного решения. Весь программный код, предназначенный для управления прототипами устройств, которые описаны в книге, полностью законченный, что позволяет сразу же применять его для решения практических задач. Поняв принципы управления датчиками в микроконтроллерных системах, вы легко сможете управиться с любым сенсорным оборудованием, даже не описанным в книге, или тем, что еще только будет производиться в обозримом будущем.

Выбирая датчики, используемые в проектах книги, мы старались охватить самый широкий спектр устройств. При этом к критериям отбора не относилась сложность применяемого оборудования. Таким образом, вы познакомитесь с готовыми решениями для Arduino и Raspberry Pi различного уровня сложности.

В каждой главе вы найдете тематические разделы, посвященные описанию применяемого оборудования, принципов управления им в микроконтроллерных системах, а также рассмотрению автономно работающих прототипов устройств, в которых применяются вышеперечисленные датчики и анализаторы.

1. **Эксперимент.** Этот раздел содержит краткое описание датчика и принципов его подключения к Arduino и Raspberry Pi. Информация данных разделов впоследствии будет использована в практических проектах, демонстрирующих возможности датчиков и очерчивающих область их практического применения.
2. **Эксперимент в окружающей среде.** Этот раздел посвящен описанию проектов, демонстрирующих принципы управления датчиками в реальной рабочей среде. Внимательно изучив весь приведенный в нем материал, вы узнаете, как датчики анализируют физическое пространство и представляют его характеристики микроконтроллеру.
3. **Пилотный проект.** Датчики изучать намного интересней, если знать, как обрабатывать получаемую от них информацию. В данных разделах описываются проекты законченных устройств, базирующихся на обработке данных, поступающих с одного датчика. Кроме того, в них вы узнаете о принципах обработки выходных сигналов и использования их для управления такими устройствами, как RGB-светодиоды, электронная бумага и сервоприводы. Пилотные проекты — это ваша отправная точка в удивительный мир собственных разработок.

Ввод, обработка и вывод данных

Работоспособность робота или другого сконструированного вами устройства обеспечивается благодаря реализации трех базовых принципов управления данными: ввод, обработка и вывод.

1. Поскольку большинство создаваемых нами прототипов устройств не оснащаются клавиатурой и мышью, то входные данные микроконтроллерная система будет получать от датчиков. Просмотрите оглавление к книге, чтобы ознакомиться с их разнообразием. Учтите, что в реальности типов датчиков существует намного больше, чем мы смогли описать в книге. Вы будете приятно удивлены, узнав, как много можно узнать об окружающем мире, изучив его с помощью специальных анализаторов.
2. Обработка данных осуществляется в программе, выполняемой микроконтроллером, который установлен на плате Arduino или Raspberry Pi. В программном коде определяются все основные действия по управлению данными и способы их последующей передачи исполняющим устройствам.
3. Вывод данных обычно сопровождается изменением отдельных характеристики среды, в которой функционирует устройство. Вы можете зажечь светодиод, включить сервопривод или воспроизвести звук. Это типичные для платформ Arduino и Raspberry Pi способы вывода информации, хотя существуют и другие методы, менее распространенные, но не менее эффективные в контексте решаемой задачи (например, тактильная отдача, реализуемая вибродвигателем, вывод графической информации на электронной бумаге или включение целевого бытового прибора).

Интерфейсы

Интерфейс определяет способ взаимодействия датчика с микроконтроллерной платой, такой как Arduino или Raspberry Pi. Интерфейс устанавливает способ физического подключения датчика к системе и методы программного управления им для получения параметров среды.

Несмотря на существование огромного количества датчиков всевозможных типов, количество протоколов обмена данными с ними весьма ограничено. С ними вы познакомитесь далее при детальном рассмотрении готовых проектов. Здесь же мы лишь вкратце остановимся на их представлении.

Базовые интерфейсы обмена данными с датчиками описаны в табл. 1.

Цифровые датчики

Некоторые датчики, к которым относится и кнопка, имеют два состояния: “Вкл.” и “Выкл.” Такие устройства просты в управлении. Включенное состояние характеризуется подачей сигнала высокого уровня (HIGH), передаваемого на входной порт микроконтроллерной системы. Как правило, такой сигнал представлен напряжением 3,3 или 5 В, в зависимости от используемой платформы.

Аналоговые датчики

Аналоговые датчики плавно изменяют свое сопротивление в зависимости от изменения характеристик среды, в которую они помещены (подобным образом работают регуляторы громкости, хотя они ничего не измеряют). В Arduino и Raspberry Pi изменения в сопротивлении представляются падением напряжения, наблюдаемом на датчике. В самом простом случае добиться изменения напряжения в цепи можно с помощью потенциометра, который всего лишь увеличивает или уменьшает свое сопротивление. Управлять потенциометром в Arduino проще простого, а вот в Raspberry Pi вам для этих целей потребуется специальный преобразователь, позволяющий обрабатывать входные аналоговые данные. С аналого-цифровым преобразователем MCP3002, используемым в Raspberry Pi, вы детально познакомитесь в главе 3. В подавляющем большинстве аналоговых датчиков выходной сигнал представлен сопротивлением, поэтому их еще называют резистивными датчиками.

Импульсные датчики

В некоторых датчиках выходной сигнал представляется импульсами определенной ширины или длительности, указывающей количество времени, в течение которого на вход микроконтроллерной системы подается высокое напряжение. Импульсный сигнал считывается функциями `pulseIn()` и `gpio.pulseInHigh()`. Так как управление сигналом осуществляется программными средствами, вам не придется заниматься низкоуровневым управлением микроконтроллерной платформы, в котором задействованы прерывания; эту задачу на себя берет соответствующая программная библиотека.

Последовательное соединение

Как правило, последовательное соединение задействуется устройствами при обмене текстовыми данными. С помощью такого же метода передачи данных компьютер подключается к Arduino через USB-порт. Вы детально ознакомитесь с последовательным соединением в проектах, в которых требуется выводить сообщения на монитор последовательного порта Arduino.

I²C

Интерфейс I²C — это популярный промышленный стандарт обмена данными между компьютерными устройствами. Он поддерживается настольными компьютерами и ноутбуками, а нам пригодится при управлении контроллером Wii Nunchuk. К шине I²C можно подключить до 128 устройств. В книге мы дважды воспользуемся готовыми решениями и программным кодом для управления датчиками, подключенными к микроконтроллерной системе через I²C.

SPI

Последовательный периферийный интерфейс (Serial Peripheral Interface — SPI) представляет собой еще один промышленный интерфейс обмена данными. Как будет показано далее, он лучше всего подходит для подключения аналого-цифрового преобразователя в Raspberry Pi. Учтите, что создание программного кода с нуля для новых устройств, подключаемых через SPI, требует немало знаний и усилий.

Ручное программирование

Встречаются ситуации, когда подключаемый к платформе датчик настолько необычен, что ни один из стандартных интерфейсов не обеспечивает его поддержки. В подобных случаях вам придется самостоятельно написать программный код управления им. Это трудоемкий процесс, поскольку в нем управление данными, поступающими с датчика, осуществляется на битовом уровне. Пример ручного программирования вы встретите в главе 12 при рассмотрении проекта управления датчиком влажности.

Изучая работу датчиков, вы вскоре познакомитесь со всеми представленными выше интерфейсами. Если вам не терпится поскорее приступить к созданию собственных устройств, не вдаваясь в коммуникационные подробности управления датчиком, то смело используйте программный код, приведенный в книге, а детальное рассмотрение тонкостей процесса оставьте “на потом”.

Таблица 1. Интерфейсы подключения датчиков к микроконтроллерным системам

Интерфейс	Примеры значений	Arduino	Raspberry Pi	Примеры датчиков
Цифровой	0 или 1	<code>digitalRead()</code>	<code>botbook_gpio.read()</code>	Кнопка, переключатель на инфракрасном датчике, датчик наклона, пассивный инфракрасный датчик движения
Аналоговый	5%, 10%, 23 °C	<code>analogRead()</code>	<code>botbook_mcp3002.readAnalog()</code>	Потенциометр, фоторезистор, датчик измерения концентрации алкоголя MQ-3, газовые анализаторы семейства MQ X (дым, углеводороды, CO и т.п.), датчик давления

Интерфейс	Примеры значений	Arduino	Raspberry Pi	Примеры датчиков
Импульсный	20 мс	<code>pulseIn()</code>	<code>gpio.pulseInHigh()</code>	FlexiForce, датчик пламени KY-026, датчик цвета HDJD-S822-QR999, температурный датчик LM35, датчик влажности грунта
Последовательный	A9B3C5B3C5	<code>Serial.read()</code>	<code>pySerial.read()</code>	Ультразвуковой датчик расстояния HC-SR04, акселерометр MX2125
I ² C	(2,11 g, 0,0 g, 0,1 g), точные значения	<code>Wire.h</code>	<code>smbus</code>	Сканер отпечатков пальцев GT-511C3, датчик радиочастотной идентификации ELB149C5M
SPI	57°, точные значения	Вручную	<code>spidev</code>	Контроллер Wii Nunchuk, акселерометр-гироскоп MPU 6050, датчик атмосферного давления GY65
Пользовательский	53%	Вручную	Вручную	Аналого-цифровой преобразователь MCP3002 Датчик влажности DHT11

Самостоятельная работа

Большинству технических изыскателей недостаточно научиться собирать робот или другое функциональное устройство из готового набора компонентов. Разработка и создание такого набора компонентов — вот задача для истинных конструкторов.

Формат книги не предусматривает рассмотрение каждого типа датчиков более чем на одном примере, хотя вам совсем не обязательно в точности следовать приведенным в ней инструкциям. Попробуйте усовершенствовать проекты, воспользовавшись другими материалами и оборудованием.

Что вы скажете об использовании упаковочного картона (рис. 2), искусственного меха (рис. 3) или трехмерной печати (рис. 4)?

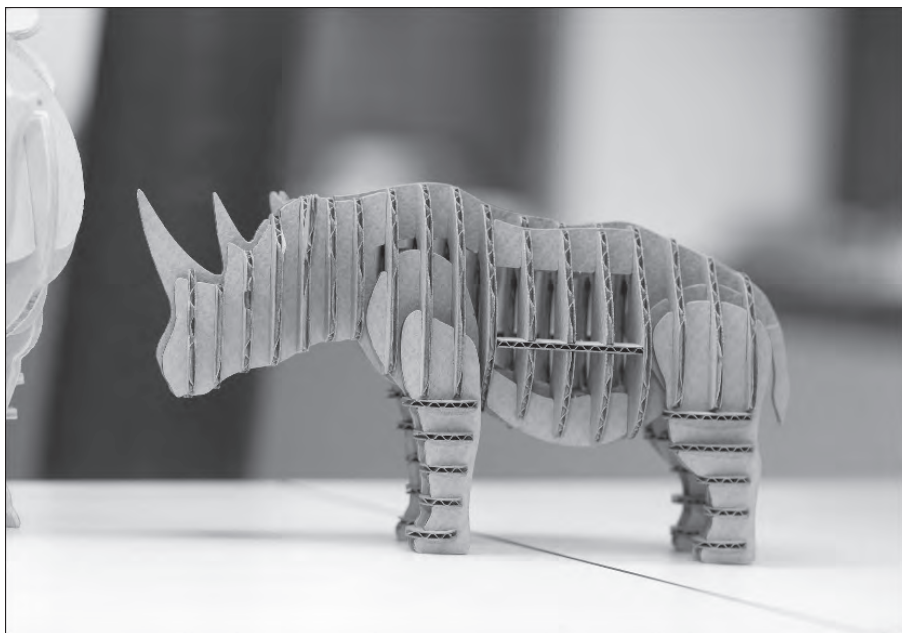


Рис. 2. Картонная модель (не наша). Фотография предоставлена художественным центром Ars Electronica, г. Линц (Австрия)



Рис. 3. “Меховой” робот (не наш). Фотография предоставлена художественным центром Ars Electronica, г. Линц (Австрия)



Рис. 4. Странный робот, распечатанный на 3D-принтере (не наш). Фотография предоставлена художественным центром Ars Electronica, г. Линц (Австрия)

Изучение и экспериментирование с новыми технологиями делает процесс конструирования увлекательнее, расширяя ваши познания во многих, казалось бы, сторонних отраслях. Вы, например, невзначай можете освоить технику лепки из глины, отливания форм, не говоря уже о такой рутине, как виртуозное владение паяльником (рис. 5).

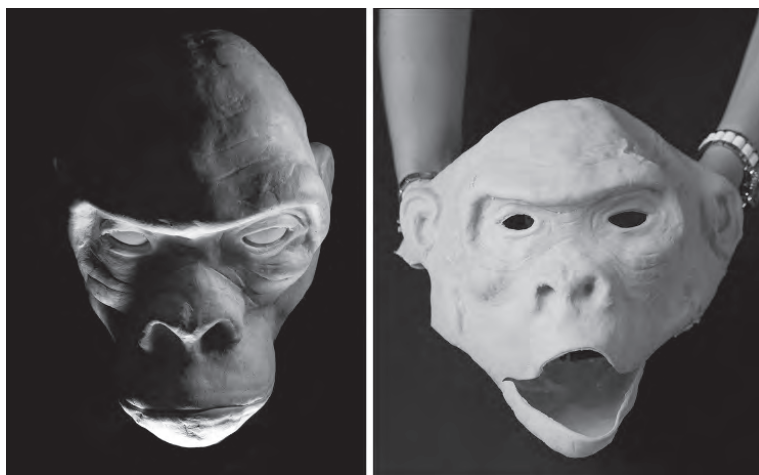


Рис. 5. Базовая модель, используемая для анимации головы гориллы, и латексная форма морды, полученная на ее основе

Можете попробовать использовать в своих изысканиях старые или выброшенные вещи. Несмотря на свою дешевизну (чаще всего они вообще бесплатны), они сделают ваш прототип совершенно уникальным.

Приобретение электронных компонентов

Если вы привыкли работать только с качественными компонентами, то обращайтесь в известные магазины электроники, поставляющие товары из Европы или США. Самые дешевые комплектующие вы традиционно найдете в магазинах, торгующих электронными компонентами, произведенными в Азии.

Дорожащие своей репутацией магазины торгуют устройствами таких торговых марок, как Maker Shed, SparkFun, Parallax и Adafruit. К слову, Maker Shed — это магазин, принадлежащий издателю данной книги. Компания SparkFun известна своими коммутационными платами, которые требуют от вас навыков работы с паяльником и припоем. Известность компании Parallax принесла Basic Stamp — микроконтроллерная система предыдущего поколения, активно закупаемая производителями электронного оборудования. В активе компании Adafruit огромное количество устройств собственной разработки. На сайтах SparkFun и Adafruit вы найдете всю необходимую информацию о производимом ими оборудовании, включая обучающие занятия по его использованию и важную техническую документацию.

В настоящее время даже крупные торговые сети магазинов электроники не брезгуют продажей микроконтроллерных платформ и компонентов к ним. Чтобы удостовериться в этом, достаточно поискать в каталогах товары по ключевым словам Arduino и Raspberry Pi.

Соглашения, принятые в этой книге

В тексте книги вы встретите такие условные обозначения.

- *Курсивом* выделяются новые понятия, ранее не определенные в тексте.
- Моноширинным шрифтом представляются электронные адреса, включая адреса почтовых ящиков, имена файлов, папок, а также расширения имен файлов. Кроме того, моноширинным шрифтом выделяется любой программный код, как представленный в виде отдельных листингов, так и отдельных ключевых слов. Также моноширинным шрифтом обозначаются команды или любой другой текст, вводимый пользователем с клавиатуры.
- Клавиши и комбинации клавиш, которые следует нажимать при выполнении описанных в книге операций, заключаются в угловые скобки, < >.

Таким форматированием в тексте книги выделяются советы, примечания и предупреждения.

Программные коды примеров

Файлы примеров, рассматриваемых в книге, можно скачать по такому адресу:

<http://makesensors.botbook.com>

Для распаковки файла архива дважды щелкните на нем или щелкните правой кнопкой мыши и выберите из контекстного меню команду Извлечь (Extract).

Кроме того, все файлы доступны также на сайте Издательского дома “Вильямс”:



go.dialektika.com/MakeSensors

Эта книга поможет вам при реализации собственных проектов. Вы всегда сможете использовать приведенный здесь программный код в собственных проектах и документации к устройствам. Для этого вам не обязательно связываться с нами и просить разрешения на применение файлов примеров в собственных решениях, за исключением случаев воспроизводства больших фрагментов книги в коммерческих целях. Например, написание собственной программы с использованием программного кода из примеров книги не требует специального разрешения с нашей стороны. А вот распространение дисков с файлами примеров или публикация их в Интернете без специального уведомления с нашей стороны запрещена. При этом размещение цитат из книги или фрагментов программного кода в собственных постах вполне легально. А вот добавление обширных фрагментов программного кода, написанного нами для своих проектов, в документации к вашим устройствам требует нашего разрешения.

Мы будем благодарны, если в своих проектах или работах вы будете ссылаться на нашу книгу. Но делать это вам не обязательно, хотя нам и будет приятно об этом знать. В ссылках лучше всего указывать полное название издания, всех его авторов и ISBN.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: info@williamspublishing.com

WWW: <http://www.williamspublishing.com>

Наши почтовые адреса:

в России: 127055, г. Москва, ул. Лесная, д. 43, стр. 1

в Украине: 03150, Киев, а/я 152

Знакомство с Raspberry Pi

1

Мы настоятельно рекомендуем начать изучение платформы Raspberry Pi с версии Model B, в которую встроен адаптер Ethernet, а USB-портов достаточно, чтобы подключить и мышь, и клавиатуру. Получив доступ к Интернету и используя привычные устройства ввода, вам будет намного проще изучать возможности этой микроконтроллерной системы.

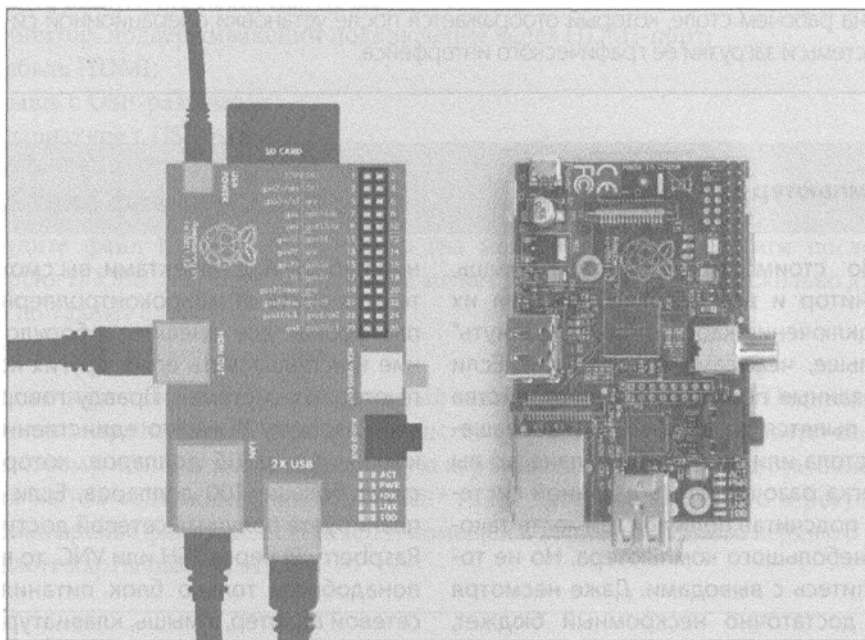


Рис. 1.1. Разъемы Raspberry Pi для подключения периферийных устройств

За исключением случаев приобретения Raspberry Pi в виде набора компонентов, система не заключена в корпус, поэтому при работе с ней вам придется проявлять осторожность. Если же у вас есть доступ к 3D-принтеру, станку с ЧПУ или лазерному резаку, то вы сможете легко создать корпус самостоятельно, воспользовавшись готовыми моделями, например, представленными на сайте <http://www.thingiverse.com>.

Карты памяти SD объемом 4 Гбайт вполне хватит для размещения на ней операционной системы. Карта памяти большего объема прослужит вам значительно дольше, поскольку количество операций записи и стирания данных в отдельных ее блоках при том же объеме рабочих данных будет меньше. Поэтому предпочтительнее все же использовать карты памяти объемом не менее 8 Гбайт.

В Raspberry Pi реализована поддержка мониторов высокого разрешения, а также вывода звука через HDMI. Телевизоры, оснащенные разъемом HDMI и поддерживающие видеосигнал высокой четкости, будут идеальным устройством вывода для проектов Raspberry Pi.

Клавиатура и мышь значительно упрощают управление платформой. К счастью, на плате Raspberry Pi Model B смонтировано два USB-порта, которых достаточно для подключения обоих устройств ввода.

Для добавления в систему адаптера Wi-Fi вам понадобится USB-концентратор с независимым питанием. Исчерпывающий список моделей адаптеров Wi-Fi, гарантированно работающих с Raspberry Pi, вы найдете по следующему адресу:

http://elinux.org/RPi_USB_Wi-Fi_Adapters

Настройка подключения Wi-Fi в Raspberry Pi производится после двойного щелчка на значке **WiFi Config** (Настройка беспроводных сетей), расположенном на рабочем столе, который отображается после установки операционной системы и загрузки ее графического интерфейса.

Компьютер за 35 долларов?

По стоимости клавиатура, мышь, монитор и все необходимые для их подключения кабели могут “потянуть” больше, чем сама Raspberry Pi. Если указанные периферийные устройства не пылятся в дальнем ящике вашего стола или на полке в чулане, то вы слегка разочаруетесь в данной системе, подсчитав полную стоимость такого небольшого компьютера. Но не топчитесь с выводами. Даже несмотря на достаточно нескромный бюджет, работая с Raspberry Pi, вы в кратчайшие сроки (не забывайте, что время — деньги) создадите комфортную среду разработки. Впоследствии, по оконча-

нии работы над проектами, вы сможете отключить от микроконтроллерной платформы все внешнее оборудование и использовать его в других компьютерных системах. Правду говорят, что Raspberry Pi — это единственный компьютер за 35 долларов, который стоит больше 100 долларов. Если вы планируете получить сетевой доступ к Raspberry Pi через SSH или VNC, то вам понадобится только блок питания и сетевой адаптер, а мышь, клавиатура и монитор будут задействоваться всего единожды — при начальной настройке системы.

Raspberry Pi — с нуля и до первой загрузки ОС

В этой главе вы вкратце ознакомитесь с настройкой и запуском Raspberry Pi, а также ее функциональными особенностями. Первое, что вам понадобится сделать, — это установить в Raspberry Pi операционную систему Linux. Эта задача выполняется следующим образом:

- загрузите и извлеките установочный пакет операционной системы на заранее отформатированную карту памяти SD;
- вставьте карту памяти в соответствующий разъем Raspberry Pi, а затем подключите к плате клавиатуру, мышь и монитор;
- подайте питание на микроконтроллерную систему, выберите устанавливаемую ОС и немного подождите.

Выполнив описанные выше действия, вы получите возможность загрузить в Raspberry Pi операционную систему Linux с графическим интерфейсом.

Для решения этой задачи понадобится такое оборудование:

- микроконтроллерная платформа Raspberry Pi версии Model B;
- зарядное устройство (блок питания) с кабелем, оснащенным разъемом Micro-USB;
- карта памяти SD емкостью не менее 4 Гбайт;
- монитор, поддерживающий подключение через HDMI-порт;
- кабель HDMI;
- мышь с USB-разъемом;
- клавиатура с USB-разъемом.

Распаковка файла NOOBS*.zip

Загрузите файл NOOBS_vX_Y_Z.zip (на момент написания книги последней была версия NOOBS_v1_3_4.zip, но вам может встретиться файл несколько другой версии) по такому адресу:

<http://raspberrypi.org/downloads>

Вставьте карту памяти SD в настольный компьютер или ноутбук. Подавляющее большинство карт памяти SD при производстве форматируются в файловую систему FAT32, поэтому, если вы используете новую карту памяти, вам будет достаточно распаковать на нее содержимое файла NOOBS*.zip. Удостоверьтесь, что в результате распаковки архива файл `bootcode.bin` помещен в корневом (самого верхнего уровня) каталоге карты памяти SD.

Для форматирования карты памяти SD воспользуйтесь одним из проверенных приложений сторонних производителей (мы рекомендуем SD Card Formatter последней версии).

В современных версиях Linux, Windows и Mac для распаковки файла NOOBS.zip достаточно дважды щелкнуть на нем или щелкнуть правой кнопкой мыши. В старых версиях Windows, чтобы распаковать файл архива, вам понадобится установить дополнительную программу — 7zip.

Подключение оборудования ввода-вывода

Подключение кабелей не должно вызвать у вас затруднений, поскольку все они вставляются в унифицированные разъемы. Мышь и клавиатура подключаются к Raspberry Pi через USB-порты. HDMI-монитор, как и следовало ожидать, подключается к Raspberry Pi с помощью HDMI-кабеля. Если вы подключаете к Raspberry Pi телевизор с системой цветности NTSC или PAL, то воспользуйтесь кабелем для композитного видеосигнала, в котором желтый разъем вставляется в соответствующее гнездо микроконтроллерной платы.

В конце подключите к Raspberry Pi кабель Micro-USB от источника питания. В гнездо Micro-USB на плате Raspberry Pi допускается подавать питание либо с USB-порта компьютера, либо с блока питания (зарядного устройства), обеспечивающего на выходе напряжение 5 В и ток не менее 700 мА.

Загрузка и установка Raspbian

Справившись с подключением к Raspberry Pi периферийных устройств и подав на плату питание, дождитесь загрузки системы. Нажимать кнопки и использовать другие переключатели при этом не нужно. Просто подождите.

Если на экран монитора ничего не выводится, то вам нужно правильно установить в Raspberry Pi режим вывода видеоданных. По умолчанию задан режим HDMI, но если экран монитора остается пустым даже при подключении к Raspberry Pi устройства HDMI, то нажмите на клавиатуре, подключенной к микроконтроллерной платформе, клавишу <2>, чтобы установить видеорежим **HDMI Safe Mode** (Безопасный режим HDMI). Если видеоданные передаются с помощью композитного (с желтым разъемом) кабеля, тогда нажмите клавишу <3> для выбора режима PAL, или <4>, чтобы выбрать режим NTSC.

Вскоре на экране появится графическое меню выбора различных операционных систем, языков и раскладок клавиатур. Выберите вариант Raspbian [RECOMMENDED] (рис. 1.2), а затем укажите предпочтительный язык и раскладку клавиатуры.

Если вам уже доводилось работать в таких системах, как Debian, Mint или Ubuntu, то далее вам все покажется до боли знакомым; если — нет, то внимательно следите за появляющейся на экране информацией и... у вас так и возникнет ощущение дежавю! Сама процедура установки занимает всего несколько минут (рис. 1.3). Вскоре на экране появится извещение о завершении установки операционной системы. Нажмите клавишу <Enter> или щелкните на кнопке ОК для перезагрузки системы.

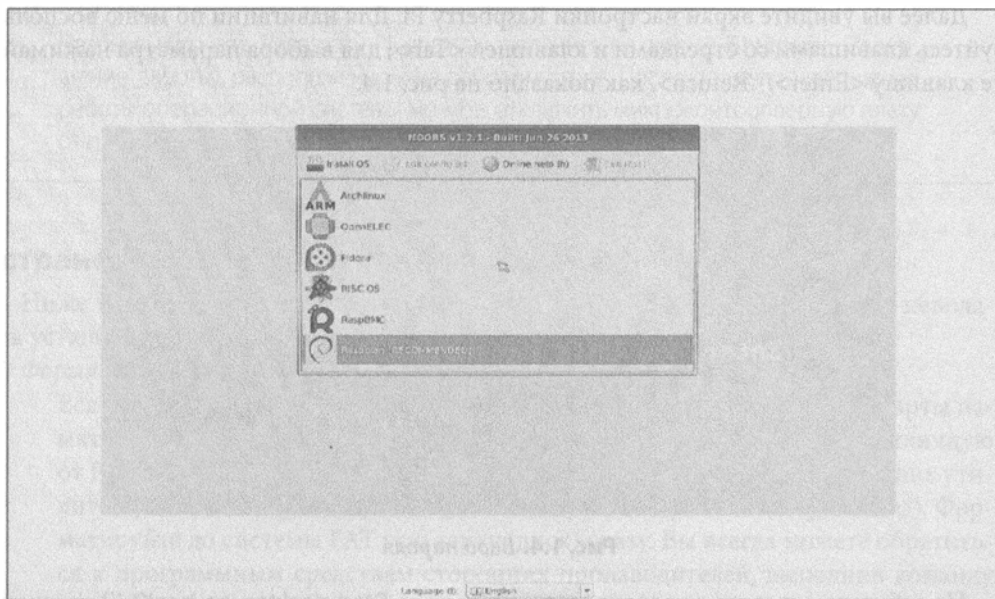


Рис. 1.2. Выбор операционной системы

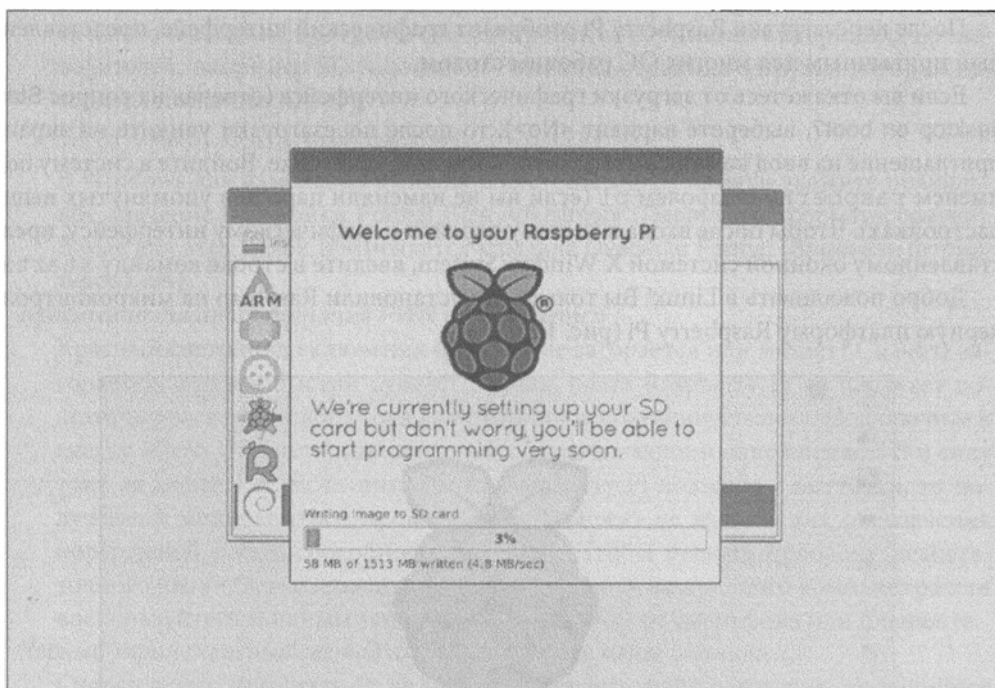


Рис. 1.3. Процесс установки Raspbian

Далее вы увидите экран настройки Raspberry Pi. Для навигации по меню воспользуйтесь клавишами со стрелками и клавишей <Tab>; для выбора параметра нажимайте клавишу <Enter>/<Return>, как показано на рис. 1.4.

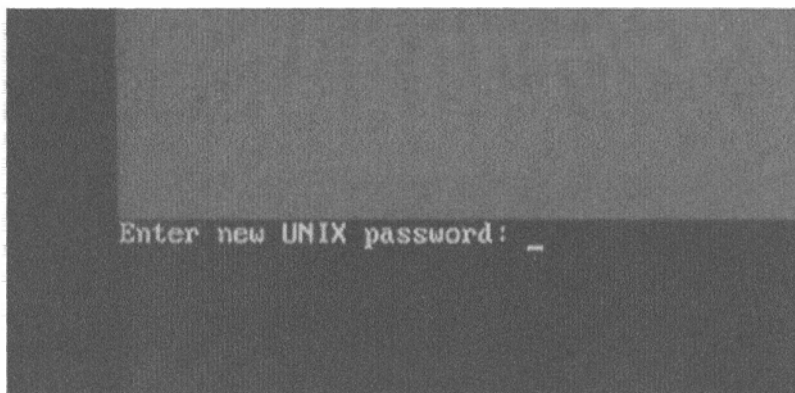


Рис. 1.4. Ввод пароля

Не забудьте ответить утвердительно на вопрос *Start desktop on boot?* (Загружать рабочий стол при запуске?). Завершив настраивать параметры системы, воспользуйтесь клавишей <Tab> для выбора кнопки *Finish* (Завершить) и перезагрузки ОС.

После перезагрузки Raspberry Pi отобразит графический интерфейс, представленный привычным для многих ОС рабочим столом.

Если вы откажетесь от загрузки графического интерфейса (отвечая на вопрос *Start desktop on boot?*, выберете вариант <No>), то после перезагрузки увидите на экране приглашение на ввод команд в стандартной командной строке. Войдите в систему под именем *raspberrypi* с паролем *pi* (если вы не изменяли пароль в упомянутых выше настройках). Чтобы после входа в систему перейти к графическому интерфейсу, представленному оконной системой X Window System, введите в строке команду *startx*.

Добро пожаловать в Linux! Вы только что установили Raspbian на микроконтроллерную платформу Raspberry Pi (рис. 1.5).

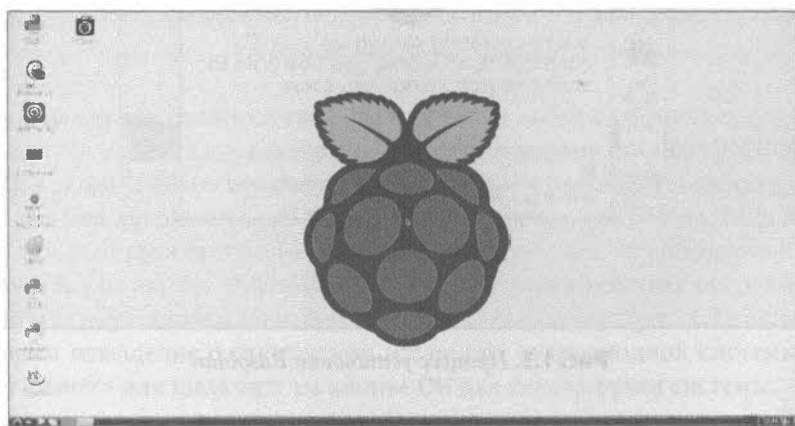


Рис. 1.5. Добро пожаловать в Linux

Чтобы выключить Raspberry Pi, дважды щелкните на значке **Shutdown** (Завершение работы), расположенном на рабочем столе. После полного завершения работы операционной системы можете отключить микроконтроллерную плату от питания.

Устранение неполадок запуска Raspberry Pi

Ниже приведены советы по устранению наиболее часто встречающихся неполадок установки и запуска ОС в Raspberry Pi.

Форматирование карты памяти как FAT32.

Если у вас возникли трудности с загрузкой операционной системы с карты памяти SD, то, скорее всего, она форматирована под файловую систему, отличную от FAT32. Чтобы исправить ситуацию, воспользуйтесь встроенной в Linux утилитой разбивки карты на диски (для ее запуска введите `sudo gparted`). Форматируйте до системы FAT всю карту (диск) сразу. Вы всегда можете обратиться к программным средствам сторонних производителей, выполнив команду `sudo palimpsest` (или `sudo gnome-disks`), хотя опытные пользователи никогда не упустят возможности отформатировать карту памяти с помощью утилит стандартной командной оболочки. В Windows и Mac используйте одну из многочисленных программ форматирования карт памяти сторонних производителей, например SD Association Formatter, страница загрузки которой находится по такому адресу:

https://www.sdcard.org/downloads/formatter_4/

В Windows для изменения файловой системы карты памяти укажите правильное значение настройки **Format Size Adjustment** (Параметры форматирования). В Mac для этих целей применяется команда **Overwrite Format** (Заменить форматирование).

Красный индикатор питания PWR не загорается

Красный светодиод включения питания не загорается или мигает? Сначала загорается, а затем быстро тухнет? Значит, плата Raspberry Pi не получает достаточно электрического тока для питания всего оборудования. Подключите к гнезду Micro-USB источник питания, обеспечивающий напряжение 5 В и силу тока не менее 1 А. Если питание на Raspberry Pi подается с ноутбука, то получаемой микроконтроллером мощности может не хватать для обеспечения нормальной работоспособности системы. Чтобы решить проблему недостаточного питания, подключите плату к USB-порту настольного компьютера или воспользуйтесь мощным зарядным устройством от смартфона или планшета.

Черный экран (красный светодиод (PWR) тем не менее светится)?

Скорее всего, Raspberry Pi не может прочитать файл-загрузчик, хранящийся на карте памяти SD. Отключите питание, извлеките из слота карту памяти и аккуратно вставьте ее снова. Удостоверьтесь, что первый в последовательности загружаемых указан файл `bootcode.bin`, а сам он содержится в корневом каталоге вставленной карты памяти SD. Если таким образом проблема не

устраняется, тогда заново отформатируйте карту памяти и повторно распакуйте на нее содержимое файла NOOBS*.zip. Если и это не помогло, то попробуйте использовать другую карту памяти SD.

На экране отображается четыре цветных квадрата

Загрузчик, находящийся на карте памяти SD, выполняется, а образ ядра операционной системы kernel.img не загружается. Повторно отформатируйте карту памяти SD и извлеките на нее содержимое архива NOOBS*.zip или же используйте другую карту памяти SD.

Загрузка прерывается сообщением об ошибках

Попробуйте отключить все подключенные к USB-портам устройства: клавиатуру, мышь и адаптер Wi-Fi (если последний используется). Оставьте подключенными к микроконтроллерной системе только карту памяти SD, монитор и источник питания. Извлеките и заново вставьте в слот карту памяти, чтобы обновить ее контакт с выводами разъема. Если проблема сохраняется, переформатируйте карту памяти SD и повторно распакуйте на нее содержимое файла NOOBS*.zip.

А где же операционная система?

Если стандартные команды не выполняются, экран постоянно засоряется ненужными элементами или Raspberry Pi внезапно прекращает свою работу, то не волнуйтесь. Удерживайте при загрузке нажатой клавишу <Shift>, а затем выберите вариант переустановки Raspbian. Это простой и быстрый способ, который тем не менее удаляет все данные на карте памяти. Если переустановка ОС не помогла решить проблему, то повторно отформатируйте карту памяти SD, извлеките на нее содержимое архива NOOBS*.zip и еще раз установите операционную систему.

Нет подключения к Интернету

Если вы подключили Ethernet-кабель к плате до загрузки операционной системы, то она будет беспрепятственно подключена к сети, как любое другое компьютерное устройство. О наличии соединения с сетью извещает загоревшийся на плате Raspberry Pi индикатор LNK. Если индикатор LNK не светится, то это означает, что Ethernet-соединение в Raspberry Pi не установлено. О наличии соединения извещает свечение индикаторов 100 (имеется в виду скорость соединения 100 Мбит/с или выше) и FDX (дуплексный режим). Если индикатор LNK загорается, а проблемы с подключением к сети сохраняются, то воспользуйтесь специальными средствами диагностики соединения: ifconfig (отображает настройки сетевого адаптера), route -n (выводит таблицу маршрутизации сетевого соединения), cat /etc/resolv.conf (выводит имя используемого сервера) и ping -c 1 google.com (показывает, доступна ли в текущий момент начальная страница Google).

Если вы столкнулись с проблемой, не перечисленной в приведенном выше списке, то скопируйте сообщение об ошибке в строку поиска Google. Указывайте сообщение в точности так, как видите на экране. Если сообщение об ошибке появляется на этапе загрузки операционной системы, то сделайте снимок экрана с помощью фотоаппарата или мобильного телефона, чтобы потом в спокойной обстановке изучить текст сообщения.

Вы уже знаете, что Raspberry Pi управляется операционной системой Linux. На самом деле платформа разрабатывалась изначально под Linux. Следует уточнить, что название “Linux” используется как для обозначения ядра операционной системы, так и самой операционной системы. Операционная система Linux состоит из ядра и множества дополнительных средств и приложений, выполняющих вспомогательные функции.

Учтите, что Raspberry Pi, хотя и относится к микрокомпьютерам, по производительности сильно уступает устройствам типа “рабочая станция”. С точки зрения общей классификации эта платформа больше тяготеет к простым планшетам и смартфонам. Несмотря на возможность загрузки в ней графического интерфейса, не стоит ожидать от Raspberry Pi функциональности и производительности настольного компьютера или ноутбука. Недостаточная производительность и ограниченный объем оперативной памяти делает невозможным запуск на ней даже таких стандартных приложений, как LibreOffice и Mozilla Firefox.

Командная оболочка (как же без нее)

Командная оболочка прошла испытание временем и вполне может стать тем, о чем вы будете рассказывать своим внукам. Текстовые команды, которые вводятся в ней, такие как `pwd`, `ls` и `cat`, возникли задолго до изобретения Linux. (Linux была разработана финским студентом в Хельсинки, работавшим всего в пяти километрах от ботанического сада, в котором писалась данная книга.) Опытные пользователи OS X и Windows давно привыкли обращаться к командному интерфейсу в тех случаях, когда мышь не в силах помочь решить поставленную задачу.

Почти все команды, выполняемые в Raspberry Pi, в точности повторяют таковые в компьютерах, работающих под управлением Mac OS и Linux, и подобны вводимым в командной строке Windows-систем.

Как вы уже, вероятно, знаете, подавляющее большинство веб-серверов работает под управлением Linux. Крупнейшие компании — Google, Facebook, Amazon — при размещении данных отдают предпочтение Linux. Многие суперкомпьютеры также

Командная оболочка управляет миром

Если ваш смартфон такой же, как у большинства других жителей планеты, то он работает под управлением Linux, и вы сможете выполнить на нем некоторые из утилит командной оболочки. Операционная система Android также поддерживает ограниченный набор средств командной строки даже без специального “разлочивания” и вскры-

тия всех функций среды (вам понадобится специальное приложение управления средой, такое как ZShaolin). Как и другие компьютеры, Apple, iPhone и iPad работают под управлением дальнего “родственника” Linux — BSD, поэтому вы сможете воспользоваться командной строкой и в iPhone, предельно “разлочив” его.

управляются Linux. Веб-серверы не требуют наличия графического интерфейса, и именно поэтому большинство программистов и системных администраторов просто обязаны уметь обращаться с командной оболочкой. Нет ничего удивительного в том, что в настольных компьютерах и ноутбуках с Linux используются такие же команды, как и при управлении интернет-серверами.

Командная оболочка — это эффективное средство автоматизации выполняемых операций. Любую команду, выполняемую из командной строки, можно быстро преобразовать в исполняемую программу. Достаточно поместить команду в текстовый файл, а в случае нескольких команд каждая из них располагается в отдельной строке. Впоследствии для выполнения команд(ы) достаточно будет выполнить соответствующий файл из командной оболочки (в Raspberry Pi командная оболочка еще называется Dash). Как только вы заметите, что вводите одни и те же команды больше десяти раз, серьезно задумайтесь над созданием для них отдельного файла сценария.

Быстрое знакомство

После загрузки в Raspberry Pi пользовательского графического интерфейса вы сможете перейти к управлению операционной системой из командной оболочки, дважды щелкнув на значке LXTerminal, расположенном на рабочем столе.

Кроме основной своей функции — просмотра информации в Интернете, браузер Midori может использоваться при копировании и вставке примеров программных кодов, рассматриваемых в данной книге.

Символ `$` в начале строки говорит о том, что Linux приглашает вас ввести команду. Введите `pwd`, чтобы определить текущий каталог (вернее, его расположение в файловой системе). Linux ответит вам, отобразив на экране точный путь расположения каталога, например `/home/pi/`.

Для вывода на экран списка файлов в текущем каталоге введите команду `ls` и нажмите клавишу `<Return>` или `<Enter>`. На экране появится список файлов, готовых для дальнейшей обработки. Если в качестве результата последняя команда возвращает ошибку выполнения, например `No such file or directory` (Файлы и каталоги отсутствуют), то сначала определите текущий каталог командой `pwd` и только после этого выполняйте команду `ls`, чтобы вывести на экране список его файлов.

После введения каждой команды, такой как `ls`, в командной строке нужно обязательно нажимать клавишу `<Return>` или `<Enter>`. Чтобы отредактировать неправильно введенную команду, перед нажатием клавиши `<Return>` или `<Enter>` воспользуйтесь клавишами со стрелками и клавишей `<Backspace>`.

Чтобы отредактировать (или создать) файл, например `foo.txt`, выполните команду `nano foo.txt`. Далее введите текст файла и нажмите комбинацию клавиш `<Ctrl+X>` для его сохранения. В ответ на подтверждение операции нажмите

клавишу <Y>. Теперь вы увидите на экране запрос на указание имени файла; ничего не вводите, а просто нажмите клавишу <Enter> или <Return>. Если возникнет необходимость в дальнейшем редактировании файла, то введите команду `nano foo.txt` еще раз.

Настройки, хранимые в текстовых файлах

В Linux большая часть настроек хранится в текстовых файлах. Воспользовавшись простыми командами, приведенными в предыдущем разделе, и командой `sudo`, опытный пользователь компьютера сможет легко изменить огромное количество параметров системы. Текстовые файлы настроек в Linux хранятся в двух каталогах. Общие настройки операционной системы вы найдете в каталоге `/etc`, а параметры учетной записи каждого пользователя — в домашнем каталоге `/home/pi/`.

С помощью текстовых файлов, хранящихся в системном каталоге `/sys`, можно даже управлять отдельными контактами входных и выходных портов GPIO, расположенных на плате Raspberry Pi. Далее мы детально рассмотрим принципы подключения датчиков и светодиодов к портам GPIO в Raspberry Pi.

Вы сможете установить в Raspberry Pi соединение с Интернетом, если до загрузки операционной системы подключите к плате кабель Ethernet. Для проверки подключения в текстовой оболочке команду `ping www.google.com`. Результат будет выведен на экран через несколько секунд. Чтобы остановить выполнение команды, спустя некоторое время нажмите <Ctrl+C>. Если сетевое подключение работает нормально, то потерянных пакетов не должно быть. Отдельные веб-страницы загружаются с помощью таких команд, как `curl botbook.com` и `wget botbook.com`.

sudo — это наше все

Операционная система Linux известна своей надежной системой защиты данных. Ключевым фактором в ней является правильное назначение пользовательских привилегий. В ней обычным пользователям разрешается вносить изменения только в те файлы, которые относятся к их рабочей среде. Буквально это означает, что они могут изменять только файлы, которые находятся в их каталоге (`/home/pi/`) и временных каталогах, таких как `/tmp/`.

Суперпользователь, или привилегированный пользователь, обладает несравненно большими полномочиями, имея возможность изменять любые системные файлы. Привилегии суперпользователя начинаются с введения перед выполняемой командой приставки `sudo`. Добавление `sudo` перед командой указывает оболочке выполнять следующую команду от имени суперпользователя.

В Raspbian за установку дополнительного программного обеспечения отвечает менеджер пакетов, запуск которого выполняется только привилегированными пользователями. Перед установкой нового программного обеспечения вам нужно обновить список доступных пакетов, для чего применяется команда `sudo apt-get update`. Подразумевается, что плата Raspberry Pi подключена к Интернету, поскольку все программные пакеты хранятся на файловых серверах.

Большая часть операционных систем, основанных на Linux и Unix (к их числу относится и OS X), устроены так, что при выполнении команды `sudo` на экран выводится запрос на ввод пароля. Это еще один уровень защиты данных в операционной системе. По умолчанию в Raspbian, установленной в Raspberry Pi, пароль не запрашивается. Поэтому будьте предельно внимательны при использовании команды `sudo` — допустить ошибку очень просто, что часто приводит к серьезным системным ошибкам, вплоть до отказа загрузки ОС.

Для установки программы из хранилища нужно лишь правильно указать название требуемого пакета. Чтобы установить программу `ipython` (интерактивную оболочку для языка Python, расширяющую синтаксические возможности языка и упрощающую управление программным кодом), используйте команду `sudo apt-get -y install ipython`. Параметр `-y` указывает автоматически отвечать утвердительно (“yes”) на все вопросы, задаваемые командой в процессе выполнения. Менеджер пакетов (`apt`) самостоятельно выполнит все необходимые операции без дальнейшего вмешательства со стороны пользователя.

Спустя несколько секунд вы сможете запустить только что установленное программное средство, введя в командной строке `ipython`. Теперь вы можете приступить к выполнению любых команд Python, но на данный момент введите `exit()`, чтобы возвратиться к командной оболочке операционной системы (приглашение `$`), в которой команды задаются в текстовом режиме.

Различные приглашения командной строки помогают пользователям определять программу, в которой они работают в текущий момент. В командной оболочке приглашение представлено символом `$`, увидев который можете смело вводить встроенные команды операционной системы, а также названия установленных в Raspberry Pi приложений. Если в командной строке отображается какое-то другое приглашение, например, программы `ipython` (`In [1]:`), то приготовьтесь к вводу команд, распознаваемых этой программой, в нашем случае — команд Python.

Установка демонов (также известных как серверы) не должна вызывать у вас особых затруднений. Давайте рассмотрим, как устанавливается самый популярный сервер на планете: Apache. Введите команду `sudo apt-get -y install apache2` и дождитесь ее выполнения. Затем укажите домашнюю страницу только что созданного веб-сервера, воспользовавшись командой `curl localhost`. Для получения доступа к серверу Apache из другого компьютера определите IP-адрес платы Raspberry Pi, воспользовавшись командой `ifconfig`. Полученный результат введите в адресной строке браузера целевого компьютера, подключенного к сети.

Ознакомившись с результатом выполнения программы `ifconfig`, вы увидите, что в системе имеется не менее двух сетевых адаптеров. Используйте адрес, указанный для адаптера Ethernet (`eth0`), или же (в случае использования адаптера Wi-Fi) — его адрес.

Вот и все! Пока что с вас достаточно команд, вводимых в командной оболочке операционной системы. Чтобы немного отдохнуть и закрепить навыки владения командной строкой, отключите питание Raspberry Pi с помощью команды `sudo shutdown -P now`. Как и в случае рабочей станции, подобный способ выключения устройства позволяет перед выключением питания сохранить в памяти все необходимые данные. После выключения системы смело отключайте источник питания от гнезда Micro-USB. Подключайте его заново только тогда, когда решите снова включить Raspberry Pi.

Все еще не до конца понимаете, зачем придумана вся эта возня с командой `sudo`? Чтобы понять, проведите поиск в сетевых ресурсах по запросу “`xkcd sudo sandwich`”. Уже первая ссылка сделает ваш обеденный перерыв незабываемым — постарайтесь отнестись с юмором к увиденным комиксам.

В приложении А вы найдете краткий список популярных команд Linux.

Ловкость одной руки, и никакого мошенничества

Иногда мы любим подшучивать над студентами, которые посещают наши курсы, посвященные Linux. Для этого задаем им простой вопрос: сколько времени нужно, чтобы ввести левой рукой следующую команду и выполнить ее.

```
supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc
```

Выслушав все прозвучавшие варианты, переходим к практической проверке поставленной задачи. Чтобы розыгрыш удался, нужно предварительно создать файл с текстом указанной команды (вводите ее в одной строке).

```
$ nano
```

```
supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc
```

Конечно, все ожидают, что сделав глубокий вдох, я постараюсь как можно быстрее ввести такую команду:

```
$ ls
```

```
supercalifragilisticexpialidocious.foo.bar.txt.botbook.com.pdf.cc
```

Но на самом деле я ввожу только `ls` и нажимаю клавишу `<Tab>`. Оболочка самостоятельно завершает ввод названия, как только вы нажмете клавишу `<Tab>`. Функция автозаполнения — это очень удобное и часто используемое средство Linux. Представляющая ее клавиша `<Tab>` интуитивно добавляет к вводимым вами командам не только названия папок и файлов, но и имена сетевых ресурсов (после ввода таких команд, как `ssh` и `ping`). Поскольку нажатие клавиши `<Tab>` позволяет завершать любые ранее используемые названия, то, взяв ее на вооружение, вам не придется утруждать себя запоминанием длинных путей.

Подключение оборудования к Raspberry Pi

В Raspberry Pi подключение электронных компонентов осуществляется через GPIO (General-Purpose Input and Output — порт ввода-вывода общего назначения). Порт называется “общего назначения”, поскольку только вам решать, что к нему подключать. Более того, одни и те же выводы этого порта могут служить как для ввода, так и для вывода сигналов, что определяется пользователем. В этой книге GPIO будет служить самым разным целям:

- цифровой выход (включение/выключение светодиода);
- цифровой резистивный вход (определяет нажатие кнопки и срабатывание датчика);
- цифровой вход для короткоимпульсных сигналов (например, выводимых датчиком расстояния);
- аналоговый резистивный вход (принимает сигналы с аналоговых резистивных датчиков давления, света и температуры);
- поддержка промышленных протоколов, например I²C и SPI (подключение контроллеров наподобие Wii Nunchuk и аналого-цифровых преобразователей).

В отличие от большинства других пособий по микроконтроллерным платформам, уже выпущенных на момент написания книги, мы остановимся на рассмотрении цифровых входов и выходов, исходя из использования их с привилегиями суперпользователя. Это, с одной стороны, упрощает нашу задачу, а с другой — понижает защищенность системы.

В своих изысканиях мы часто будем подключать к цифровому входу Raspberry Pi подтягивающий резистор, значительно упрощающий считывание сигналов, передаваемых внешним оборудованием.

Для измерения аналогового сопротивления в Raspberry Pi в обязательном порядке используется внешний аналого-цифровой преобразователь. Кроме этого, многие современные электронные устройства обмениваются данными через общепромышленные интерфейсы, такие как I²C и SPI. Примеры их практического использования вы также найдете в последующих главах.

Но вначале рассмотрим самый простой вариант применения интерфейса GPIO — в качестве цифрового выхода.

Привет всем! Мигание светодиодом

В примере, приведенном в этом разделе, вы научитесь подключать к Raspberry Pi внешний светодиод и заставите его мигать.

Это классический пример управления микроконтроллерной платой внешним устройством, применимый для всех платформ и языков программирования. Какие бы сложные проекты вы ни планировали реализовать в будущем, знакомясь с новой платформой, не ленитесь вначале выполнить это простое задание. Кроме всего прочего, реализуя его, вы удостоверитесь в работоспособности аппаратной и программной частей проекта. Если рассматриваемый ниже пример в вашем случае не выполняется, то вначале устраните неполадки в оборудовании или среде разработки и только после этого переходите к реализации сложных проектов.

Итак, вам понадобятся:

- плата Raspberry Pi;
- навесные или монтажные провода с разъемами обоих типов (проволочные перемычки);
- плата для беспаячного макетирования (еще называемая платой для прототипирования), или просто макетная плата;
- резистор с сопротивлением 470 Ом (обозначенный желтой, фиолетовой и коричневой полосками);
- светодиод.

Если у вас нет в наличии всего описанного выше оборудования, то обратитесь к разделу “Устранение неполадок”. Что касается проволочных перемычек, то они могут иметь на своих концах как гнездовой (разъем типа “мама”), так и штыревой (разъем типа “папа”) наконечники. При выборе перемычек ориентируйтесь на обозначения F (female — гнездовой) и M (male — штыревой), которые указывают совместно с другими характеристиками, такими как длина провода. В большинстве случаев вам понадобятся перемычки с обоими типами наконечников на одном проводе (F-M).

Порты GPIO не защищены от перегрузок и закорачивания контактов (рис. 1.6). В отличие от Arduino, плата Raspberry Pi не рассчитана на грубое с ней обращение. Входы и выходы GPIO рассчитаны на рабочее напряжение +3,3 В. Подача на контакты напряжения +5 В может привести к повреждению платы Raspberry Pi или (если сильно повезет) к выходу из строя только поврежденного входа/выхода. Несколько раз проверяйте вашу цепь на макетной плате перед подключением ее к GPIO. Также будьте предельно внимательны при использовании мультиметра для замера характеристик портов GPIO.

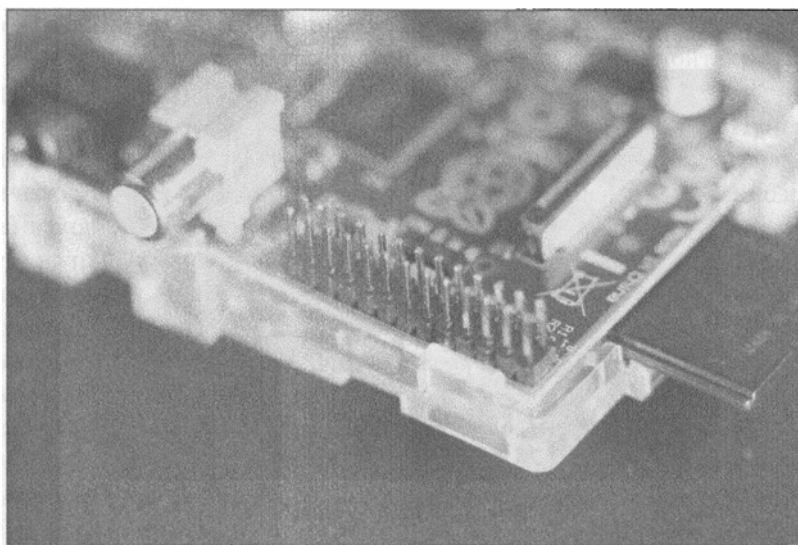


Рис. 1.6. Контактная колодка GPIO

Построение электрической цепи

Цепь для этого простого проекта состоит всего лишь из последовательно соединенных светодиода и ограничивающего ток через него резистора, которые подключены между выводом с номером 27 и выводом GND (земля) колодки GPIO (рис. 1.7). Короткий (катод) вывод светодиода соедините с темным проводом, а длинный вывод (анод) — с резистором. Подключение светодиода проводится при отключенной от питания плате Raspberry Pi.

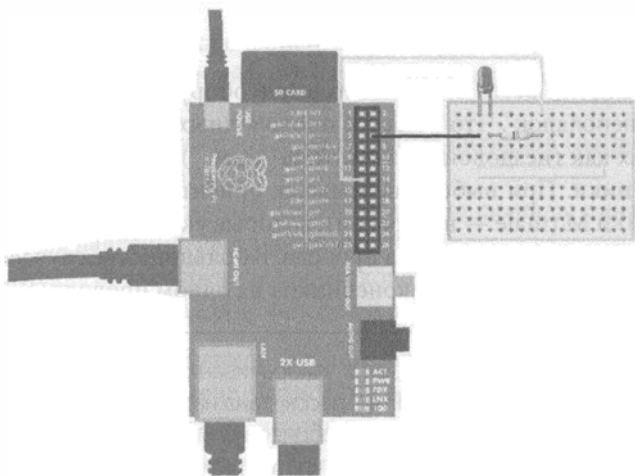


Рис. 1.7. Макетная плата с построенной электрической цепью управления светодиодом

Соберите электрическую цепь на макетной плате, как показано на рис. 1.8. Внимательно проверьте правильность подключения компонентов, чтобы исключить повреждение платы Raspberry Pi. Только убедившись в правильности соединения элементов цепи, подавайте питание на Raspberry Pi.

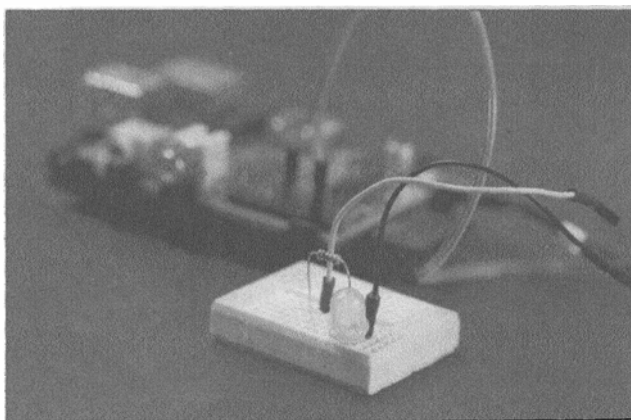


Рис. 1.8. Заработало!

Далее мы поговорим о системах обозначения выводов GPIO в Raspberry Pi.

Две системы обозначения: функциональная и последовательная

Каждый контакт колодки GPIO имеет двойное обозначение: функциональное и последовательное числовое. Чтобы быстро ориентироваться в выводах GPIO, вам нужно научиться безошибочно сопоставлять значения обеих систем обозначения.

Взгляните на колодку с выводами GPIO в Raspberry Pi (см. рис. 1.6). Таблица сопоставления обоих типов обозначения выводов приведена на рис. 1.9.

The diagram shows the Raspberry Pi 1 GPIO pin header. On the left, there are labels for 'SD CARD', 'USB', 'POWER', and 'HD'. The Raspberry Pi logo and 'Raspberry Pi 1' are also visible. The table below lists the functional names and their corresponding pin numbers (1 to 26).

Functional Name	Pin Number
3.3V	1
5V	2
gpio2 u/sda	3
gpio3 u/scl	4
gnd	5
gpio14/tx	6
gpio15/rx	7
gpio17	8
gpio27	9
gpio22	10
3.3V	11
gpio10/mosi	12
gpio9/miso	13
gpio11/sclk	14
gnd	15
gpio7/cel	16
gpio8/ce0	17
gpio25	18
gnd	19
gpio24	20
gpio23	21
gnd	22
gpio21	23
gpio20	24
gpio19	25
gpio18	26

Рис. 1.9. Функциональное и числовое обозначение выходов GPIO

Слева в таблице указаны функциональные названия выводов (GND, gpio27 и т.п.), а справа приведены числовые значения, определяющие номера контактов на колодке по порядку их следования (от 1 до 26).

Как уже указывалось, всего на колодке находится 26 контактов. Нумерация начинается с контакта, обозначенного крошечным белым квадратиком и надписью P1. Это физический номер штыревого контакта, на который непосредственно надевается гнездовой наконечник проволочной перемычки. Физические номера еще известны как номера на плате.

В функциональной нумерации контактов поддерживается несколько другой подход к обозначению. В примере подключения к плате Raspberry Pi светодиода используется вывод gpio27, который физически имеет номер 13 (к этому контакту с помощью перемычки подключается резистор). В таблице обозначений, показанной на рис. 1.9, вы увидите такие функциональные обозначения выводов, как GND (земля), 5V и 3.3V. Некоторые выводы имеют несколько функций. Например, вывод gpio10 применяется как порт шины SPI. Функциональная нумерация выводов еще известна как BCM-нумерация.

В первом примере, приведенном в данной книге, используются всего два контакта GPIO, обозначенные в обеих системах нумерации так, как показано в табл. 1.1. В следующих проектах для правильного обозначения номеров, используемых в цепях, см. таблицу на рис. 1.9.

Таблица 1.1. Выводы, используемые в первом проекте

Вывод GPIO (BCM-номер)	№ п/п
gpio27	13
GND	6

Управление GPIO из командной оболочки

Настало время разобраться с тем, как управлять выводами GPIO, к которым подключена цепь со светодиодом, из командной строки. Для начала сделаем это от имени суперпользователя, а затем разберемся, как можно управлять светодиодом без привлечения команды `sudo`.

Как вы уже знаете, в Linux все настройки задаются в виде текстовых файлов. Драйвер ядра GPIO (программа, определяющая способ взаимодействия Linux с GPIO-портом), без которого вы не получите доступ к выводам GPIO, располагается в виртуальной файловой системе `/sys`. Чтобы определить необходимые операции по управлению GPIO, внесите необходимые изменения в соответствующие текстовые файлы.

На данном этапе вам не потребуется графический интерфейс, поскольку с помощью командной оболочки ОС вносить изменения в настройки системы намного проще. Дважды щелкните на значке **LXTerminal**, расположенном на рабочем столе, чтобы запустить интерфейс командной оболочки.

Если вы ранее настроили Raspberry Pi на запуск в графическом режиме или если подключаетесь к ней через SSH-соединение с другого компьютера, то для выполнения операций из командной строки переходить к графическому интерфейсу не нужно.

Чтобы сделать контакт доступным, сначала нужно его “экспортировать” или представить виртуальным файлом, затем перевести в режим выхода и только после этого установить для него значение 1. Все эти задачи легко решаются путем редактирования текстовых файлов.

Запись в файл, минуя редактор

Ранее мы уже показывали, как изменить содержимое текстового файла с помощью редактора `nano`. Ниже приведен еще один способ редактирования файлов без привлечения текстового редактора.

Базовая операция управления текстом — это его вывод. Текст выводится в строке терминала следующей командой (символ `$` вводить не нужно — он обозначает приглашение командной оболочки):


```
$ echo "Hello BotBook"
```

Любой выводимый на экран таким способом текст можно занести в файл. Содержимое файла при этом заменяется на указываемый вами текст, поэтому будьте внимательны и не удалите с помощью следующей команды важную информацию:

```
$ echo "Hello BotBook" > foo.txt
```

Если файл `foo.txt` ранее не создавался, то он будет создан автоматически. Если файл уже существует, то его содержимое обновится без лишнего предупреждения. Можете воспользоваться оператором `>` (перенаправление), чтобы занести в текстовый файл результат выполнения практически любой команды. Для просмотра содержимого файла применяется команда `cat`.

```
$ cat foo.txt  
Hello BotBook
```

А почему нельзя использовать команду `nano foo.txt`? Как это нельзя? Можно! Но в таком случае вам придется вводить больше текста, и нельзя будет автоматизировать выполняемые операции.

Управление светодиодом

Настало время использовать команду `sudo` для управления светодиодом. Вам все еще кажется неправильным использовать привилегии суперпользователя для выполнения неадминистративных задач? Как минимум сомнения у вас должны возникнуть, ведь, неправильно вводя команды после `sudo`, вы потенциально можете вызвать сбой в операционной системе, что чревато полной ее переустановкой.

Поспешу вас успокоить: использовать команду `sudo` мы будем только в этом простом примере и то лишь для того, чтобы детально ознакомить вас с ней. В дальнейшем мы зададим полномочия доступа к файлам в Linux так, чтобы иметь возможность управлять выводами GPIO с привилегиями обычного пользователя.

Введите команду `sudo -i`, чтобы получить привилегии суперпользователя. В режиме суперпользователя вы можете работать сколько угодно до завершения всех необходимых операций. Чтобы покинуть режим суперпользователя, выполните команду `exit`. Обратите внимание на то, что символ приглашения изменился — теперь это знак решетки (`#`). Внимательно следите за тем, что вводите, так как ошибки могут вызвать сбой в работе операционной системы.

Получив привилегии суперпользователя, представьте вывод `gpio27` виртуальным файлом, чтобы в дальнейшем иметь возможность управлять им (текст команды вводится после приглашения `#`):

```
# echo "27">/sys/class/gpio/export
```

Вы только что создали новый виртуальный файл, который будет применяться для настройки мигания светодиода. Далее переведите вывод `gpio27` в режим вывода сигнала, чтобы управлять подачей сигнала на него:

```
# echo "out" > /sys/class/gpio/gpio27/direction
```

Подайте на контакт сигнал:

```
# echo "1" > /sys/class/gpio/gpio27/value
```

Светодиод должен загореться. Налюбовавшись результатом своей деятельности, прекратите подачу сигнала:

```
# echo "0" > /sys/class/gpio/gpio27/value
```

У вас получилось? Вне всяких сомнений!

Закончив работу от имени привилегированного пользователя, не забудьте выполнить команду `exit`. Символ приглашения должен измениться на значок \$, указывая на возвращение к режиму обычного пользователя.

Устранение неполадок

Если у вас все же возникли трудности с построением электрической цепи, настройкой системы и управлением светодиодом, то ознакомьтесь со следующими рекомендациями.

У вас нет под рукой резистора номиналом 470 Ом?

Подойдет любой резистор, сопротивление которого составляет несколько сотен ом. Резистор того же порядка сопротивления наверняка не вызовет повреждения микроконтроллерной платформы, поскольку он применяется для ограничения тока, протекающего через светодиод, и предотвращения перегрева светодиода и контактов на плате Raspberry Pi. Если вам кажется, что при подключении другого резистора светодиод имеет слабую или, наоборот, очень большую яркость, то остановитесь на выборе резистора с сопротивлением в диапазоне от 100 Ом до 1 кОм. Такие резисторы маркируются третьей коричневой полоской на корпусе. (О маркировке электронных компонентов детально рассказано во врезке “Правило третьей полоски”).

Не можете найти проволочные перемычки с разъемами необходимых типов?

Воспользуйтесь старым 40-жильным кабелем для подключения жестких дисков IDE (но не 80-жильным). Подключите разъем на одном из концов кабеля к контактной колодке GPIO на плате Raspberry Pi. Срежьте разъем на другом конце кабеля. Следите за нумерацией проводов кабеля. Отделите от кабеля ровно столько проводов, сколько необходимо для работы, а остальные оставьте соединенными в шлейф. Если вы все же разделили весь кабель на отдельные провода, то обязательно изолируйте провода, на которые подается напряжение +5 В. Увидеть, как применяется старый IDE-кабель, можно по такому адресу:

<http://bit.ly/1f0GWfV>

На рынке имеется большое количество других кабелей с разъемами, пригодными для подключения к контактной колодке Raspberry Pi, например, производства компании Adafruit. Если вы приобретали Raspberry Pi в виде начального набора компонентов, то кроме микроконтроллерной платы у вас есть и макетная плата, и проволочные перемычки.

Светодиод не светится!

Вы уверены в правильности его подключения? Светодиоды относятся к полярным устройствам. Если их включить в цепь неправильно, то они не будут работать. Длинный положительный контакт светодиода через резистор подключается к выводу `gpio27` платы. Короткий отрицательный контакт светодиода, выходящий из корпуса со стороны, обозначенной срезанной кромкой,

необходимо заземлить. На рис. 1.10 стрелка указывает расположение в светодиоде катода, из которого выходит короткая “ножка” катода. Справедливости ради стоит заметить, что разглядеть срезанную кромку на корпусе светодиода не так уж и просто. Лучше всего взять светодиод в руки и нащупать ее. Кроме того внутри прозрачного корпуса светодиода легко разглядеть большой плоский электрод. Это и будет катод.

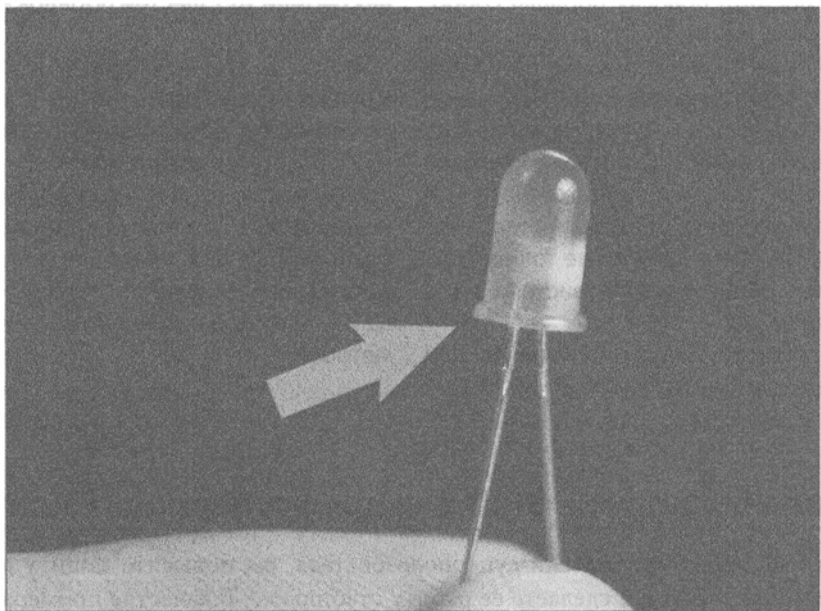


Рис. 1.10. Следите за полярностью светодиода

При вводе команд с привилегиями суперпользователя ничего не происходит

Ни в коем случае не вводите символ приглашения командной строки `#`. В рассматриваемом ранее примере он приводился для наглядности, чтобы показать, что вы будете наблюдать на мониторе, подключенном к Raspberry Pi. Это приглашение появляется на экране при переходе в режим суперпользователя и говорит о том, что операционная система ожидает от вас команды. Таким же образом, не нужно вручную вводить символ приглашения `$` при возврате к привилегиям обычного пользователя. Если вы введете символ `#` вручную, то остальная часть строки будет воспринята операционной системой как комментарий (ремарка, оставленная для себя и других пользователей), а потому просто проигнорирована.

Сообщение об ошибке

В точности скопируйте сообщение об ошибке и вставьте его в окно поисковой системы, например Google. Можете попробовать заключить все сообщение в кавычки. При поиске используйте дополнительные ключевые слова “Raspbian” и “Raspberry Pi”. Найдя решение возникшей проблемы, обязательно поделитесь им с общественностью в сети, чтобы помочь другим пользователям сэкономить рабочее время на преодоление подобных трудностей.

Цвет третьей полоски на корпусе резистора позволяет быстро определить его номинал. Зная, как правильно найти и оценить полоску порядка сопротивления, вы будете быстро находить резисторы с необходимым номиналом. Полоска порядка значения, как правило, указывается на корпусе резистора третьей (у резисторов, маркированных пятью полосками, порядок — это четвертая полоска). В большинстве электрических цепей важно не точное значение сопротивления резистора, а его порядок.

Например, сопротивление 10 кОм определяется в тысячах ом (или 10^3 Ом);

третий порядок — это оранжевая полоска. Резисторы с сопротивлением от 10 до 50 МОм определяются в миллионах ом (или 10^6); шестой порядок — это синяя полоска.

Чтобы детально ознакомиться со значением всех полосок на корпусе резисторов и научиться использовать их при расчете электрических цепей, проведите поиск в Интернете по ключевой фразе “цветовая маркировка резисторов”. Для проверки правильности вашей оценки порядка номинала резистора воспользуйтесь мультиметром.

Управление портом GPIO без прав суперпользователя

Отказавшись от привилегий суперпользователя, вы повысите защиту данных в операционной системе и сделаете ее работу стабильнее. В качестве примера рассмотрим программу, отправляющую показания некоего датчика на сервер. Решитесь ли вы выполнять такую программу с привилегиями суперпользователя, зная, что к ней могут получить доступ злоумышленники?

Прежде чем продолжить наше знакомство с привилегиями пользователей, убедитесь, что электрическая цепь управления светодиодом, собранная ранее, работает.

В последних версиях Linux подключенные к системе устройства управляются менеджером `udev`. С помощью этого средства запускаются сценарии управления, выполняемые в пространстве пользователя при добавлении и отключении внешних устройств. Если вам доводилось разрабатывать приложения для Android под Linux, то вы, скорее всего, создавали с помощью `udev` правила, определяющие полномочия по управлению подключенного к компьютеру смартфона. Если вы имеете опыт разработки приложений для Arduino в Linux, то, возможно, добавляли пользователя в группу `dialout`, чтобы обеспечить соединение через последовательный порт USB.

Обычно файлы управления выводами GPIO располагаются в каталоге `/sys/class/gpio/`, владельцем которого является суперпользователь и группа `root`. Далее мы покажем, как с помощью `udev` создается правило, заменяющее группу владельцев на `dialout`. Группе будут назначены привилегии на чтение и запись файлов в каталоге `/sys/class/gpio/`. Наконец, мы изменим атрибуты указанного каталога так, чтобы создаваемые в нем каталоги и файлы также принадлежали к группе `dialout`.

Все важные системные настройки в Linux хранятся в каталоге /etc. Неудивительно, что настройки менеджера udev находятся по адресу /etc/udev/. Сначала с помощью команды `sudoedit` запустите редактор, чтобы создать файл правила:

```
$ sudoedit /etc/udev/rules.d/88-gpio-without-root.rules
```

Добавьте в новый файл текст из листинга 1.1. Вводите программный код в точности в том виде, в котором он представлен в листинге 1.1 (не вводите числовые символы в кружках; они обозначают сноски, которые описывают операции, выполняемые в строках кода). При задании правила с помощью udev нельзя допускать ни одной опечатки.

Листинг 1.1. 88-gpio-without-root.rules

```
# /etc/udev/rules.d/88-gpio-without-root.rules - управление
# GPIO обычным пользователем в Raspberry Pi # ❶
# Copyright 2013 http://BotBook.com
# ❷
SUBSYSTEM=="gpio", RUN+="/bin/chown -R root.dialout /sys/class/gpio/"
SUBSYSTEM=="gpio", RUN+="/bin/chown -R root.dialout
❧ /sys/devices/virtual/gpio/"
# ❸
SUBSYSTEM=="gpio", RUN+="/bin/chmod g+s /sys/class/gpio/"
SUBSYSTEM=="gpio", RUN+="/bin/chmod g+s /sys/devices/virtual/gpio/"
# ❹
SUBSYSTEM=="gpio", RUN+="/bin/chmod -R ug+rw /sys/class/gpio/"
SUBSYSTEM=="gpio", RUN+="/bin/chmod -R ug+rw
❧ /sys/devices/virtual/gpio/"
```

- ❶ Этот комментарий описывает назначение файла.
- ❷ Делаем владельцев двух каталогов суперпользователями и назначаем группу dialout.
- ❸ Назначаем двум каталогам атрибут sticky bit.
- ❹ Изменяем права для двух каталогов, предоставляя членам группы dialout возможность чтения и записи файлов в них.

Правила обрабатываются последовательно, но, поскольку каталогам для GPIO других правил не назначено, порядковый номер не играет большой роли. В азбуке Морзе число 88 символически представляет понятие “обнимашки-целовашки”. Нам оно нравится больше, чем чаще встречаемое значение 99, обозначающее в азбуке Морзе грубое “отвали!”.

Чтобы избежать опечаток при вводе кода, загрузите готовый файл `88-gpio-withoutroot.rules` со страницы книги, адрес которой был указан во введении.

Сохраните файл (нажмите комбинацию клавиш <Ctrl+X>, затем клавишу <Y>, а затем — <Enter>).

Чтобы применить только что созданные правила, перезапустите демон udev и включите новое правило такими командами.

```
$ sudo service udev restart
$ sudo udevadm trigger --subsystem-match=gpio
```

В последних версиях Linux все демоны (серверы) управляются с помощью сценариев. Это позволяет просто обновлять настройки в веб-сервере Apache, для чего выполняется команда `sudo service apache2 reload`. Вы также можете перезапустить (остановить и запустить заново) SSH-сервер командой `sudo service ssh restart`.

Теперь нужно проверить владельца каталога:

```
$ ls -lR /sys/class/gpio/
```

В выведенном на экран списке название группы `dialout` должно отображаться многократно. Параметр `-l` предназначается для вывода длинных записей (включая сведения о владельце, группе и привилегиях), а параметр `-R` к тому же указывает отображать содержимое каталога в обратном порядке.

Можно попробовать изменить настройки GPIO и без прав суперпользователя. Обратите внимание на приглашение в виде символа доллара, \$, обозначающего командную оболочку обычного пользователя.

```
$ echo "27" > /sys/class/gpio/unexport
$ echo "27" > /sys/class/gpio/export
$ echo "out" > /sys/class/gpio/gpio27/direction
$ echo "1" > /sys/class/gpio/gpio27/value
```

Первая команда отменяет “экспорт” контакта GPIO, чтобы последующая команда добавления виртуального файла не выдала ошибку.

Светодиод должен загореться. Выключение светодиода выполняется следующей командой:

```
$ echo "0" > /sys/class/gpio/gpio27/value
```

Вот видите, управлять светодиодом можно и с привилегиями обычного пользователя.

Поздравляем! Вы только что узнали, как управлять GPIO почти в любой среде разработки.

Устранение неполадок при работе с GPIO

Владелец файла не изменился

Если команда `ls -lR /sys/class/gpio/` не возвращает `dialout`, то, скорее всего, в файл правил `/etc/udev/rules.d/88-gpio-without-root.rules` закралась опечатка. Загрузите готовый файл с сайта книги, указанного во введении, и поместите его в “правильный” каталог (например, `sudo mv 88-gpio-withoutroot.rules /etc/udev/rules.d/`). Чтобы применить в Linux новые правила, выполните команду `sudo service udev restart` или перезагрузите систему с помощью команды `sudo shutdown -r now`.

Светодиод не горит

Попробуйте, будет ли загораться светодиод при управлении им с привилегиями суперпользователя.

На экране появляется сообщение об ошибке

Поищите решение проблемы в Интернете, скопировав текст сообщения в окно одной из поисковых систем.

Управление GPIO из Python

Вы всегда можете получить доступ к выводам GPIO из среды разработки Python, считывая и записывая файлы в каталоге `/sys`. Этот же способ применялся ранее при управлении портами в Raspberry Pi из командной оболочки.

Старый служака Python

Как обычно, начнем с тестирования среды на примере операции вывода сообщения. С помощью текстового редактора создайте файл:

```
$ nano hello.py
```

Если вы не можете сохранить файл с таким именем, то с помощью команды `cd` перейдите в раздел файловой системы, на которую ваши привилегии не распространяются. Затем выполните команду `cd ~`, чтобы вернуться в исходный каталог, и попробуйте повторно сохранить файл.

В самом файле будет храниться всего одна текстовая строка:

```
print "Привет всем!"
```

Сохраните файл (в редакторе nano нажмите комбинацию клавиш `<Ctrl+X>`, потом клавишу `<Y>`, а затем — `<Enter>` или `<Return>`).

Выполните программу.

```
$ python hello.py
Привет всем!
```

Выполнение команды `python` без дополнительных параметров приводит к запуску интерактивной консоли. Детально о консоли Python речь пойдет в главе 8.

Доступ к портам ввода-вывода в Python

Немного усложним задачу: научим мигать светодиод, подключенный к выводу `gpio27` с помощью Python. Подключите светодиод к плате Raspberry Pi (см. рис. 1.7),

если еще не сделали этого. Сохраните исходный код, приведенный в листинге 1.2, в файле `led_hello.py` и выполните его из Raspberry Pi.

```
$ python led_hello.py
```

Светодиод GPIO 27 подмигивает вам...

Светодиод загорелся на пару секунд? Привет всем!

Возникли проблемы с выполнением этого задания? Детально изучите описание исходного кода, приведенное после листинга 1.2, а также прочитайте следующий раздел, посвященный устранению неполадок. Очень важно научиться находить ошибки в этом простом примере, чтобы при выполнении сложных проектов окончательно не запутаться в самых простых ситуациях.

Листинг 1.2. `led_hello.py`

```
# led_hello.py - включение светодиода в Raspberry Pi
```

```
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time    # ❶
import os

def writeFile(filename, contents):    # ❷
    with open(filename, 'w') as f:    # ❸
        f.write(contents)

# Основной код

print "Светодиод GPIO 27 подмигивает вам..."    # ❹
if not os.path.isfile("/sys/class/gpio/gpio27/direction"):    # ❺
    writeFile("/sys/class/gpio/export", "27")    # ❻

time.sleep(0.1)
writeFile("/sys/class/gpio/gpio27/direction", "out")    # ❼

writeFile("/sys/class/gpio/gpio27/value", "1")    # ❽
time.sleep(2)    # секунд    # ❾
writeFile("/sys/class/gpio/gpio27/value", "0")    # ❿
```

- ❶ Импорт необходимых библиотек. Каждая библиотека имеет свое пространство имен с названием, повторяющим имя библиотеки, поэтому с него будут начинаться все соответствующие команды, например `time.sleep(2)`.
- ❷ Определение вспомогательной функции, используемой при записи файлов. Вызывается далее в программе несколько раз.
- ❸ Современный способ доступа к файлам в Python с использованием синтаксиса `with`. Он позволяет автоматически управлять закрытием файла в случае возникновения ошибок. В функции `open()` параметр `filename` принимает значение `/sys/class/gpio/gpio27/direction`, а параметр `w` указывает открывать файл для редактирования и последующего сохранения. В процессе выполнения этой функции создается дескриптор файла `f`, над которым и выполняются реальные операции по управлению файлом.

- 4 Хотя основная наша задача — заставить мигнуть светодиод, неплохо вывести что-то на экран, подтверждающее работоспособность программы. В Python последняя операция не столь важна, поскольку среда разработки оснащается прекрасным инструментом отслеживания ошибок от результата к источнику, который замечательно справляется с определением причин неполадок в программном коде.
- 5 Проверка того, что контакт не “экспортирован” (для него не создавались файлы настроек). Если ее не выполнить, то при повторном выполнении будет выдана ошибка `IOError: [Errno 16] Device or resource busy` (Ошибка устройств ввода-вывода: [16] Устройство или ресурс занят). Этот стандартный тип проверки называется “запросом полномочий”. Другой тип проверки, не используемый здесь, заключается в использовании конструкции `try...except`. Мы выбрали первый вариант проверки, поскольку с ним условная конструкция и вся программа намного проще.
- 6 “Экспорт” контакта. В результате этой операции создаются все необходимые для управления контактом файлы — в них указывается направление передачи сигнала и его величина.
- 7 Поскольку нам необходимо подавать и сбрасывать сигнал на контакте, то устанавливается режим `out` (выход). Если требуется прочитать значение сигнала, подаваемого на контакт, то нужно выбрать режим `in` (вход). Если вы уже знакомы с платформой Arduino, то наверняка проведете аналогию с командой `pinMode()`.
- 8 Выходное значение 1 соответствует подаче на светодиод напряжения 3,3 В. Светодиод начинает светиться, поскольку 1 соответствует сигналу высокого уровня в Raspberry Pi.
- 9 Задержка в пару секунд. Будет хорошей практикой указывать единицы измерения в комментариях, хотя бы один раз для каждой функции или переменной. Это позволит впоследствии не только другим программистам, но и вам самим быстрее разобраться с особенностями программного кода. В течение указанного времени вывод сохраняет текущее состояние: контакт `gpio27` принимает значение 1, а светодиод светится.
- 10 Назначение выводу `gpio27` сигнала низкого уровня (0); светодиод гаснет.

Устранение неполадок

Отказано в доступе

Если вы видите на экране ошибку `IOError: [Errno 13] Permission denied` (Ошибка ввода-вывода: [13] отказано в доступе) или `IOError: [Errno 2] No such file` (Ошибка ввода-вывода: [2] нет такого файла), то временно пренебрегите защитой данных и попытайтесь получить доступ к файлу как суперпользователь:

```
$ sudo python led_hello.py # тестирование
```

Если для суперпользователя файл доступен, то приступайте к исправлению ошибки назначения полномочий для виртуальных файлов GPIO (см. раздел “Управление портом GPIO без прав суперпользователя”). Если проблема все еще не устраняется, то перезагрузите Raspberry Pi, сбросив и заново подав питание на плату. Как только проблема будет решена, выполните файл Python, как обычный пользователь:

```
$ python led_hello.py
```

Светодиод не светится, хотя программа не выводит сообщение об ошибке

Проверьте полярность светодиода и правильность его подключения к плате, а также правильность соединения компонентов с помощью проволочных перемычек и подключения их к выводам GPIO (см. рис. 1.9). Если все подключено правильно, а проблема остается нерешенной, то воспользуйтесь мультиметром для проверки сопротивления резистора (в нашем случае 470 Ом). Кроме того, вы всегда можете протестировать цепь без микроконтроллерной платы: соедините последовательно батарейку, светодиод и резистор. Возможно, не работает сам светодиод?

Что дальше?

Вот мы и закончили настройку в Linux среды разработки стоимостью всего 35 долларов. Мало того, мы научились управлять из нее оборудованием. Вы освоили в Linux программные средства управления самым разным электронным оборудованием.

Вы также узнали о системном администрировании платформы, определяющем методы управления файлами проекта. Старайтесь максимально дистанцироваться от полномочий суперпользователя, чтобы избежать случайного повреждения файлов операционной системы.

В последующих главах мы рассмотрим проекты устройств на базе Raspberry Pi, управление которыми осуществляется с помощью сигналов, поступающих с датчиков. Простые примеры подключения к микроконтроллерной платформе одних только датчиков призваны познакомить вас с особенностями обработки входных и выходных цифровых сигналов, применения стандартных общепромышленных протоколов и управления импульсными сигналами.

Вы будете применять принципы администрирования и управления выводами GPIO, с которыми ознакомились выше, во всех остальных главах. Платформа Raspberry Pi поддерживает огромное количество датчиков, работающих с самыми разными интерфейсами обмена данными. Их преимущества вы в полной мере оцените при использовании стандарта I²C для подключения, например, контроллера Wii Nunchuk, акселерометра-гироскопа MPU 6050 и датчика атмосферного давления GY65.

Добро пожаловать в мир Linux!

Знакомство с Arduino

2

Arduino — это простая и производительная микроконтроллерная платформа, предназначенная для создания прототипов робототехнических и электронных устройств (рис. 2.1). Она считается самой простой платформой, с помощью которой можно создавать программируемые электронные устройства, имеющие высокий уровень надежности.

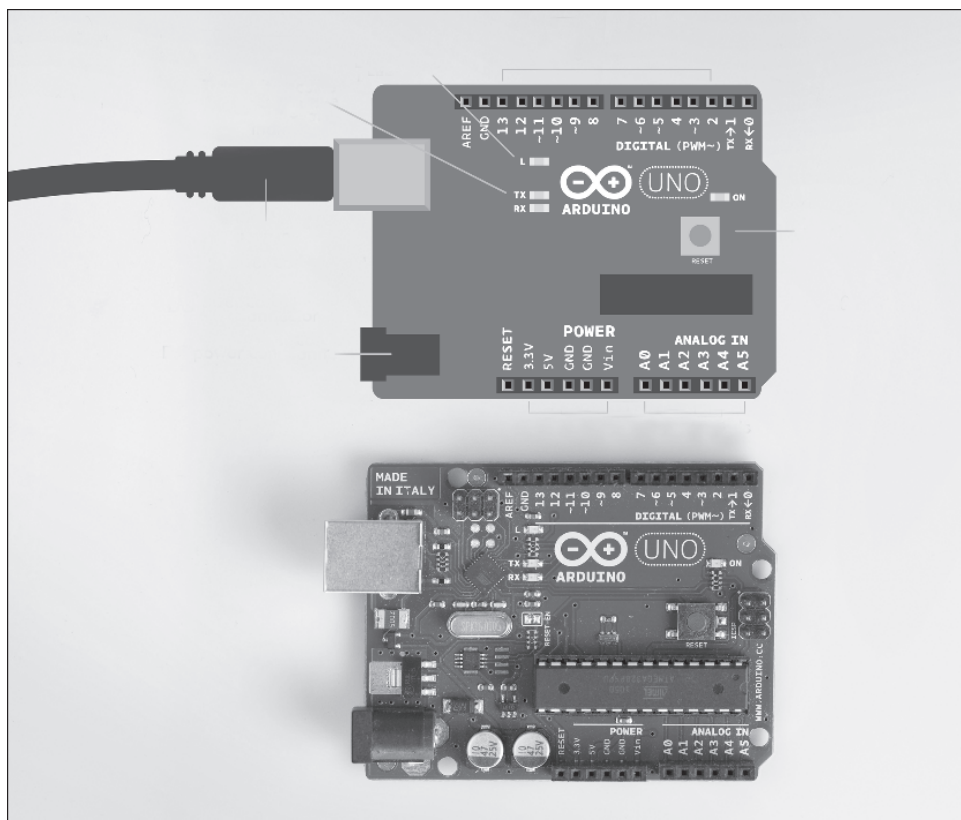


Рис. 2.1. Разъемы Arduino

Начать работу с Arduino очень просто. Все, что вам понадобится, — это самая простая версия Arduino — Uno и кабель USB для подключения платы к компьютеру. Все вместе вам обойдется не более чем в 35–40 долларов. Программное обеспечение для Arduino распространяется бесплатно (исходный код разрешается свободно использовать в обучающих целях, собственных проектах, а также распространять на свое усмотрение).

Первым делом в вашем компьютере нужно настроить интегрированную среду разработки (Integrated Development Environment — IDE) для Arduino. После этого через USB-порт в Arduino можно загрузить программу (на жаргоне Arduino-сообщества она называется “скетчем”). Одновременно в Arduino можно установить только одну программу — ту, которая будет непосредственно выполняться. Вот и все, что вам нужно вкратце знать об Arduino, поскольку, в отличие от Raspberry Pi, у нее нет операционной системы, а потому нечего дополнительно настраивать. Только вы, ваша программа и электрическая цепь, управление которой осуществляется загруженным в Arduino “скетчем”.

Все же существует еще одна деталь, о которой стоит упомянуть, описывая Arduino. Это загрузчик, занимающий небольшой объем памяти микроконтроллера. Загрузчик представляет собой небольшую программу, которая запускается непосредственно при подаче питания или перезагрузке платы контроллера. Эта программа позволяет устанавливать внешнее исполняемое приложение, загружаемое через USB-порт, исключая необходимость применения отдельного аппаратного программатора.

Небольшая по размерам Arduino Uno имеет высокую надежность. Вам вряд ли удастся повредить электронику платы вследствие неправильного подключения внешнего оборудования (не стоит умышленно проверять платформу на крепость, ведь при должном старании все же можно “выжечь” один или несколько контактов платы).

Обучаться работе с Arduino очень интересно. Начинающие пользователи будут удивлены широкими возможностями платформы по подключению внешнего оборудования. В отличие от Raspberry Pi, можно напрямую подключать к ней аналоговые резистивные датчики без использования дополнительного оборудования, поскольку Arduino оснащается встроенным аналого-цифровым преобразователем.

Базовая настройка Arduino

Ниже показано, как правильно настроить Arduino в Linux, Windows и Mac.

Ubuntu Linux

Подключите Arduino к компьютеру с помощью кабеля USB. Питание на Arduino подается непосредственно из USB-порта компьютера, поэтому внешний блок питания для него не потребуется. Запустите терминальное приложение.

Командный терминал запускается различными способами. В Xubuntu и других основанных на технологии XFCE дистрибутивах, например Debian, выполните из главного меню команду **Applications⇒Accessories⇒Terminal** (**Приложения⇒Стандартные⇒Терминал**). Клавиша <Super>, также известная как клавиша <Windows> или "ненужная" клавиша, поддерживается во многих операционных системах. Если в стандартном дистрибутиве Ubuntu вы используете оболочку Unity, то найдете терминал в главном меню (**Dash**), расположенном в левом верхнем углу экрана.

Для того чтобы добавить в компьютер среду разработки Arduino, установите пакет **arduino**. Ниже показано, как эта задача выполняется в Ubuntu Linux.

```
$ sudo apt-get update
$ sudo apt-get -y install arduino
```

Откройте своему пользователю доступ к USB-порту (без выполнения этого требования среда разработки Arduino не будет корректно функционировать). Первая из приведенных ниже команд добавляет пользователя в группу **dialout**, а вторая переключает командную среду на эту группу, избавляя вас от необходимости выходить из системы и регистрироваться в ней повторно.

```
$ sudo adduser $(whoami) dialout
$ newgrp dialout
```

Запустите среду разработки Arduino следующей командой:

```
$ arduino
```

Среда разработки готова к использованию. При выходе и последующем входе в систему вы сможете запустить среду разработки Arduino из меню.

Теперь можно приступить к тестированию среды разработки Arduino и запуску первой программы.

Windows 7 и Windows 8

Загрузите последнюю версию программного пакета Arduino по следующему адресу:
<http://arduino.cc/en/Main/Software>

Распакуйте загруженный архив в любое удобное для вас место (на рабочий стол или в папку Загрузки, например).

Подключите плату Arduino Uno к компьютеру с помощью кабеля USB. Как уже упоминалось, плата Arduino получает питание из USB-порта, поэтому ей не требуются дополнительные источники электроэнергии. Операционная система Windows автоматически найдет новое оборудование и начнет процесс установки драйверов для Arduino. Однако процесс установки далеко не всегда завершается успешно, и в подобных случаях на экране появляется сообщение об ошибке.

Если драйверы для Arduino автоматически установить не удалось, то выполните следующие действия.

1. Запустите программу Проводник (Windows Explorer), щелкните правой кнопкой мыши на значке Компьютер и выберите в контекстном меню команду Управление.
2. В окне Управление компьютером перейдите в раздел Диспетчер устройств, выбрав соответствующую команду на левой панели. Найдите в списке оборудования Arduino Uno, щелкните на нем правой кнопкой мыши и выберите команду Обновить драйверы.
3. Выберите вариант Выполнить поиск драйверов на этом компьютере. Перейдите в папку, в которую ранее распаковывался дистрибутив программного пакета Arduino, откройте папку с драйверами, выделите файл `arduino.inf` и щелкните на кнопке Далее.
4. Windows установит драйверы для Arduino.

Запустите среду разработки Arduino, дважды щелкнув на значке Arduino в папке, в которую был распакован программный пакет Arduino.

Можно приступить к тестированию среды разработки Arduino и запуску первой программы.

OS X

Загрузите последнюю версию программного пакета Arduino по такому адресу:

<http://arduino.cc/en/Main/Software>

Распакуйте загруженный архив и скопируйте его содержимое в папку `/Applications`.

Подключите Arduino Uno к компьютеру с помощью USB-кабеля. Как уже упоминалось, плата Arduino получает питание от USB-порта, поэтому ей не нужны дополнительные источники электроэнергии. Операционная система OS X не требует установки специальных драйверов для Arduino.

Запустите среду разработки Arduino, дважды щелкнув на значке Arduino в каталоге `/Applications`.

Можно приступить к тестированию среды разработки Arduino и запуску первой программы.

Приветствие светодиодом

Теперь, когда вы установили в операционной системе среду разработки Arduino, давайте выполним на этой платформе эквивалент программы приветствия, с которой вы уже познакомились при тестировании Raspberry Pi в предыдущей главе.

Для начала удостоверьтесь, что среда разработки правильно определила вашу платформу. По умолчанию в ней указывается вариант Arduino Uno. Если у вас другая версия платформы Arduino, например Mega или Leonardo, то укажите ее в меню `Tools⇒Board` (`Сервис⇒Плата`).

Загрузите тестовую программу Blink (Мигание). Выполните команду `File⇒Examples⇒1.Basics⇒Blink` (Файл⇒Примеры⇒1.Базовые⇒Мигание). Щелкните на кнопке Upload (Загрузить) или выполните команду `File⇒Upload` (Файл⇒Загрузить), чтобы скомпилировать и загрузить программу в Arduino.

При загрузке первой программы в среде разработки Arduino на экране может отобразиться сообщение об ошибке: `Serial port COM1 not found` (Последовательный порт COM1 не найден). Это происходит потому, что вы не указали, какой именно последовательный порт будете использовать для передачи данных (подключение Arduino к компьютеру представляется как последовательное соединение USB). Выберите необходимый порт в раскрывающемся меню. В Linux это будет `/dev/ttyACM0`, в Mac — скорее всего `/dev/usbmodem1234`, а в Windows — один из COM-портов.

Если на экране появилось сообщение об ошибке, отличной от ошибки последовательного соединения, то все равно укажите необходимый порт в меню **Tools** ⇒ **Port** (**Сервис** ⇒ **Последовательный порт**). Если вам сложно определить, к какому порту подключена плата Arduino, то сначала запомните все порты, перечисленные в списке, а затем отключите Arduino от компьютера и посмотрите, какой порт перестал отображаться в списке. Именно через этот порт Arduino и подключается к компьютеру. В OS X каждый порт перечисляется в списке дважды, например `/dev/cu.usbmodem1234` и `/dev/tty.usbmodem1234`. Вам подойдет любой из вариантов.

При загрузке программы в Arduino на его плате быстро мигают индикаторы TX (передача данных) и RX (прием данных). А при выполнении программы мигать начинает небольшой индикатор, обозначенный символом L.

Мигание светодиода L извещает об успешной загрузке и выполнении первой тестовой программы.

Примите наши поздравления! Зачем вообще нужно было запускать встроенную тестовую программу? Все просто: в будущем, при реализации сложных проектов, у вас может сложиться ситуация, когда вы засомневаетесь, а выполняет ли Arduino написанный вами исходный код? Вот тогда вспомните свой первый опыт работы с Arduino и повторите его, чтобы удостовериться в работоспособности платформы.

Структура программы Arduino

В Arduino весь единожды выполняемый программный код находится в теле функции `setup()`. Вместе с этим код функции `loop()` выполняется повторно бесконечное количество раз (до прекращения подачи питания на плату). Рассмотрим листинг 2.1.

Листинг 2.1. `blink.ino`

```
// blink.ino - мигание светодиода L при проверке среды
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void setup() { // ❶
    pinMode(13, OUTPUT); // ❷
}

void loop() { // ❸
    digitalWrite(13, HIGH); // ❹
    delay(1000); // мс // ❺
    digitalWrite(13, LOW);
    delay(1000);
}
```

- ❶ При загрузке программы в Arduino код, содержащийся в теле функции `setup()`, выполняется один раз.
- ❷ Установка вывода D13 в режим вывода сигнала (OUTPUT); предоставляется возможность управлять им из программы.
- ❸ После завершения функции `setup()` вызывается функция `loop()`. После выполнения всех включенных в нее операторов функция `loop()` вызывается повторно. Потом еще раз. И так до бесконечности.
- ❹ Выводу D13 назначается сигнал высокого уровня (HIGH), что соответствует подаче на этот контакт платы Arduino напряжения +5 В.
- ❺ В течение указанной временной задержки состояние вывода не изменяется. Поскольку выводу D13 присвоено значение HIGH, встроенный в Arduino индикатор L продолжает светиться. В течение следующей задержки на вывод передается значение LOW (сигнал низкого уровня), поэтому индикатор L гаснет. Поочередно: светится одну секунду (100 мс) и не светится — тоже одну секунду. И так вечно (шутка)!

Дополнительные модули: простота и удобство использования

Дополнительный модуль, или плата расширения¹, — это плата, устанавливаемая поверх платы Arduino и расширяющая функциональные возможности последней или повышающая удобство работы с ней (рис. 2.2). Существует огромное количество самых разнообразных плат расширения для Arduino, начиная с модулей макетирования (прототипирования) и заканчивая сложными устройствами, такими как адаптеры Ethernet или Wi-Fi. Главное достоинство дополнительных модулей для Arduino заключается в их, как бы странно это ни звучало, беспроводном монтаже; они буквально надеваются на плату Arduino, подключаясь своими разъемными колодками к штекерным колодкам платы контроллера. Конечно, далеко не все задачи можно решать с помощью плат расширения, но и забывать об их существовании тоже неправильно.

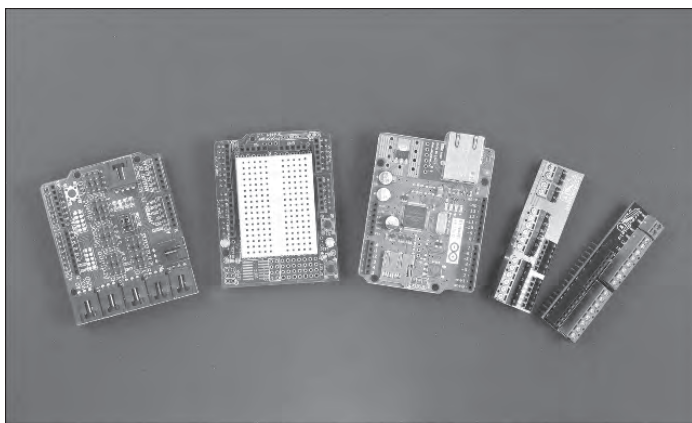


Рис. 2.2. Платы расширения

¹ Известная также как шилд, от англ. “shield” — щит. — *Примеч. ред.*

Некоторые модули вообще не содержат электронных компонентов, а предназначены для удобства подключения макетной платы: они снабжают Arduino штыревым разъемом, позволяющим надевать на него макетную плату безопасного монтажа, что избавляет от необходимости применять проволочные перемычки. Нашим любимым дополнительным модулем стал бесценный Screw Shield, снабжающий плату Arduino (с обеих сторон!) колодками с зажимными клеммами. Он предотвращает выпадение проводов из гнезд, что является самым раздражающим моментом при создании прототипов устройств на макетной плате.

Можете самостоятельно заняться разработкой и созданием плат расширения, внося посильный вклад в развитие платформы Arduino (рис. 2.3). Самое главное — это припаять к своей печатной плате разъем, совпадающий по форме и размеру с контактной колодкой Arduino.



Рис. 2.3. Дополнительные модули, созданные Андреасом Зингерле

Определение расстояния — это важнейший вопрос пространственного ориентирования. Именно поэтому ультразвуковые датчики расстояния безоговорочно рассматриваются нами во всех проводимых университетских курсах. Их незаменимость легко понять. Любой робот должен уметь определять, насколько близки к нему объекты окружающей среды. Конечно, всегда можно “прощупать” пространство перед собой, чтобы узнать о находящихся преградах, но такой способ не только малоэффективный, но и сложно реализуемый. В охранных сигнализациях датчики расстояния применяются для выявления движущихся объектов. (Еще движущиеся объекты можно определить по термометрии.) В любом многоквартирном доме, офисе и школе точно найдется хотя бы одна охранная сигнализация.

В настоящее время существуют два эффективных способа измерения расстояния до объекта: по времени возвращения отраженного звукового сигнала и по углу отражения светового пучка. Чтобы избавиться от справедливых нареканий на постоянное жужжание и мигание своих устройств, производители датчиков используют оборудование, рабочие характеристики которого находятся вне диапазонов, воспринимаемых нашими органами чувств. Именно поэтому в датчиках применяются анализаторы ультразвуковых сигналов (звуковых волн высокой частоты) и инфракрасного излучения (световых волн низкой частоты).

Несмотря на то что инфракрасное излучение невидимо для человеческого глаза, его можно легко наблюдать с помощью некоторых бытовых устройств.

Ультразвуковые датчики измеряют абсолютные значения расстояний. Например, они могут определить, что расстояние до ближайшего объекта составляет 36 см.

Для определения близости человека или другого живого организма датчик регистрирует рассеиваемое им тепловое излучение. Работающие по такому принципу приборы могут всего лишь указать на наличие в измеряемой области теплых объектов, но не в силах измерить точное расстояние до них. Существует несколько способов передачи тепла от объекта другому объекту: теплопроводность, конвекция и

тепловое излучение. Пассивный датчик инфракрасного излучения определяет наличие в окружающем пространстве объектов по их тепловому излучению.

В отличие от пассивных, активные датчики инфракрасного излучения сначала излучают невидимый для человеческого глаза пучок инфракрасного излучения, а затем анализируют возвращающийся обратно (отраженный от расположенных рядом объектов) сигнал. Но они тоже не в состоянии определить точное расстояние до объектов, а всего лишь указывают на их наличие в определенной области. Например, активный датчик инфракрасного излучения точно укажет на наличие или отсутствие объекта на расстоянии 30 см, но не сможет определить, на каком расстоянии этот объект находится: 5 или 29 см. Но в каждом правиле есть исключения; отдельные датчики инфракрасного излучения таки умеют определять расстояние до объекта по отраженному сигналу, но встречаются они крайне редко.

Активные инфракрасные датчики нашли свое практическое применение... в общественных туалетах — их монтируют в устройствах автоматического слива и сушилках для рук. Отдельные модели мусорных баков умеют автоматически открывать крышку при приближении к ним человека — это тоже заслуга активных датчиков инфракрасного излучения. Такое “санитарно-гигиеническое” применение датчиков инфракрасного излучения весьма оправдано — они устраняют необходимость прикасаться и брать в руки вещи, которые могут служить источником опасных инфекций.

Еще в одном типе устройств — видеоискателях и дальномерах — для определения расстояния до объектов применяются лазерные технологии. Некоторые из них определяют расстояние до объектов по времени возвращения отраженного от объекта луча. Поскольку скорость распространения лазерного луча в воздухе очень большая, подобные датчики должны уметь предельно точно вычислять временные интервалы. Что, конечно же, предопределяет их высокую цену (не менее 100 долларов). Именно поэтому при создании прототипов роботизированных и автоматических устройств на базе Arduino и Raspberry Pi они применяются реже, чем ультразвуковые и инфракрасные датчики.

Эксперимент: измерение расстояния ультразвуковым датчиком Ping

Все вы знаете, что такое эхо. Этот, казалось бы, забавный эффект находит применение во многих приборах, среди которых и ультразвуковой датчик. Принцип действия такого датчика предельно прост: он излучает ультразвуковые волны и определяет время, через которое возвращается отраженный эхо-сигнал. Зная скорость распространения звука в воздухе (330 м/с), рассчитать расстояние, которое пройдет ультразвуковая волна в обоих направлениях, не составит большого труда.

На сегодняшний день производится огромное количество дешевых ультразвуковых датчиков, основанных на датчике серии Ping компании Parallax (рис. 3.1). Далее вы познакомитесь с HC-SR04 — недорогим аналогом датчика Ping, а также программой управления им. Но чтобы лучше понять принцип управления ультразвуковыми датчиками, необходимо детально изучить “родоначальника” серии. Именно

поэтому знакомство с программным кодом управления ультразвуковыми датчиками мы начнем с изучения базового датчика Ping компании Parallax. Это очень популярное устройство, которое повсеместно встречается не только в торговой сети, но и во многих профессиональных мастерских, учебных заведениях и даже в гаражных кладовых.

Математические расчеты, лежащие в основе работы ультразвуковых датчиков, приведены в разделе "Расчет времени возвращения эхо-сигнала".

Датчик Ping — это "ветеран" индустрии производства Parallax. Он далеко не дешев по сравнению с аналогами — его стоимость составляет чуть более 30 долларов. Если в вашем проекте применяется несколько ультразвуковых датчиков, то остановите свой выбор на модели подешевле. Если проектом предусмотрено использование только одного датчика, то Ping будет идеальным решением. Подобный ему датчик HC-SR04 стоит всего несколько долларов и отличается от Ping только одним контактом. (В HC-SR04 за отправку ультразвукового сигнала и прием эхо-сигнала отвечают разные выводы.) При этом программы управления обоими датчиками во многом подобны.

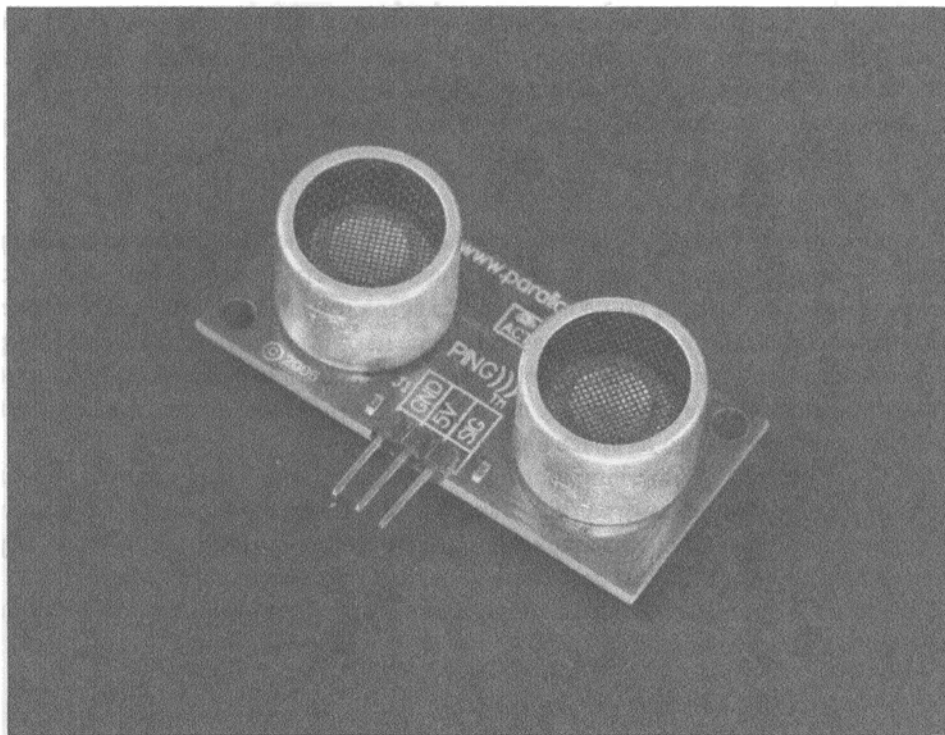


Рис. 3.1. Датчик Ping компании Parallax

Подключение к Arduino и программа управления датчиком Ping

На рис. 3.2 показана схема подключения датчика Ping к плате Arduino. Постройте показанную цепь, а затем скомпилируйте и загрузите программный код из среды разработки в Arduino.

Программный код этого примера можно загрузить с сайта книги, адрес которого был указан во введении.

Чтобы увидеть данные, передаваемые датчиком, воспользуйтесь монитором последовательного порта — в среде разработки Arduino выполните команду Tools⇌Serial Monitor (Сервис⇌Монитор порта). Если вместо текста на монитор последовательного порта выводятся непонятные данные, то удостоверьтесь, что в программном коде (функция `Serial.begin()`) и настройках последовательного порта Arduino применяются одни и те же единицы измерения пропускной способности соединения (бит/с или бод).

Не расстраивайтесь, если приведенный программный код покажется вам очень громоздким, — вы сможете без проблем применять его для определения расстояний в любых других проектах. Достаточно скопировать основную часть кода (функцию `distanceCm()` и раздел объявления глобальных переменных) в целевую программу. Для сохранения измеряемого расстояния в виде некой переменной добавьте в программный код следующую строку:

```
int d=distanceCm();
```

Поскольку датчик Ping работает на отраженном сигнале, то важно научиться правильно его использовать, вернее, научиться использовать в правильном месте. Если вы определяете расстояния до небольших объектов (около 2 см), то удостоверьтесь, что за этими объектами в поле обзора датчика не находятся другие предметы, например край стола или макетной платы. Чтобы не принимать отраженный от макетной платы сигнал, никогда не монтируйте датчик в ее центре. Возьмите за правило всегда размещать датчик у одного из краев платы.

Чтобы установить датчик Ping в стороне от Arduino и макетной платы, воспользуйтесь штекерно-гнездовым удлинительным кабелем, поставляемым в комплекте с сервоприводом. Датчик Ping оснащен тремя штыревыми контактами, поэтому обращайте особое внимание на тип разъемов на концах удлинительного кабеля.

В листинге 3.1 приведен полный код программы получения данных от ультразвукового датчика расстояния Ping.

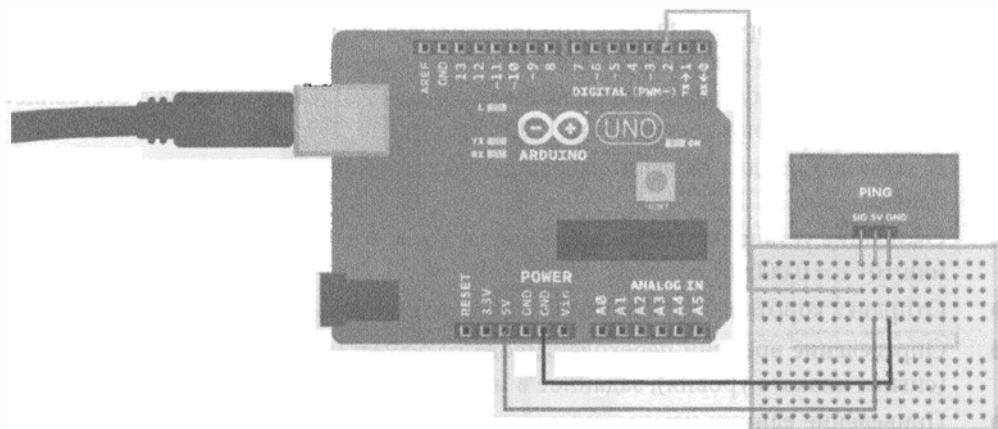


Рис. 3.2. Схема подключения датчика Ping к Arduino

Листинг 3.1. distance_ping.ino

```
// distance_ping.ino - измерение расстояния датчиком Ping
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
int pingPin = 2;
float v=331.5+0.6*20; // м/с    // ❶

void setup()
{
  Serial.begin(115200);
}

float distanceCm(){
  // отправка звукового сигнала
  pinMode(pingPin, OUTPUT); // ❷
  digitalWrite(pingPin, LOW);
  delayMicroseconds(3); // ❸
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5); // ❹
  digitalWrite(pingPin, LOW);

  // прием эхо-сигнала
  pinMode(pingPin, INPUT);
  float tUs = pulseIn(pingPin, HIGH); // мкс // ❺
  float t = tUs / 1000.0 / 1000.0 / 2; // с    // ❻
  float d = t*v; // м    // ❼
  return d*100; // см
}

void loop()
{
  int d=distanceCm(); // ❽
  Serial.println(d, DEC); // ❾
  delay(200); // мс    // ❿
}
```

- ❶ Вычисление скорости звука v при температуре $20\text{ }^{\circ}\text{C}$ (если в вашем случае температура окружающей среды значительно отличается от $20\text{ }^{\circ}\text{C}$, то укажите в расчетной формуле свое значение вместо числа 20). Скорость звука составляет около 340 метров в секунду, или 1200 км/ч.
- ❷ В датчике Ping для ввода и вывода данных используется один и тот же вывод платы.
- ❸ Ожидание установки состояния вывода; $1\text{ мкс} = 1\text{ миллионная часть секунды}$, или $10^{-6}\text{ с} = 0,000001\text{ с}$.
- ❹ Отправка короткого звукового сигнала длительностью 5 мкс или $5 \times 10^{-6}\text{ с}$.
- ❺ Определение времени перехода вывода pingPin (вывод D2) в состояние LOW (сигнал низкого уровня), заданное в микросекундах.
- ❻ Приведение единиц измерения времени к стандарту СИ (Международная система единиц, или метрическая система единиц измерения); в ней время измеряется в секундах. Обратите внимание на использование значения с плавающей запятой (1000.0) вместо целочисленного значения 1000. Этот прием позволяет получить результат в виде числа с плавающей запятой. Преобразование необратимо.
- ❼ Расстояние рассчитывается как время, умноженное на скорость.
- ❽ Вычисление расстояния и сохранение его в новой переменной d. В дальнейшем в программном коде расстояние представляется этой переменной.
- ❾ Вывод значения переменной d на мониторе последовательного порта.
- ❿ В циклических структурах всегда устанавливайте задержки. Если выполнять программный код без задания временных пауз, то в конечном счете можно перегрузить микропроцессор Arduino и понапрасну истратить много электроэнергии (загрузка одноядерного микропроцессора на 100% всегда сопровождается его сильным нагревом).

Подключение к Raspberry Pi и программа управления датчиком Ping

Подключите датчик Ping к плате Raspberry Pi, реализовав электрическую цепь, схема которой показана на рис. 3.3, а затем выполните программный код из листинга 3.2.

Будьте предельно внимательны, подключая компоненты к контактам GPIO. Неправильное подключение чревато в лучшем случае повреждением только одного вывода, а в худшем — поломкой всей платы Raspberry Pi. Чтобы избежать проблем, отключите питание и подключите цепь к другим выводам или же многократно проверяйте правильность подключения компонентов перед подачей питания на плату.

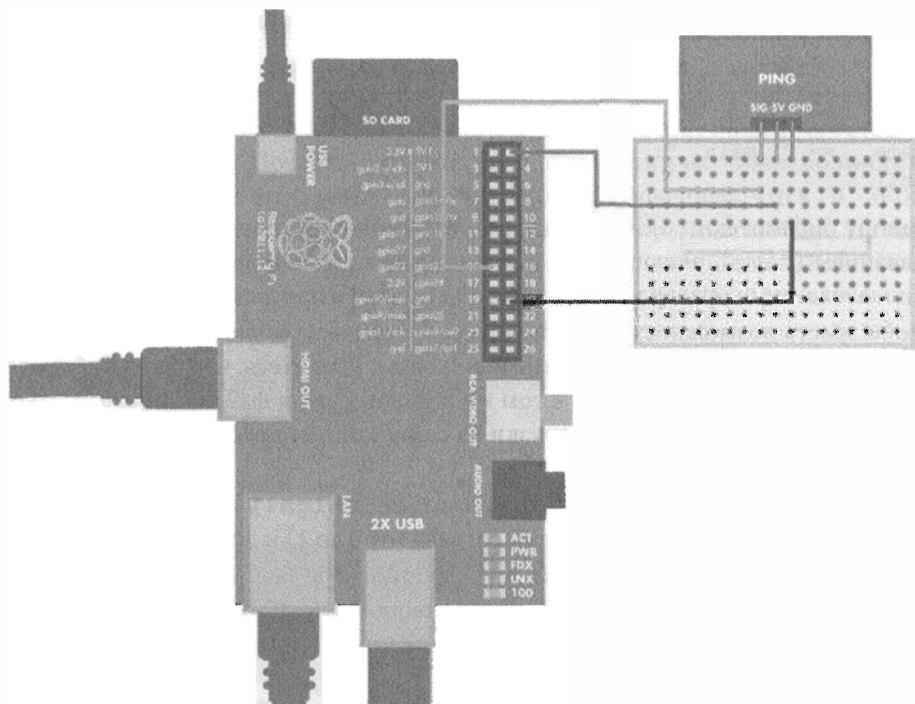


Рис. 3.3. Схема подключения датчика Ping к Raspberry Pi

Листинг 3.2. `distance_ping.py`

```
# distance_ping.py - измерение расстояния
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time # ❶
import botbook_gpio as gpio # ❷

def readDistanceCm():
    sigPin=22
    v=(331.5+0.6*20)

    gpio.interruptMode(sigPin, "both") # ❸

    gpio.mode(sigPin, "out") # ❹
    gpio.write(sigPin, gpio.LOW) # ❺
    time.sleep(0.5) # c

    gpio.write(sigPin, gpio.HIGH) # ❻
    time.sleep(1/1000.0/1000.0) # ❼
    gpio.mode(sigPin, "in") # ❽

    # Считывание длительности сигнала высокого уровня
    t = gpio.pulseInHigh(sigPin) # c # ❶
    d = t*v
    d = d/2 # ❿
    return d*100 # cm
```

```
def main():
    d = readDistanceCm() # ❶
    print "Расстояние составляет %.2f см" % d # ❷
    time.sleep(0.5)

if __name__ == "__main__":
    main()
```

- ❶ Импорт библиотеки `time` приводит к созданию пространства имен с таким же названием (`time`), включаемым в имена функций библиотеки, что определяет использование оператора `time.sleep(1)` далее в программном коде.
- ❷ Чтобы иметь возможность подключать собственные библиотеки, их нужно разметить в каталоге файла с программным кодом. Убедитесь, что файл библиотеки `botbook_gpio.py` находится в том же каталоге, что и файл `distance_ping.py`. Оба файла доступны для загрузки на сайте книги, адрес которого был указан во введении.
- ❸ Режим прерывания позволяет функции `pulseInHigh()` измерить полный импульс, начиная с увеличения уровня сигнала (от 0 до 1) и до его спада (от 1 до 0).
- ❹ В датчике `Ping` у одного и того же контакта по мере необходимости можно изменить режим обработки сигнала на противоположный. В остальных ультразвуковых датчиках, например `HC-SR04`, для входного и выходного сигнала применяются различные выводы.
- ❺ Выключение контакта и ожидание установки нового состояния. Полсекунды вполне достаточно.
- ❻ Подача импульса (сигнал растёт от минимального уровня до максимально возможного). Тщательно подходите к заданию временных интервалов.
- ❼ Ожидание в течение микросекунды (10^{-6} с), или одной миллионной доли секунды.
- ❸ Перевод вывода в режим входа. В качестве побочного эффекта вы сбрасываете сигнал, создавая спадающий фронт небольшой длительности.
- ❹ Считывание длительности сигнала в секундах. Функция `gpio.pulseInHigh()` измеряет длительность всего импульса, с начала возрастания сигнала и до завершения его спада. Raspberry Pi находится под управлением своей операционной системы, поэтому точность измерения временных интервалов в ней несколько ниже, чем в Arduino. На точность замеров значительно влияют другие программы, запущенные в среде.
- ❽ Расстояние в одну сторону рассчитывается как половина всего пути, пройденного звуковой волной.
- ❾ Эта строка кода понадобится вам, если вы планируете использовать результаты проведенных измерений в собственных программах.
- ❿ Вывод значения расстояния в окне терминала, из которого запускается программный код. Оператор `%.2f` является частью форматирующей строки. Он определяет формат вывода значения переменной `d`. Оператор `%f` указывает на значение с плавающей запятой (десятичное), `.2` — определяет количество разрядов после запятой. Если использовать простой оператор `print d`, то будет выведено длинное числовое значение, представленное десятичной дробью.

Эксперимент: измерение расстояния ультразвуковым датчиком HC-SR04

Ультразвуковой датчик HC-SR04 во многом подобен датчику Ping, но его стоимость в несколько раз меньше. Программный код управления этим датчиком во многом повторяет код управления датчиком Ping, за тем лишь исключением, что в HC-SR04 для обработки выходного и входного сигналов применяются разные выводы. Детальнее об этом будет рассказано ниже, при рассмотрении принципов подключения HC-SR04 к Arduino и Raspberry Pi.

Физические принципы определения расстояний до объектов с помощью ультразвуковых датчиков изложены в разделе "Расчет времени возвращения эхо-сигнала".

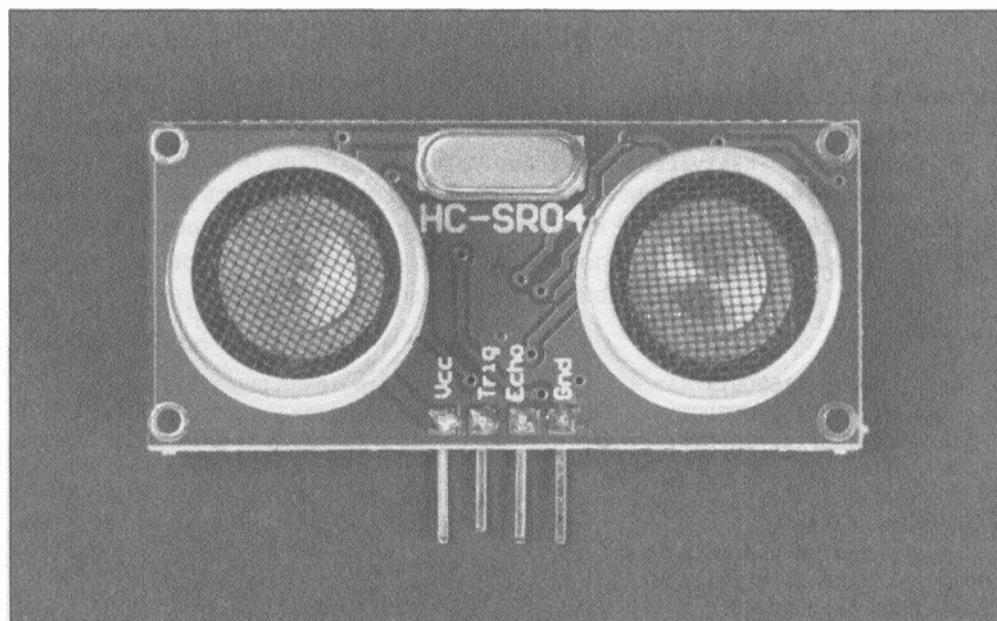


Рис. 3.4. Ультразвуковой датчик HC-SR04

Подключение к Arduino и программа управления датчиком HC-SR04

Постройте цепь, схема которой показана на рис. 3.5, и воспользуйтесь программным кодом из листинга 3.3 для управления датчиком HC-SR04.

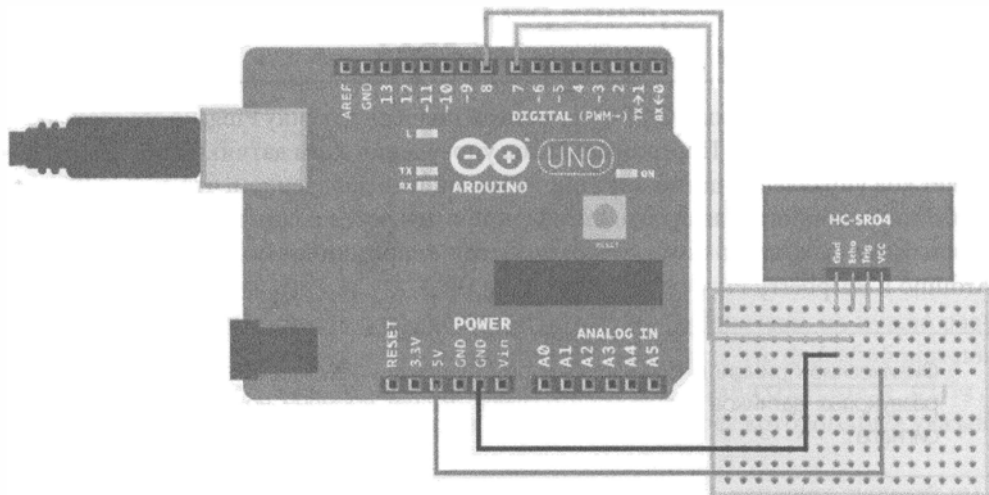


Рис. 3.5. Схема подключения датчика HC-SR04 к Arduino

Листинг 3.3. hc-sr04.ino

```
// hc_sr04.ino - измерение расстояния
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int trigPin = 8;
int echoPin = 7;
float v=331.5+0.6*20; // м/с

void setup()
{
    Serial.begin(115200);
    pinMode(trigPin, OUTPUT); // ❶
    pinMode(echoPin, INPUT); // ❷
}

float distanceM(){
    // отправка звукового сигнала
    digitalWrite(trigPin, LOW);
    delayMicroseconds(3);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(trigPin, LOW);

    // прием эхо-сигнала
    float tUs = pulseIn(echoPin, HIGH); // мкс
    float t = tUs / 1000.0 / 1000.0 / 2; // с
    float d = t*v; // м
    return d*100; // см
}

void loop() // ❸
{

```

```
int d=distanceM();
Serial.println(d, DEC);
delay(200); // мс
}
```

- ❶ В случае датчика Ping в функции `setup()` рабочее состояние контакта не определяется, поскольку в дальнейшем он все равно изменяется (в Ping один и тот же вывод применяется как для отправки звукового сигнала, так и считывания отраженного сигнала). В HC-SR04 управление отправкой звукового сигнала осуществляется через вывод Trig.
- ❷ Вывод Echo используется для получения времени, которое проходит перед регистрацией отраженного сигнала.
- ❸ В отличие от функции `setup()`, здесь основная часть программного кода, предназначенного для управления датчиком HC-SR04, повторяет таковой для датчика Ping.

Подключение к Raspberry Pi и программа управления датчиком HC-SR04

Постройте цепь согласно схеме, показанной на рис. 3.6, и выполните в Raspberry Pi программный код из листинга 3.4. Обратите внимание на то, что кроме проволочных перемычек вам дополнительно понадобятся два резистора с сопротивлением 10 кОм. (Чтобы быстро оценить порядок сопротивления резистора, воспользуйтесь правилом третьей полоски, описанным в главе 1.) Программный код управления датчиком HC-SR04 очень похож на код управления датчиком Ping.

Листинг 3.4. `hc-sr04.py`

```
# hc-sr04.py - измерение расстояния до объекта в сантиметрах
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio

def readDistanceCm():
    triggerPin = 22 # ❶
    echoPin = 27

    v=(331.5+0.6*20) # м/с

    gpio.mode(triggerPin, "out")

    gpio.mode(echoPin, "in")
    gpio.interruptMode(echoPin, "both")

    gpio.write(triggerPin, gpio.LOW)
    time.sleep(0.5)

    gpio.write(triggerPin, gpio.HIGH)
    time.sleep(1/1000.0/1000.0)
    gpio.write(triggerPin, gpio.LOW)
```

```

t = gpio.pulseInHigh(echoPin) # c

d = t*v
d = d/2
return d*100 # см

def main():
    d = readDistanceCm() # ❶
    print "Расстояние: %.2f см" % d
    time.sleep(0.5)

if __name__ == "__main__":
    main()

```

- ❶ Единственное отличие HC-SR04 от Ping заключается в замене функций одного контакта двумя (Trig и Echo).
- ❷ Считывание результата в HC-SR04 производится так же, как и в Ping.

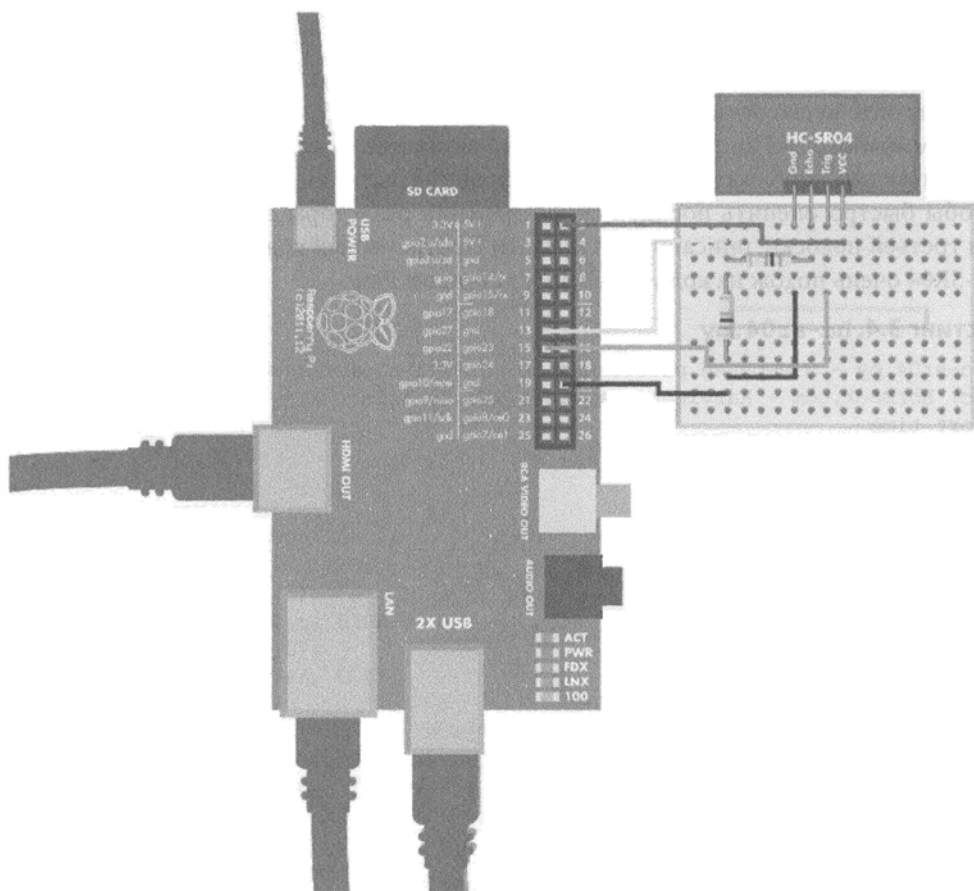


Рис. 3.6. Схема подключения датчика HC-SR04 к Raspberry Pi (см. цветную вклейку)

Вы могли заметить, что при подключении HC-SR04 в цепь добавляется резистор, а в случае Ping — нет. В чем же разница? Согласно технической документации, выходной сигнал в HC-SR04 формируется на уровне ТТЛ, а потому составляет +5 В. В технических характеристиках датчика Ping указывается, что выходной сигнал высокого уровня представлен напряжением +3,3 В. Если сопоставить рабочие характеристики Raspberry Pi и датчика HC-SR04, то получается, что порт GPIO рассчитан на считывание сигнала высокого уровня с напряжением +3,3 В, а датчик подает на выход сигнал с напряжением +5 В, что может вызвать повреждение платы Raspberry Pi.

Расчет времени возвращения эхо-сигнала

За окном непогода, и надвигается гроза? Воспользовавшись случаем, вы можете измерить расстояние до места удара молнии в землю, сравнив задержку во времени между появлением вспышки и раскатами грома. Сосчитайте количество секунд от момента вспышки до первых раскатов грома. Каждая сосчитанная секунда соответствует пройденному звуком расстоянию в 330 метров (на самом деле скорость звуковой волны зависит от температуры воздуха, на чем мы детально остановимся ниже).

Математические расчеты не очень сложные: если раскаты грома вы услышали через 3 секунды после вспышки молнии, то расстояние до места удара составляет $330 \text{ м/с} \times 3 \text{ с} = 990 \text{ м}$. Округлив полученное значение, можно утверждать, что за 3 секунды звук проходит целый километр (милю звуковая волна преодолеет приблизительно за 5 секунд). Как оказывается, даже на таких расстояниях вспышка видна мгновенно, а звук приходит с заметным опозданием.

Ультразвуковые датчики обычно применяются для измерения коротких расстояний, от 3 см до 6 м. Строительные дальномеры (также работающие по принципу эхо-локации ультразвуковых волн), приобретаемые в хозяйственных магазинах, рассчитаны на измерение несколько больших расстояний — до 20 м (они оснащаются термометрами для калибровки скорости звука согласно температуре воздуха).

Время, которое требуется звуку для преодоления расстояния в 1 см, совсем небольшое — около 30 мкс: 30 миллионных долей секунды. Непривычно работать со столь малыми величинами?

При решении любой сложной математической задачи с очень маленькими и/или очень большими числовыми значениями проще всего смоделировать ситуацию на примере аналогичной задачи, представленной привычными, встречающимися в повседневной жизни величинами. Например, двигаясь в автомобиле со скоростью 50 км/ч на протяжении двух часов по однотипному ландшафту, вы сильно устанете от столь скучной поездки.

Как бы там ни было, зная скорость (v) автомобиля и время, проведенное в пути (t), вы быстро узнаете пройденное расстояние (d):

$$t = 2 \text{ ч};$$

$$v = 50 \text{ км/ч};$$

$$d = t \times v = 2 \text{ ч} \times 50 \text{ км/ч} = 2 \times 50 \text{ ч} \times \text{км/ч} = 100 \text{ км}.$$

Все не так уж и сложно. Как видите, в нашей модели два часа в пути соответствуют пройденному расстоянию в 100 км. Давайте применим эту же формулу для расчетов коротких временных интервалов (3,33 мс) и очень высоких значений скорости, проведя вычисления в метрической системе измерений. (Приставка милли- обозначает тысячные доли, миллисекунда — это одна тысячная доля секунды.)

$$t = 3,33 \text{ мс} = 0,00333 \text{ с};$$

$$v = 330 \text{ м/с};$$

$$d = t \times v = 0,00333 \text{ с} \times 330 \text{ м/с} = 1,10 \text{ м}.$$

Именно по таким формулам и определяются расстояния, измеренные ультразвуковыми датчиками: за 3,33 мс отраженный от преграды звуковой сигнал пройдет расстояние 1,1 м.

Изучая программный код, написанный другими разработчиками, вы можете встретить несколько необычные величины и единицы измерения. Вместо скорости, измеряемой в метрах в секунду, часто применяется инверсная величина — миллисекунд на метр, чаще представляемая как мс/м:

$$1/v = 1/(330 \text{ м/с}) = 0,00303 \text{ с/м} = 3,03 \text{ мс/м}.$$

Эта величина показывает, что на прохождение метра расстояния у звуковой волны уходит приблизительно 3 миллисекунды.

Звук лучше распространяется в теплом воздухе. Звук — это колебания, а любые колебания лучше распространяются в среде, в которой молекулы уже колеблются под воздействием температуры. Если вы проживаете в южной климатической зоне, то беспокоиться о калибровке датчика вам скорее всего не придется. В северных широтах, например в Финляндии, температура в жилом помещении составляет +22 °С, а снаружи может опускаться до -40 °С. Таким образом, разница температур внутри и вне помещения может составлять до 60 °С. Эта разница настолько существенна, что требует корректировки проведенных выше вычислений. Температура среды (T) напрямую влияет на скорость звука (v) в этой среде следующим образом:

$$v = (331,3 + 0,606 \times T) \text{ м/с}.$$

Проведя вычисления по этой формуле, вы увидите, что скорость звука в воздухе при температуре +22 °С составляет 343 м/с. Принимая во внимание приведенную выше формулу, вы быстро откалибруете датчик при любой температуре воздуха. Кроме того, взобравшись высоко в горы или опустившись на большую глубину в батискафе, вам придется учитывать изменение атмосферного давления. В пустыне с сухим воздухом и во влажной прачечной скорости звуковых волн будут разными. Очень жаль, что ультразвуковые датчики расстояния общего назначения можно калибровать только согласно изменению температуры окружающей среды.

Решив проводить калибровочные расчеты в программном коде, обязательно включайте их в начало программы. Тем самым эти вычисления в Arduino и Raspberry Pi будут проводиться только один раз, хотя любой программный код вне цикла `loop()` и так выполняется единожды. Не забудьте снабдить вычисления детальными комментариями, чтобы не запутаться в расчетных формулах, когда вернетесь к рассмотрению программного кода спустя несколько недель после его написания.

Эксперимент в окружающей среде: невидимые объекты

Вы можете легко одурачить ультразвуковой датчик движения так, что он перестанет “замечать” объекты перед собой. Закрепите датчик в штативе или другом устройстве (обеспечив его неподвижность) и наведите на твердый, плоский объект. Загрузите код управления датчиком, рассмотренный выше, и понаблюдайте за показаниями монитора последовательного порта. Удостоверьтесь, что на монитор выводятся правильные значения расстояния до объекта.

Далее закрепите между датчиком и твердым телом некий мягкий объект, например подушку или плюшевую игрушку, как показано на рис. 3.7. Взгляните на монитор последовательного порта. “Видит” ли датчик твердый объект сейчас?

Еще одна “ахиллесова пята” ультразвуковых датчиков — это наклонные поверхности. Уберите мягкое тело, закрывающее твердый объект от датчика, и начинайте поворачивать его. Внимательно следите за показаниями на мониторе при повороте плоскости твердого тела относительно датчика.

Почему датчик перестает наблюдать объекты? Мягкие вещи (как плюшевый кролик на рис. 3.7) поглощают большую часть звуковых волн, и от них почти ничего не отражается в обратном направлении. В случае поворота твердой поверхности звуковые волны отражаются, но не обратно, а под таким углом, что отраженный сигнал проходит мимо приемника датчика. По такому же принципу построена антирадарная система некоторых моделей военных самолетов.

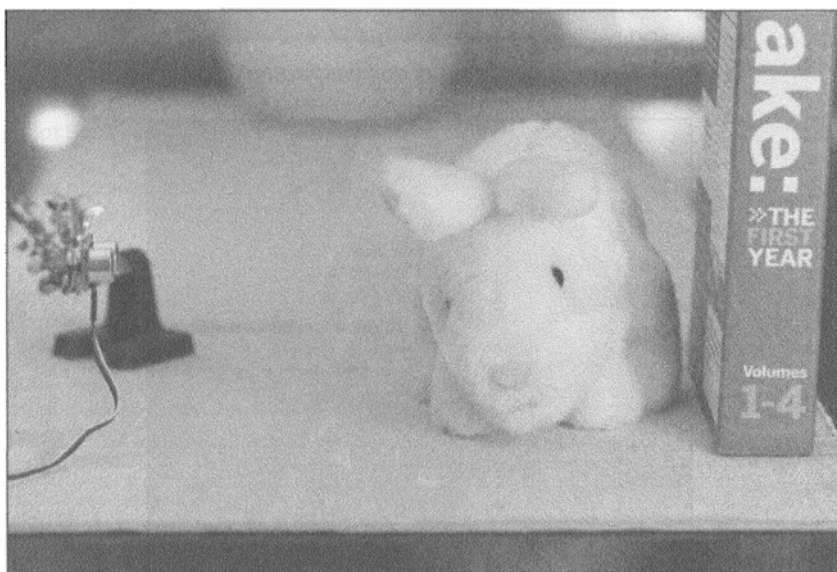


Рис. 3.7. Тестирование ультразвукового датчика на простых объектах

Эксперимент в окружающей среде: обнаружение преград датчиком инфракрасного излучения

Датчик инфракрасного излучения (рис. 3.8) гораздо надежнее ультразвукового датчика, но область его применения сильно ограничена. Вы не сможете так же просто обмануть датчик инфракрасного излучения, как ультразвуковой датчик. Датчик ИК-излучения представляет качественную характеристику пространства перед собой, а не количественную. Он показывает наличие чего-то перед собой, но не расстояние до него (рис. 3.9). А так как Солнце является самым большим источником инфракрасного излучения на планете, повредить датчик ИК-излучения на открытом воздухе не составит большого труда.

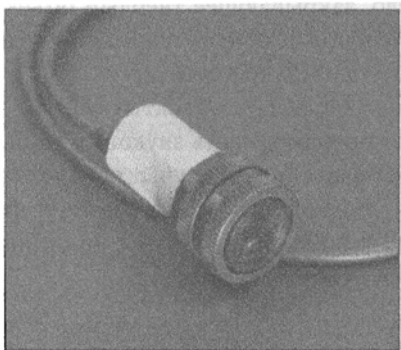


Рис. 3.8. Типичный датчик инфракрасного излучения

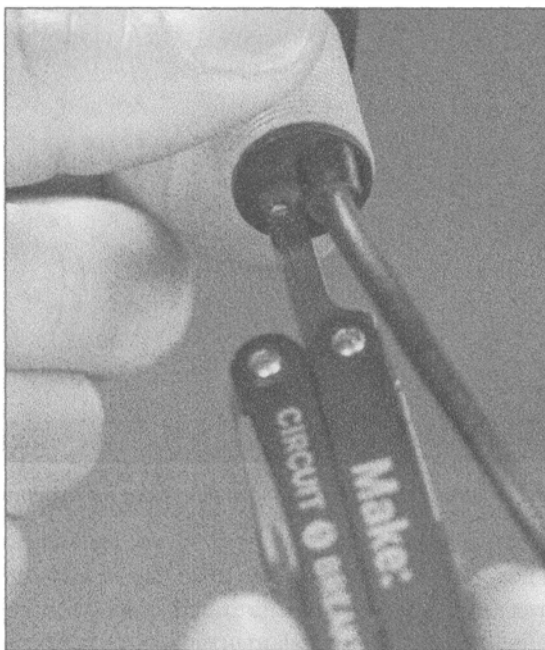


Рис. 3.9. Настройка расстояния, до которого определяются преграды

Подключение к Arduino и программа управления датчиком инфракрасного излучения

На рис. 3.10 показана цепь подключения датчика ИК-излучения к Arduino. “Скетч” управления этим устройством в Arduino приведен в листинге 3.5.

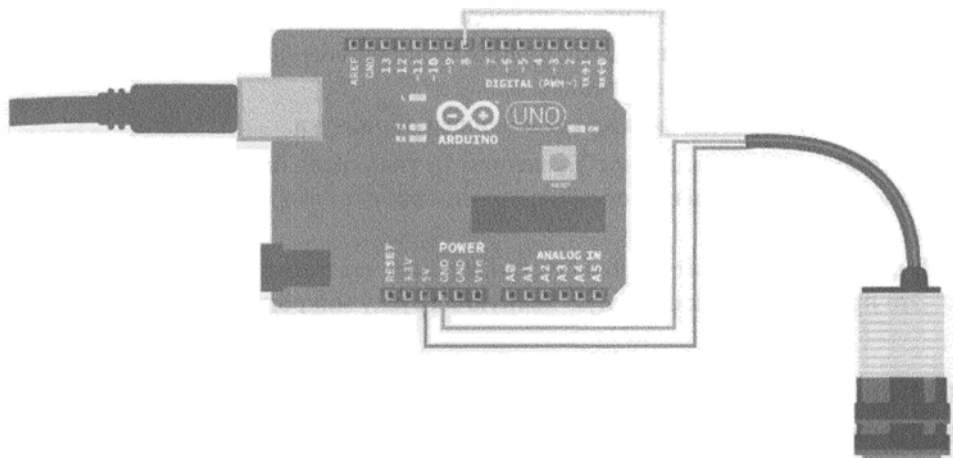


Рис. 3.10. Схема подключения датчика инфракрасного излучения к Arduino

Листинг 3.5. `adjustable_infrared_sensor_switch.ino`

```
// adjustable_infrared_sensor_switch.ino - вывод результатов
// обнаружения объекта и включение светодиода
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 8;
const int ledPin = 13;

// Начальное состояние
int switchState = 0;

void setup() {
  Serial.begin(115200); // ❶
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  switchState = digitalRead(sensorPin); // ❷
  Serial.println(switchState); // ❸
  if(switchState == 0) {
    digitalWrite(ledPin, HIGH);
    Serial.println("Обнаружен объект!"); // ❹
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(10); // мс // ❺
}
```

- ❶ Запуск монитора последовательного порта Arduino (команда Tools⇒Serial Monitor (Сервис⇒Монитор порта)). Вам нужно синхронизировать пропускную способность монитора и скорость передачи данных, заданную в программном коде. Самое высокое значение пропускной способности устанавливается на уровне 115 200 бит/с. В случае использования кабеля USB неизвестного качества (или большой длины) уменьшите ее до уровня 9 600 бит/с.
- ❷ Функционально датчик ИК-излучения подобен обычной кнопке. В этой строке считывается состояние датчика.
- ❸ Вывод состояния датчика для дальнейшей обработки.
- ❹ Состояние, представленное значением 0, указывает на обнаружение датчиком объекта. Для извещения об обнаружении объекта загорается светодиод, встроенный в Arduino.
- ❺ В цикле loop() всегда нужно устанавливать задержку, хоть и незначительную. Это предотвратит постоянную полную загрузку микропроцессора Arduino.

Подключение к Raspberry Pi и программа управления датчиком инфракрасного излучения

На рис. 3.11 показана схема подключения датчика ИК-излучения к Raspberry Pi. Соответствующий программный код Python управления датчиком приведен в листинге 3.6.

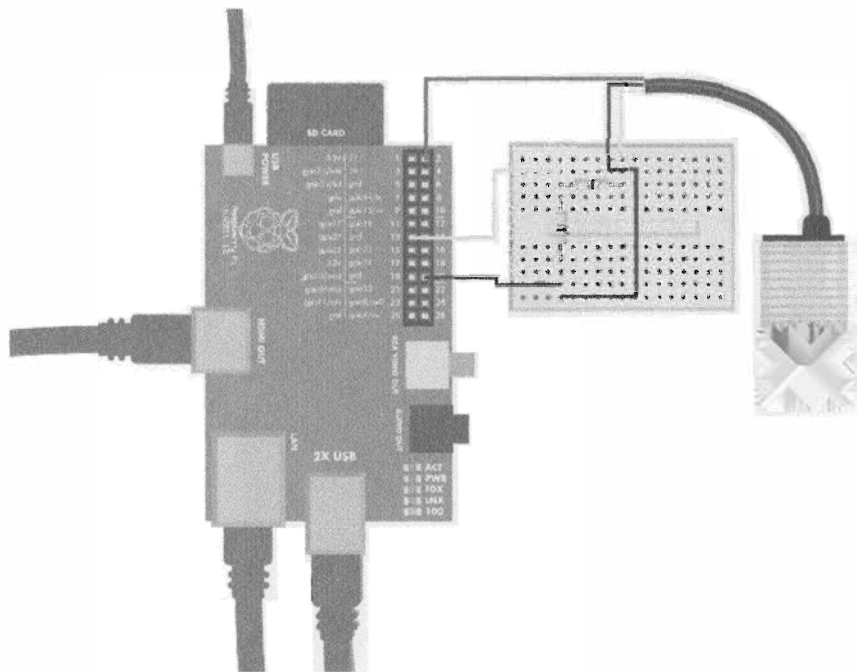


Рис. 3.11. Схема подключения датчика инфракрасного излучения к Raspberry Pi (см. цветную вклейку)

Листинг 3.6. adjustable-infrared-sensor-switch.py

```
# adjustable-infrared-sensor-switch.py - считывание сигнала
# с ИК-датчика
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio # ❶

def main():
    switchPin = 27
    gpio.mode(switchPin, "in") # ❷
    x = gpio.read(switchPin) # ❸
    if( x == gpio.LOW ): # ❹
        print "Что-то здесь все же есть"
    else:
        print "Полное отсутствие чего-либо"
    time.sleep(0.1)

if __name__ == "__main__":
    main()
```

- ❶ Загрузка библиотеки `botbook_gpio`. Ее нужно сохранить в том же каталоге, что и файл программы. Удостоверьтесь лишний раз, что файлы `botbook_gpio.py` и `adjustable-infraredsensor-switch.py` хранятся вместе. Вы найдете необходимый каталог в архиве с примерами книги, доступном на сайте, адрес которого был указан во введении.
- ❷ Настройка вывода, к которому подключается датчик. Контакт переводится в режим входа.
- ❸ Считывание получаемого сигнала и сохранение его в переменной `x`.
- ❹ Если на вывод подается сигнал низкого уровня, то это означает, что датчик обнаружил перед собой объект(ы).

Эксперимент: инфракрасное зрение

Как вы уже знаете, инфракрасное излучение лежит вне видимого диапазона световых волн. Как же нам увидеть это ИК-излучение, если в этом возникнет острая необходимость? Нам помогут приборы ночного видения или, если вы не разжились столь дорогостоящим прибором, простая цифровая камера.

Взгляните на датчик инфракрасного излучения через камеру смартфона. Вокруг ИК-излучателя устройства вы будете наблюдать фиолетовое свечение (рис. 3.12). Выключите в комнате освещение и запахните шторы, чтобы сделать его сильнее. Конечно, ИК-излучатель в датчике не сравнится с таковым в приборах ночного видения, но позволит получить вам представление об инфракрасных источниках света.



Рис. 3.12. Таким “видит” датчик инфракрасного излучения камера смартфона

А что, если проделать такой же трюк с дорогим зеркальным фотоаппаратом? Боюсь, что ничего неожиданного не произойдет. Многие дорогие фотоаппараты оснащаются инфракрасными фильтрами, предотвращающими прохождение через объектив световых волн нежелательной длины (в дешевых камерах, в том числе тех, которыми оснащается большинство смартфонов, такие фильтры не устанавливаются, поскольку инфракрасное излучение востребовано во многих датчиках аппарата). В таком случае зайдите в полностью неосвещенное помещение, закрепите фотоаппарат на штативе и настройте его на очень длинную выдержку (по возможности, несколько секунд), после чего сделайте несколько снимков. Поскольку инфракрасное освещение представляет в темной комнате единственный источник света, то именно оно и будет зарегистрировано вашей камерой.

Являясь счастливым обладателем приборов ночного видения, вы в полной мере сможете познакомиться со свечением ИК-излучателя датчика. Прибор ночного видения усиливает любое излучение из видимого диапазона, но особенно сильно эта функция проявляется в инфракрасной части спектра. Самые дешевые модели подобных устройств вообще оснащаются встроенными ИК-излучателями, улавливая собственное отраженное инфракрасное излучение. Если у вас в руках прибор ночного видения именно такого типа, то не забудьте его включить перед наблюдением свечения датчика ИК-излучения. Но все же основное назначение приборов ночного видения заключается не в изучении свечения ИК-излучателей, а в наблюдении отраженного от окружающих объектов инфракрасного света (рис. 3.13).



Рис. 3.13. Датчик инфракрасного излучения в окулярах прибора ночного видения

Эксперимент в окружающей среде: слежение за перемещением объекта (составной датчик инфракрасного излучения)

Составной датчик представляет собой набор ИК-фоторезисторов и ИК-светодиодов, смонтированных на одной плате в единое устройство. Он способен отслеживать движение объектов на расстоянии 20 сантиметров от себя. Несмотря на то что составной ИК-датчик работает как единое устройство, сигнал с каждого из ИК-фоторезисторов можно обрабатывать отдельно. Коррекция рассеянного внешнего освещения осуществляется в результате последовательного отключения ИК-светодиодов и последующего сравнения полученных результатов.

Точное название датчика звучит как **IR Compound Eye** (инфракрасный фасеточный глаз). Несколько обескураживающе, не правда ли? Если в будущем вы планируете посвятить себя производству или торговле датчиками, то обязательно дополняйте название модели уникальным производственным номером, чтобы избежать неоднозначности в трактовке их предназначения.

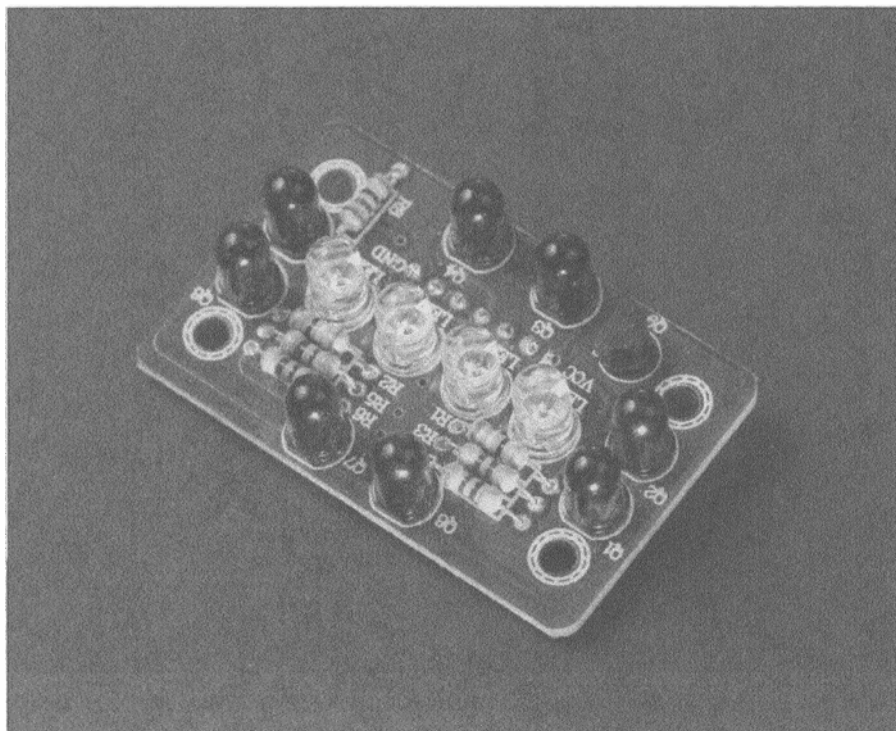


Рис. 3.14. Модуль составного датчика инфракрасного излучения

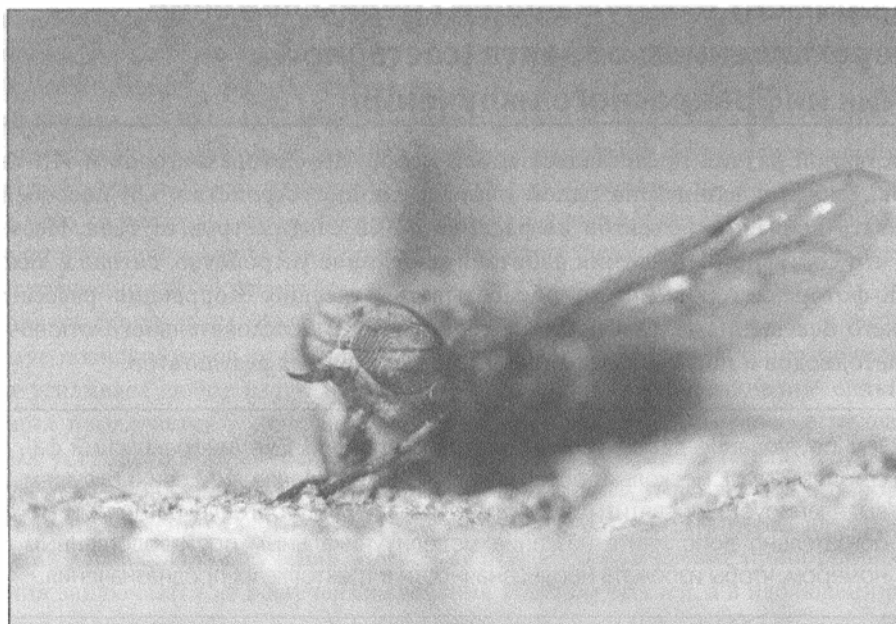


Рис. 3.15. Составной (фасеточный) глаз легко встретить в живой природе

Чтобы повысить точность работы составного ИК-датчика, вначале нужно его откалибровать. Дождитесь ночи, когда рассеянное инфракрасное освещение полностью отсутствует (ведь даже через плотно закрытые шторы в помещение проникает часть дневного света). Если не хотите ждать глубокой ночи, то спуститесь в подвал или комнату без окон и плотно закройте за собой дверь. Постройте цепь, схема которой показана рис. 3.16, чтобы измерить сигналы, выводимые составным датчиком. Поместите лист бумаги перед составным датчиком (на расстоянии около 20 см) и проанализируйте отличия в уровнях сигналов, выводимых на каждый контакт устройства. Значения должны совпадать с небольшими отличиями (± 100). Если одно из значений слишком большое, то заклейте соответствующий ИК-светодиод непрозрачным скотчем или обтяните термоусаживаемой пленкой. Если отдельные значения слишком малы, то прикройте ИК-светодиоды соответствующих датчиков.

Этот составной датчик в общей классификации относится к аналоговым резистивным устройствам. Простейший пример считывания аналогового сопротивления приведен в главе 5.

Подключение к Arduino и программа управления составным датчиком инфракрасного излучения

На рис. 3.16 показана схема цепи подключения составного ИК-датчика к Arduino. Программа управления таким датчиком приведена в листинге 3.7.

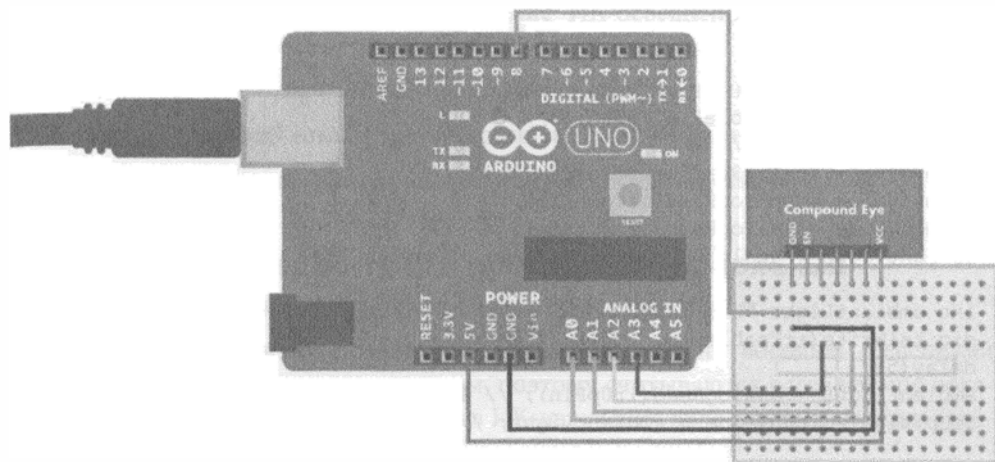


Рис. 3.16. Схема подключения составного датчика инфракрасного излучения к Arduino (см. цветную вклейку)

Листинг 3.7. `compound_eye.ino`

```
// compound_eye.ino - вывод расстояний и направлений
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```

const int irEnablePin = 8; // ❶
const int irUpPin = 0;
const int irDownPin = 2;
const int irLeftPin = 1;
const int irRightPin = 3;

int distance = 0; // ❷
int irUpValue = 0;
int irDownValue = 0;
int irLeftValue = 0;
int irRightValue = 0;

void setup() {
    Serial.begin(115200);
    pinMode(irEnablePin, OUTPUT);
}

void loop() {
    readSensor(); // ❸
    Serial.print("Значения: "); // ❹
    Serial.print("irUpValue"); Serial.print(irUpValue);
    Serial.print(",");
    Serial.print("irDownValue"); Serial.print(irDownValue);
    Serial.print(",");
    Serial.print("irLeftValue"); Serial.print(irLeftValue);
    Serial.print(",");
    Serial.print("irRightValue"); Serial.print(irRightValue);
    Serial.print(",");
    Serial.print("расстояние"); Serial.println(distance);
    delay(100);
}

void readSensor() {
    digitalWrite(irEnablePin, HIGH); // ❺
    delay(5); // мс // ❻
    irUpValue = analogRead(irUpPin);
    irDownValue = analogRead(irDownPin);
    irLeftValue = analogRead(irLeftPin);
    irRightValue = analogRead(irRightPin);

    int ambientLight = 0; // ❼
    digitalWrite(irEnablePin, LOW); // ❽
    delay(5);
    ambientLight = analogRead(irUpPin); // ❾
    irUpValue = irUpValue - ambientLight; // ❿

    ambientLight = analogRead(irDownPin);
    irDownValue = irDownValue - ambientLight;

    ambientLight = analogRead(irLeftPin);
    irLeftValue = irLeftValue - ambientLight;

    ambientLight = analogRead(irRightPin);

```

```

    irRightValue = irRightValue - ambientLight;

    distance = (irUpValue+irDownValue+irLeftValue+irRightValue) / 4; // ❶
}

```

- ❶ Номера выводов объявлены как константы (`const`), поэтому далее в программе они не изменяются (если вы попытаетесь создать код, изменяющий номера выводов — умышленно или ошибочно, — то при его компиляции и выгрузке в Arduino получите сообщение об ошибке).
- ❷ Значения, возвращаемые датчиками, сохраняются в глобальных переменных. Глобальные переменные разрешается использовать в любых функциях. В языках C и C++ (на последнем основан язык программирования Arduino) считается хорошей практикой присваивать переменным значения на этапе объявления (например, `int foo = 0;`).
- ❸ Функция `readSensor()` не возвращает значений. В ее задачу входит изменять глобальные переменные. При ее выполнении одновременно вносятся изменения во многие значения.
- ❹ Вывод данных. Функция `Serial.print()` не создает новую строку (для вывода данных в новой строке применяется функция `Serial.println()`).
- ❺ Включение ИК-светодиодов для освещения цели измерения.
- ❻ Ожидание установки состояния выводов.
- ❼ Подготовка к измерению естественного освещения (например, невидимого инфракрасного солнечного излучения).
- ❽ Выключение ИК-светодиодов. Измерение ИК-освещения будет проводиться только от естественных источников.
- ❾ Применение ИК-фоторезисторов для измерения естественного освещения.
- ❿ Вычитание значений естественного освещения из показаний каждого датчика.
- ⓫ Вычисление среднего расстояния для четырех ИК-фоторезисторов.

Подключение к Raspberry Pi и программа управления составным датчиком инфракрасного излучения

Рассматриваемый нами составной датчик оснащен восемью сенсорами ИК-излучения, скомпонованными в пары, поэтому считываются всего четыре показателя. Каждый из этих четырех датчиков ИК-излучения является аналоговым резистивным устройством.

Таким образом, чтобы считать показания с составного датчика в Raspberry Pi, требуется внешний аналого-цифровой преобразователь (Analog-to-Digital Converter — ADC). Мы будем использовать микросхему MCP3002, которая способна одновременно преобразовывать в цифровые два входных аналоговых сигнала. Поскольку нам требуется одновременно обрабатывать входные сигналы с четырех датчиков, то в цепь подключения составного ИК-датчика добавлены две микросхемы MCP3002.

Сама цепь, показанная на рис. 3.17, несколько сложнее всех рассматриваемых ранее, хотя принцип ее работы остается простым: к Raspberry Pi подключается четыре аналоговых резистивных датчика, показания которых считываются последовательно. Постройте цепь согласно приведенной схеме, а затем выполните программу из листинга 3.8.

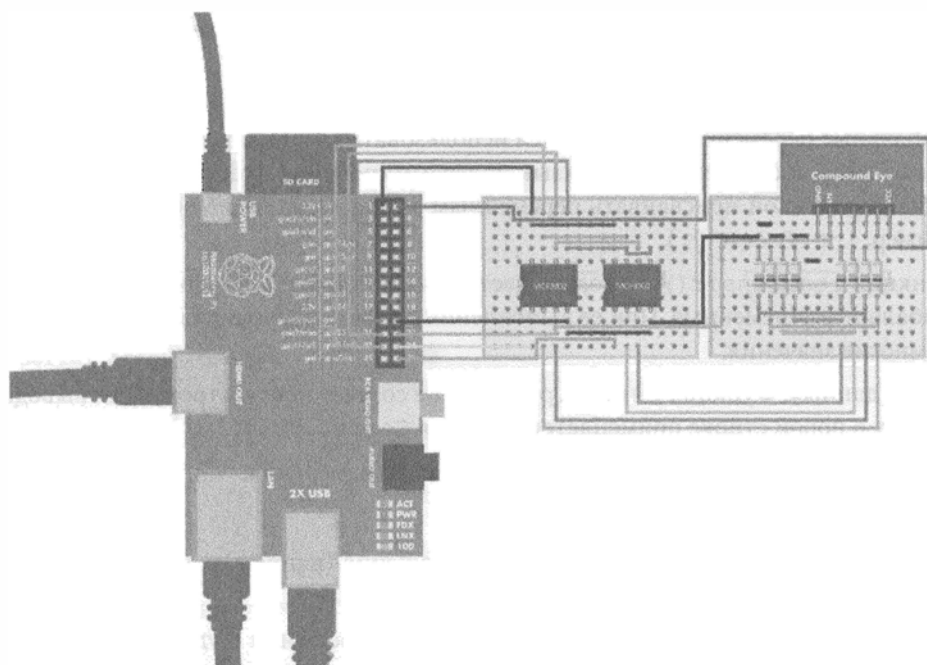


Рис. 3.17. Схема подключения составного датчика инфракрасного излучения к Raspberry Pi (см. цветную вклейку)

Листинг 3.8. `compound_eye.py`

```
# compound_eye.py - считывание расстояния и направления
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import botbook_gpio as gpio # ❶
import botbook_mcp3002 as mcp # ❷

irUpValue = 0 # ❸
irDownValue = 0
irLeftValue = 0
irRightValue = 0
distance = 0

def readCompoundEye():
    global irUpValue, irDownValue, irLeftValue, irRightValue, distance # ❶
    ledPin = 25
    gpio.mode(ledPin, "out") # ❹
    gpio.write(ledPin, gpio.HIGH)
```

```

# Ожидание готовности датчиков
time.sleep(0.05) # ⑤

irUpValue = mcp.readAnalog(0, 0) # ⑦
irDownValue = mcp.readAnalog(1, 0)
irLeftValue = mcp.readAnalog(0, 1)
irRightValue = mcp.readAnalog(1, 1)

ambientLight = 0
gpio.write(ledPin, gpio.LOW) # ⑥
time.sleep(0.05)
ambientLight = mcp.readAnalog(0, 0) # ⑧
irUpValue = irUpValue - ambientLight # ⑩
ambientLight = mcp.readAnalog(1, 0) # ⑩
irDownValue = irDownValue - ambientLight
ambientLight = mcp.readAnalog(0, 1)
irLeftValue = irLeftValue - ambientLight
ambientLight = mcp.readAnalog(1, 1)
irRightValue = irRightValue - ambientLight

distance = (irUpValue+irDownValue+irLeftValue+irRightValue)/4 # ⑪

def main():
    global irUpValue,irDownValue,irLeftValue,irRightValue,distance
    while True: # ⑪
        readCompoundEye() # ⑪
        print "Значения:"
        print "верх: %f" % irUpValue
        print "вниз: %f" % irDownValue
        print "влево: %f" % irLeftValue
        print "вправо: %f" % irRightValue
        print "расстояние: %f" % distance
        time.sleep(0.5) # с # ⑪

if __name__ == "__main__":
    main()

```

- ① Импорт библиотеки `botbook_gpio` для получения возможности включения и выключения цифровых выводов (в нашем случае — `gpio25`). Файл `botbook_gpio.py` должен располагаться в том же каталоге, что и текущая программа (`compound_eye.py`).
- ② Импорт библиотеки `botbook_mcp3002`, обеспечивающей считывание сигналов с аналоговых датчиков с помощью аналого-цифровых преобразователей MCP3002. Она нужна для получения значений от каждого ИК-фоторезистора. Библиотека `botbook_mcp3002.py` должна располагаться в том же каталоге, что и файл `compound_eye.py`. Кроме того, вам также необходимо установить библиотеку `spidev`, которая используется в `botbook_mcp3002.py`. Подробнее об этом рассказывается в комментариях к программным кодам файла `botbook_mcp3002.py` и в разделе “Подключение библиотеки `Spidev`”.
- ③ Объявление глобальных переменных.

- ④ Чтобы иметь возможность использовать глобальные переменные в функции, их нужно объявить в начале программного кода этой функции.
- ⑤ Включение вывода `gpio25`, подключенного к ИК-светодиодам. С его помощью освещается целевая область измерений.
- ⑥ Ожидание установки состояния контакта.
- ⑦ Считывание данных с каждого ИК-фоторезистора. Поскольку в Raspberry Pi не встроено аналого-цифровой преобразователь, применяется внешний модуль MCP3002.
- ⑧ Чтобы исключить влияние естественного освещения, ИК-подсветка выключается.
- ⑨ ⑪ Данные с каждого ИК-фоторезистора считываются повторно.
- ⑫ Естественное освещение вычитается из первичных результатов (полученных с включенной ИК-подсветкой).
- ⑬ Расстояние определяется как усредненный результат для четырех ИК-фоторезисторов.
- ⑭ Во встраиваемых решениях для бесконечного выполнения одних и тех же операций обычно применяется циклическая структура `while (True)`. Многие подключаемые устройства рассчитаны на непрерывное выполнение действий, не предполагая прекращения выполнения программы или отключения питания. Чтобы прервать выполнение программы в цикле `while (True)`, нажмите комбинацию клавиш `<Ctrl+C>` в том сеансе работы с терминалом, из которого она запускалась.
- ⑮ Функция `readSensor()` не возвращает значений, а изменяет значения глобальных переменных. В Python множественные значения допускается возвращать с помощью конструкции `a, b, c = foo()`.
- ⑯ Как обычно, при использовании в программном коде циклических структур необходимо установить небольшую задержку, чтобы предотвратить постоянную полную загрузку микропроцессора.

Составной датчик инфракрасного излучения оснащен относительно небольшим количеством выводов. Но после включения в конечную цепь аналого-цифровых преобразователей она становится весьма запутанной из-за большого количества проводочных перемычек.

Подключение библиотеки SpiDev

Аналого-цифровой преобразователь MCP3002 подключается к микроконтроллеру через интерфейс SPI. Это достаточно сложный интерфейс для ручного управления, поэтому для его использования в своих собственных проектах загрузите библиотеку SpiDev.

Библиотека SpiDev подключается во всех программах, в которых встречается оператор `import spidev`. Среди них программный код управления потенциометром в Raspberry Pi и любым другим аналоговым резистивным датчиком, описанным в данной книге, поскольку SpiDev задействуется в файле библиотеки `botbook_mcp3002`.

В операционной системе Raspberry Pi запустите терминал. Для начала задайте начальные настройки среды.

```
$ sudo apt-get update
$ sudo apt-get -y install git python-dev
```

Загрузите последнюю версию библиотеки SpiDev с соответствующего сайта поддержки.

```
$ git clone https://github.com/doceme/py-spidev.git
$ cd py-spidev/
```

Установите библиотеку в системе следующей командой:

```
$ sudo python setup.py install
```

Далее нужно подключить интерфейс SPI в Raspberry Pi. Вначале убедитесь, что он не отключен. Откройте файл `/etc/modprobe.d/raspi-blacklist.conf` для внесения в него изменений, воспользовавшись командой `sudoedit /etc/modprobe.d/raspi-blacklist.conf`. Удалите в файле такую строку:

```
blacklist spi-bcm2708
```

Сохраните изменения в файле: нажмите комбинацию клавиш `<Ctrl+X>`, а затем — клавишу `<Y>` и клавишу `<Enter>` или `<Return>`.

Чтобы получить доступ к интерфейсу SPI без привилегий суперпользователя, скопируйте файл правил из листинга 3.9 или соответствующий файл из каталога с файлом `botbook_mcp3002`, как показано ниже:

```
$ sudo cp 99-spi.rules /etc/udev/rules.d/99-spi.rules
```

Листинг 3.9. 99-spi.rules

```
# /etc/udev/rules.d/99-spi.rules - доступ к SPI в Raspberry Pi
# Copyright 2013 http://BotBook.com
```

```
SUBSYSTEM=="spidev", MODE="0666"
```

Перезагрузите Raspberry Pi, запустите LXTerminal, разрешите доступ к SPI-устройствам и подтвердите владельца файла:

```
$ ls -l /dev/spi*
```

В списке должны указываться два файла с атрибутами `crw-rw-rwT`. Если это не так, то повторите описанные выше действия еще раз.

Теперь вы сможете использовать микросхему MCP3002 и другие устройства, подключаемые через интерфейс SPI к Raspberry Pi.

Другие варианты подключения датчика инфракрасного излучения к Raspberry Pi

Конечная цепь подключения составного датчика к Raspberry Pi выглядит весьма запутанно. Несмотря на то что разобраться в ней несложно, при подключении всех ее компонентов применяется огромное количество перемычек. Чтобы упростить конструкцию, попробуйте использовать другой аналого-цифровой преобразователь или же Arduino (как описано во врезке “Raspberry Pi + Arduino”).

Все аналого-цифровые преобразования можно выполнить с помощью всего одной микросхемы — MCP3008, оснащенной восьмью входами. Но ее применение требует внесения изменений в библиотеку `botbook_mcp3002`. К счастью, соответствующий код подключения MCP3008 уже создан разработчиками Adafruit и доступен для загрузки по следующему адресу:

<https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code>

Raspberry Pi + Arduino

Если вы находите количество переключателей в текущем примере чрезмерно большим, то рассмотрите компромиссный вариант: подключение составного датчика к Arduino с отправкой результатов в Raspberry Pi через последовательное соединение USB. Еще один вариант — это использование

AlaMode для Raspberry Pi — Arduino-совместимой платы, которая монтируется непосредственно поверх платы Raspberry Pi, создавая функциональный гибрид обеих платформ.

Детально о взаимодействии обеих платформ рассказывается в главе 12.

Пилотный проект: контроль осанки (Arduino)

Каждый пользователь знаком с такой проблемой: чем дольше работаешь за компьютером, тем ближе голова наклоняется к экрану. Длительное сидение на одном месте, особенно перед монитором, не предусмотрено физиологией человеческого тела, а потому вызывает изменение осанки и тяжелые заболевания позвоночника. Совместив в одном приборе датчик ИК-излучения и пьезоэлектрический зуммер, вы получите простое устройство предупреждения ситуаций чрезмерного приближения головы к экрану монитора (рис. 3.18).

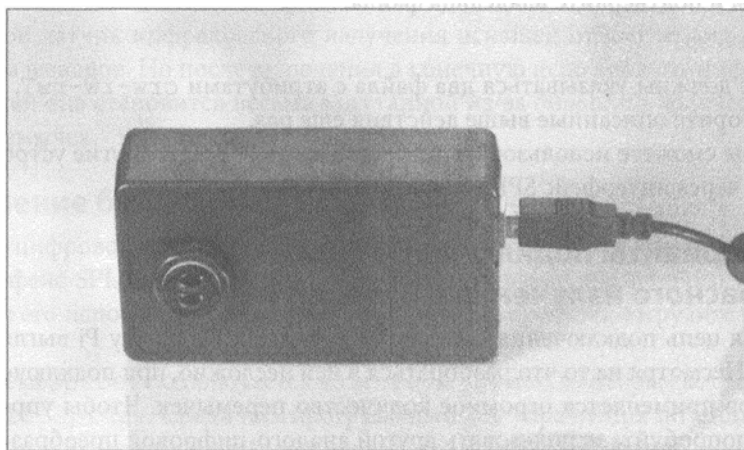


Рис. 3.18. Готовый прибор контроля осанки

Получаемые навыки

Реализуя проект прибора контроля осанки, вы научитесь:

- проводить полный цикл управления данными: ввод, обработка, вывод;
- воспроизводить звуки пьезоэлектрическим зуммером;
- создавать корпус для прибора.

Пьезоэлектрический зуммер

Особенность пьезоэлектрического кристалла заключается в изменении геометрической формы при подаче на него напряжения. Подавая на пьезоэлектрический кристалл постоянный ток или простой прямоугольный сигнал, вы заставите его вибрировать (колебаться). Колебания будут передаваться воздуху, а колебания воздуха — это не что иное, как звук. В данном случае — противный, раздражающий звук.

Большая часть пьезоэлектриков при подаче на них прямоугольного сигнала издает зуммер (рис. 3.19). Прямоугольный сигнал представляет собой последовательное чередование сигналов высокого (5 В в Arduino) и низкого (0 В) уровней.

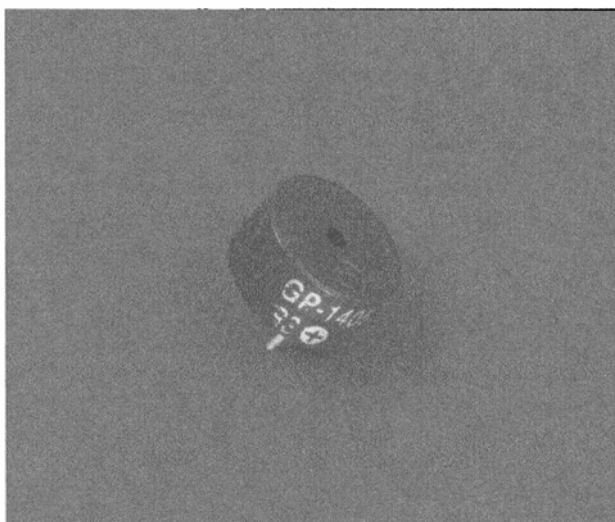


Рис. 3.19. Пьезоэлектрический зуммер

Вы легко создадите прямоугольный сигнал, если будете последовательно включать и выключать вывод с помощью функции `digitalWrite()`. В качестве другого варианта можете попробовать воспользоваться встроенной функцией `tone()`, которая создает сигнал необходимой формы более сложным и совершенным способом.

Пьезоэлектрический эффект обратим: вы всегда можете сжимать и разжимать кристалл для получения электрического сигнала. Подобное поведение пьезоэлектриков делает их идеальным материалом при производстве зажигалок.

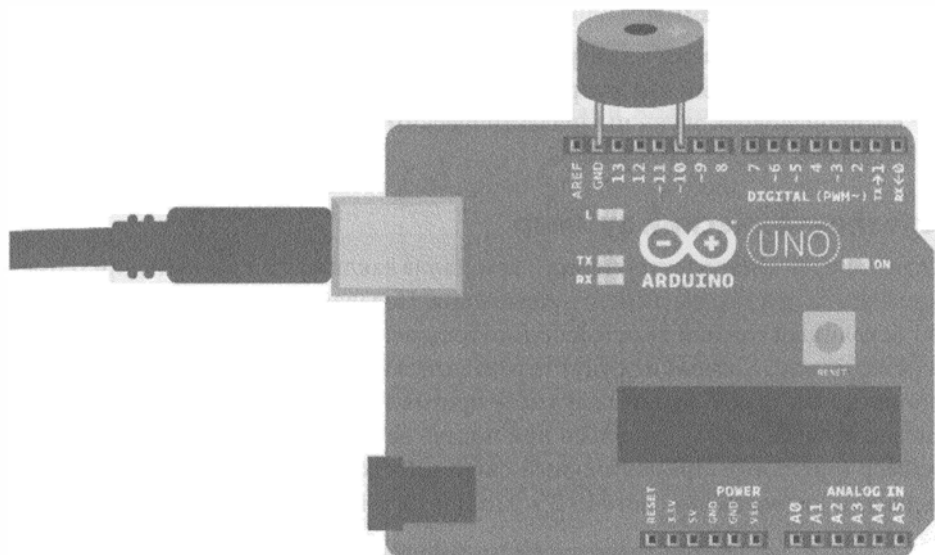


Рис. 3.20. Подключение зуммера к Arduino

Листинг 3.10. piezo_beep.ino

// piezo_beep.ino - в оспроизв едениезуммера требу емойчастоты
 // (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
int speakerPin = 10;

void wave(int pin, float frequency, int duration) // ❶
{
    float period=1/frequency*1000*1000; // мкс // ❷
    long int startTime=millis(); // ❸
    while(millis()-startTime < duration) { // ❹
        digitalWrite(pin, HIGH); // ❺
        delayMicroseconds(period/2);
        digitalWrite(pin, LOW);
        delayMicroseconds(period/2);
    }
}

void setup()
{
    pinMode(speakerPin, OUTPUT);
}

void loop()
{
    wave(speakerPin, 440, 500); // ❻
    delay(500);
}
```

- ❶ Функция `wave` принимает в качестве параметров переменные вывода, частоты звучания и длительность звука в миллисекундах. Выполняемые в функции операции не представляют большого интереса: они познавательные, но не основополагающие для проекта.
- ❷ Вычисление периода сигнала `T`: времени, за которое одна волна совершает полное колебание (сигнал высокого уровня продолжается сигналом низкого уровня). Это обратная величина к частоте `f`: $T = 1/f$. Например, если за секунду волна совершает 2 полных колебания (частота 2 Гц), то одно колебание происходит в течение 1/2 секунды. Обратите внимание на единицы измерения частоты (количество циклов в секунду) колебания; герц или Гц.
- ❸ Стандартный прием в программном коде, применяемый при отсчете временного интервала: начальное время сохраняется в переменной (`millis` представляет количество миллисекунд с момента подачи питания на плату Arduino)...
- ❹ ...а действия выполняются до момента, когда временной интервал становится больше заранее определенной величины, отсчитанной от начального времени.
- ❺ Создание одного полного колебания. Сначала создается сигнал высокого уровня, а затем низкого.
- ❻ Вызов функции, управляющей воспроизведением звукового сигнала.

Сирена

Настало время перейти от воспроизведения монотонного звука к более сложным звуковым композициям, а именно: к сигналу сирены.

Листинг 3.11. `piezo_alarmtone.ino`

```
// piezo_alarmtone.ino - воспроизведение сигнала сирены
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
int speakerPin = 10;

void wave(int pin, float frequency, int duration) // ❶
{
    float period=1 / frequency * 1000 * 1000; // мкс
    long int startTime=millis();
    while(millis()-startTime < duration) {
        digitalWrite(pin, HIGH);
        delayMicroseconds(period/2);
        digitalWrite(pin, LOW);
        delayMicroseconds(period/2);
    }
}

void setup()
{
    pinMode(speakerPin, OUTPUT);
}

void loop()
```

```

{
  wave(speakerPin, 440, 40); // ②
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
  delay(25);
  wave(speakerPin, 440, 20);
  wave(speakerPin, 640, 40);
  delay(25);
  wave(speakerPin, 540, 20);
}

```

- ① Это та же функция `wave()`, что использовалась при генерировании монотонного звука.
- ② Воспроизведение ноты, ожидание в течение непродолжительного времени, воспроизведение еще одной ноты и т.д.

Сочетание зуммера с инфракрасным датчиком

Создайте цепь для прибора контроля осанки, воспользовавшись схемой, показанной на рис. 3.21. Загрузите в Arduino программный код из листинга 3.12 и приготовьтесь зажить новой жизнью здорового человека.

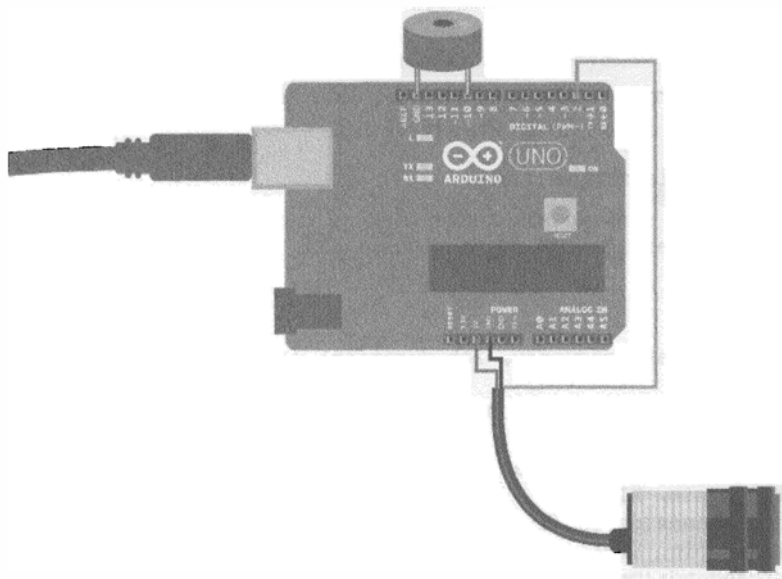


Рис. 3.21. Конструкция прибора контроля осанки предельно проста

Листинг 3.12. `posture_alarm.ino`

```

// posture_alarm.ino - звучание сирены при изменении осанки
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int speakerPin = 10;

```

```

const int sensorPin = 2;
int switchState = 0;

void wave(int pin, float frequency, int duration) // ❶
{
    float period=1/frequency*1000*1000; // мкс
    long int startTime=millis();
    while(millis()-startTime < duration) {
        digitalWrite(pin, HIGH);
        delayMicroseconds(period/2);
        digitalWrite(pin, LOW);
        delayMicroseconds(period/2);
    }
}

void alarm() // ❷
{
    wave(speakerPin, 440, 40);
    delay(25);
    wave(speakerPin, 300, 20);
    wave(speakerPin, 540, 40);
    delay(25);
    wave(speakerPin, 440, 20);
    wave(speakerPin, 640, 40);
    delay(25);
    wave(speakerPin, 540, 20);
}

void setup()
{
    pinMode(speakerPin, OUTPUT);
    Serial.begin(115200);
    pinMode(sensorPin, INPUT);
}

void loop()
{
    switchState = digitalRead(sensorPin);
    Serial.println(switchState,BIN);
    if (switchState==0) { // ❸
        alarm(); // ❹
    }
    delay(10);
}

```

- ❶ Используется такая же функция `wave()`, как и в предыдущих проектах.
- ❷ Применяется функция `alarm()`, аналогичная рассматриваемой ранее.
- ❸ Функция `digitalRead()` получает значение от ИК-датчика, определяющего расстояние, таким же способом, как и ранее.
- ❹ При обнаружении объектов перед монитором звучит сигнал сирены. Буквально это означает следующее: как только ваша голова приблизится к экрану монитора, послышится звук сирены.

Заключение сигнализации в корпус

Создаваемые вами прототипы устройств будут служить дольше и выглядеть лучше, если заключить их в прочный и красивый корпус. Для нашего текущего проекта Arduino используется корпус от компании SmartProjects, совпадающий по размерам с габаритами собранного ранее прототипа. Вначале разместите все компоненты собранного устройства в корпус без закрепления, удостоверившись, что они туда помещаются, и прорежьте резцом диаметром 19 мм отверстие для вывода датчика ИК-излучения (рис. 3.22).

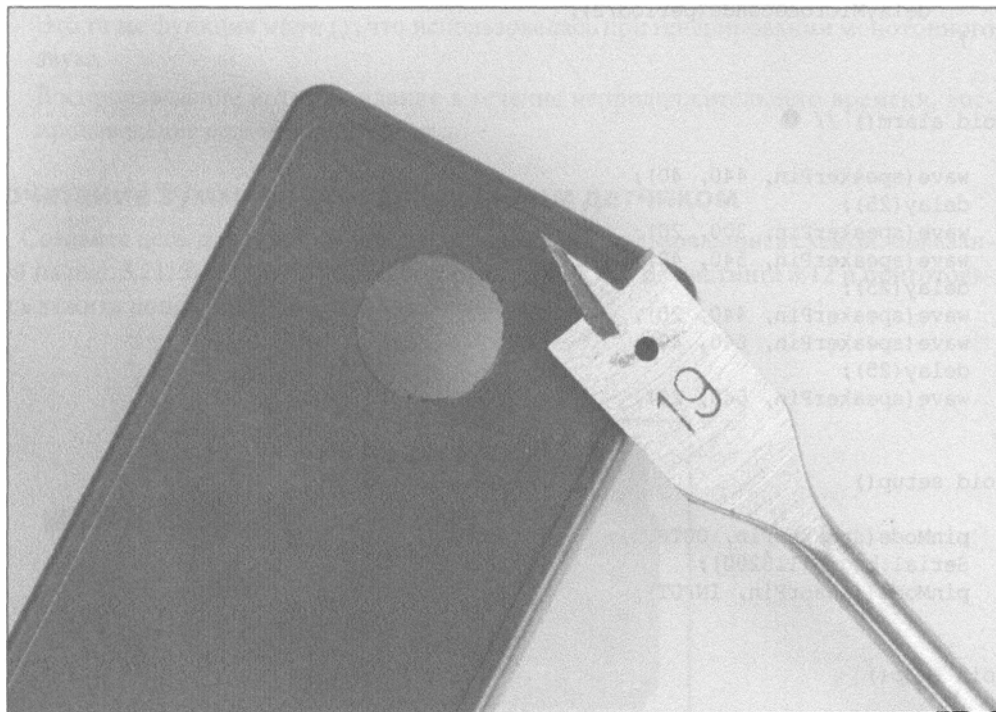


Рис. 3.22. Отверстие для монтирования датчика инфракрасного излучения

В задней части корпуса, как раз под отверстием, мы вырезали специальный лючок, чтобы упростить монтаж датчика и получить доступ к его регулятору чувствительности (рис. 3.23). Сам датчик фиксируется внутри корпуса специальной пластиковой гайкой, входящей в комплект поставки корпуса. Плата Arduino надежно фиксируется защелками, оставаясь неподвижной даже при подключении к ее портам кабелей (рис. 3.24).

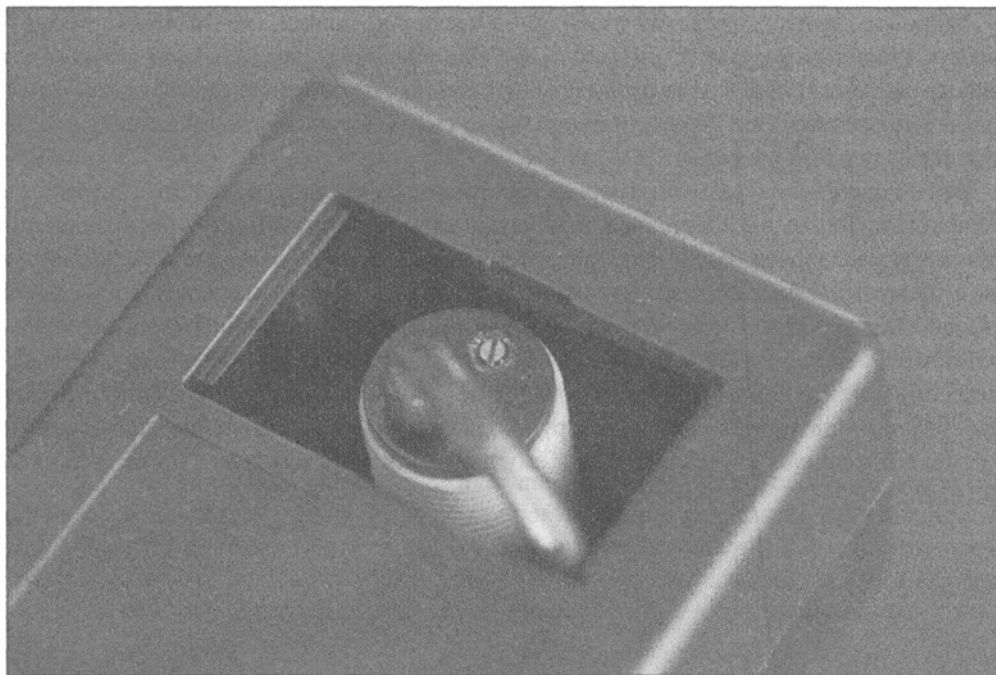


Рис. 3.23. С обратной стороны нужно оставить отверстие для доступа к регулятору датчика

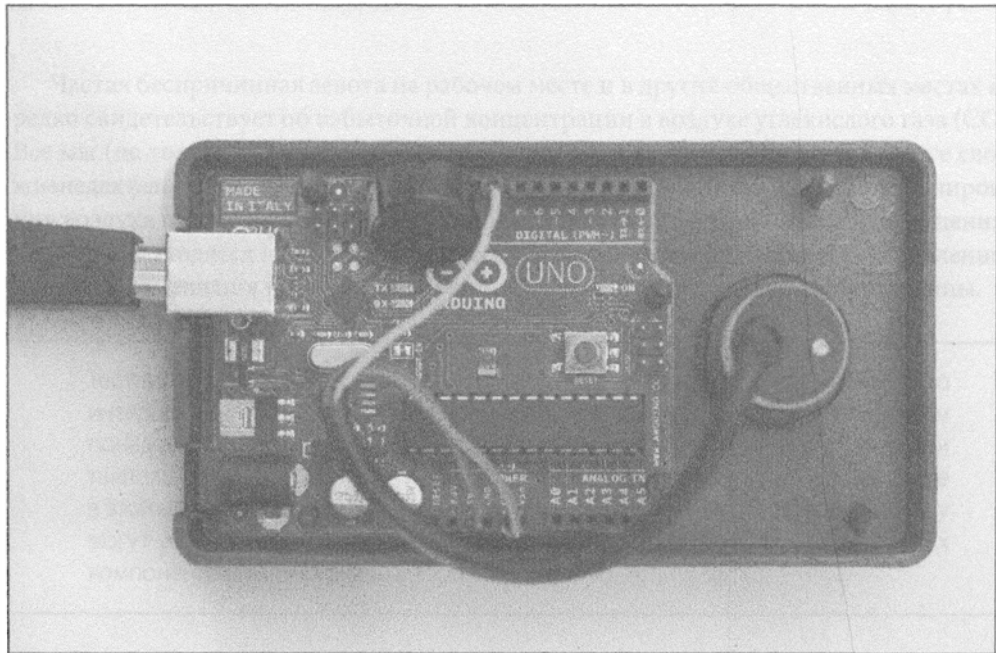


Рис. 3.24. Все компоненты компактно размещаются в корпусе

Закройте корпус, совместив нижнюю ее часть с крышкой, и внимательно осмотрите полученный результат (см. рис. 3.18). Изначально корпус снабжен лючком для вывода разъема USB, и вам не придется специально беспокоиться еще об одном функциональном отверстии. Теперь прибор можно смело использовать по назначению, не опасаясь случайно повредить.

В этой главе вы научились измерять расстояния с помощью различных датчиков и технологий. Теперь вы знаете, как включать в свои проекты средства распознавания объектов, расположенных в непосредственной близости и на отдалении. Добавив в свои прототипы более одного датчика, вы можете запрограммировать их на более сложное поведение. Например, с использованием двух датчиков инфракрасного излучения и электродвигателя можно создать прототип робота,двигающегося к ближайшему объекту. Подобным образом, но уже с помощью сервопривода, конструируется простейший повторитель движений руки. Два датчика ИК-излучения, добавленные к самодвижущемуся роботу, позволят ему двигаться за источником света. А какие у вас идеи?

Громкий вой пожарной сирены врывается в каждый дом и не оставляет никого равнодушным! Он прорывается даже через закрытые окна, вызывает неприкрытый интерес зевак, будит соседей и приводит в чувство нерадивых заложников пожара, вселяя в души надежду и заставляя активнее прорываться к аварийным выходам, подальше от клубов дыма и угарного газа, грозящего смертельным удушьем.

Сирена полицейского автомобиля заставляет прижаться к обочине и заглушить двигатель. После короткого знакомства с документами вам предлагают пройти уникальную, но крайне важную процедуру проверки выдыхаемого воздуха на наличие алкоголя. Маленький газовый анализатор, скрывающийся внутри, казалось бы, нехитрого прибора, спасает здоровье и жизни огромному количеству людей по всему миру.

Частая беспричинная зевота на рабочем месте и в других общественных местах нередко свидетельствует об избыточной концентрации в воздухе углекислого газа (CO_2). Все мы (не только люди, но и животные) выдыхаем углекислый газ в процессе своей жизнедеятельности. Датчики, встроенные в системы вентиляции и кондиционирования воздуха городских домов, постоянно следят за концентрацией CO_2 в помещениях, в которых находятся люди. Как только его уровень превышает заранее установленный предел, в помещения начинает нагнетаться свежий воздух, поступающий с улицы.

Тестировать датчики газа и дыма на примере реальных прототипов чрезвычайно интересно. Чтобы поставить их использование на промышленную основу, вам понадобится выполнить большое количество жестких требований, провести тщательные испытания и сертифицировать конечный продукт. Рассматриваемые в этой главе проекты не претендуют на роль коммерческих продуктов, а потому могут использоваться только для знакомства с возможностями электронных компонентов газовых анализаторов.

Пожарные службы обязывают владельцев офисов и других общественных зданий обвешивать свои помещения целыми гирляндами датчиков, определяющих наличие соединений углеводородов (природного газа), взрывающихся при определенной

концентрации в воздухе. Анализаторами газа оснащаются все современные автомобили; с их помощью определяется степень насыщения горючей смеси, подаваемой в камеры сгорания двигателя, и не только. Правильное соотношение смеси зависит от многих факторов, поэтому требует постоянной корректировки в процессе работы двигателя. Более того, неправильно подобранное соотношение топлива и воздуха приводит к заглошению двигателя или же чрезмерному расходу бензина и загрязнению окружающей среды.

Газовые датчики серии MQ — это недорогие устройства, предназначенные для выявления в воздухе широкого спектра газов, встречающихся в бытовых условиях. Некоторые из датчиков этой серии приведены в табл. 4.1.

Таблица 4.1. Некоторые газовые анализаторы серии MQ

Датчик серии MQ	Регистрируемые соединения
MQ-2	Дым и другие газы, возникающие в процессе горения
MQ-3, MQ-303A	Пары алкоголя (этиловый спирт)
MQ-4	Метан (CH_4)
MQ-7	Угарный газ (CO)
MQ-8	Водород
MQ-9	Угарный газ, метан, природный газ (пропан или бутан)

Эксперимент: выявление дыма (аналоговый газовый датчик)

Датчик дыма MQ-2 определяет наличие дыма в воздухе и извещает об этом увеличением напряжения на выходе. Чем больше дыма в воздухе, тем выше напряжение на выходе. В используемый нами датчик MQ-2 встроен специальный потенциометр, позволяющий корректировать чувствительность анализатора (рис. 4.1).

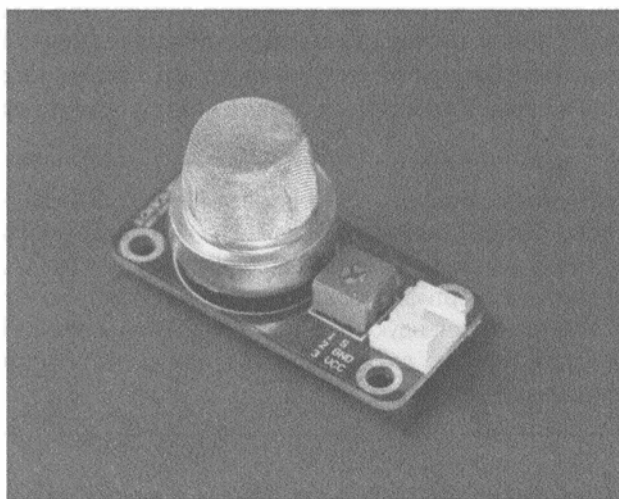


Рис. 4.1. Аналоговый датчик дыма

Какой из газов самый ядовитый, спросите вы? Это зависит от системы подсчета. Если по количеству жертв, то несомненный лидер — угарный газ (CO). При пожаре большая часть пострадавших гибнет от удушья дымом, а не сгорает в огне, как может показаться на первый взгляд.

Угарный газ, или CO, несмотря на схожесть по названию с менее вредным углекислым газом (CO₂), очень токсичен для человеческого организма. А все потому, что наш организм умеет выводить углекислый газ наружу, ведь он естественным образом образуется внутри нас при высвобождении в тканях энергии и выводится с кровью через легкие, высвобождаясь при выдохе. В случае угарного газа ситуация не столь оптимистичная. Токсичность CO во многом определяется отсутствием естественных механизмов вывода этого газа из организма. Он прикрепляется к красным кровяным тельцам, блокируя в крови функцию переноса кислорода и углекислого газа на несколько часов. Как только весь гемоглобин в крови вместо кислорода и углекислого газа начинает перемещать CO, в организме наступает кислородное голодание, неизбежно приводящее к летальному исходу.

Так как датчик MQ-2 оснащается тремя выводами, при установке в электрические цепи он не нуждается в дополнительных подтягивающих и согласующих резисторах. Arduino рассматривает датчик MQ-2 как потенциометр с тремя выводами.

На два из трех выводов датчика MQ-2 подается питание: напряжение +5 В и 0 В (земля). Напряжение на третьем контакте датчика (вывод S) изменяется в зависимости от концентрации дыма (ближе к +5 В).

Подключение к Arduino и программа управления датчиком MQ-2

Плата Arduino оснащена встроенным аналого-цифровым преобразователем, поэтому показания датчика MQ-2 считываются функцией `analogRead()`. Возвращаемое ею значение непосредственно указывает на концентрацию дыма возле датчика. Для настройки чувствительности датчика используется потенциометр, расположенный на плате датчика (рис. 4.2).

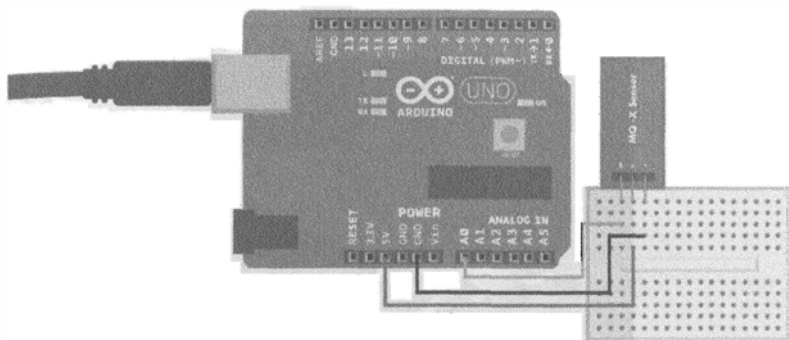


Рис. 4.2. Схема подключения датчика MQ-2 к Arduino

Листинг 4.1. `mq_x_smoke_sensor.ino`

```
// mq_x_smoke_sensor.ino - определение концентрации дыма
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = A0;
int smoke_level = -1; // ❶

void setup() {
    Serial.begin(115200); // бит/с
    pinMode(sensorPin, INPUT);
}

void loop() {
    smoke_level = analogRead(sensorPin); // ❷
    Serial.println(smoke_level);
    if(smoke_level > 120) { // ❸
        Serial.println("Утечка газа!");
    }
    delay(100); // мс
}
```

- ❶ Установка уровня дыма в невозможное значение, чтобы облегчить дальнейшую отладку. Если на монитор выводится `-1`, то вы будете знать, что датчик нерабочий, поскольку это значение в случае работающего датчика должно заменяться значением, возвращаемым функцией `analogRead()`.
- ❷ MQ-2 — это обычный аналоговый резистивный датчик, поэтому выводимое им значение определяется функцией `analogRead()`. Она возвращает значение в диапазоне от 0 (0 В) до 1023 (+5 В).
- ❸ Несмотря на установку граничного значения в коде, для настройки оптимального рабочего диапазона датчика лучше всего использовать потенциометр на плате.

Пожарная сигнализация не обязательно должна выглядеть как безликая коробка неприглядной расцветки. Это доказывает Паоло Сухонен своим творением, показанным на рис. 4.3. Корпус в виде мотылька делает привычную вещь необычной и даже непривычно броской. Помните об этом, разрабатывая дизайн для собственных изделий.

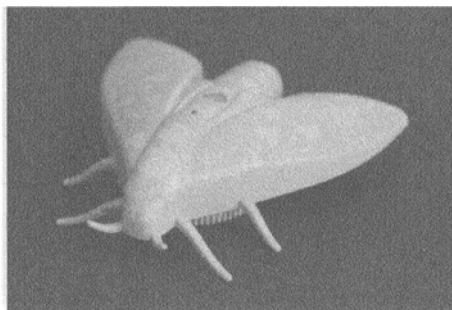


Рис. 4.3. Необычный дизайн сигнализации, разработанной Паолой Сухонен

Подключение к Raspberry Pi и программа управления датчиком MQ-2

Для подключения датчика MQ-2 к Raspberry Pi вам понадобится отдельный аналого-цифровой преобразователь (АЦП). Как и в случае других аналоговых резистивных датчиков (см. описание составного датчика ИК-излучения в главе 3), для выполнения необходимых преобразований мы будем использовать микросхему MCP3002.

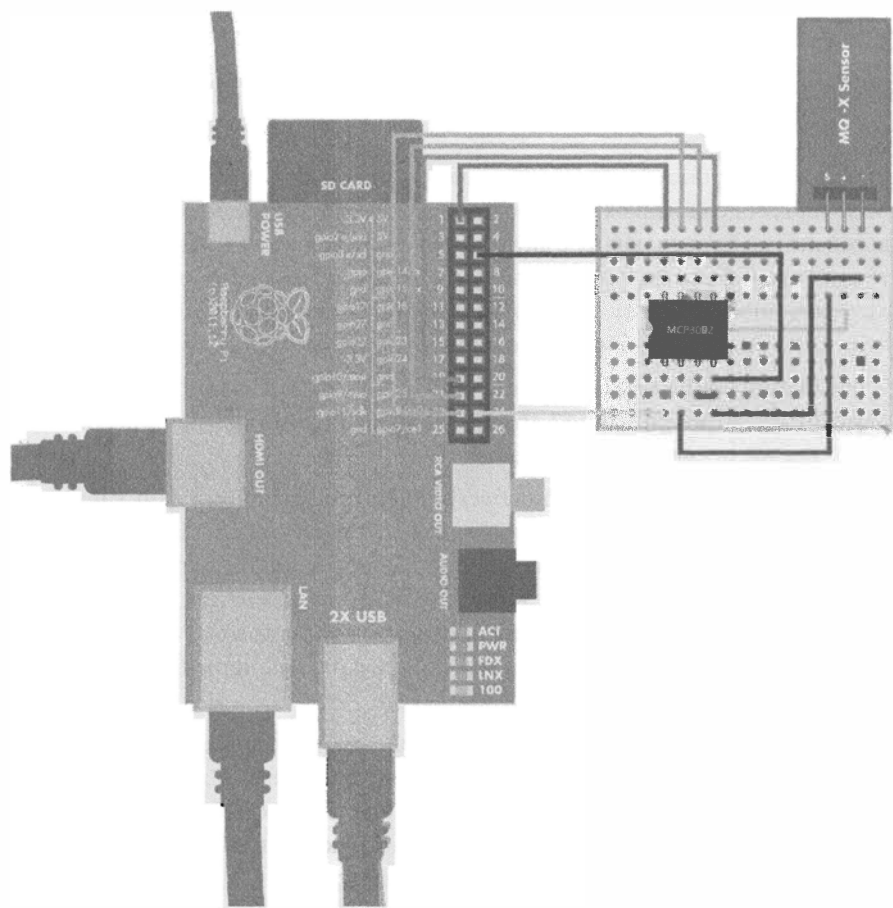


Рис. 4.4. Схема подключения датчика MQ-2 к Raspberry Pi (см. цветную вклейку)

Листинг 4.2. `mq_x_smoke_sensor.py`

```
# mq_x_smoke_sensor.py - определение концентрации дыма
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp # ❶

smokeLevel = 0
```

```
def readSmokeLevel():
    global smokeLevel
    smokeLevel = mcp.readAnalog() # ❷

def main():
    while True: # ❶
        readSmokeLevel() # ❸
        print("Концентрация газа: %i " % smokeLevel) # ❹
        if smokeLevel > 120:
            print("Утечка газа!")
            time.sleep(0.5) # с

if __name__ == "__main__":
    main()
```

- ❶ Библиотека `botbook_mcp3002`, подключаемая для управления микросхемой МСР3002, содержит важный программный код и упрощает обработку аналоговых данных, поступающих с датчика. Ее файл должен размещаться в том же каталоге, что и файл `mcp_x_smoke_sensor.py`. Кроме нее вам также нужно подключить библиотеку `spidev`, которая запрашивается библиотекой `botbook_mcp3002`. Детально об использовании этих библиотек см. в главе 3.
- ❷ Считывание напряжения первого устройства, подключенного к МСР3002. Параметры функции `readAnalog()`, установленные по умолчанию — `device=0`, `channel=0`, оставлены без изменений.
- ❸ Во встраиваемых решениях для бесконечного выполнения одних и тех же операций обычно применяется циклическая структура `while(True)`. Многие подключаемые устройства рассчитаны на непрерывное выполнение действий, не предполагая прекращения выполнения программы или отключения питания. Чтобы прервать выполнение программы в цикле `while(True)`, нажмите комбинацию клавиш `<Ctrl+C>` в том сеансе работы с терминалом, из которого она запускалась.
- ❹ Хорошим тоном считается выделять решение основных задач проекта в отдельную функцию. Назначение функции `readSmokeLevel()` однозначно указывается ее названием: это очень удобно в сложных проектах, в которых применяются длинные программные конструкции и анализируется большой массив данных, поступающих с разных датчиков.
- ❺ Создание выводимого сообщения с помощью операторов форматирующей строки. Значение переменной `smokeLevel` имеет целочисленный тип данных и подставляется в форматирующей строке вместо оператора `%i`.

Эксперимент в окружающей среде: задымление помещения

В неэстетичном монтаже датчиков пожарной сигнализации к потолку есть своя веселая причина. Дым обычно является следствием возгорания, а открытый огонь нагревает окружающий его воздух, который вместе с дымом поднимается вверх, к потолку.

Вы спросите: "Почему горячий воздух легче холодного?" Все дело в подвижности молекул. Молекулы воздуха постоянно двигаются и сталкиваются друг с другом. Чем теплее воздух, тем большее количество молекул начинает сталкиваться. Частое и сильное взаимодействие молекул приводит к их расталкиванию на большие расстояния. Чем дальше молекулы находятся друг от друга, тем более разреженным становится воздух. Поскольку вокруг области разрежения (высокой температуры) воздух остается прежней высокой плотности (низкой температуры), то он выталкивает более легкий воздух вверх.

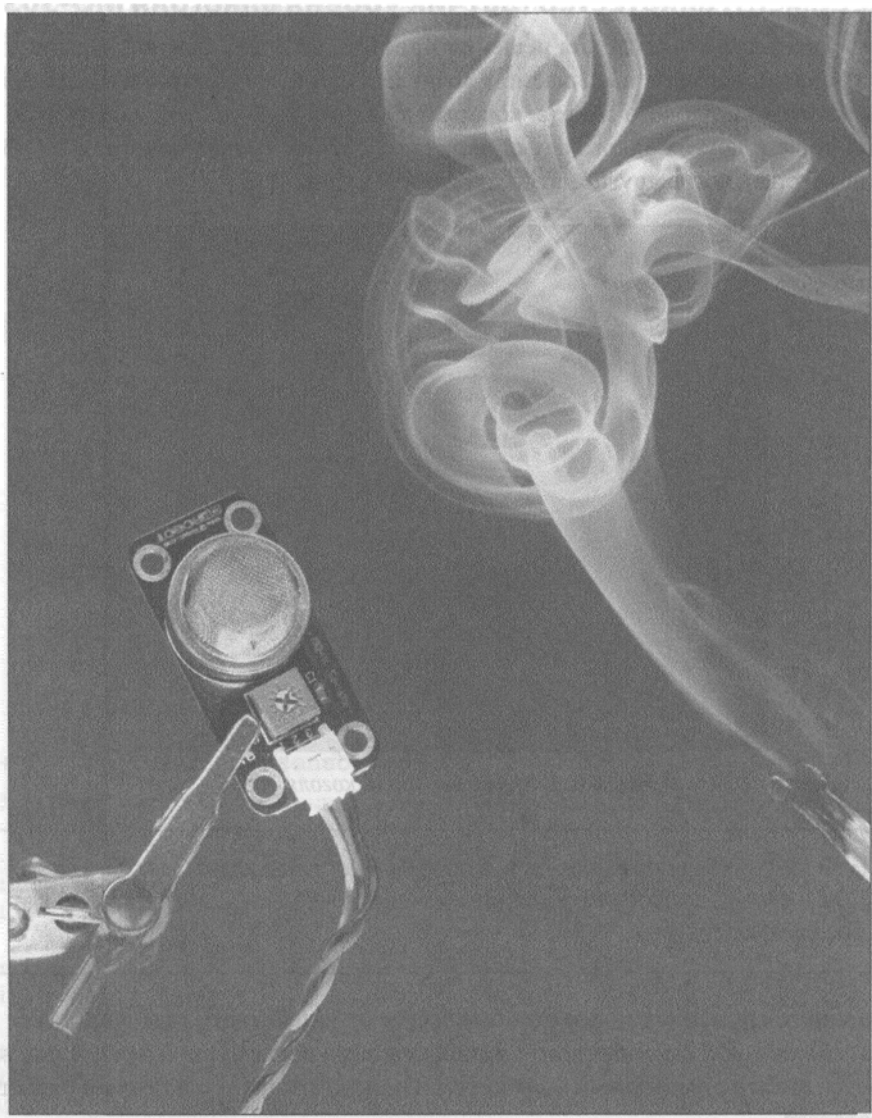


Рис. 4.5. Дым поднимается вверх

Проведите эксперимент: выполните программный код управления датчиком дыма, поднеся к нему гаснущую спичку. Насколько высоко вы можете поднять над датчиком тухнущую спичку и все еще регистрировать дым с помощью датчика? Скорее всего, дымящая спичка над датчиком никак не будет влиять на считываемые показания, — они будут такими же, как и для обычного воздуха. Только расположив тухнущую спичку под датчиком, вы заметите резкое изменение сигнала на его выходе. Этот эксперимент доказывает простую истину: дым поднимается вверх. Оказывается, важно не только подобрать нужный датчик, но и правильно его расположить в помещении, чтобы в случае пожара сигнализация сработала своевременно.

Эксперимент: алкотестер (датчик уровня алкоголя MQ-303A)

Алкотестер применяется для решения простой, но очень важной задачи: определения уровня алкоголя в выдыхаемом воздухе. Если быть точным, то определяются пары этанола — этилового спирта, который в разных количествах содержится в водке, вине, пиве и других алкогольных напитках. Датчик уровня алкоголя MQ-3 визуально похож на многие другие датчики серии MQ (рис. 4.6).

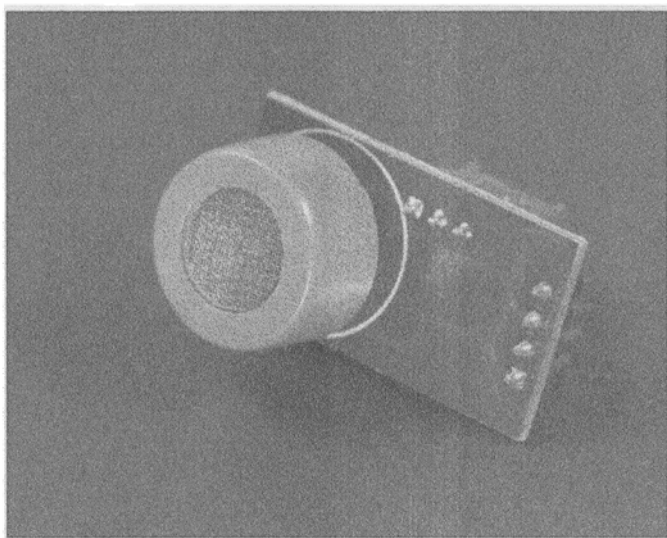


Рис. 4.6. Датчик уровня алкоголя MQ-3

Если вы привыкли праздновать все победы и достижения с бокалом пива или вина в руке, то попробуйте изменить жизненную концепцию, не пропустив следующий раздел главы.

Подобно тому, как организм освобождается от углекислого газа в процессе дыхания, часть алкоголя, накопленного в крови человека после бурной вечеринки, высвобождается вместе с выдыхаемым воздухом. Именно эти пары алкоголя и регистрирует алкотестер.

Чем больше этилового спирта в крови, тем выше его концентрация в выдыхаемом воздухе. Таким образом, алкотестер с большой долей вероятности определяет правильное значение уровня алкоголя в крови человека.

Стоит заметить, что алкоголь действует по-разному на людей. Справедливо утверждение: чем больше выпьешь, тем сильнее захмелеешь. Вот только количество выпитого алкоголя для достижения одинаковой степени опьянения у всех людей разное. В законе, запрещающем вождение в пьяном виде, прописываются некие средние ограничения. Например, в Финляндии разрешается вождение автомобиля с уровнем алкоголя в крови, не превышающим значение 0,5%.

Официально разрешенные алкотестеры должны периодически проходить процедуру калибровки, поскольку они определяют уровень алкоголя в крови по содержанию выдыхаемого воздуха автоматически, согласно встроенному программному коду.

На рис. 4.7 показана схема подключения датчика уровня алкоголя MQ-3 к Arduino, а программный код управления этой цепью представлен в листинге 4.3. Схема подключения датчика уровня алкоголя к Raspberry Pi приведена на рис. 4.8, а программный код Python управления такой цепью — в листинге 4.4.

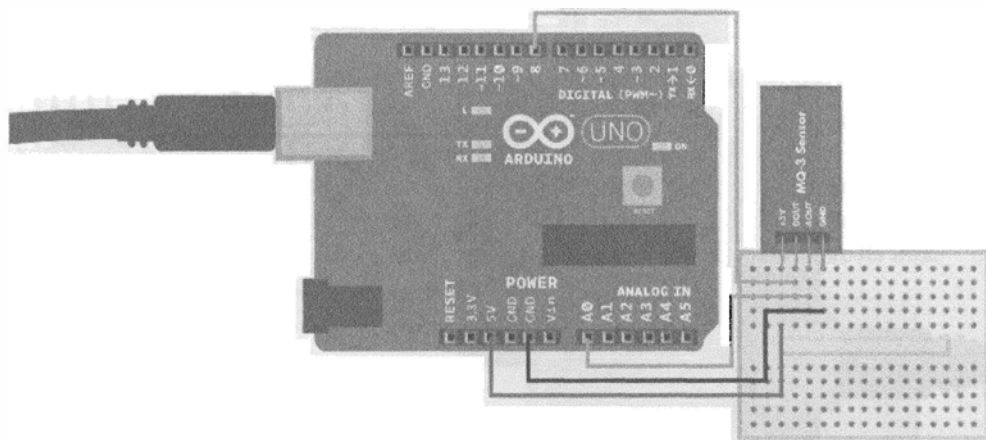


Рис. 4.7. Схема подключения датчика MQ-3 к Arduino

Листинг 4.3. `mq_3_alcohol_sensor.ino`

```
// mq_3_alcohol_sensor.ino - вывод концентрации алкоголя
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int analogPin = A0;
const int digitalPin = 8;

int limit = -1;
int value = 0;

void setup() {
    Serial.begin(115200);
    pinMode(digitalPin, INPUT);
}
```

```

void loop()
{
    // Считывание аналоговых данных
    value = analogRead(analogPin);
    // Проверка превышения граничного значения
    limit = digitalRead(digitalPin);
    Serial.print("Уровень алкоголя: ");
    Serial.print(value);
    Serial.print("Допустимо: ");
    Serial.println(limit);
    delay(100);
}

```

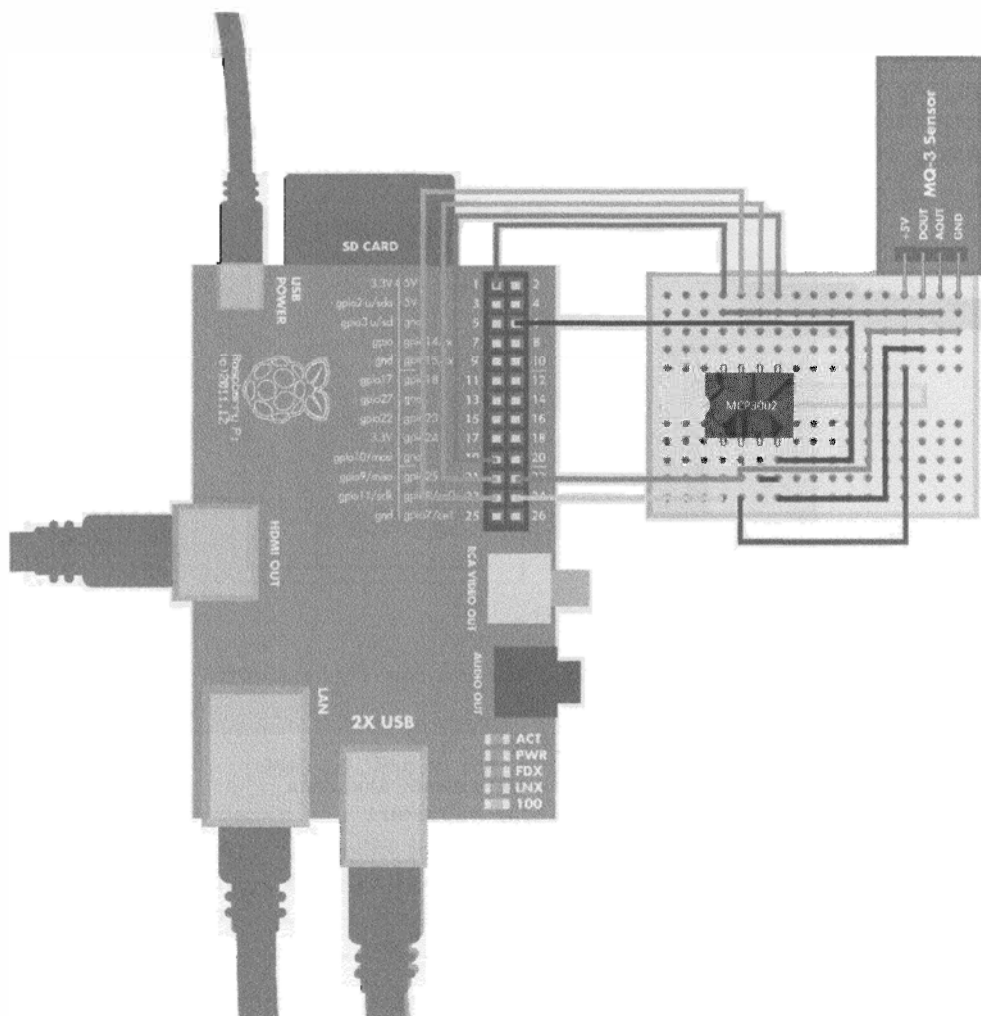


Рис. 4.8. Схема подключения датчика MQ-3 к Raspberry Pi (см. цветную вклейку)

Листинг 4.4. `mq_3_alcohol_sensor.py`

```
# mq_3_alcohol_sensor.py - вывод концентрации алкоголя
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio

def readLimit():
    limitPin = 23
    gpio.setMode(limitPin, "in")
    return gpio.read(limitPin)

def main():
    while True:
        if readLimit() == gpio.HIGH:
            print("Больше допустимого!")
            time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Эксперимент в окружающей среде: жизнь без алкоголя

Употребление алкоголя в процессе обучения или на производстве не сулит ничего хорошего. Хорошая новость заключается в том, что для тестирования алкотестера совсем не обязательно соблазнять себя покупкой бутылкой виски или водки. Вы будете удивлены, узнав, сколько товаров, принесенных из супермаркета, содержат алкоголь. Ополаскиватель для рта и конфеты с ликером — это самые очевидные из них.



Запустите программный код управления алкотестером и отобразите монитор последовательного порта. Съешьте пару конфет с ликером или прополощите рот специальной жидкостью (ни в коем случае не глотайте ее) и подышите на датчик.

Выводимые значения должны резко увеличиться. Показатели будут оставаться высокими в течение нескольких последующих минут, а затем выводимые значения начнут уменьшаться.

Пилотный проект: отправка извещения о задымленности по электронной почте

Задача проста: отправить почтовое извещение при обнаружении дыма в помещении.

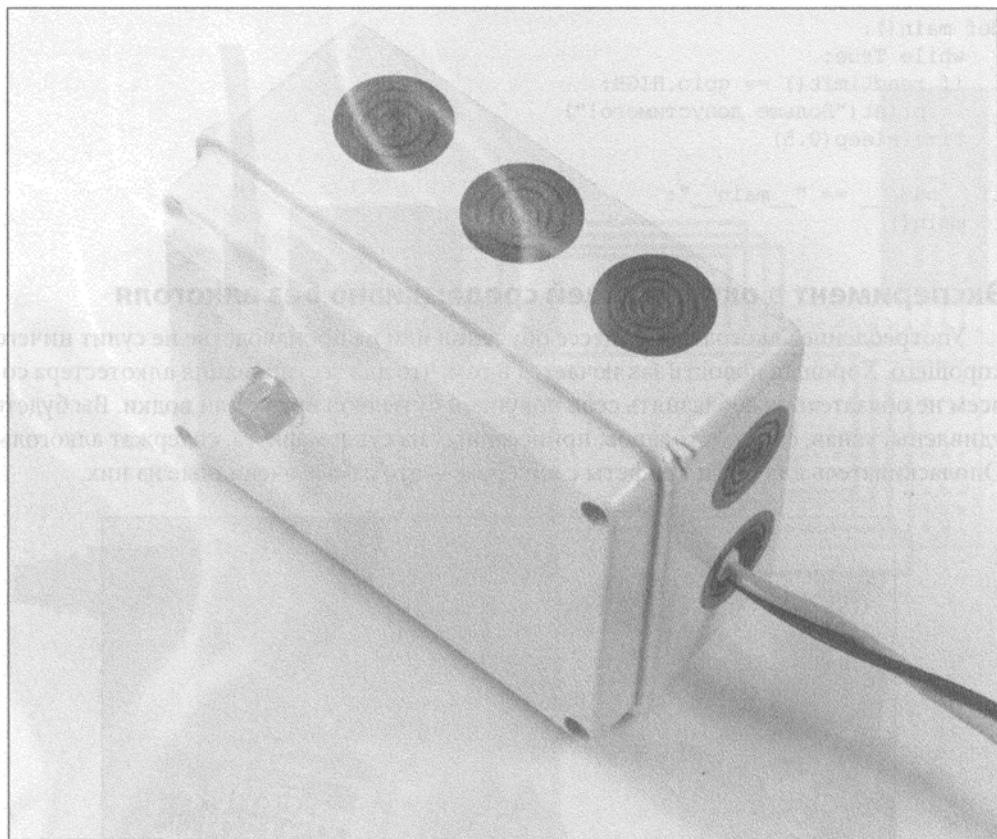


Рис. 4.9. Готовая к использованию дымовая сигнализация, умеющая отправлять извещения через Интернет

Получаемые навыки

В этом проекте вы научитесь следующему:

- выполнять действия в ответ на возникновение неких событий, в частности, отправлять почтовое сообщение при задымлении помещения;

- автоматически генерировать сообщения в Raspberry Pi;
- отправлять извещения по электронной почте.

Отправка электронных писем и извещений с помощью Python

Ниже мы рассмотрим проект, который проще реализовать в Raspberry Pi. Автоматическая отправка почтовых сообщений в Raspberry Pi ничем не отличается от такой в любой другой среде Linux.

Язык программирования Python известен своей “библиотечной” концепцией, подразумевающей существование программных библиотек для всего без исключения. Отправка почтовых сообщений — это рутинная задача, решаемая с помощью уже существующих библиотек. Неудивительно, что программисты уже давно пользуются библиотеками, позволяющими отправлять и получать данные через Интернет с помощью протокола HTTP.

А как же социальные сети? Адаптировать готовые программы для отправки сообщений в Твиттер, Фейсбук и другие социальные сети не составит большого труда. Но хотя во всех социальных сетях для обмена данными и используются протоколы, основанные на открытых технологиях, все они патентуются и вносят определенные ограничения на использование внутренних сервисов. При этом решения об изменениях в доступе к сервисам могут приниматься чуть ли не каждый день. Так, например, в Твиттере часто изменяется количество запросов, подаваемых вами за день, или удаляются возможности, без которых эта социальная сеть становится просто ненужной. И тут ничего не поделаешь, приходится либо принимать условия игры, либо вообще отказаться от социальной сети, правила которой вас не устраивают.

Тестирование оборудования

Подключите датчик дыма MQ-2 к Raspberry Pi, как было показано на рис. 4.4. Желательно сразу проверить работоспособность датчика, выполнив готовую программу, код которой приведен в листинге 4.2.

Убедившись в работоспособности оборудования, улучшите программный код, добавив в него команды отправки почтовых сообщений. Проверьте свою почтовую учетную запись прежде, чем использовать ее в текущей программе. Подойдет любой надежный почтовый клиент, желательно тот, в котором вы уже работали. В нашем случае используется учетная запись Gmail, но вы вольны использовать любую другую, которая у вас есть.

В целях тестирования работоспособности программы не стоит использовать рабочую почтовую учетную запись. В случае ошибки в программном коде и отправки из почтового ящика слишком большого количества сообщений почтовый сервер может расценить это как попытку рассылки писем со спамом. Не поленитесь и создайте отдельный почтовый ящик исключительно для тестирования текущих проектов (если такового у вас еще нет).

Почтовый клиент

Вы еще не забыли, как пользоваться электронной почтой? На пике своей популярности такие гиганты почтовых служб, как Gmail, практически стандартизировали интерфейс почтовых клиентов. Годы, проведенные перед экраном за деловой перепиской, не дают забыть основы почтового дела.

Ниже мы опишем принципы работы почтового клиента, запускаемого на локальном компьютере, а не сетевой почтовой службы. Для простоты изложения будем рассматривать все на тривиальных примерах.

Как вам, наверное, известно, за отправку и получение почтовых сообщений отвечают два разных сервера (отправителя и получателя). Само собой разумеется, что отправляющий и получающий почтовые сообщения серверы могут находиться в разных сетях и даже на разных континентах.

Щелкая на кнопке Send (Отправить) в почтовом клиенте своего компьютера, вы отправляете электронное письмо на SMTP-сервер. Он может располагаться в вашей локальной сети — его поддержку осуществляет ваш провайдер (например, `smtp.verizon.net` или `smtp.comcast.net`). В таком случае доступ к почтовой учетной записи может предоставляться без ввода пароля. Если SMTP-сервер располагается на специальном почтовом домене (например, `smtp.gmail.com` или `mail.gmx.com`), то он обеспечивает шифрование данных по технологии TLS/SSL и требует обязательного ввода учетных данных. Как бы там ни было, получив почтовое сообщение, SMTP-сервер отправляет его получателю.

Проверяя наличие в почтовом ящике новым писем, вы даете указание почтовому клиенту обратиться к IMAP-серверу. Этот сервер также находится либо в локальной сети, либо на специальном почтовом домене. Чтобы прочитать почтовые сообщения, вам, разумеется, нужно ввести имя учетной записи и пароль, а в случае передачи зашифрованных данных (большинство почтовых служб автоматически шифруют отправляемые сообщения) — расшифровать их (по упомянутой выше технологии SSL/TLS). Получив разрешение, почтовый клиент загружает с IMAP-сервера заголовки сообщений и полные копии самих сообщений. Вкратце процедура такова: при отправке вам почтового сообщения почтовый клиент отправляющей стороны связывается со своим SMTP-сервером, который перенаправляет сообщение на ваш почтовый сервер, а при проверке вами своего почтового ящика вы получаете копию электронного письма со своего IMAP-сервера.

По умолчанию все электронные письма, отправленные на ваш адрес, хранятся на соответствующем IMAP-сервере; именно поэтому, когда вы проверяете свой почтовый ящик из нескольких устройств, в каждом из них отображается список всех принятых почтовых сообщений.

Трудности отправки извещений из Arduino

При попытке выполнения рассматриваемого проекта с помощью Arduino вам потребуется полноценный компьютер. В самом простом варианте данные с датчика дыма считываются Arduino, а затем передаются через последовательное соединение USB в настольный компьютер. Полученные данные в настольном компьютере обрабатываются программой, написанной на Python (с использованием библиотеки pySerial), которая и занимается отправкой почтового сообщения необходимому адресату.

Напрямую отправить почтовое сообщение из Arduino без использования средств настольного компьютера вам, скорее всего, не удастся даже при подключении к платформе дополнительного адаптера Ethernet или Wi-Fi. Несмотря на то что протоколы, используемые для передачи почтовых сообщений, кажутся простыми, их полноценное использование требует знания большого количества тонкостей применяемых технологий, поэтому создать надежную библиотеку SMTP для Arduino очень сложно, а порой и невозможно, поскольку многие почтовые серверы используют SSL-шифрование, а в Arduino Uno и некоторых других моделях платформы оно не поддерживается.

Становится очевидным, что лучше реализовать данный проект с помощью Raspberry Pi или другой аналогичной микроконтроллерной платформы, например BeagleBone (или же гибрида Arduino/Linux — Arduino Yun).

Программа отправки извещений из Raspberry Pi

Листинг 4.5. smoke_alarm.py

```
# smoke_alarm.py - отправка извещений о задымленности
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import botbook_mcp3002 as mcp
import smtplib # ❶
from email.mime.text import MIMEText # ❷

# Почтовые адреса
email_to = 'example@gmail.com' # ❸
email_from = 'example@gmail.com' # ❹

# Настройка SMTP-сервера
server = 'smtp.gmail.com' # ❺
mail_port = 587
user = 'example@gmail.com' # ❻
password = 'password' # ❼
gracePeriod = 5 * 60 # секунд # ❽

def sendEmail(subject, msg): # ❹
    msg = MIMEText(msg) # ❶
    msg['Subject'] = subject # ❷
    msg['To'] = email_to # ❸
    msg['From'] = email_from
    smtp = smtplib.SMTP(server, mail_port) # ❶
    smtp.starttls()
```

```

smtp.login(user, password) # 10
smtp.sendmail(email_from, email_to, msg.as_string()) # 11
smtp.quit() # 12

def main():
    while True:
        smokeLevel = mcp.readAnalog()
        print("Концентрация дыма: %i " % smokeLevel)
        if smokeLevel > 120:
            print("Больше допустимого")
            sendEmail("Задымление", "Концентрация дыма: %i" %
smokeLevel) # 13
            time.sleep(gracePeriod) # 14
            time.sleep(0.5) # c # 15

if __name__ == "__main__":
    main()

```

- ❶ Принцип “библиотечности” Python предполагает подключение специальной библиотеки, обеспечивающей поддержку SMTP-технологий обмена сообщениями. Вам не придется самостоятельно заниматься низкоуровневым программированием сокетов.
- ❷ Почтовые сообщения состоят из простого текста: строки заголовков (каждый заголовок в одной строке), пустой строки и тела сообщения. Но в текстовых сообщениях повсеместно используются символы, отсутствующие в ASCII-кодировке. Именно поэтому тело сообщения обычно представляется в MIME-кодировке, как и вложения в электронные письма. Вам может показаться странным, но в большинстве европейских языков имеются буквы, не включенные в ASCII-кодировку.
- ❸ В поле To указывается получатель сообщения. Скорее всего, вам нужно указать в этом поле адрес своего почтового ящика, чтобы иметь возможность первым получать извещения. Адреса получателей хранятся в глобальных переменных, доступных всем функциям.
- ❹ В поле From теоретически можно указать адрес любого почтового ящика. Но во времена активного использования фильтров спама ввод произвольного адреса может вызвать блокирование отправки сообщения. По возможности используйте адрес своего почтового ящика, поскольку его доменное имя с высокой степенью вероятности повторяет имя используемого SMTP-сервера.
- ❺ SMTP-сервер отвечает только за отправку сообщений. Поскольку настоящая программа только отправляет сообщения и никогда не принимает их, то SMTP-сервер — это единственный сервер, указываемый в ней.
- ❻ Имя учетной записи на SMTP-сервере. Оно может представляться почтовым адресом (example@gmail.com) или коротким регистрационным именем (jdoe).
- ❼ Пароль доступа к SMTP-серверу должен повторять пароль, который вы вводите, регистрируясь в почтовом клиенте или почтовой интернет-службе.

- ❶ Даже в случае обнаружения дыма в помещении вам не захочется получать по 60 извещений в минуту. Генерируя почтовые сообщения с такой частотой, вы рискуете попасть в “черный список” сервера и быть заблокированным. Именно поэтому вам нужно установить специальную задержку: временной интервал, в течение которого программа будет простаивать перед отправкой очередного сообщения. Задержки в 5 минут вполне достаточно; для преобразования минут в секунды это значение умножается на 60. Проведите вычисления в коде вместо того, чтобы делать это вручную и допустить ошибку. Этот пример не показательный, но общее правило вам понятно (избегайте соблазна повторять вычисления в комментариях).
- ❷ Код отправки почтового извещения для большей наглядности выделен в отдельную функцию. Тема письма (subject) и его основной текст (msg) принимаются функцией в качестве аргументов.
- ❸ Основной текст (тело) письма представляется кодировкой MIME, чтобы позволить вводить сообщения с использованием символов, не представленных в ASCII-кодировке. Как видно по первой большой букве, MIMEText — это название класса. Конструкция MIMEText () возвращает новый объект, сохраняемый в переменной msg. Класс MIMEText отвечает только за создание сообщения и не участвует в установке соединения и отправке сообщения.
- ❹ Чтобы изменить заголовки сообщения, можете использовать объект msg класса MIMEText, подобно тому, как применяются переменные с типом данных dictionary.
- ❺ Отправитель и получатель указываются в глобальных переменных.
- ❻ Создание нового объекта класса SMTP. Он нужен для установки подключения к SMTP-серверу. Начальная часть конструкции (smtpplib) указывает на пространство имен, автоматически создаваемое оператором import smtpplib.
- ❼ Подключение к SMTP-серверу с использованием регистрационного имени и пароля, хранимых в глобальных переменных.
- ❽ Вызов метода sendmail () объекта smtp класса smtpplib.SMTP для отправки сообщения. Метод smtp.sendmail () принимает в качестве параметра строку, поэтому объект msg преобразуется в строку с помощью отдельного встроенного метода.
- ❾ Сбрасывание подключения и очистка памяти. Функция quit () представляет собой метод объекта smtp.
- ❿ Использование форматирующей строки для создания сообщения msg, в котором %i заменяется значением переменной smokeLevel.
- ⓫ Задержка в 5 минут перед отправкой следующего почтового сообщения.
- ⓬ Предотвращение выполнения цикла с максимальной производительностью; задержка перед следующей итерацией в полминуты.

Корпус для дымовой сигнализации

В качестве корпуса нашей дымовой сигнализации мы использовали распределительную коробку общего назначения (рис. 4.10). Несмотря на утилитарный вид, она продается в хозяйственных магазинах и снабжена большим количеством отверстий, закрытых резиновыми заглушками, что делает ее применимой в различных проектах.



Рис. 4.10. Универсальный контейнер для разных электротехнических нужд

В этой коробке нужно просверлить всего одно отверстие — в верхней крышке — под датчик дыма (рис. 4.11). Остальные отверстия вы получите, если прорежете в резиновых заглушках контур необходимого диаметра. Отверстие для датчика дыма можно также прорезать в резиновой заглушке, но после закрепления на потолке такое устройство будет выглядеть неопрятно, поэтому боковые отверстия мы использовали только под вывод проводов. Для вывода датчика понадобится отверстие диаметром 19 мм. Если датчик входит в высверленное отверстие плотно, то нет необходимости закреплять его специальными фиксаторами (рис. 4.12).

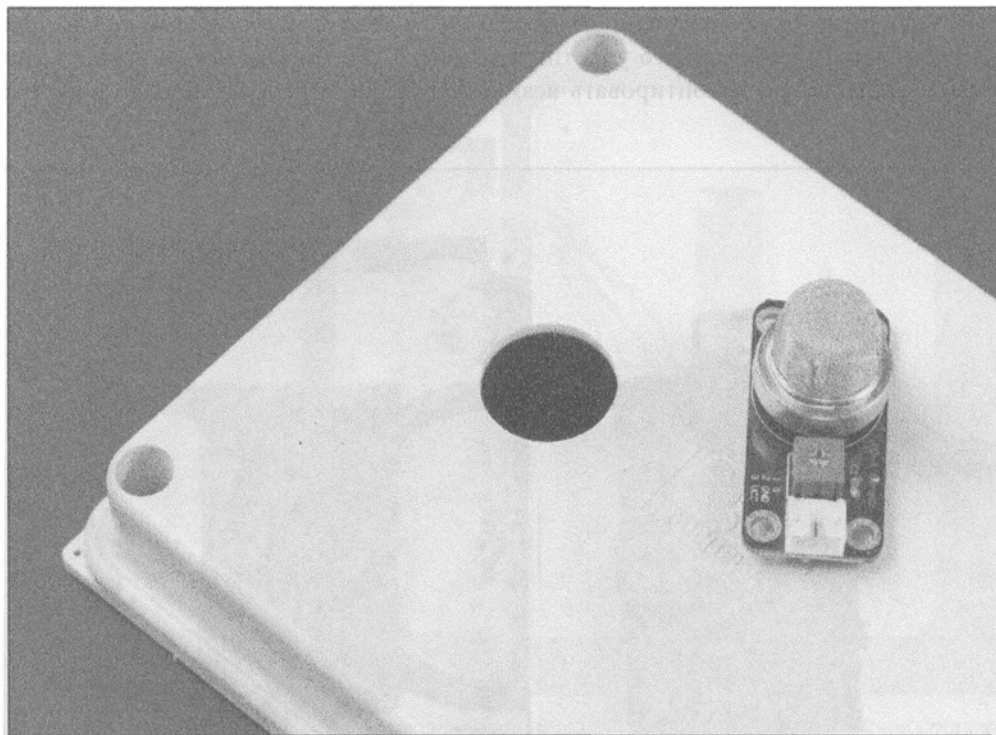


Рис. 4.11. *Отверстие для установки датчика*

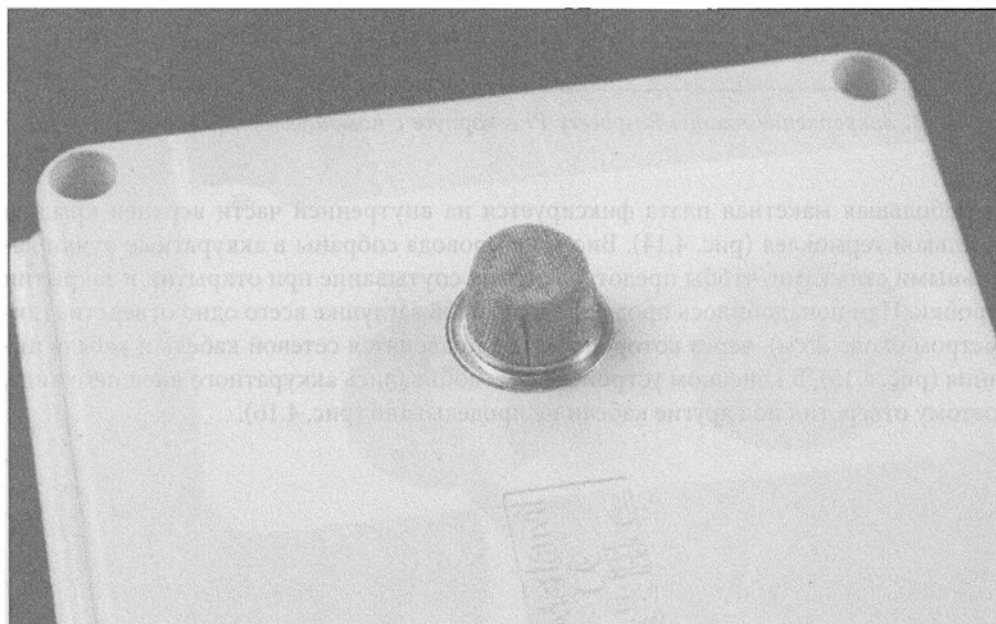


Рис. 4.12. *Датчик на своем месте*

Для неподвижного закрепления платы Raspberry Pi внутри корпуса мы подложили под нее толстый кусок мягкого уплотнителя (рис. 4.13). Это очень удобно, поскольку позволяет быстро демонтировать всю конструкцию и использовать ее в других проектах.

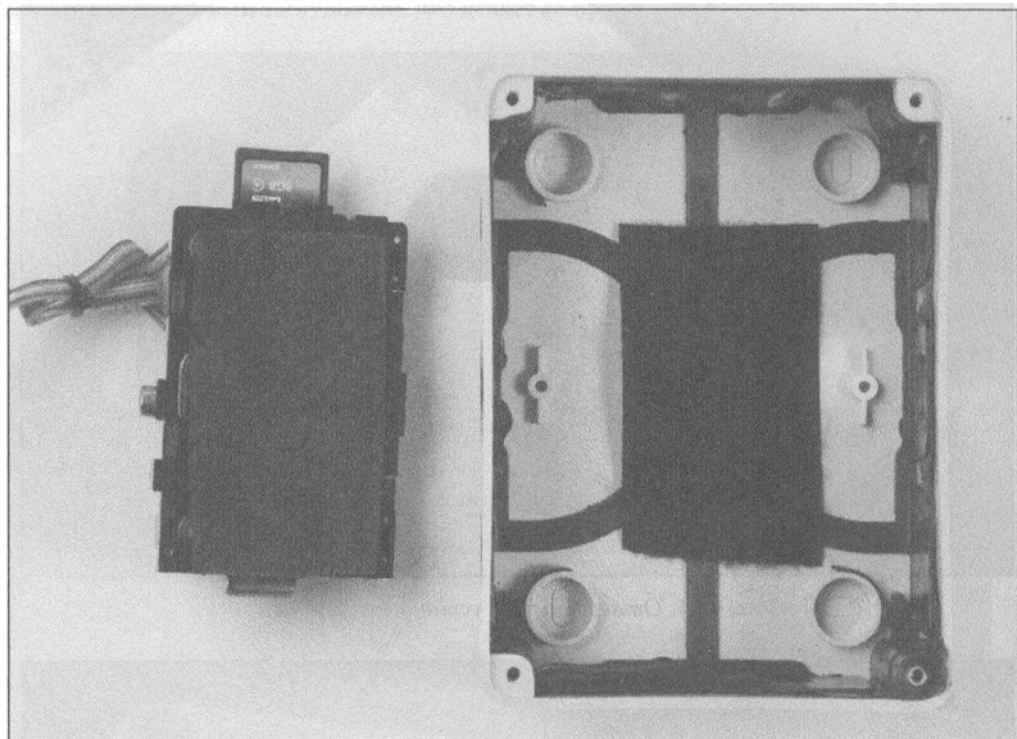


Рис. 4.13. Закрепление платы Raspberry Pi в корпусе с помощью самоклеящегося уплотнителя

Небольшая макетная плата фиксируется на внутренней части верхней крышки капелькой термоклея (рис. 4.14). Висящие провода собраны в аккуратные пучки кабельными стяжками, чтобы предотвратить их спутывание при открытии и закрытии коробки. Нам понадобилось проделать в боковой заглушке всего одно отверстие (диаметром около 2 см), через которое к плате подводится сетевой кабель и кабель питания (рис. 4.15). В конечном устройстве мы добивались аккуратного внешнего вида, поэтому отверстия под другие кабели не проделывали (рис. 4.16).

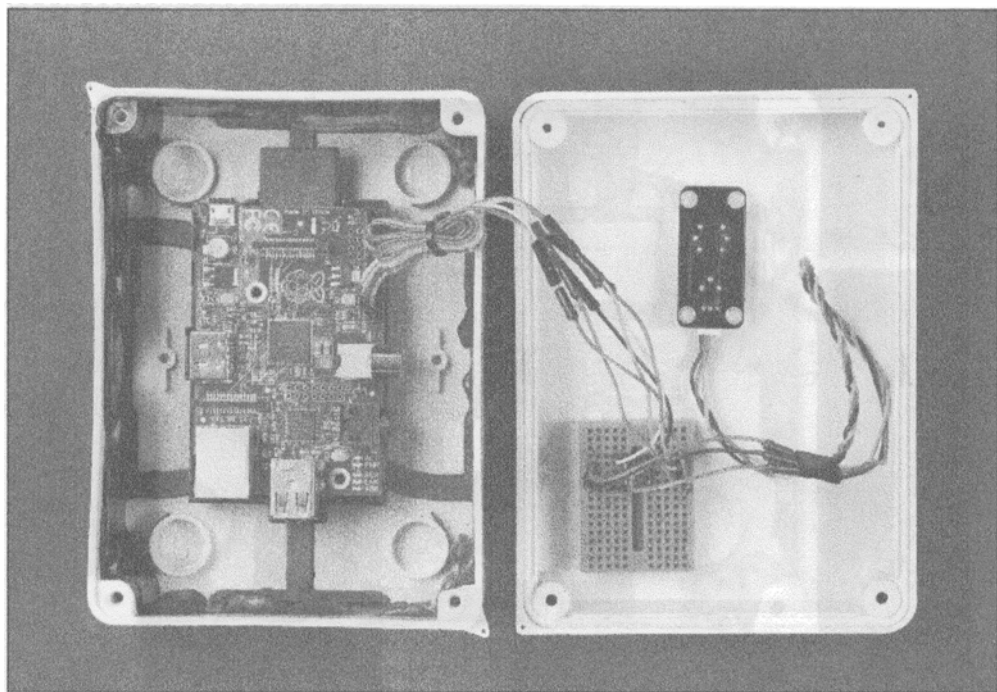


Рис. 4.14. Установка небольшой макетной платы и проводов, собранных в пучок стяжкой

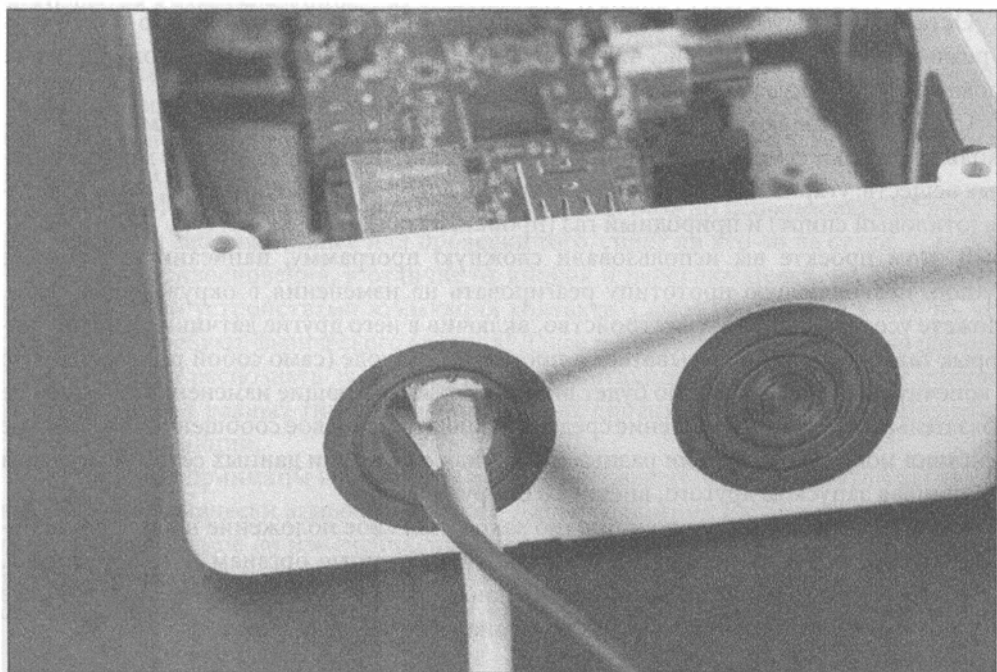


Рис. 4.15. Отверстие в корпусе под кабели

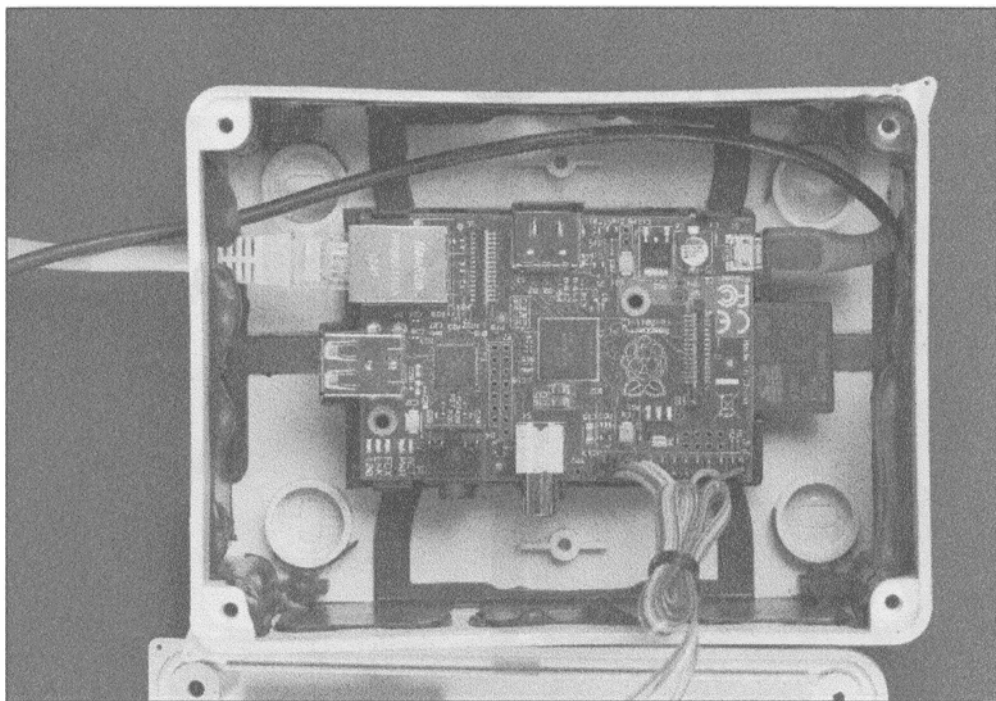


Рис. 4.16. Подключение кабелей

Осталось закрыть все крышкой, и можно приступать к тестированию устройства в реальных условиях (см. рис. 4.9). При размещении сигнализации в помещении не забывайте о том, куда стелется дым при пожаре.

Созданный только что прибор можно смело назвать “электрическим носом”, поскольку датчики семейства MQ умеют распознавать наличие в воздухе многих опасных веществ: угарный газ (CO), углеводороды, водород (взрывоопасен), пары алкоголя (этиловый спирт) и природный газ (пропан, бутан).

В этом проекте вы использовали сложную программу, написанную на языке Python, позволяющую прототипу реагировать на изменения в окружающей среде. Можете усовершенствовать устройство, включив в него другие датчики, данные с которых также будут обрабатываться в программном коде (само собой разумеется, что в конечную программу нужно будет внести соответствующие изменения). Совсем не обязательно в ответ на изменение среды отправлять почтовое сообщение — конечные действия могут быть самыми разными, начиная с передачи данных сетевой службе и заканчивая запуском другого, внешнего оборудования.

Газовые датчики уже давно и прочно закрепили свое положение в индустрии, поскольку позволяют распознавать вещества, недоступные органам чувств человека. В следующей главе мы поговорим еще об одном важном чувстве — осязании, а также о тактильных датчиках, призванных распознавать прикосновение.

Прикосновение

5

Прикосновение уже много лет считается привычным способом управления компьютерными устройствами. Без переключателей вы не включите освещение в кабинете, а уж про кнопки и говорить нечего — пульты управления бытовыми приборами и всевозможные клавиатуры просто усыпаны ими.

В этой главе вы познакомитесь с устройствами, без которых сложно представить современную жизнь: кнопками, датчиками давления и емкостными датчиками прикосновения. Вы узнаете о назначении подтягивающих (нагрузочных) резисторов и их применении в электротехнических устройствах. В конце главы вы сможете закрепить полученные знания, выполнив проект сенсорного звонка.

Всем привычная кнопка — это тоже датчик, поскольку она реагирует на нажатие. Кнопка нажимается, контакты соприкасаются, и цепь замыкается. Микропереключатель тоже относят к кнопкам, хотя он и выделяется в отдельный тип устройств благодаря высокой износостойчивости.

Датчики FlexiForce определяют степень нажатия. Можно использовать их для измерения силы сжатия пальцев или проверки того, сидит ли кто-то на стуле.

Датчики прикосновения, в отличие от кнопок и переключателей, не оснащаются механическими устройствами замыкания контакта. Самое интересное в них то, что они даже не требуют физического касания! Далее вы узнаете, как с помощью датчика прикосновения распознать ладонь через деревянную поверхность. Датчики прикосновения бывают разных типов, хотя в конечных проектах, как правило, выполняют одинаковые функции.

Физические принципы всех датчиков — самые разные. Самый простой датчик — кнопка — механически замыкает контакт. Датчик давления FlexiForce измеряет давление, определяемое по изменению проводимости тонких слоев материала, чувствительного к нажиму. Датчик прикосновения реагирует на изменение емкости — способности материалов накапливать электрический заряд.

Эксперимент: нажатие кнопки

Перед нами стоит задача научиться включать светодиод после нажатия кнопки.

Кнопка, показанная на рис. 5.1, условно относится к семейству датчиков. После нажатия кнопки ее контакты замыкаются, и она начинает функционировать как неразрывный проводник. Отжатие кнопки приводит к размыканию контактов, а следовательно, разрыву цепи. Все цифровые переключатели на базе датчиков работают по такому же принципу (например, герконовый переключатель и датчик наклона). Можно утверждать, что кнопка — это простейший цифровой датчик.

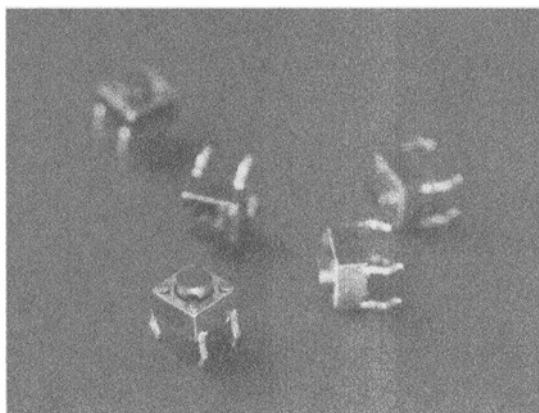


Рис. 5.1. Кнопка

Кнопки бывают разных размеров и форм. Создавая прототипы устройств на макетной плате, удобнее использовать четырехвыводные кнопки. Соседние выводы кнопки соединены друг с другом, поэтому замыкаются и размыкаются они попарно. Если вы перевернете кнопку обратной стороной к себе, то сможете увидеть, какие контакты объединены между собой. Несложно догадаться, что контакты, соединенные пластиковой перемычкой, имеют общий вывод (рис. 5.2).

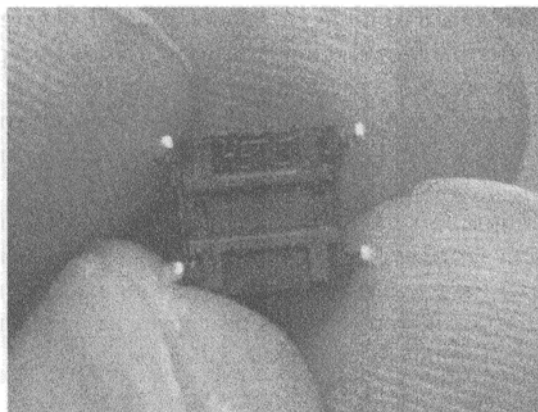


Рис. 5.2. Вид кнопки снизу

Подтягивающий (нагрузочный) резистор

Чтобы получить сигнал с вывода, его сначала нужно к чему-нибудь подключить. Невозможно прочитать сигнал с неподключенного к нагрузке, висящего в воздухе вывода. Назначение подтягивающего резистора — предотвратить образование такой неопределенности в случае отсутствия нагрузки, обеспечивая подачу на него сигнала высокого уровня (HIGH).

Текущее состояние вывода считывается командами `digitalRead()`, `analogRead()`, `botbook_gpio.read()`, `botbook_mcp2003.readAnalog()`.

Состояние выводов цепи, подключенных к шине заземления или питания, не вызывает сомнений. Если вывод заземлен (GND, 0 В), то возвращается цифровое значение LOW. Если вывод подключен к шине питания (+5 В или +3,3 В), то возвращается сигнал высокого уровня HIGH.

При попытке прочитать состояние ненагруженного вывода вы получите неопределенное значение: с одинаковой степенью вероятности вы получите сигнал HIGH или LOW, который может постоянно изменяться на противоположный или оставаться на исходном уровне бесконечно долго. Такое неопределенное состояние невозможно измерить, а потому использовать в практических целях.

Для наглядности рассмотрим электрическую цепь, состоящую из кнопки, подключенной своими контактами с одной стороны к шине заземления, а с другой — к сигнальному выводу платы Arduino (рис. 5.3). Когда кнопка нажата, сигнальный вывод заземляется, что равнозначно подаче на него сигнала низкого уровня.

А можете ли вы точно определить, какое состояние устанавливается на выводе при размыкании контактов кнопки? Чтобы однозначно задать сигнал высокого уровня, вам нужно включить в цепь между шиной питания и сигнальным выводом подтягивающий резистор.

Сопротивление подтягивающего резистора очень большое. Обычно применяются резисторы номиналом в несколько десятков килоом. В подобном случае после нажатия кнопки вывод подключается и к шине заземления, и к подтягивающему резистору, а потому короткое замыкание между выводом с напряжением +5 В и землей исключается. (Ток течет по пути наименьшего сопротивления, в нашем случае — точно не через резистор.)

Стоит заметить, что в примере, показанном на рис. 5.3, задействуются встроенные в плату Arduino подтягивающие резисторы. Для их включения используется специальный программный код. В Raspberry Pi подтягивающим (нагрузочным) резистором (постоянно задействованным) снабжаются только определенные выводы.

Подключение к Arduino и программа управления кнопкой

Создайте электрическую цепь, схематически показанную на рис. 5.3, и выполните программный код из листинга 5.1. Нажмите кнопку, чтобы зажечь встроенный в плату Arduino светодиод.

Если после нажатия кнопки ничего не происходит, то убедитесь в правильности подключения к кнопке проводочных перемычек.

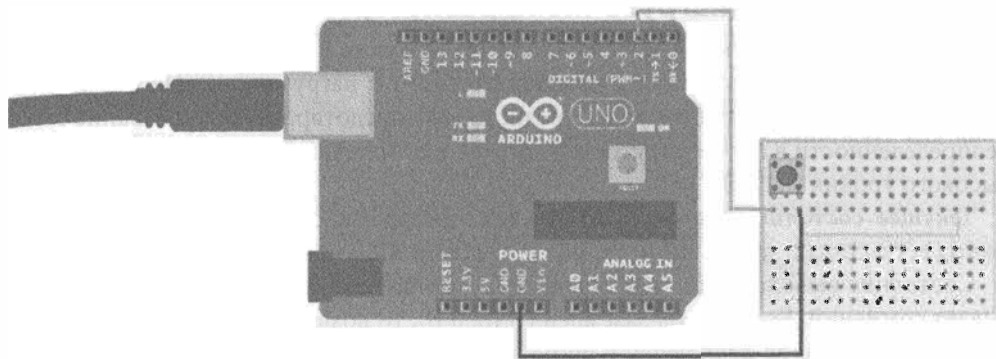


Рис. 5.3. Схема подключения кнопки к плате Arduino

Листинг 5.1.button.ino

```
// button.ino - зажигание светодиода после нажатия кнопки
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int buttonPin=2;
int ledPin=13;
int buttonStatus=-1;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT); // ❶
  digitalWrite(buttonPin, HIGH); // встроенный резистор // ❷
}

void loop() {
  buttonStatus=digitalRead(buttonPin); // ❸
  if (LOW==buttonStatus) { // ❹
    digitalWrite(ledPin, HIGH); // ❺
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(20); // ❻
}
```

- ❶ Перевод вывода в режим INPUT, чтобы получить возможность считывать сигнал с помощью функции `digitalRead()`.
- ❷ Подключение вывода D2 к “земле” (вывод GND) вместе со встроенным резистором большого сопротивления (20 кОм). Подтягивающий резистор предотвращает возникновение неопределенности состояния вывода при размыкании контактов кнопки. Наличие внутреннего подтягивающего резистора очень удобно, поскольку избавляет вас от необходимости вручную подбирать и включать в цепь дополнительный резистор. Последние версии платформы Arduino обеспечивают поддержку обновленной функции `pinMode()` с параметром `INPUT_PULLUP`, которая включает в себя задачи, возложенные на текущую строку кода: `pinMode(buttonPin, INPUT_PULLUP);`. Это не означает, что старый метод

отныне не поддерживается; от него никто не отказывается, и вы его встретите во многих программах, написанных до появления конструкции `pinMode()`.

- ❸ Считывание напряжения на выводе D2. Значение HIGH соответствует +5 В, а значение LOW — 0 В.
- ❹ Если вывод кнопки переведен в состояние LOW (кнопка нажата)...
- ❺ ...то светодиод загорается. Светодиод, встроенный в Arduino, соединен с выводом D13.
- ❻ Небольшая задержка, предотвращающая перегрузку микропроцессора.

В этом коде установлено, что светодиод загорается после каждого нажатия кнопки. Если нужно при каждом нажатии кнопки изменять состояние светодиода на противоположное, то сначала вам следует побороть проблему дребезга контактов, возникающую при включении механических переключателей. Эффект дребезга контактов заключается в частых паразитных изменениях уровня сигнала с HIGH в LOW при включении/выключении механических переключателей. Он часто приводит к ошибкам последующего определения состояния кнопки, без чего невозможно настроить функцию отключения светодиода при повторном ее нажатии. Пример программы борьбы с дребезгом сигнала в Arduino вы найдете в среде разработки: **File⇒Examples⇒2.Digital⇒Debounce** (Файл⇒Примеры⇒2.Digital⇒Debounce).

Подключение к Raspberry Pi и программа управления кнопкой

Подключите кнопку к Raspberry Pi, как показано на рис. 5.4, и выполните программу Python, код который представлен в листинге 5.2.

Листинг 5.2. `button.py`

```
# button.py - вывод на экран состояния кнопки
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio # ❶

def main():
    buttonpin = 3 # встроенный подтягивающий резистор # ❷
    gpio.mode(buttonpin, "in") # ❸

    while (True): # ❹
        buttonUp = gpio.read(buttonpin) # ❺
        if(buttonUp == gpio.HIGH):
            print "Кнопка отжата"
        else:
            print "Кнопка нажата"
            time.sleep(0.3) # секунд # ❻

if __name__ == "__main__":
    main()
```

- ❶ Убедитесь, что в каталоге с кодом текущей программы находится файл библиотеки `botbook_gpio.py`. Можете загрузить файл библиотеки, а также файлы других примеров книги по адресу, указанному во введении. Детально о настройке доступа к выводам интерфейса GPIO в Raspberry Pi рассказывалось в главе 1.
- ❷ Вывод `gpio3` снабжается встроенным подтягивающим резистором: выход микросхемы подключается (подтягивается) к источнику питания +3,3V через сопротивление 1800 Ом.
- ❸ Переведите вывод `gpio3` в состояние входа, чтобы иметь возможность прочитать на нем входной сигнал функцией `gpio.read()`.
- ❹ Выполнение программы продолжается до ее завершения пользователем с помощью комбинации клавиш `<Ctrl+C>`.
- ❺ Считывание состояния вывода. Возможен вариант `True` (Истина) или `False` (Ложь).
- ❻ Небольшая задержка, чтобы избежать полной загрузки микропроцессора циклом `while (True)`. Кроме того, задержка уменьшает скорость появления текста на мониторе.

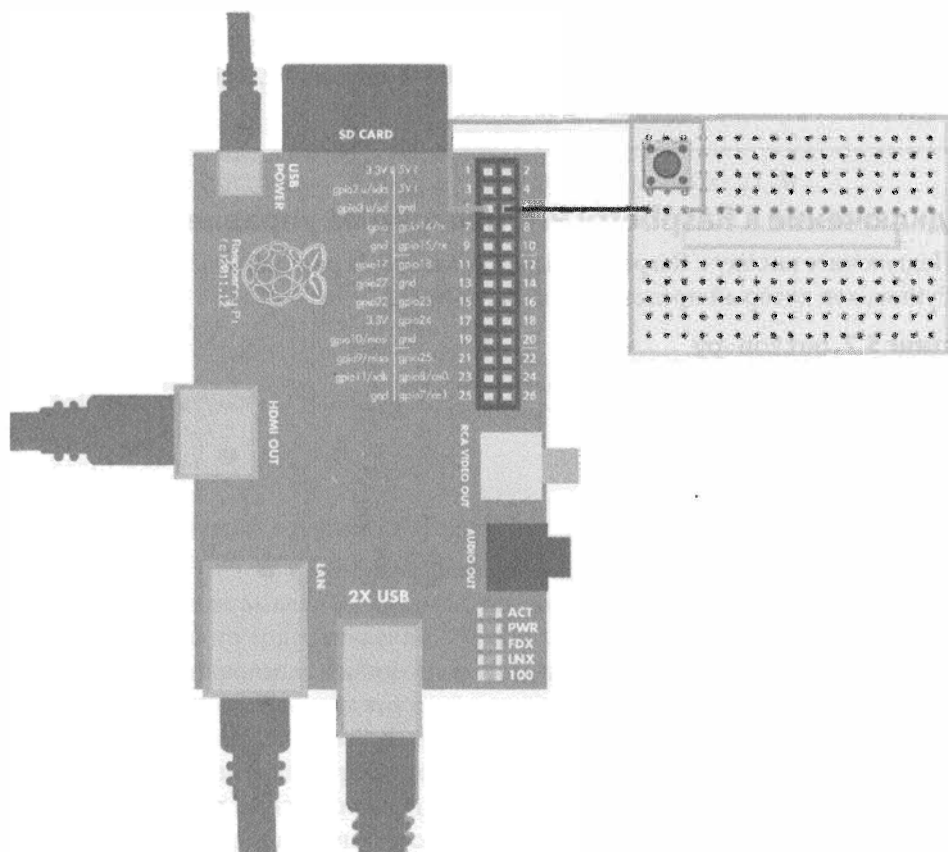


Рис. 5.4. Схема подключения кнопки к Raspberry Pi (см. цветную вклейку)

Эксперимент: микропереключатель

Микропереключатель ничем особенным не отличается от кнопки (рис. 5.5). Как вы знаете, после нажатия кнопки ее контакты замыкаются. В рассмотренных ниже примерах после нажатия кнопки микропереключателя на монитор последовательного порта выводится значение 0.

Микропереключатели из-за подобия кнопкам применяются в электротехнических схемах так же, как и кнопки (см. предыдущий раздел). Как и в случае кнопок, модель и производитель микропереключателя не имеют большого значения. Подключение и управление в программном коде стандартизировано для большей части микропереключателей общего назначения.

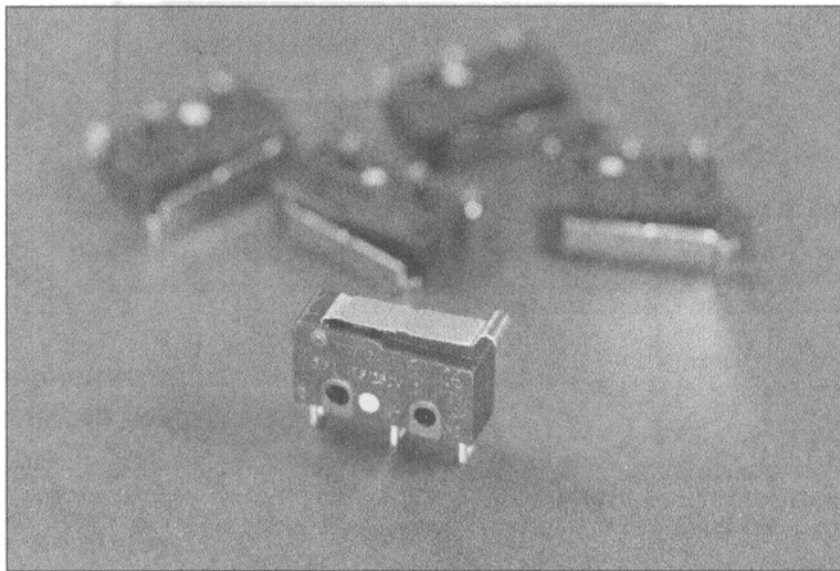


Рис. 5.5. Микропереключатель

Микропереключатели приобрели большую популярность благодаря невысокой стоимости, небольшому размеру и высокой отказоустойчивости. Срок службы типичного микропереключателя составляет более миллиона нажатий. Щелчок при нажатии и слабая отдача (обратная связь) исходит от рычажка, поворачивающегося относительно толкателя, что приводит к его удару при замыкании контактной группы.

Благодаря такому технологическому исполнению в схемах с переключателями отпадает необходимость включения подтягивающего резистора, играющего важную роль при использовании в электрических цепях кнопок.

Подключение к Arduino и программа управления микропереключателем

Постройте электрическую цепь согласно схеме, показанной на рис. 5.6, и выполните программный код, приведенный в листинге 5.3.

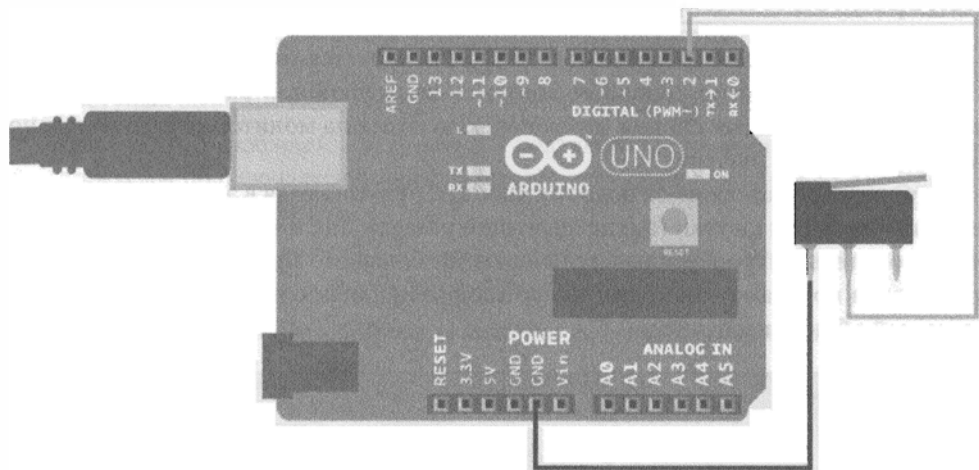


Рис. 5.6. Схема подключения микропереключателя к Arduino

Листинг 5.3. `microswitch.ino`

```
// microswitch.ino - вывод на монитор состояния
// микропереключателя
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int switchPin = 2;
int switchState = -1; // ❶

void setup() {
    Serial.begin(115200);
    pinMode(switchPin, INPUT);
    digitalWrite(switchPin, HIGH); // встроенный резистор // ❷
}

void loop() {
    switchState = digitalRead(switchPin);
    Serial.println(switchState); // ❸
    delay(10);
}
```

- ❶ Начальная установка невозможных значений.
- ❷ Перевод вывода, к которому подключен микропереключатель, в состояние HIGH, что соответствует подаче на него напряжения +5 В через встроенный подтягивающий резистор 20 кОм.
- ❸ При замыкании контактной группы микропереключателя вывод D2 напрямую заземляется, и переменная `switchState` принимает значение LOW. В подобном случае сопротивление подтягивающего резистора настолько большое, что ток через него не проходит, а потому на выводе D2 устанавливается сигнал низкого уровня. В случае разомкнутой контактной группы подтягивающий резистор "подстраивает" сигнал до уровня HIGH (+5 В).

Из микропереключателя удобно создавать щупы для своих роботизированных устройств. Для фиксации стяжки необходимой длины на кнопке переключателя воспользуйтесь клеевым пистолетом (рис. 5.7).

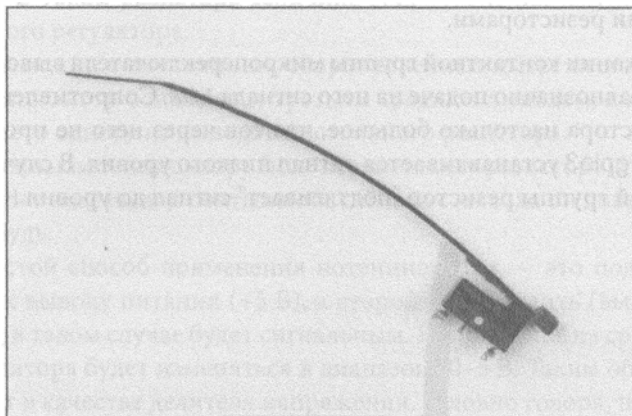


Рис. 5.7. Выносной щуп на базе микропереключателя, устанавливаемый впоследствии на самодвижущиеся роботы

Подключение к Raspberry Pi и программа управления микропереключателем

Подключите микропереключатель к Raspberry Pi согласно схеме, показанной на рис. 5.8, а затем выполните программу Python из листинга 5.4.

Листинг 5.4. `microswitch.py`

`microswitch.py` – вывод на монитор состояния микропереключателя
(c) BotBook.com – Karvinen, Karvinen, Valtokari

```
import time
import botbook_gpio as gpio # ❶

def main():
    switchpin = 3 # встроенный подтягивающий резистор # ❷
    gpio.mode(switchpin, "in")

    while (True):
        switchState = gpio.read(switchpin) # ❸
        if(switchState == gpio.LOW):
            print "Переключатель включен"
        else:
            print "Переключатель выключен"
            time.sleep(0.3) # секунд

if __name__ == "__main__":
    main()
```

- ❶ Убедитесь, что в каталоге с файлом текущей программы находится файл библиотеки `botbook_gpio.py`. Можете загрузить файл библиотеки, а также файлы других примеров книги по адресу, указанному во введении. Детально о настройке доступа к выводам интерфейса GPIO в Raspberry Pi рассказано в главе 1.
- ❷ Выводы `gpio2` и `gpio3` на постоянной основе снабжаются встроенными подтягивающими резисторами.
- ❸ При замыкании контактной группы микропереключателя вывод `gpio3` заземляется, что равнозначно подаче на него сигнала LOW. Сопротивление подтягивающего резистора настолько большое, что ток через него не проходит, а потому на выводе `gpio3` устанавливается сигнал низкого уровня. В случае разомкнутой контактной группы резистор "подтягивает" сигнал до уровня HIGH.

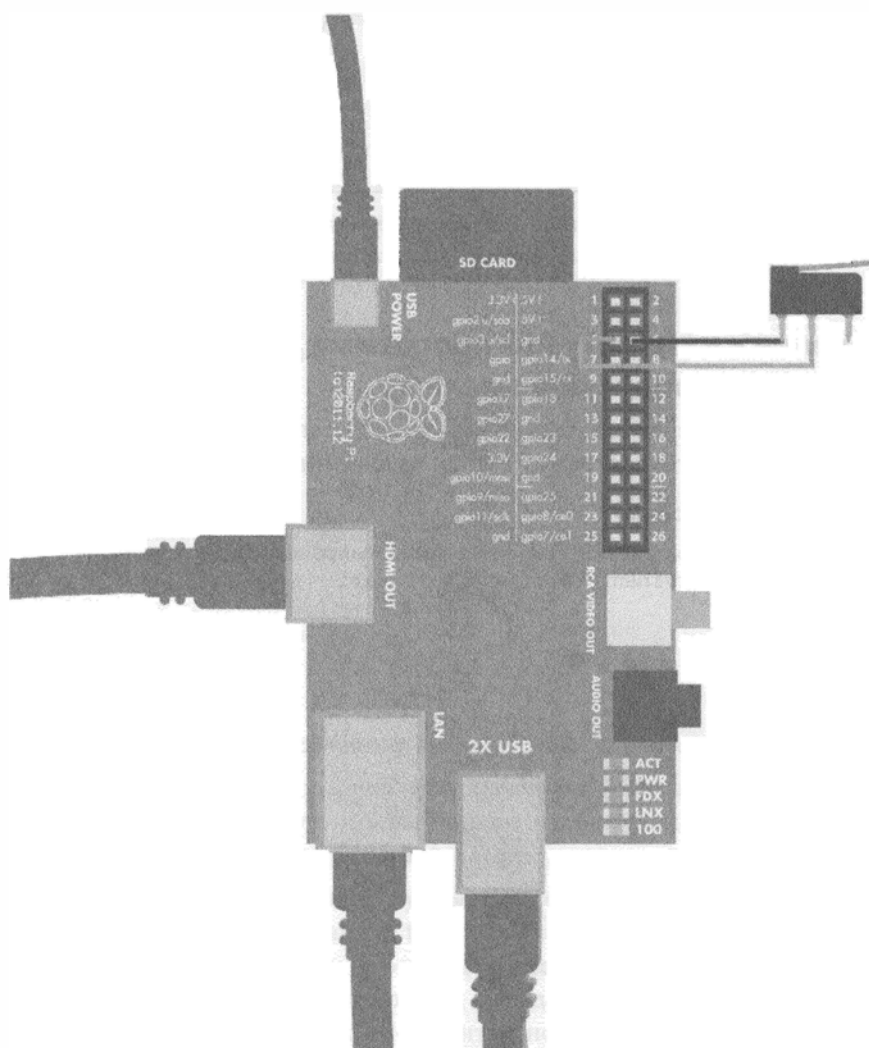


Рис. 5.8. Схема подключения микропереключателя к Raspberry Pi

Эксперимент: потенциометр (переменный резистор)

В этом эксперименте вы научитесь управлять скоростью мигания светодиода с помощью потенциометра.

Потенциометр представляет собой не что иное, как резистор с переменным сопротивлением. В таком резисторе величина сопротивления изменяется при вращении специального регулятора.

Обычный резистор с постоянным сопротивлением имеет всего два вывода. В то же время потенциометр оснащается третьим средним выводом, который подключается в цепь вместо одного из выводов постоянного резистора.

Зачем же нужны потенциометру (или переменному резистору) три вывода? Ответ прост. Как вы уже знаете, для считывания сигнала с вывода он должен быть подключен к чему-нибудь.

Самый простой способ применения потенциометра — это подключить один из его контактов к выводу питания (+5 В), а второй — заземлить (вывод GND). Третий средний вывод в таком случае будет сигнальным. Напряжение на среднем выводе при повороте регулятора будет изменяться в диапазоне 0–5 В. Таким образом, потенциометр выступает в качестве делителя напряжения. Условно говоря, потенциометр разделяет исходный резистор на два резистора меньшего номинала, один из которых подключен к шине питания, а второй — к “земле”. Сопротивления каждого из резисторов изменяются по мере вращения регулятора.

На рис. 5.9 схематически показан принцип работы потенциометра. Со среднего вывода считывается сигнал, и напряжение на нем зависит от перераспределения сопротивлений в потенциометре. На один боковой вывод подается напряжение +5 В, а второй подключается к “земле”. В данном случае резистор схематически показан как дуга, соединяющая боковые выводы, сопротивление которой изменяется линейным образом.

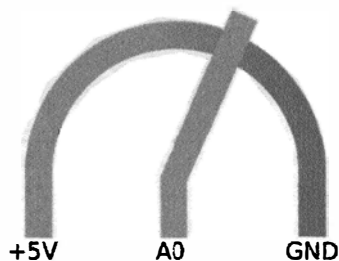


Рис. 5.9. Принципиальная схема устройства потенциометра

Поворачивая регулятор, вы определяете место соприкосновения скользящего контакта с дуговым резистором. Вывод с напряжением +5 В не закорачивается напрямую на общий провод, поскольку между ним и заземленным выводом располагается длинный дуговой резистор с большим сопротивлением.

Повернув регулятор максимально вправо, вы соприкоснете скользящий контакт с выводом GND, при этом со среднего вывода потенциометра считывается сигнал 0 В. В таком случае ток на сигнальный вывод будет проходить через всю дугу потенциометра.

Если повернуть регулятор потенциометра максимально влево, то скользящий контакт соприкоснется с выводом +5 В. Со среднего контакта при этом считывается сигнал +5 В, поскольку ток при прохождении через потенциометр не встретит никакого сопротивления.

Установив регулятор потенциометра в промежуточном положении, вы получите на среднем выводе сигнал, находящийся в диапазоне от 0 до +5 В.

Рассмотренный выше потенциометр условно подключается к плате Arduino с шиной питания +5 В. Принцип работы такого же потенциометра в Raspberry Pi остается прежним, но в ней напряжение питания составляет +3,3 В, соответственно на среднем выводе вы получите напряжение в диапазоне 0–3,3 В.

Подключение к Arduino и программа управления потенциометром

На рис. 5.10 показана схема подключения потенциометра к Arduino. Постройте согласно этой схеме электрическую цепь и выполните программный код, приведенный в листинге 5.5.

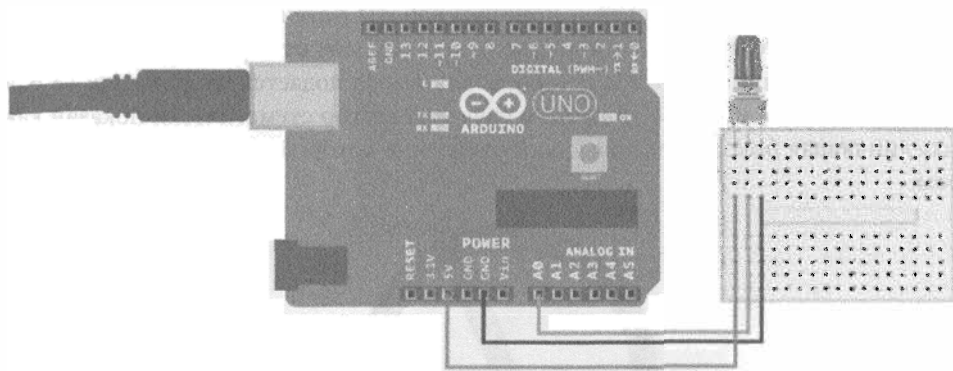


Рис. 5.10. Схема подключения потенциометра к Arduino

Листинг 5.5. pot.ino

```
// pot.ino - Изменение частоты мигания светодиода потенциометром  
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
int potPin=A0; // ①  
int ledPin=13;  
int x=0; // 0...1023
```

```
void setup() {
```

```

pinMode(ledPin, OUTPUT);
pinMode(potPin, INPUT);
}

void loop() {
  x=analogRead(potPin); // ❶
  digitalWrite(ledPin, HIGH);
  delay(x/10); // мс // ❷
  digitalWrite(ledPin, LOW);
  delay(x/10);
}

```

- ❶ Средний вывод потенциометра подключается к одному из выводов Arduino, поддерживающих считывание аналоговых данных.
- ❷ Переменной `x` присваивается значение, возвращаемое функцией `analogRead()`, в диапазоне от 0 (0 В) до 1023 (5 В).
- ❸ Сопротивление потенциометра определяет задержку перед подачей напряжения на светодиод. Задержка задается в диапазоне от 0 до приблизительно 100 мс ($1023/10$).

Подключение к Raspberry Pi и программа управления потенциометром

На рис. 5.11 показана схема применения потенциометра при подключении к плате Raspberry Pi. Создайте с ее помощью электрическую цепь, а затем выполните программу, код которой приведен в листинге 5.6.

В отличие от Arduino, платформа Raspberry Pi не оснащается встроенным аналого-цифровым преобразователем. Это, в частности, означает, что подключать аналоговые датчики к Raspberry Pi проблематичнее, чем к Arduino.

Потенциометр — это очень простое устройство, что видно по программному коду управления им в Arduino. Но получать аналоговый сигнал с потенциометра в Raspberry Pi много сложнее.

Ранее вы уже познакомились с методикой считывания аналоговых данных в Raspberry Pi (см. описание проекта подключения составного датчика ИК-излучения в главе 3), а потому знаете, что для этих целей используется библиотека `botbook_mcp3002`. Разве вам не интересно, как работает эта библиотека?

Программный код библиотеки предназначен для обеспечения поддержки в Raspberry Pi внешнего аналого-цифрового преобразователя MCP3002, без которого невозможно подключить к платформе аналоговый датчик. Если отказаться от использования библиотеки, то нужно включить в конечную программу управления потенциометром код управления данными, которые считываются с микросхемы MCP3002, подключенной к Raspberry Pi через интерфейс SPI. Научившись это делать, вы поймете, как работает внешняя подключаемая библиотека (встроенный программный код не требует подключения библиотеки, но выглядит громоздко и сложнее для понимания).

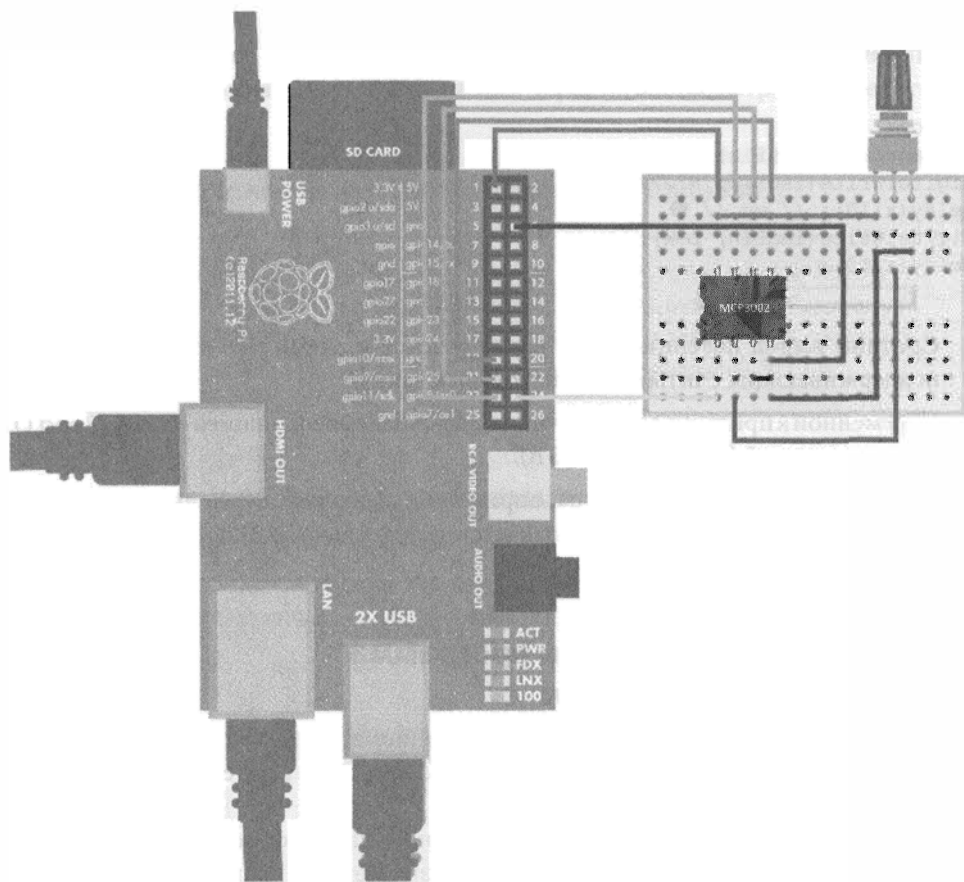


Рис. 5.11. Схема подключения потенциометра к Raspberry Pi (см. цветную вклейку)

В остальных примерах главы применяется программная конструкция с подключаемой библиотекой, что делает код компактнее.

Листинг 5.6. `pot.py`

```
# pot.py - подключение потенциометра через АЦП mcp3002
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import spidev # Используется в библиотеке botbook_mcp3002.py # ❶
import time

def readPotentiometer():
    spi = spidev.SpiDev() # ❷
    spi.open(0, 0) # ❸
    command = [1, 128, 0] # ❹
    reply = spi.xfer2(command) # ❺
    # Извлечение необходимых 10 битов из 24 битов ответа
    data = reply[1] & 31 # ❻
    data = data << 6 # ❼
    data = data + (reply[2] >> 2) # ❽
```

```

spi.close() # ❹
return data

def main():
    while True:
        potentiometer = readPotentiometer() # ❺
        print("Положение потенциометра: %i " % potentiometer)
        time.sleep(0.5)

if __name__ == "__main__":
    main()

```

- ❶ Библиотека `spidev` призвана упростить управление интерфейсом SPI.
- ❷ Создание нового объекта `Spidev` и сохранение его в переменной `spi`.
- ❸ Открытие первого канала для первого подключенного устройства.
- ❹ Первый канал соответствует 128, а второй — 128+64.
- ❺ Передача одного байта и получение ответа.
- ❻ Теперь нам нужно проанализировать ответ и выделить из 24 битов необходимые нам 10 битов. Формат ответа нам известен из описания микросхемы MCP3002: берем второй байт (1) из ответа и выполняем операцию побитового "И" над ним и числом 31 (в битовом виде представляется как 00011111, также часто записывается как 0b 0001 1111). В результате выполнения этой операции переменная `data` будет содержать пять младших битов значения `reply[1]`. Детально побитовые операции рассматриваются в главе 8.
- ❼ Сдвиг на шесть битов влево и присвоение полученного результата переменной `data`. Например, в результате такой операции над 0b 0001 1001 вы получите 0b 0110 0100 0000 (пробелы между битами добавлены для того, чтобы упростить анализ длинных битовых чисел). Шесть младших битов теперь представлены нулями.
- ❸ Заполнение шести младших битов третьим байтом данных, `data[2]`.
- ❹ Освобождение шины SPI.
- ❺ В основной программе выполняемые в представленном выше коде операции в явном виде не отображаются. Для получения необходимых данных в ней вызывается всего одна функция — `readPotentiometer()`.

Выглядит сложно? Не волнуйтесь! В остальных примерах книги мы будем использовать программный код, в котором для выполнения побитовых операций задействуется библиотека `botbook_mcp3002`, поэтому вы сможете сконцентрироваться на общих операциях, а побитовые расчеты пусть проводятся уже созданной ранее специальной программой.

Эксперимент: касание без прикосновения (емкостный датчик прикосновения QT113)

Главная особенность емкостных датчиков прикосновения заключается в том, что они не реагируют на само прикосновение. На самом деле они измеряют время накопления проводником заряда (емкость) при приближении к нему руки. Поскольку все мы состоим большей частью из воды, то расположение руки возле проводника усложняет (затягивает) накопление на нем заряда.



Рис. 5.12. Емкостный датчик прикосновения QT113, представленный отдельной микросхемой

Для распознавания прикосновения используется интегральная микросхема QT113. Принцип ее действия очень прост: при распознавании прикосновения на выход подается сигнал низкого уровня (LOW).

Хорошие емкостные датчики прикосновения требуют использования поверхностей определенного типа и площади. В нашем примере мы воспользуемся стальной проволокой, свернутой в спираль. Кусок алюминиевой проволоки также будет работать. В нашей электрической цепи для накопления заряда применяется конденсатор емкостью 10–500 нФ.

Полный датчик прикосновения создается несколькими способами:

- кусок проволоки и обычный таймер;
- кусок проволоки и библиотека CapSense;
- специальная микросхема (например, QT113).

Вы можете и не знать, но подобные датчики встречаются в быту повсеместно. В вашем смартфоне сенсорный экран может работать по такому же принципу, как и простой емкостный датчик прикосновения.

Надежно заземлите датчик. В данном случае недостаточно подключить его к нулевому выводу аккумуляторной батареи. Например, при подключении такого датчика к системе Arduino, питание на которую подается с ноутбука, мы не добились устойчивой работы, пока не подключили ноутбук к электрической сети (в розетку с центральным заземлением).

Подключение к Arduino и программа управления датчиком прикосновения QT113

На рис. 5.13 показана схема подключения микросхемы QT113 и самодельного датчика прикосновения к Arduino. Соберите воедино все компоненты и выполните программный код из листинга 5.7.

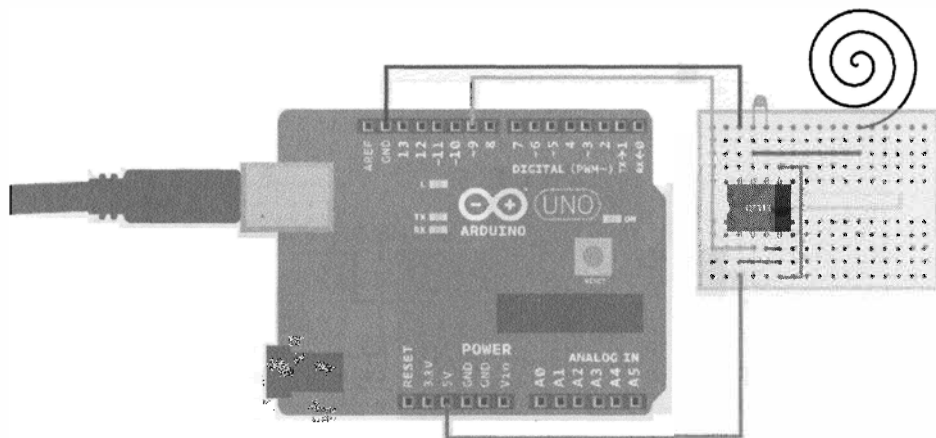


Рис. 5.13. Схема подключения датчика прикосновения QT113 к Arduino

Листинг 5.7. qt113.ino

```
// qt113.ino - управление датчиком прикосновения QT113
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
int sensorPin = 9;

void setup()
{
    pinMode(sensorPin, INPUT);
    Serial.begin(115200);
}

void loop()
{
    int touch = digitalRead(sensorPin); // ❶
    if(touch == LOW) {
        Serial.println("Прикосновение");
    } else {
        Serial.println("Без прикосновения");
    }
    delay(100);
}
```

- ❶ Простейший датчик, срабатывающий как переключатель при возникновении события.

Подключение к Raspberry Pi и программа управления датчиком прикосновения QT113

На рис. 5.14 показана схема подключения датчика прикосновения, созданного на базе микросхемы QT113, к Raspberry Pi. Постройте соответствующую электрическую цепь и выполните программу, представленную в листинге 5.8.

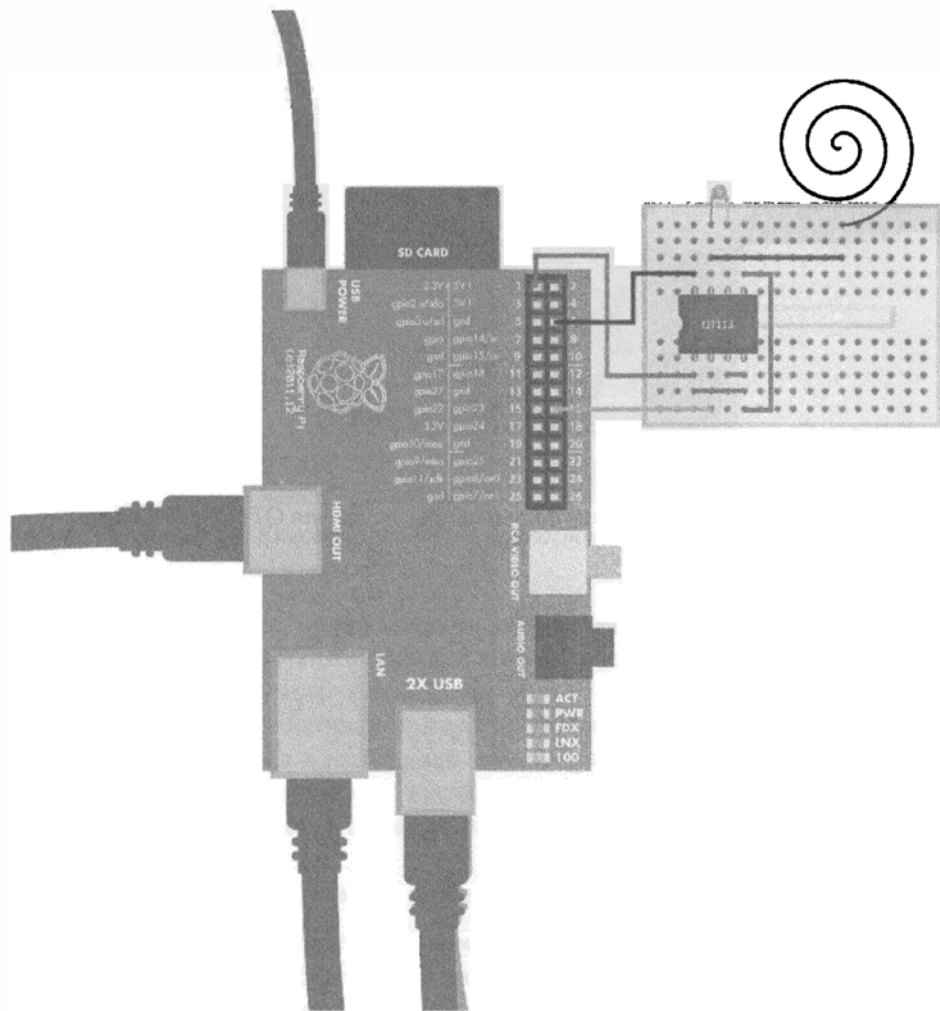


Рис. 5.14. Схема подключения датчика прикосновения QT113 к Raspberry Pi

Листинг 5.8. qt113.py

```
# qt113.py - считывание сигнала с микросхемы QT113
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio
```



```
def main():
    limitPin = 23
    gpio.setMode(limitPin, "in")
    while True:
        if gpio.read(limitPin) == gpio.LOW: # ❶
            print("Касание!")
            time.sleep(0.5)

if __name__ == "__main__":
    main()
```

- ❶ Простейший датчик, срабатывающий как переключатель при возникновении события.

Эксперимент в окружающей среде: распознавание прикосновения через дерево

При правильной подготовке и настройке оборудования емкостный датчик прикосновения может распознать приближение руки даже через твердые предметы. Результат для многих невероятный, поскольку даже толстый кусок дерева может превратиться в сенсорную поверхность (рис. 5.15). Таким способом мы создали книжную полку, прикосновение к которой приводит к изменению цвета ее подсветки.

Как вы понимаете, само дерево вряд ли участвует в распознавании прикосновения. Все обязанности выполняет емкостный датчик прикосновения; ситуация сходна с приближением руки к датчику на открытом воздухе, только теперь между рукой и датчиком располагается дерево.

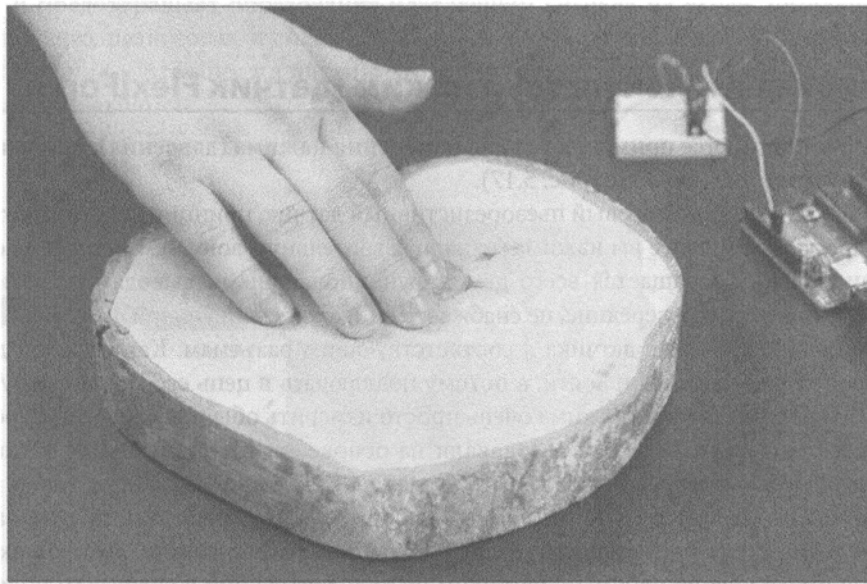


Рис. 5.15. Деревянная сенсорная поверхность: почему бы и нет?

Попробуйте создать прототип устройства, обеспечивающего описанный выше способ подключения датчика прикосновения, для чего воспользуйтесь схемой подключения датчика QT113 и программным кодом, рассмотренными ранее. Немного попрактиковавшись, вы заметите, что для получения отклика от датчика вам нужно усилить его действие по распознаванию изменений в электромагнитном поле. В общем случае для этого требуется использовать более длинный кусок проволоки или другого проводника. Создание большего количества витков или применение алюминиевой фольги также способно помочь в данной ситуации (рис. 5.16).

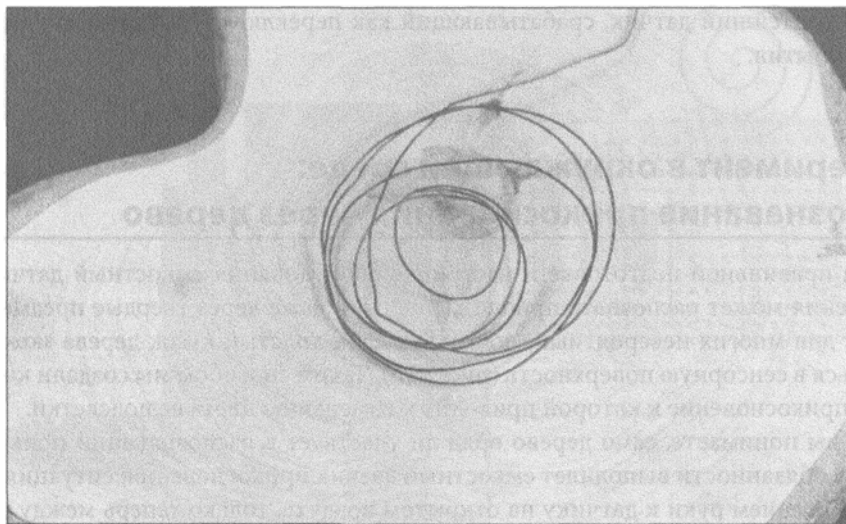


Рис. 5.16. Спираль из проволоки, которую нужно скрыть от любопытных глаз

Эксперимент: почувствуй нажим (датчик FlexiForce)

Датчики FlexiForce применяются для измерения нажима (давления), приложенного к их чувствительной зоне (рис. 5.17).

FlexiForce — это аналоговый пьезорезистивный датчик, принцип действия которого очень прост: чем сильнее вы нажимаете на чувствительную зону, тем меньше ее сопротивление. Датчик оснащается всего двумя функциональными выводами. Третий вывод, расположенный посередине, не снабжается никакой специальной функцией, кроме упрощения подключения датчика к соответствующим разъемам. Как и любой другой резистор, он не имеет полярности, а потому подключать в цепь его можно как угодно. Рабочий диапазон датчика нажима очень просто измерить обычным мультиметром.

Наши студенты шутки ради создавали на основе датчиков FlexiForce сигнализации, которые включались, когда кто-то (угадайте, кто?) сел на стул. Еще с их помощью можно создать простой силомер для спортивных соревнований, основанный на измерении степени сжатия его руками. Охранная сигнализация, которая включается при прохождении злоумышленником по чувствительной к нагрузкам поверхности, также легко реализуется с помощью технологии FlexiForce.

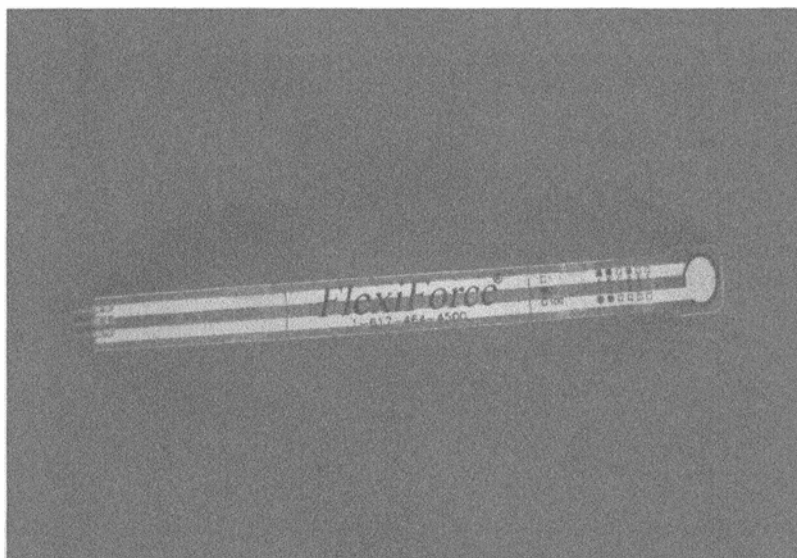


Рис. 5.17. Датчик *FlexiForce*

Подключение к Arduino и программа управления датчиком FlexiForce

Как уже упоминалось выше, FlexiForce — это аналоговый пьезорезистивный датчик. В Arduino встроен аналого-цифровой преобразователь, поэтому считывание сигналов с такого датчика выполняется привычной уже функцией `analogRead()`.

Включенный в схему резистор номиналом 1 МОм используется как подтягивающий и предотвращает образование плавающего сигнала на входе. Детально о подтягивающих резисторах и сигналах см. в начале главы. На рис. 5.18 показана схема подключения к Arduino, а в листинге 5.9 представлен программный код управления датчиком FlexiForce.

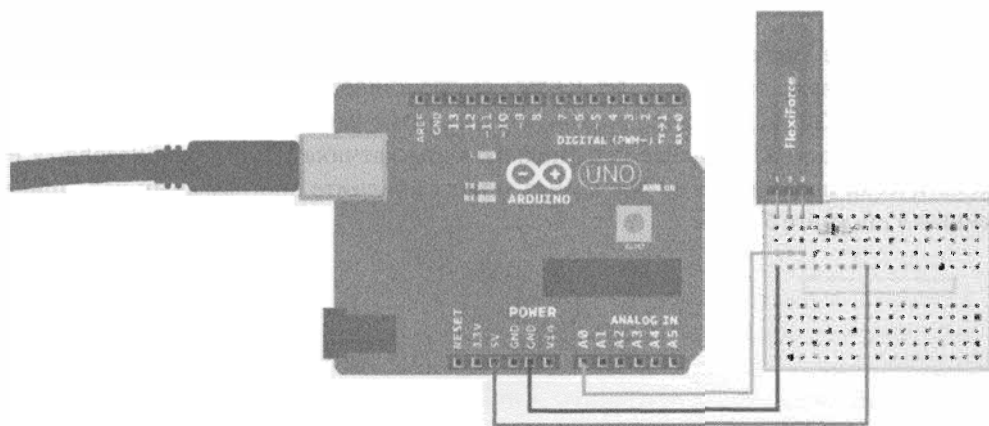


Рис. 5.18. Схема подключения датчика *FlexiForce* к Arduino

Листинг 5.9. flexiforce_25.ino

```
// flexiforce_25.ino - вывод степени сжатия на монитор
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int squeezePin=A0; // ❶
int x=-1; // 0..1023

void setup() {
  pinMode(squeezePin, INPUT);
  Serial.begin(115200); // бит/с
}

void loop() {
  x=analogRead(squeezePin); // ❷
  Serial.println(x);
  delay(500); // мс
}
```

- ❶ Предопределенные в Arduino константы (A0, A1, A2...) лучше всего использовать для управления аналоговыми входами в Arduino.
- ❷ Как и для любого другого аналогового резистивного датчика, сигнал на выходе с датчика считывается функцией `analogRead()`, которая возвращает значение напряжения, представленное диапазоном от 0 (0 В) до 1023 (+5 В).

Подключение к Raspberry Pi и программа управления датчиком FlexiForce

Подключение датчика FlexiForce к Raspberry Pi выполняется подобно тому, как это реализовано в схемах с аналоговыми резистивными датчиками любых типов (рис. 5.19). Поскольку Raspberry Pi не оснащается встроенным аналого-цифровым преобразователем, в качестве АЦП воспользуемся внешне подключаемой микросхемой MCP3002. А так как датчик FlexiForce имеет всего два функциональных вывода, то во избежание получения неопределенного сигнала включим в цепь подтягивающий резистор.

Не забудьте также, что программное управление внешним аналого-цифровым преобразователем, таким как MCP3002, осуществляется с помощью библиотеки `botbook_mcp3002`, делающей основную программу, из которой вызывается библиотека, очень простой. Создайте цепь подключения датчика FlexiForce к Raspberry Pi, схематически показанную на рис. 5.19, а затем выполните программный код, приведенный в листинге 5.10.

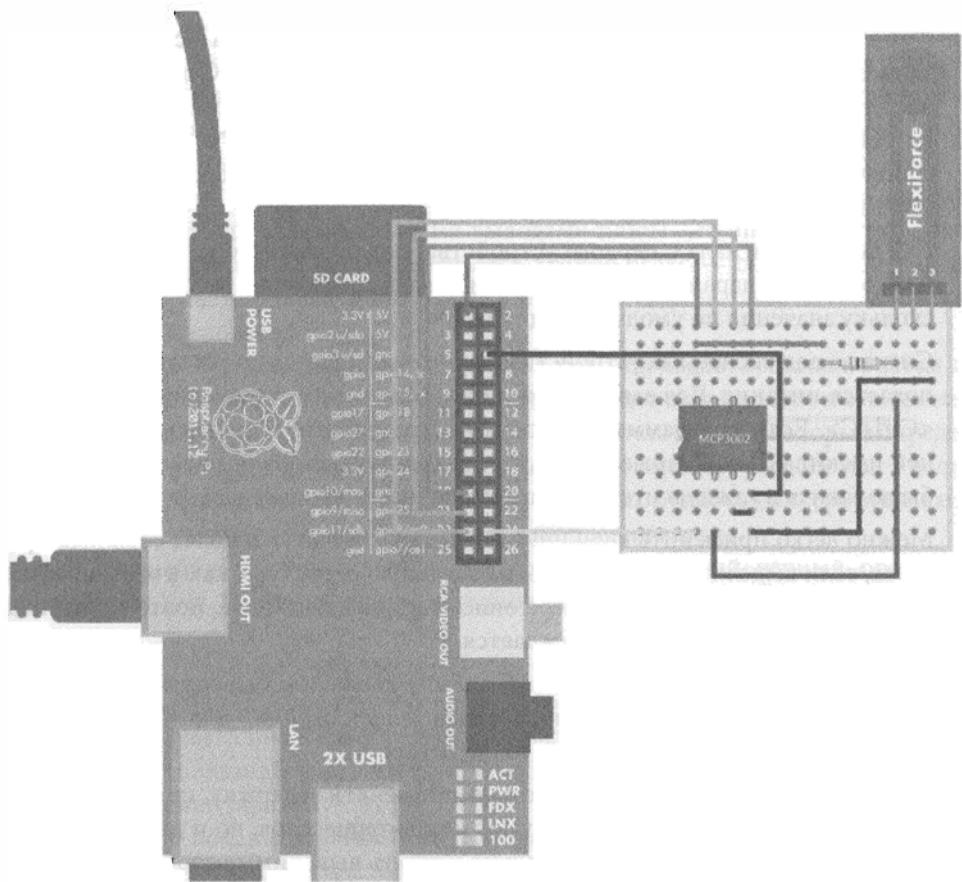


Рис. 5.19. Схема подключения датчика FlexiForce к Raspberry Pi (см. цветную вклейку)

Листинг 5.10. flexi force .py

flexiforce.py - измерение давления и вывод на монитор
(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_mcp3002 as mcp # ❶

def readFlexiForce():
    return mcp.readAnalog() # ❷

def main():
    while True: # ❸
        f = readFlexiForce() # ❹
        print("Сила нажима: %i " % f) # ❺
        time.sleep(0.5) # c # ❻

if __name__ == "__main__":
    main()
```

- ❶ Файл библиотеки `botbook_mcp3002.py` должен располагаться в том же каталоге, что и файл основной программы `flexiforce.py`. Кроме того, в этом же каталоге требуется разместить библиотеку `spidev`, которая вызывается из библиотеки `botbook_mcp3002`. Детально об этом см. в комментариях к файлу `botbook_mcp3002.py` или в главе 3 при описании составного датчика ИК-излучения.
- ❷ Считывание сигнала с первого устройства, подключенного к MCP3002. В подобном случае параметры устройства и канала специально не указываются, поскольку значения по умолчанию для них — 0.
- ❸ Сигнал с подключаемых устройств обычно считывается до тех пор, пока на них подается питание. Для завершения программы нажмите комбинацию клавиш `<Ctrl+C>`. Если программа настроена на продолжительное выполнение или в ней имеются бесконечные циклы, то воспользуйтесь функцией `delay()`, чтобы добавить в каждую итерацию цикла небольшую задержку.
- ❹ Можно легко применить функцию `readFlexiForce()` в более сложных проектах, если выделить ее операторы в отдельную структуру, как в этом проекте. Название функции в полной мере описывает ее назначение, поэтому в дополнительных комментариях не нуждается.
- ❺ Выведите полученную от датчика величину давления. Выводимое сообщение формируется с использованием форматирующей строки, в которой значение переменной `f` подставляется вместо оператора `%i`.
- ❻ В длинных циклах нужно обязательно применять задержку, не только во избежание постоянной полной загрузки микропроцессора, но и для того, чтобы позволить пользователям внимательно изучить выводимые сообщения.

Эксперимент: создание собственного датчика прикосновения

Если емкостные датчики прикосновения всего лишь измеряют время, которое требуется для накопления электрического заряда до определенного уровня, то почему бы не создать такой датчик самому вместо того, чтобы использовать отдельную микросхему QT113? Ничего сложного в этом нет, и для построения такого датчика вам понадобятся алюминиевая фольга, резистор и устройство надежного заземления платы Arduino.

Для правильной реализации проекта вам также понадобятся усидчивость и опыт в выполнении подобных работ. Несмотря на то что принцип работы емкостных датчиков прикосновения очень простой, их создание требует определенной сноровки. В дополнение к хорошему заземлению вам потребуется на этапе вычислений научиться усреднять полученные результаты, чтобы исключить сильные отклонения в получаемых зависимостях.

Чтобы обеспечить полное заземление платы, питание на Arduino нужно подать от розетки (с помощью зарядного устройства с USB-разъемом). При использовании ноутбука для подачи питания на Arduino достаточно подключить к розетке сам ноутбук. В случае применения для “запитывания” Arduino настольной системы вообще беспокоиться не о чем, поскольку системный блок всегда подключен к розетке.

Измерительная часть устройства, алюминиевая проволока или фольга, подключается в цепь между двумя сигнальными выводами платы Arduino. Такой тип подключения оборудования в Arduino встречается крайне редко — обычно к сигнальному выводу подключается только один контакт внешнего устройства, а на его второй контакт подается питание +5 В.

Принцип работы датчика такой: на один из выводов подается сигнал, а затем замедляется время, в течение которого на второй вывод будет подано такое же напряжение, как и на первый. В данном случае скорость образования заряда (который определяет напряжение на втором выводе) зависит от конечной емкости устройства, а большие объекты, расположенные возле проводника, такие как руки, только увеличивают емкость, тем самым увеличивая общее время накопления заряда.

Резистор небольшого номинала, около 10 кОм (коричневый–черный–оранжевый), применяется для предотвращения влияния на устройство статического электричества.

Резистор с большим сопротивлением, от 1 до 50 Мом, определяет чувствительность датчика. Чем больше сопротивление этого резистора, тем на большем расстоянии датчик будет способен определить присутствие человека. Учтите, что чем на большем расстоянии датчик сохраняет чувствительность, тем больше времени уходит на накопление заряда.

На рис. 5.20 показан конечный вид электрической цепи подключения собственного датчика к Arduino, а в листинге 5.11 приведен программный код управления им. Подключите все компоненты устройства к Arduino, как показано на рисунке, а затем выполните программный код.

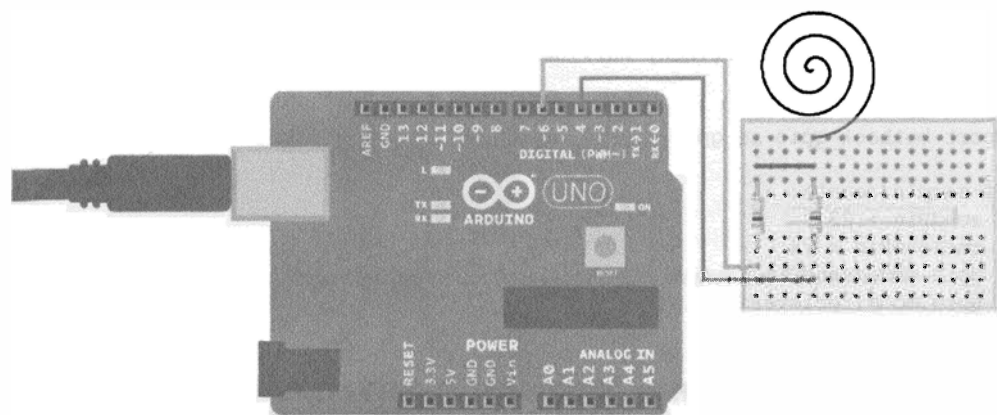


Рис. 5.20. Схема подключения емкостного датчика к Arduino

Листинг 5.11. `diy_capacitive_sensor.ino`

```
// diy_capacitive_sensor.ino - распознавание руки
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sendPin = 4;
const int readPin = 6;

void setup() {
    Serial.begin(115200);
    pinMode(sendPin, OUTPUT);
    pinMode(readPin, INPUT);
    digitalWrite(readPin, LOW);
}

void loop() {
    int time = 0;

    digitalWrite(sendPin, HIGH);
    while(digitalRead(readPin) == LOW) time++;
    Serial.println(time);
    digitalWrite(sendPin, LOW);
    delay(100);
}
```

Подключение к Raspberry Pi и программа управления собственным датчиком прикосновения

На рис. 5.21 показана схема подключения собственного датчика прикосновения (в данном случае присутствия) к Raspberry Pi. Соберите согласно ей электрическую цепь и выполните программный код, представленный в листинге 5.12.

Листинг 5.12. `diy_capacity_sensor_simple.py`

```
# diy_capacity_sensor_simple.py - считывание данных с датчика
# присутствия
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio

def sample(count):
    sendPin = 23
    recievePin = 24
    gpio.mode(sendPin, "out")
    gpio.mode(recievePin, "in")
    gpio.write(sendPin, 0)
    total = 0
    # Установка сигнала низкого уровня
    for x in xrange(1, count):
        time.sleep(0.01)
        gpio.write(sendPin, gpio.HIGH)
        while(gpio.read(recievePin) == False):
            total += 1
```



```

gpio.write(sendPin,gpio.LOW)
return total

def main():
    while True:
        touch = sample(30)
        print("Прикосновение: %d" % touch)
        time.sleep(0.5)

if __name__ == "__main__":
    main()

```

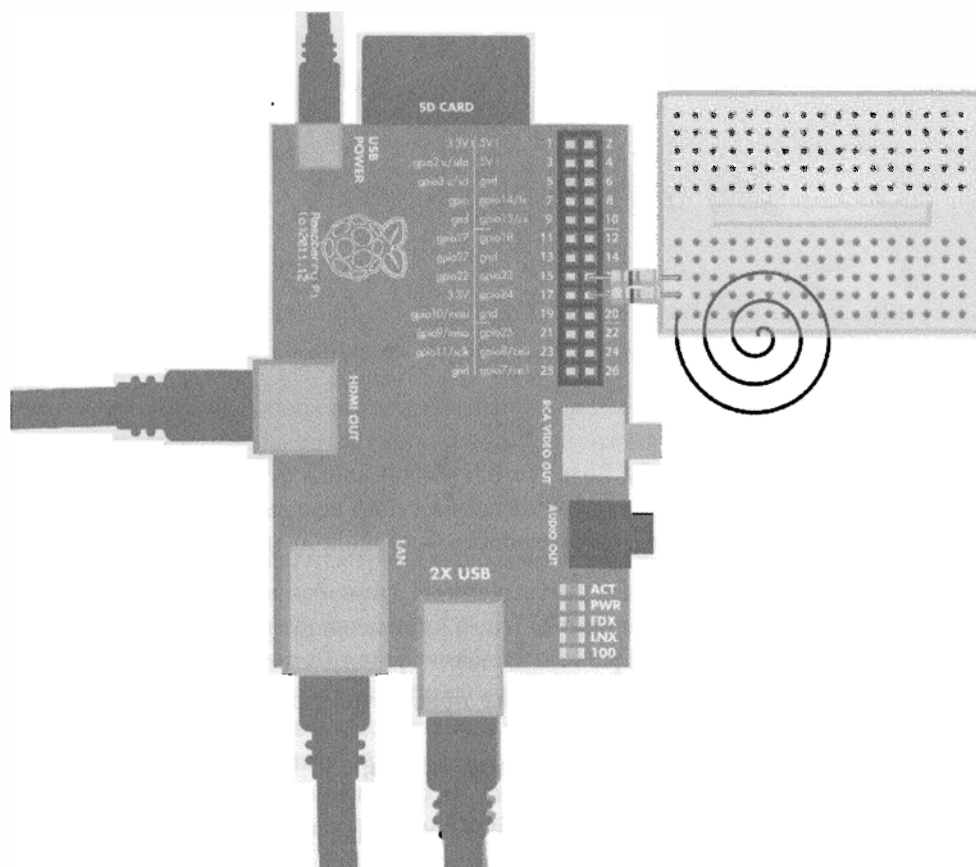


Рис. 5.21. Схема подключения датчика прикосновения к Raspberry Pi

Пилотный проект: сенсорный звонок

Вы заходите в пустующий холл небольшого отеля. Вокруг ни души. Вы подходите к стойке и тянете руку к звонку вызова администратора (рис. 5.22), но звонок начинает звенеть до того, как вы нажимаете кнопку!



Рис. 5.22. Звонок вызова администратора отеля

В этом проекте мы объединим емкостный датчик прикосновения и старый добрый отельный звонок. В результате получим звонок, который начинает звенеть до того, как вы нажмете его кнопку. Более того, кнопка звонка будет нажиматься самостоятельно, создавая в помещении ауру мистичности. А еще вы научитесь управлять новым типом чрезвычайно полезных устройств — сервоприводом.

Получаемые навыки

В этом проекте вы научитесь:

- создавать гаджеты, реагирующие на присутствие руки до прикосновения к ним;
- перемещать объекты с помощью сервоприводов;
- управлять сервоприводами;
- маскировать необычные устройства под обыденные, ничем не примечательные объекты.

Сервоприводы

Сервопривод, или серводвигатель, — это тип двигателей, вращением вала которых можно управлять очень точно (рис. 5.23). Вал сервопривода можно повернуть на точно заданный угол, например 90 градусов. Существуют, правда, и полнооборотные сервоприводы, в которых регулируется только направление и скорость вращения.

Лучше всего начать знакомство с двигателями, используемыми в проектах на базе микроконтроллеров, именно с сервоприводов. Они функциональнее обычных

коллекторных электромоторов, в которых даже изменение направления вращения требует использования дополнительного оборудования. В проектах на базе Arduino и Raspberry Pi большая часть задач по перемещению объектов реализуется с помощью именно сервоприводов.

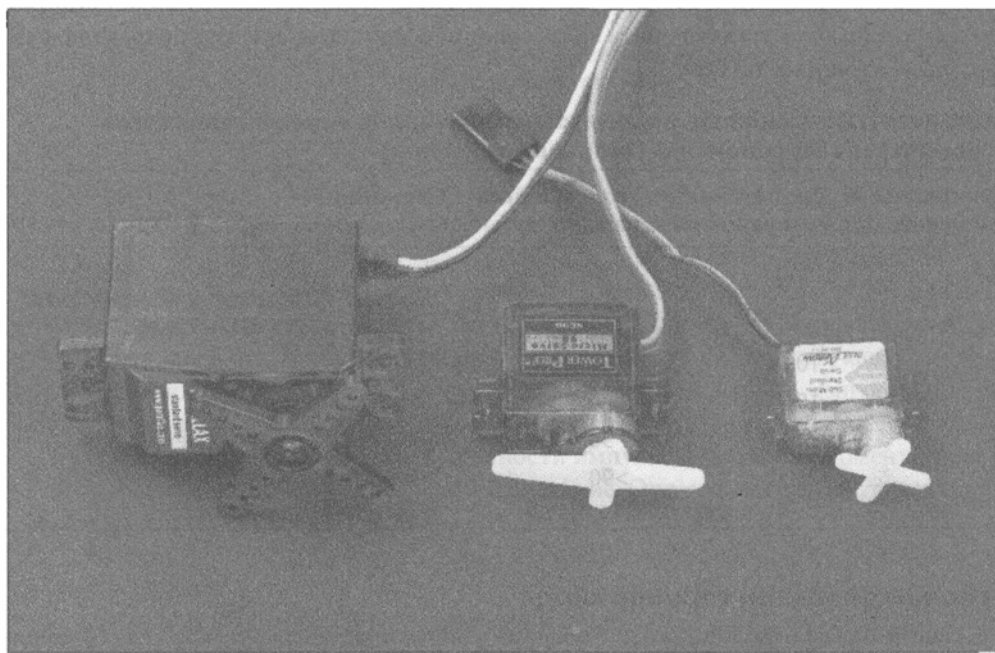


Рис. 5.23. Сервоприводы

Сервоприводы оснащаются тремя выводами: черный (0 В, “земля”), красный (+5 В) и сигнальный (желтый или белый). На сигнальный вывод сервопривода, подключенного к Arduino, подается управляющий импульсный сигнал. Длительность импульсов определяет угол, на который поворачивается вал сервопривода. После поворота на указанный угол вал сервопривода остается в таком положении в течение всего времени подачи на сигнальный вывод исходного управляющего импульса.

Прямоугольный импульсный сигнал очень просто сгенерировать, если часто изменять выходной сигнал с низкого (LOW) на высокий (HIGH) уровень и наоборот. Для управления сервоприводами в Arduino необходимо обеспечить подачу такого сигнала с частотой 50 раз в секунду. Другими словами, частота прямоугольного импульсного сигнала должна составлять 50 Гц:

$$50/\text{с} = 50 \times 1/\text{с} = 50 \text{ Гц}$$

Сами импульсы имеют длительность 1-2 мс. Чем длиннее импульс, тем на больший угол поворачивается вал сервопривода. Центрирование вала двигателя происходит при подаче импульса средней длительности (среднее значение между минимальным и максимальным). Чаще всего центрирование осуществляется при подаче импульсов длительностью 1,5 мс.

Вы не поверите, но для большинства сервоприводов техническая документация просто не выпускается. При этом частота управляющего сигнала (количество импульсов в секунду) у них обычно одинаковая, иногда с небольшими отличиями.

Длительность же управляющих импульсов у каждого сервопривода своя, но ее очень просто определить экспериментально. Чтобы понять, в каком виде представляется зависимость угла поворота вала сервопривода от длительности управляющего импульса, взгляните на табл. 5.1.

Таблица 5.1. Длительность управляющих импульсов, определяющих угол поворота вала сервопривода (типичные значения)

Длительность импульса, мс	Длительность импульса, мкс	Угол поворота вала, градусы	Описание
0,5	500	<-90	Попытка повернуть вал на угол вне рабочего диапазона; неприятный звук передаточного механизма
1	1000	-90	Крайнее левое положение
1,5	1500	0	По центру
2	2000	90	Крайнее правое положение
2,5	2500	>90	Вне рабочего диапазона; неприятный звук

Рабочий диапазон сервопривода

Даже если вам известно, что длительность управляющих импульсов для сервопривода задается в диапазоне от 1 до 2 мс, то как определить точные значения? Можно сделать это экспериментально. Приведенный ниже код можно считать отправной точкой для любого проекта на базе Arduino или Raspberry Pi, в котором задействованы сервоприводы.

За годы конструирования новых устройств мы выработали устойчивую привычку самостоятельно определять рабочие характеристики любых приобретаемых сервоприводов. Теоретически технические характеристики, особенно такие ключевые, как длительность управляющего импульса и углы поворота вала, должны указываться на сайте производителя. В действительности большинство сервоприводов не сопровождается технической документацией вообще, или ее очень сложно достать.

Мы тестируем чувствительность сервоприводов к изменению длительности импульса, начиная с очень малых значений и заканчивая заведомо большими (листинг 5.13). Указанная программа выводит длительности подаваемых импульсов на экран, мы замечаем, при каких значениях и на какой угол поворачивается вал сервопривода, и представляем конечный результат в виде таблицы (пример приведен в табл. 5.2). Схема подключения сервопривода к Arduino показана на рис. 5.24.

Выполните программу тестирования сервопривода. Отобразите на экране монитор последовательного порта в среде разработки Arduino (команда Tools⇒Serial Monitor (Сервис⇒Монитор порта)), чтобы ознакомиться с выводимыми значениями. Настройте одинаковую скорость передачи данных в мониторе и программном коде (в бодах или бит/с).

Таблица 5.2. Ключевые значения длительности управляющих импульсов сервопривода

Длительность импульса, мс	Угол, градусы	Описание
	-90	Крайнее левое положение (определяется программно)
	0	По центру, среднее между крайним левым и крайним правым положениями
	90	Крайнее правое положение (определяется программно)

Исходно управляющий импульс настолько короткий, что гарантированно выпадает из рабочего диапазона сервопривода. По мере его увеличения вал все еще не вращается, но уже начинает немного “подергиваться”, а передаточный механизм издает неприятный звук. Если звук непродолжительный, то это означает, что ничего страшного с сервоприводом не происходит (так вы его точно не сломаете). Как только вы увидите, что вал двигателя провернулся, запишите или запомните соответствующее значение, представленное на мониторе. Вы только что нашли длительность импульса, определяющего поворот вала сервопривода на угол -90 градусов (крайнее левое положение).

При дальнейшем увеличении длительности импульса в какой-то момент вал перестает поворачиваться — запомните и это значение. Оно соответствует углу поворота +90 градусов или крайнему правому положению.

Положение по центру соответствует длительности импульса, рассчитанной как среднее значение импульсов, полученных для предыдущих двух положений: крайнего левого и крайнего правого. Например, если крайние положения соответствуют длительности импульсов 1 и 2 мс, то среднее этих двух значений — 1,5 мс — и будет соответствовать длительности импульса центрирования вала сервопривода.

В некоторые сервоприводы встраиваются потенциометры для настройки центрального положения вала. Если у вас такой сервопривод, то поэкспериментируйте над установкой положения регулятора потенциометра и определением длительности импульса для центрирования вала сервопривода.

Листинг 5.13. `servo_range.ino`

```
// servo_range.ino - вывод длительности управляющего импульса
// для сервопривода
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int servoPin=2;

void pulseServo(int servoPin, int pulseLenUs) // ❶
{
    digitalWrite(servoPin, HIGH); // ❷
    delayMicroseconds(pulseLenUs); // ❸
    digitalWrite(servoPin, LOW); // ❹
    delay(15); // ❺
}

void setup()
{
    pinMode(servoPin, OUTPUT);
```

```

Serial.begin(115200);
}

void loop()
{
  for (int i=500; i<=3000; i=i+2) { // ❹
    pulseServo(servoPin, i); // ❶
    Serial.println(i);
  }
}

```

- ❶ Отправка одного короткого импульса осуществляется с помощью функции `pulseServo()`. Функция вызывается не в месте определения, а далее в программе. В качестве параметра в функцию передается номер вывода управления сервоприводом, что позволяет одновременно управлять несколькими сервоприводами, для чего функция вызывается многократно, и каждый раз ей передаются новые значения.
- ❷ Исходно на управляющий вывод подается сигнал низкого уровня (LOW), обозначающий отсутствие сигнала. Подача импульса отсчитывается от момента установки вывода в значение HIGH.
- ❸ Небольшая задержка. Как видно по названию переменной, задержка указывается в микросекундах (мкс), миллионных долях секунды. Стандартные значения см. в табл. 5.1. Назначение текущего программного кода — определение точных значений длительности импульса.
- ❹ Подача на вывод значения LOW, конец импульса.
- ❺ Небольшая задержка. Для нас 15 мс ничего не значит, а это в десять раз больше, чем длительность импульса. Как вы помните, импульсы должны подаваться 50 раз в секунду, а потому каждый следующий импульс должен отправляться не позже каждых $1/50 \text{ с} = 0,02 \text{ с} = 20 \text{ мс}$. Мы выбрали несколько меньшее значение.
- ❻ Выполнение тела цикла; в каждой последующей итерации длительность импульса несколько увеличивается. Начинается цикл с заведомо небольшого значения, 500 мкс (0,5 мс), а заканчивается чрезвычайно большим — 3000 мкс (3 мс). Если вы забыли, как выполняются операторы цикла, прочитайте врезку “До каких пор?”.
- ❼ Подача короткого импульса на сервопривод. Поскольку задержка добавлена в функцию `pulseServo()`, дополнительно устанавливать ее в основном цикле нет смысла.

До каких пор?

Циклические конструкции `for` простыми словами можно описать так: “выполнять следующие операции указанное количество итераций”. Синтаксис такого цикла простой.

```
for (инициализация; условие; постоперация) {
    Тело цикла
}
```

Например:

```
for(int i=0; i<3; i++){
    Serial.print(i);
}
```

Инициализация проводится единожды, в самом начале выполнения цикла, и заключается в объявлении переменной цикла.

Перед началом выполнения тела цикла проверяется условие. Если условие ложно, то на этом выполнение цикла прекращается, и программа выходит из него. Если условие истинно, то выполняется тело цикла.

В самом конце, после операторов тела цикла, выполняется постоперация. Обычно она заключается в увеличении (уменьшении) счетчика цикла. Впоследствии начинается новая итерация цикла, и условие проверяется снова.

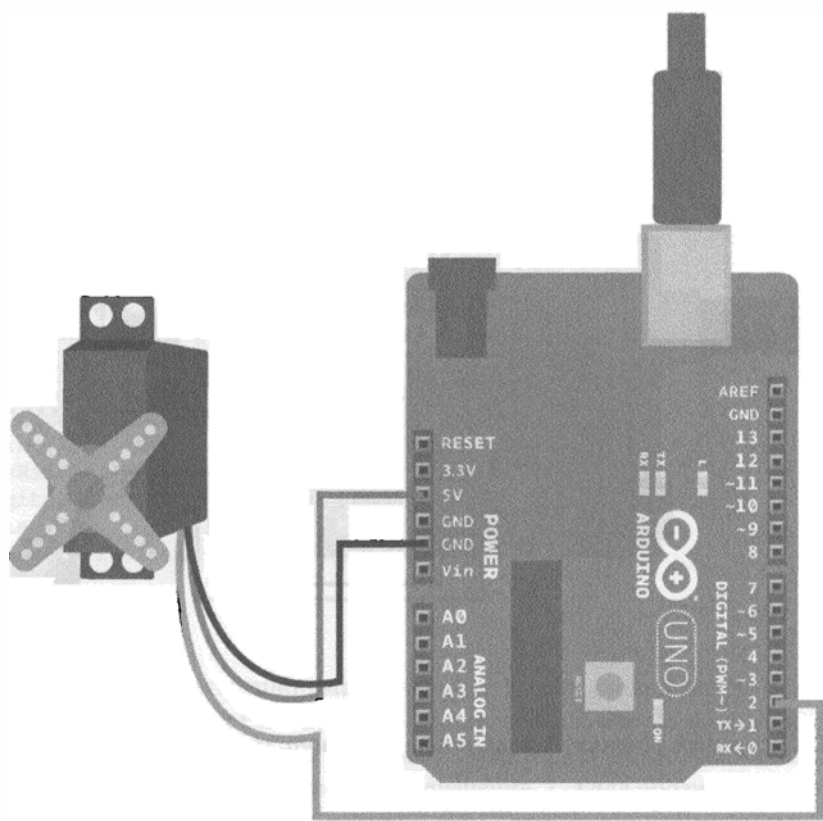


Рис. 5.24. Сервопривод подключен ко второму цифровому выходу Arduino, чтобы обеспечить совместимость с программой `servo_range.ino`

Для управления сервоприводами всегда можно воспользоваться встроенной в Arduino библиотекой `Servo.h`. Она поставляется вместе со средой разработки Arduino и предоставляет объектно-ориентированный интерфейс управления сервоприводами. С его помощью вы сможете напрямую указывать углы поворота вала сервопривода, например `myservo.write(180)`. Вы найдете ее удобной, если планируете использовать в одном проекте сразу несколько стандартных сервоприводов. Тем не менее подключение встроенной библиотеки `Servo.h` делает невозможным использование функции `analogWrite()` для передачи сигналов через выводы 9 и 10. Мы не приверженцы урезания функциональных возможностей среды разработки, поэтому никогда не пользуемся этой библиотекой в своих проектах.

Подключение к Arduino и программа управления сенсорным звонком

Подключите в одну цепь сервопривод и емкостный датчик прикосновения, как вы это уже делали в предыдущих примерах (рис. 5.25). Вам снова понадобится конденсатор емкостью 10–500 нФ, используемый совместно с микросхемой QT113. Лучше всего найти конденсатор емкостью 30 нФ.

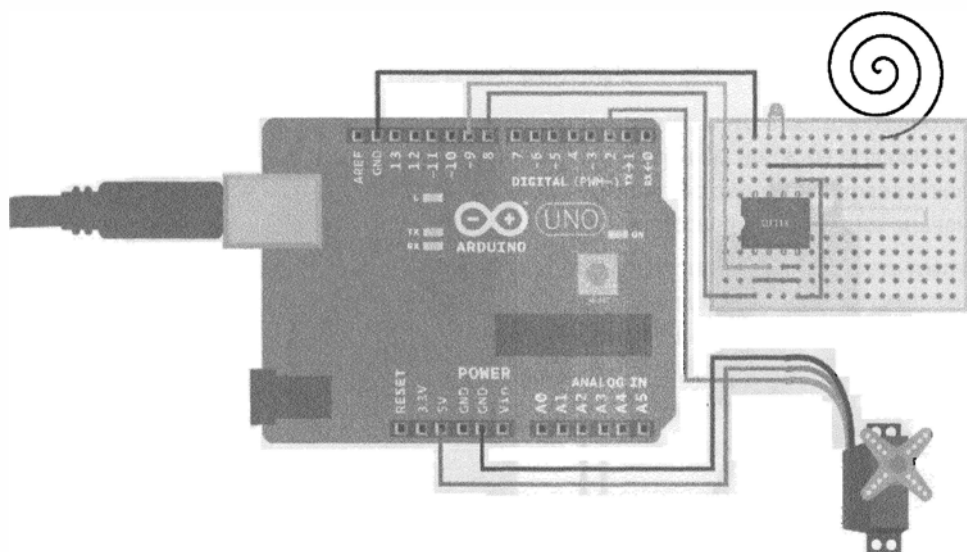


Рис. 5.25. Схема подключения компонентов отдельного звонка к Arduino (см. цветную вклейку)

Листинг 5.14. `haunted_bell.ino`

```
// haunted_bell.ino - включение звонка перед прикосновением
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```



```

int servoPin=2;
int sensorPin = 9;
int sensorPowerPin = 8;
int hasRang = 0; // ❶

void pulseServo(int servoPin, int pulseLenUs) // ❷
{
    digitalWrite(servoPin, HIGH);
    delayMicroseconds(pulseLenUs);
    digitalWrite(servoPin, LOW);
    delay(20);
}

void cling() // ❸
{
    for (int i=0; i<=3; i++) { // ❹
        pulseServo(servoPin, 2000);
    }
    for (int i=0; i<=100; i++) {
        pulseServo(servoPin, 1000);
    }
}

void setup()
{
    pinMode(servoPin, OUTPUT);
    pinMode(sensorPowerPin, OUTPUT);
    digitalWrite(sensorPowerPin,HIGH);
    pinMode(sensorPin, INPUT);
}

void loop()
{
    int touch = digitalRead(sensorPin); // ❺
    if(touch == HIGH) { // ❻
        hasRang = 0;
    }
    if(touch == LOW && hasRang == 0) { // ❼
        cling(); // ❶
        hasRang = 1; // ❷
        digitalWrite(sensorPowerPin,LOW);
        delay(1);
        digitalWrite(sensorPowerPin,HIGH);
    }
    delay(100);
}

```

- ❶ Переменная `hasRang` принимает значение 1, если звонок звенел без нажатия кнопки. Это позволяет избежать повторного звонка, когда он активизируется и при приближении руки, и нажатием кнопки. Для обозначения истинности события применяется целочисленное значение 1, а для обозначения ложности условия — целочисленное значение 0. Мы умышленно избежали использования булевой логики в программе, но вы всегда можете прибегнуть к ней в случае такой необходимости

- ⑦ Управление сервоприводом детально описано при рассмотрении листинга 5.13.
- ⑧ Функция однократного включения звонка. Эта операция выносится в отдельную функцию, чтобы упростить структуру основного цикла `loop()`.
- ⑨ Чтобы предоставить сервоприводу достаточно времени для включения звонка, нужно отправить ему несколько управляющих импульсов. Выполнение функции `pulseServo()` занимает приблизительно 20 мс, а 100 итераций занимает всего 2 с. Поэтому каждый звонок будет длиться около 2 секунд.
- ⑩ При срабатывании датчика переменной `SensorPin` присваивается значение `LOW`.
- ⑪ Если прикосновения нет (подается сигнал `HIGH`), то значение переменной `hasRang` сбрасывается. Звонок будет звенеть при последующем приближении к нему руки.
- ⑫ Только если рука находится у звонка и звонок еще не звенел...
- ⑬ ...он звонит один раз.
- ⑭ Если звонок уже звенел в текущем подходе, то нужно предотвратить его включение при повторном возникновении подходящих условий.

Подключение сервопривода к звонку

Используйте клей или термоклей для закрепления сервопривода в корпусе отельного звонка (рис. 5.26). Перед тем как склеивать компоненты в единое устройство, убедитесь, что рычаг сервопривода при повороте вала толкает механизм звонка так, что тот производит громкий звук.

Кроме того, движение рычажка сервопривода должно вызывать нажатие кнопки в верхней части звонка. У вас может уйти несколько попыток на правильное размещение сервопривода внутри звонка. Завершив установку сервопривода, приготовьтесь наблюдать реакцию ничего не подозревающих посетителей.

В этой главе вы познакомились с различными датчиками прикосновения и приближения, узнали об особенностях управления с помощью микроконтроллерных плат кнопками, микропереключателями и всевозможными емкостными датчиками. Когда вам надоест разыгрывать друзей реагирующим на приближение руки звонком, принимайтесь за реализацию собственных идей. Посмотрите вокруг: вас окружает огромное количество предметов и устройств, так неужели они не достойны получить сенсорное управление?

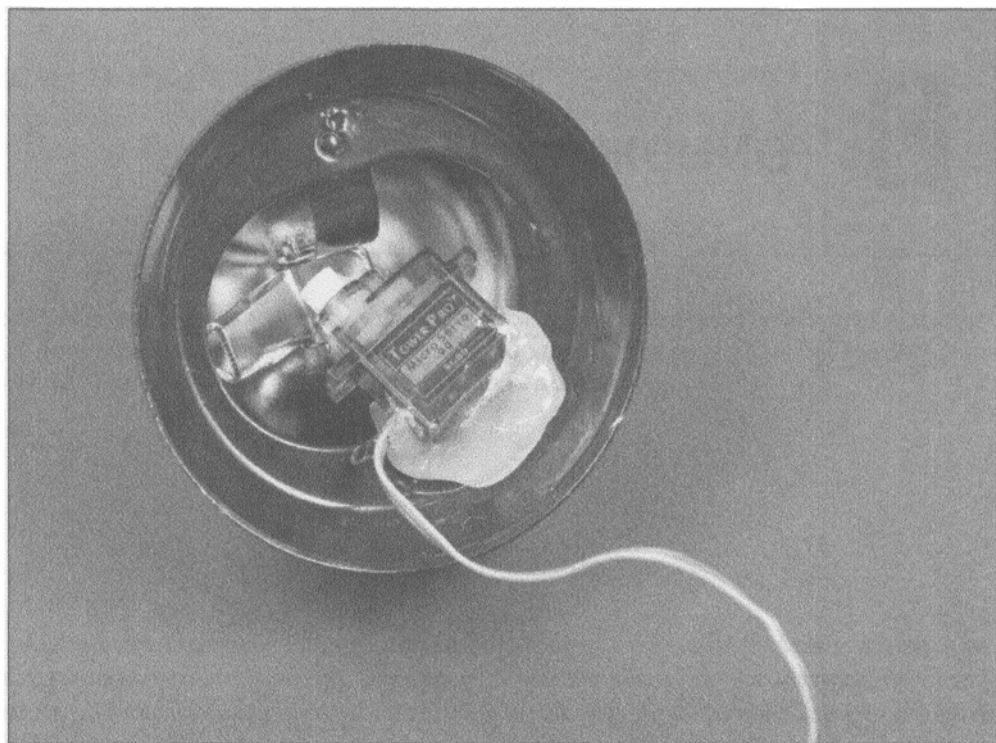


Рис. 5.26. Установка сервопривода в отельный звонок

Когда вы выходите на задний дворик своего дома, автоматически включается освещение. Пора немного передохнуть, взяв в руки любимую игровую консоль (мы предпочитаем Оиуа). Только уменьшите громкость, чтобы поберечь уши.

Автоматическое освещение в парках включается при обнаружении движущихся источников тепла, к которым относится и тело человека. Джойстики, без которых нельзя представить ни один игровой контроллер, преобразуют движение вашей руки в выходной сигнал, характеризующийся изменением сопротивления. Регулятор громкости — это самый обычный потенциометр, известный также как переменный резистор.

Эксперимент: где верх, а где низ (датчик наклона)?

Если вы решили собрать из имеющихся средств игровой автомат типа пинбола, то прежде следует научиться распознавать чрезмерно сильные удары и завершать ход игрока при недопустимом наклоне устройства. В последнем случае вам нужно будет совместить датчик наклона с уже применяемой ранее охранной сигнализацией.

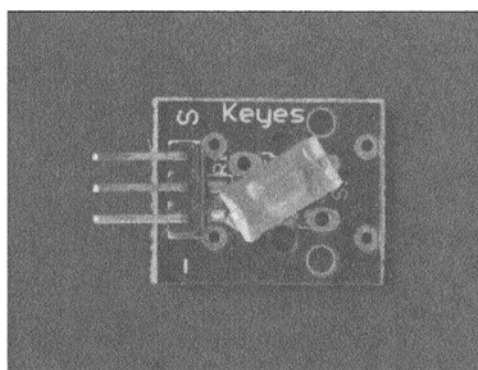


Рис. 6.1. Датчик наклона

Подключение к Arduino и программа управления датчиком наклона

На рис. 6.2 показана схема подключения датчика наклона к плате Arduino. Создайте соответствующую цепь и выполните программный код, приведенный в листинге 6.1.

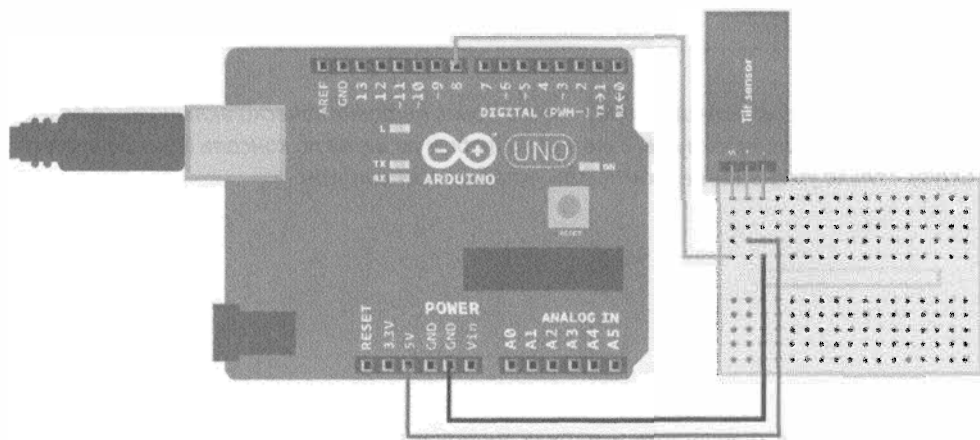


Рис. 6.2. Схема подключения датчика наклона к Arduino

Листинг 6.1. tilt_sensor.ino

```
// tilt_sensor.ino - определение наклона и вывод его значений  
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const int tiltPin = 8;  
int tilted = -1;  
  
void setup() {  
    Serial.begin(115200);  
    pinMode(tiltPin, INPUT);  
    digitalWrite(tiltPin, HIGH);  
}  
  
void loop() {  
    tilted = digitalRead(tiltPin); // ❶  
    if(tilted == 0) {  
        Serial.println("Датчик наклонен");  
    } else {  
        Serial.println("Датчик выровнен");  
    }  
  
    delay(100);  
}
```

- ❶ Простейший датчик, срабатывающий в определенный момент как переключатель.

Подключение к Raspberry Pi и программа управления датчиком наклона

На рис. 6.3 показана схема электрической цепи подключения датчика наклона к Raspberry Pi. После ее реализации выполните программный код из листинга 6.2.

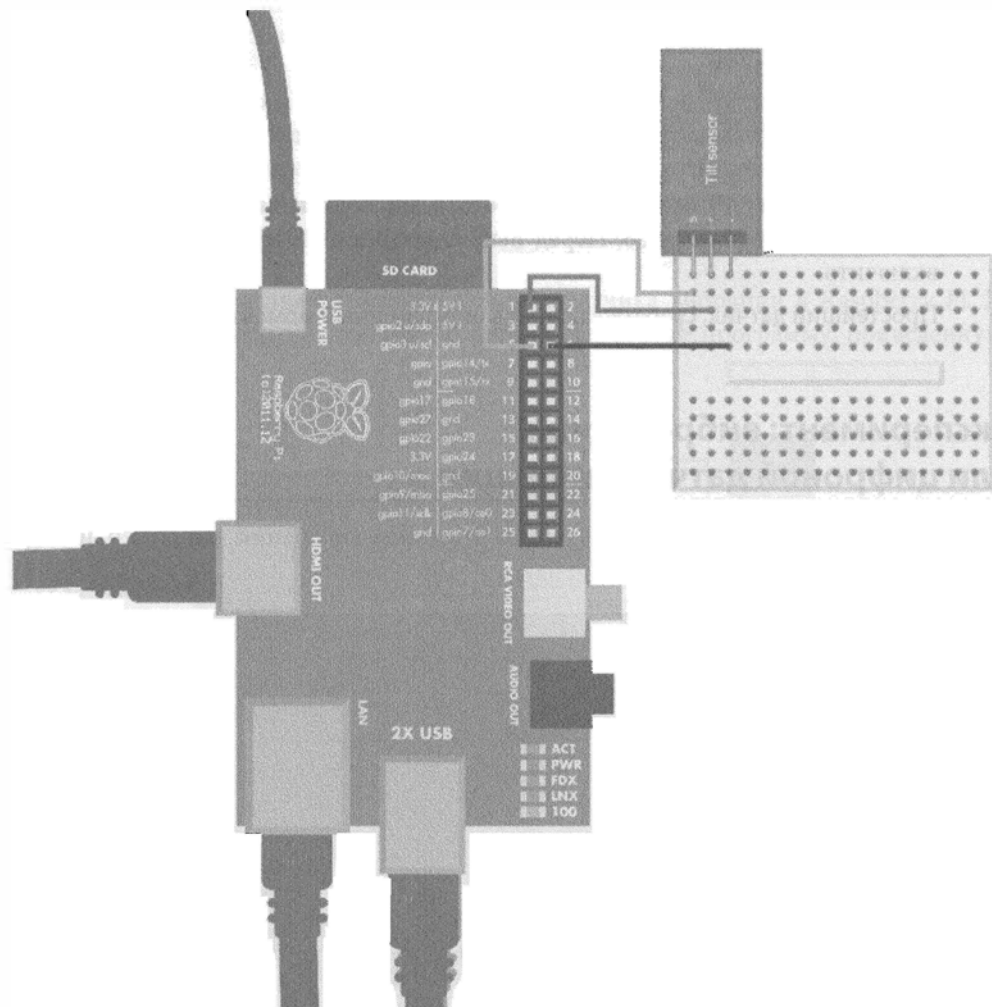


Рис. 6.3. Схема подключения датчика наклона к Raspberry Pi

Листинг 6.2. `tilt_sensor.py`

```
# tilt_sensor.py - определение наклона датчика
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio

def main():
```

```

tiltpin = 3 # встроенный подтягивающий резистор # ❶
gpio.mode(tiltpin, "in")
while (True):
    isNotTilted = gpio.read(tiltpin) # ❷
    if(isNotTilted == gpio.LOW):
        print "Датчик наклонен"
    else:
        print "Датчик выровнен"
    time.sleep(0.3) # секунд

if __name__ == "__main__":
    main()

```

- ❶ Плата Raspberry Pi оснащается встроенными подтягивающими резисторами для выводов gpio2 и gpio3. Перед использованием их нужно предварительно включить.
- ❷ Простейший датчик, срабатывающий в определенный момент как переключатель.

Эксперимент: вибродатчик, или цифровой датчик вибрации

Датчик вибрации распознает даже незначительные колебания, например, исходящие от земли (рис. 6.4).

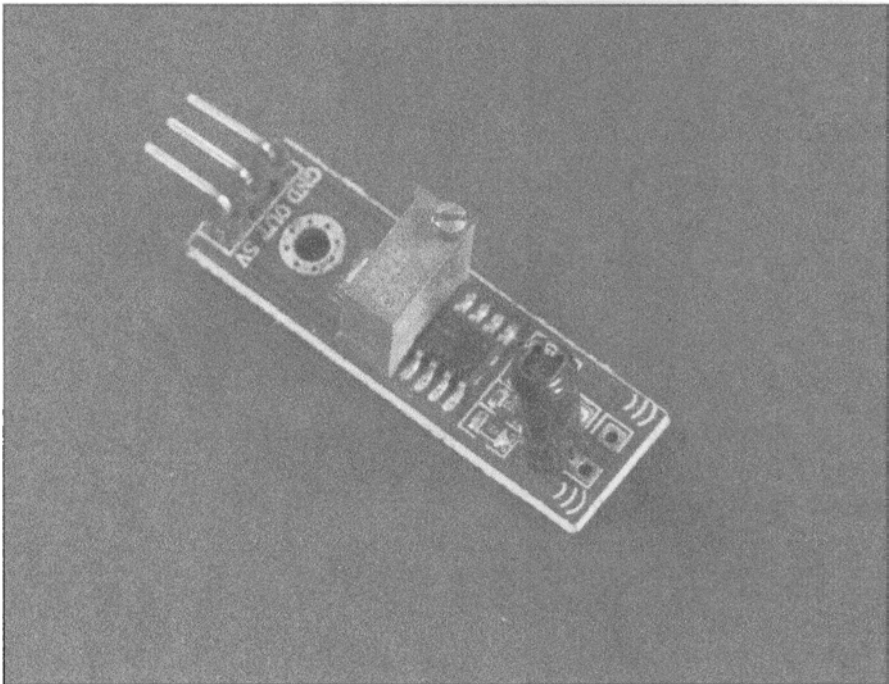


Рис. 6.4. Датчик вибраций

Подключение к Arduino и программа управления датчиком вибрации

Сигнал, передаваемый датчиком вибрации, очень короткий. Для его обработки необходимо научиться работать с прерываниями.

Для того чтобы получить сигнал с большинства датчиков, достаточно опросить их. Опрос заключается в проверке состояния цифрового вывода, ожидании и повторной проверке.

В случае использования прерывания вы указываете Arduino выполнять необходимую функцию непосредственно после наступления определенного события. Прерывания нарушают привычную последовательность выполнения строк программного кода, делая ее запутаннее для стороннего наблюдателя, но они позволяют распознавать кратковременные события, например, наблюдать быстрый скачок напряжения на выводе.

На рис. 6.5 показана схема подключения датчика вибрации к Raspberry Pi. Постройте на ее основе электрическую цепь и выполните программу, приведенную в листинге 6.3.

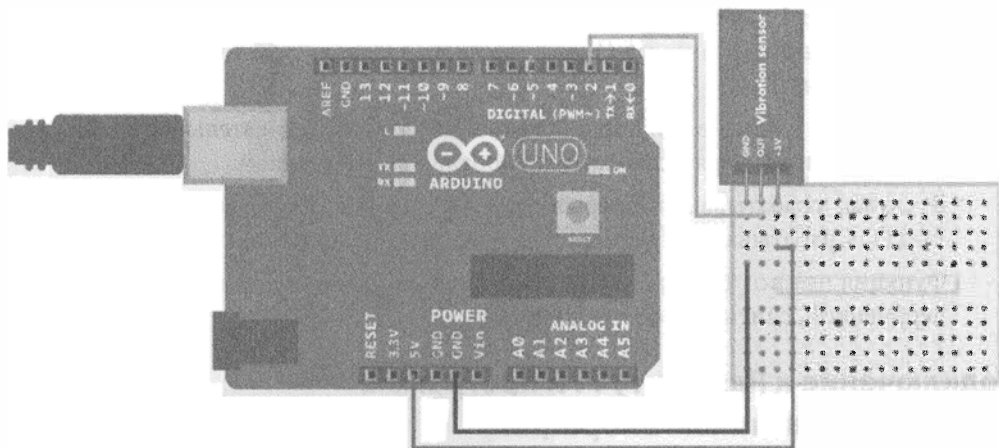


Рис. 6.5. Схема подключения датчика вибраций к Arduino

Листинг 6.3. vibration_sensor.ino

```
// vibration_sensor.ino - распознавание вибраций с помощью прерываний
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 0; // Uno, Mega - вывод 2, Leonardo -
                          // вывод 3
volatile int sensorState = -1;

void setup() {
    Serial.begin(115200);
    attachInterrupt(sensorPin, sensorTriggered, RISING); // ❶
}
```

```

void loop() {
    if(sensorState == 1) { // ❷
        Serial.println("Вибрирует!");
        delay(1); // мс
        sensorState = 0;
    }
    delay(10);
}

void sensorTriggered() { // ❷
    sensorState = 1; // ❶
}

```

- ❶ В результате срабатывания прерывания функция `sensorTriggered()` вызывается, как только переменная `sensorPin` изменяет свое значение с LOW на HIGH. Такой прием в программировании называется обратным вызовом. Обратите внимание на то, что при обратном вызове название исполняемой функции указывается без круглых скобок. А все потому, что при обратном вызове в качестве аргумента указывается не сама функция, а указатель на нее. Если вместо указателя подставить название функции с круглыми скобками, то она обязательно будет выполнена и вернет соответствующий результат. В данном случае `attachInterrupt()` должна лишь знать о существовании функции, но не вызывать ее.
- ❷ В основном цикле нет ничего необычного. Через последовательные временные промежутки проверяется значение глобальной переменной.
- ❸ В случае срабатывания прерывания вызывается функция `sensorTriggered()`. Срабатывание прерывания происходит без дополнительных задержек.
- ❹ Функция `sensorTriggered()` всего лишь запрашивает значение глобальной переменной, считываемое в основном цикле.

Подключение к Raspberry Pi и программа управления датчиком вибрации

На рис. 6.6 показана схема электрической цепи подключения датчика вибрации к Raspberry Pi. После подключения датчика к плате выполните программный код из листинга 6.4.

Листинг 6.4. `vibration_sensor.py`

```

# vibration_sensor.py - распознавание вибраций
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio # ❶

def main():
    vibPin = 3
    gpio.mode(vibPin, "in")
    while (True):
        vibrationInput = gpio.read(vibPin) # ❷

```

```

if(vibrationInput == gpio.LOW):
    print "Вибрации обнаружены"
    time.sleep(5) # ❶
    time.sleep(0.01) # секунд # ❷

if __name__ == "__main__":
    main()

```

- ❶ Как и в предыдущем примере для Raspberry Pi, указанная библиотека должна находиться в том же каталоге, что и файл `vibration_sensor.py`.
- ❷ Датчик вибрации относится к цифровым переключателям, поэтому сигнал с него считывается как с обычной кнопки.
- ❸ В случае обнаружения вибраций извещение выводится только один раз, а не заполняет своими копиями весь экран.
- ❹ Используется короткая задержка в 10 мс, поэтому переменная `vibPin` опрашивается 100 раз в секунду (с частотой 100 Гц).

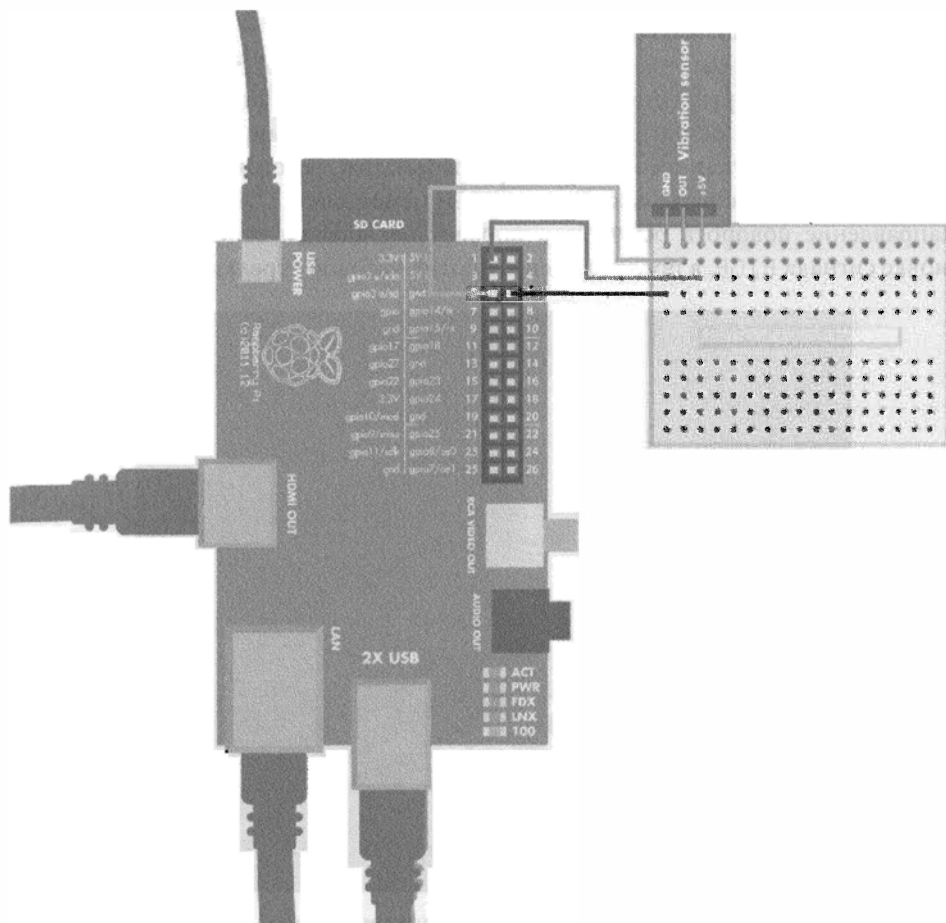


Рис. 6.6. Схема подключения датчика вибраций к Raspberry Pi

Приведенный выше программный код управления датчиком вибрации в Raspberry Pi не предназначен для распознавания коротких вибраций, что характерно для соответствующей программы Arduino. В данном коде не применяются конструкции типа прерываний, поскольку они сильно усложнят конечную программу, но все равно не обеспечат высокой чувствительности распознавания вибраций, достигнутой в Arduino. Чтобы добиться высокой точности распознавания вибраций в Raspberry Pi, попробуйте совместно использовать обе платформы (подробно об этом — в главе 12).

Эксперимент: поверни до упора (датчик угла поворота)

Датчик угла поворота (или кодовый датчик угла поворота), как и следовало ожидать, применяется для измерения угла поворота регулятора (рис. 6.7). Кодовые датчики угла поворота бывают двух типов: абсолютные, которые определяют точное значение угла поворота регулятора относительно исходного положения (например, 83 градуса), и накопительные (относительные), указывающие угол поворота относительно предыдущего положения (например, 23 градуса).

В текущем проекте мы будем использовать накопительный датчик угла поворота. При повороте ручки регулятора на отдельный контакт датчика подаются синхросимпульсы. Импульсы на сигнальном выводе датчика формируются как по нарастающему фронту (сигнал с состояния LOW переходит в состояние HIGH), так и по спадающему фронту (сигнал с состояния HIGH меняется на LOW).

Направление поворота регулятора вправо (по часовой стрелке) соответствует установке сигнала HIGH, а поворот влево (против часовой стрелки) — сигнала LOW.

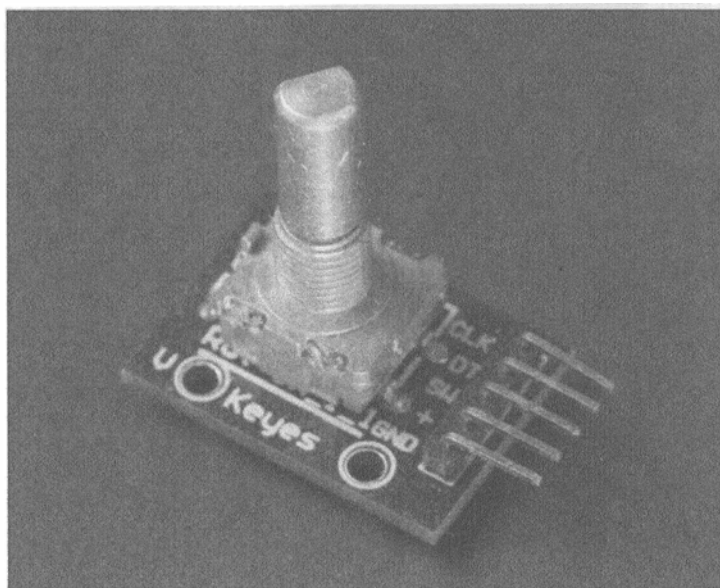


Рис. 6.7. Кодовый датчик угла поворота

Подключение к Arduino и программа управления кодовым датчиком угла поворота

В приведенном ниже программном коде применяются прерывания, поэтому он структурно отличается от большинства других программ для Arduino, с которыми вам уже приходилось иметь дело в ранних проектах. Подключите оборудование так, как показано на схеме, приведенной на рис. 6.8, а затем выполните код, приведенный в листинге 6.5.

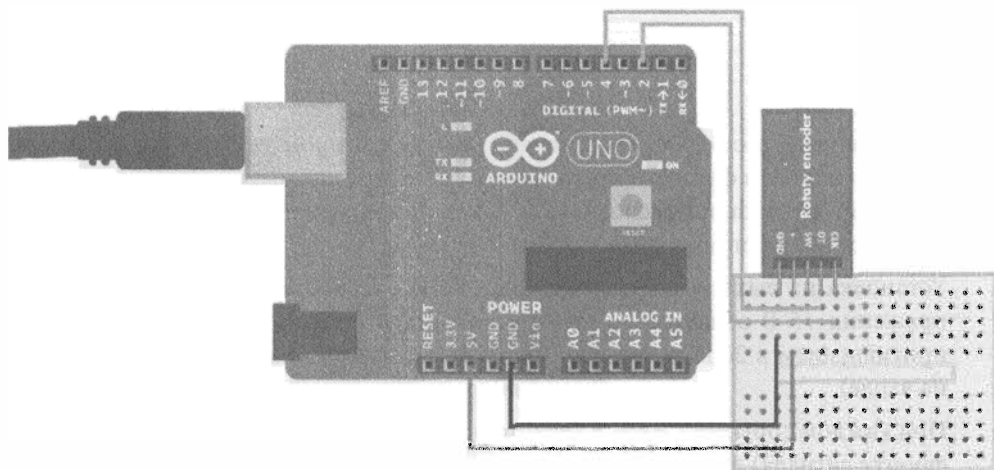


Рис. 6.8. Схема подключения датчика угла поворота к Arduino

Листинг 6.5. `rotary_encoder.ino`

```
// rotary_encoder.ino - вывод положения регулятора
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int clkPin = 2;
const int dtPin = 4;

volatile unsigned int encoderPos = 0; // ❶

void setup()
{
  Serial.begin(115200);
  pinMode(clkPin, INPUT);
  digitalWrite(clkPin, HIGH); // подтягивающий резистор // ❷
  pinMode(dtPin, INPUT);
  digitalWrite(dtPin, HIGH); // подтягивающий резистор

  attachInterrupt(0, processEncoder, CHANGE); // ❸
}
```

```

void loop()
{
    Serial.println(encoderPos);
    delay(100);
}

void processEncoder() // ❶
{
    if(digitalRead(clkPin) == digitalRead(dtPin)) // ❷
    {
        encoderPos++; // вращение вправо
    } else {
        encoderPos--;
    }
}

```

- ❶ Переменная `EncoderPos` обозначена квалификатором `volatile`, поскольку она изменяется в функции прерывания. Ключевое слово `volatile` позволяет компилятору Arduino распознавать переменные, значения которых могут изменяться вне программы, т.е. основного цикла “скетча” Arduino.
- ❷ Установка на выходе сигнала уровня HIGH соответствует подаче на него напряжения +5 В обязательно через встроенный подтягивающий резистор сопротивлением в 20 кОм (во избежание образования плавающего напряжения на входе).
- ❸ При каждом событии `CHANGE` (возникновении нарастающего или спадающего фронта сигнала) вызывается функция `processEncoder`. В Arduino Uno прерывание с номером 0 назначается для второго цифрового вывода. Опять-таки, название внешне вызываемой функции `processEncoder` указано без скобок, поскольку она сама на данном этапе не выполняется. Она будет вызвана и выполнена далее, в результате срабатывания прерывания.
- ❹ На этапе выполнения функции прерывания выполнение остального программного кода приостанавливается.
- ❺ Если значение переменной `dtPin` такое же, как и у `clkPin` (значения синхросигнала и сигнала на выводе датчика совпадают), то регистрируется поворот регулятора вправо. В противном случае регулятор поворачивается влево.

Подключение к Raspberry Pi и программа управления кодовым датчиком угла поворота

На рис. 6.9 показана схема подключения кодового датчика угла поворота к Raspberry Pi. После подключения всех необходимых компонентов в единую цепь выполните программу из листинга 6.6.

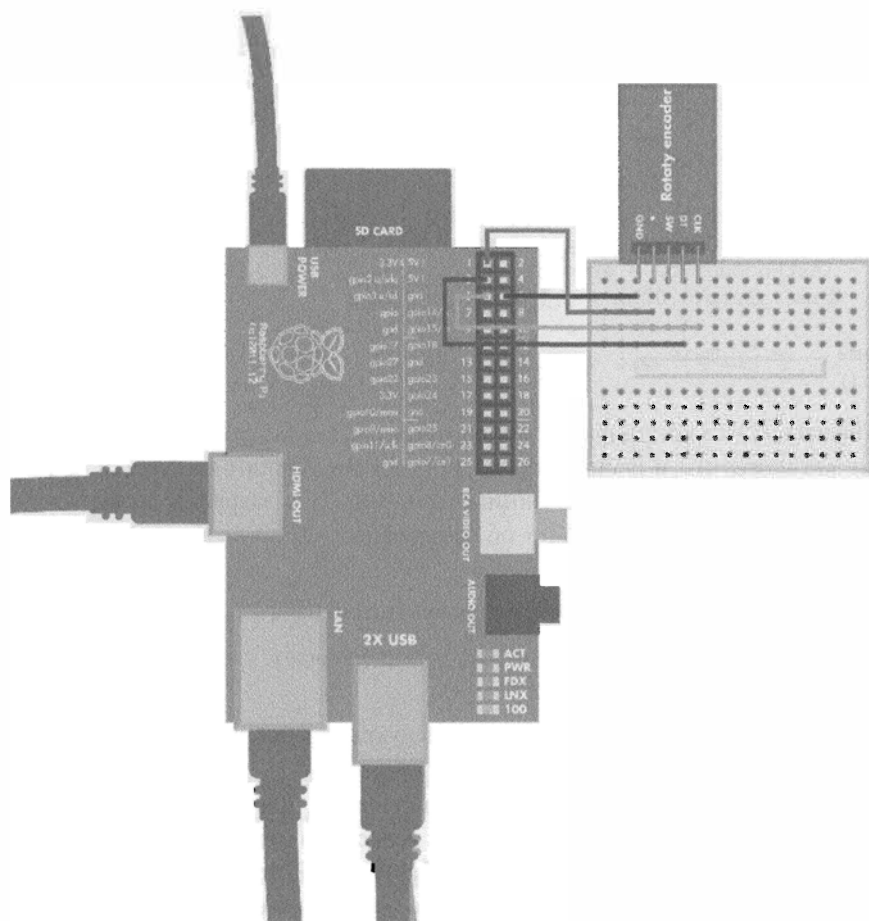


Рис. 6.9. Схема подключения датчика угла поворота к Raspberry Pi (см. цветную вклейку)

Листинг 6.6. `rotary_encoder.py`

`rotary_encoder.py` - вывод положения регулятора
 # (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_gpio as gpio # ❶

def main():
    encoderClk = 3
    encoderDt = 2
    gpio.mode(encoderClk, "in")
    gpio.mode(encoderDt, "in")
    encoderLast = gpio.LOW
    encoderPos = 0
    lastEncoderPos = 0
    while True:
```

```

clk = gpio.read(encoderClk)
if encoderLast == gpio.LOW and clk == gpio.HIGH: # ❶
    if (gpio.read(encoderDt) == gpio.LOW): # ❷
        encoderPos += 1
    else:
        encoderPos -= 1
encoderLast = clk
if encoderPos != lastEncoderPos:
    print("Положение регулятора: %d" % encoderPos)
lastEncoderPos = encoderPos
time.sleep(0.001) # с # ❸

if __name__ == "__main__":
    main()

```

- ❶ Убедитесь, что копия файла библиотеки `botbook_gpio.py` находится в том же каталоге, что и файл текущей программы. Файл библиотеки, а также все файлы примеров из данной книги доступны для загрузки по адресу, указанному во введении. Детально об управлении GPIO-выводами в Raspberry Pi см. в главе 1.
- ❷ Если при нарастающем фронте (измерение сигнала с LOW на HIGH)...
- ❸ ...синхросигнал имеет уровень LOW, то это означает, что регулятор поворачивается влево (против часовой стрелки).
- ❹ Задержка в 1 мс, чтобы не упустить следующий такт синхроимпульса.

Даже несмотря на большую производительность, Raspberry Pi выполняет программу управления датчиком угла поворота несколько медленнее, чем Arduino (в реальном времени). Если вы замечаете, что Raspberry Pi пропускает слишком много импульсов при быстром повороте регулятора (что обычно происходит при одновременном запуске в ней нескольких приложений), то рассмотрите вариант совместного применения Arduino и Raspberr Pi (подробно о взаимодействии платформ рассказывается в главе 12).

Эксперимент: джойстик под большой палец (аналоговый двухкоординатный резистивный джойстик)

Если вы играли (или продолжаете это делать) в консольные видеоигры, то наверняка знаете, что такое джойстик. В старые времена, когда деревья были большими, а трава зеленее, управление игрой выполнялось с помощью джойстика, обхватываемого всей кистью. В современных игровых приставках, таких как Xbox, PlayStation и Оцуа, манипуляторы komponуются несколькими двухкоординатными джойстиками, рассчитанными на управление большими пальцами рук.

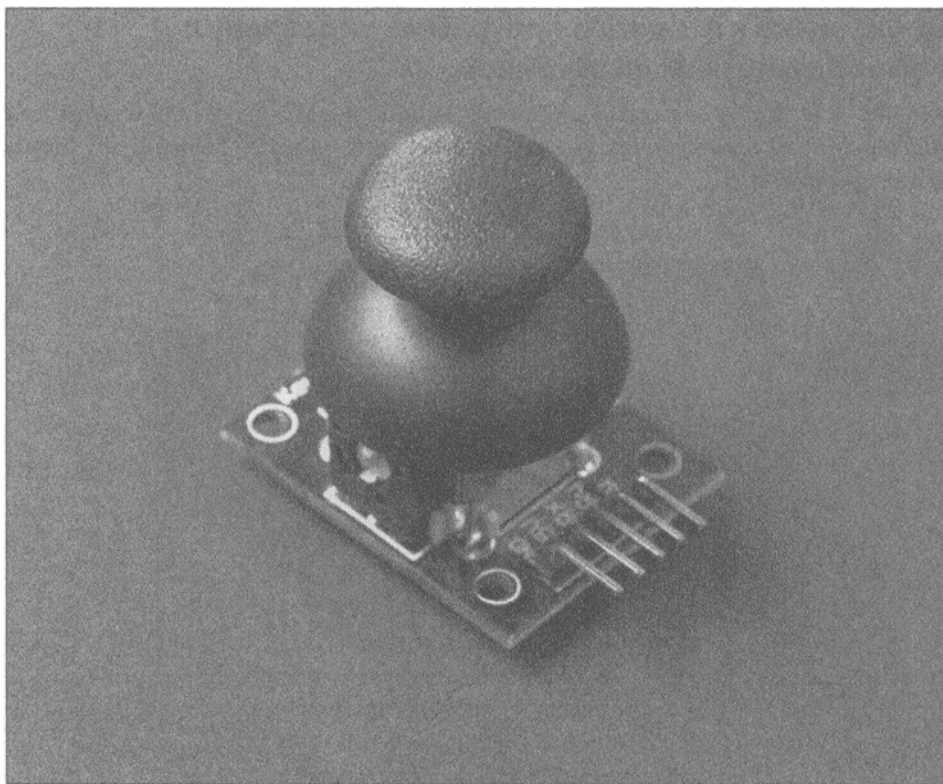


Рис. 6.10. Двухкоординатный джойстик под большой палец

Чаще всего подобный джойстик оснащается двумя потенциометрами, представляющими собой обычные переменные резисторы. Наклон рычага джойстика в одном направлении (например, вдоль оси Y) приводит к изменению сопротивления одного из потенциометров. За изменение выходного сигнала джойстика при наклоне рычага в другом координатном направлении (теперь вдоль оси X) отвечает второй потенциометр.

В мобильных устройствах и игровых приставках Nintendo Wii для управления используются не джойстики, а акселерометры (совместно с гироскопами). Управление играми в них осуществляется в результате изменения ориентации всего устройства в трехмерном пространстве. В данном случае акселерометр измеряет гравитационное ускорение, а потому не требует применения механических манипуляторов. (Детальнее об этом рассказывается в главе 8.)

В большинстве джойстиков применяются трехвыводные потенциометры. На один из выводов такого потенциометра подается напряжение +5 В, второй вывод заземляется, а на третьем среднем выводе считывается сигнал (в диапазоне напряжений от 0 до +5 В). В подобной конфигурации нет нужды использовать дополнительные подтягивающие и согласующие резисторы.

Подключение к Arduino и программа управления двухкоординатным джойстиком

На рис. 6.11 показана схема оснащения Arduino двухкоординатным джойстиком. Подключите джойстик к платформе так, как показано на ней, и выполните программный код из листинга 6.7.

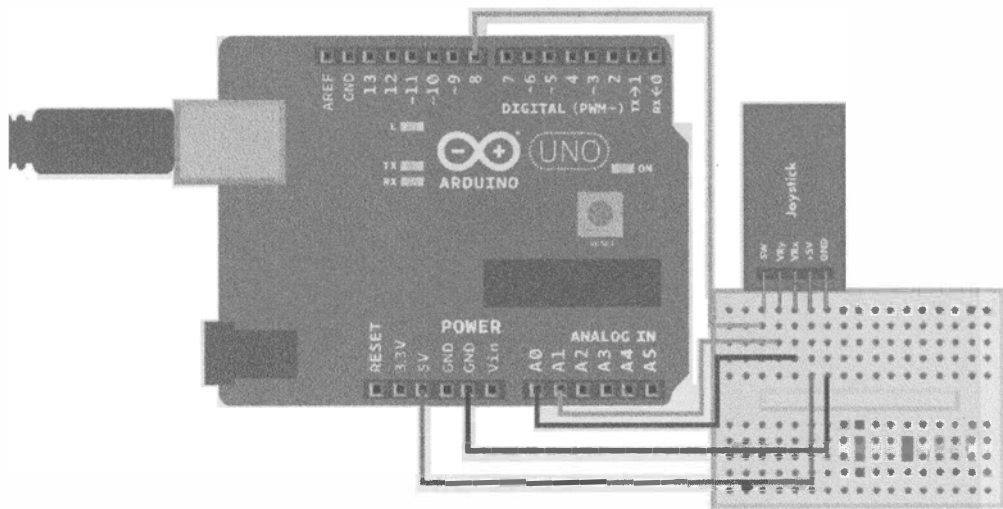


Рис. 6.11. Схема подключения джойстика к Arduino

Листинг 6.7. `ky_023_xyjoystick.ino`

```
// ky_023_xyjoystick.ino - вывод положения джойстика
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int VRxPin = 0; // ❶
const int VRyPin = 1;
const int SwButtonPin = 8;

int button = -1; // LOW или HIGH // ❷
int x = -1; // 0..1023
int y = -1; // 0..1023

void readJoystick() { // ❸
    button = digitalRead(SwButtonPin); // ❹
    x = analogRead(VRxPin);
    y = analogRead(VRyPin);
}

void setup() {
    pinMode(SwButtonPin, INPUT);
    digitalWrite(SwButtonPin, HIGH); //подтягивающий резистор // ❺
    Serial.begin(115200);
}
```

```

void loop() {
    readJoystick(); // ❶
    Serial.print("X: ");
    Serial.print(x);
    Serial.print(" Y: ");
    Serial.print(y);
    Serial.print(" Кнопка: ");
    Serial.println(button);
    delay(10);
}

```

- ❶ Сохранение номеров выводов в виде глобальных констант. К этим выводам подключаются потенциометр для направления X, потенциометр для направления Y и кнопка.
- ❷ В глобальных переменных хранятся сведения о состоянии кнопки и наклоне рычага джойстика в направлении X и Y. Исходно этим переменным назначаются невозможные значения (т.е. такие, которые не могут быть сгенерированы функциями `analogRead()` и `digitalRead()`). Этот прием значительно упрощает последующую отладку программы. Предполагаемый диапазон возможных значений указан в комментариях.
- ❸ Функция `readJoystick()` не возвращает значений (имеет тип `void`). В языке C++, на котором основана среда разработки Arduino, функциям не разрешается возвращать множественные значения. Чтобы обойти это ограничение, функция `readJoystick()` обновляет глобальные переменные.
- ❹ Состояние кнопки хранится в виде значения переменной `button`.
- ❺ Включение встроенного подтягивающего резистора во избежание образования плавающего напряжения на выходе.
- ❻ Обновление глобальных переменных, которыми представляется состояние джойстика.

Подключение к Raspberry Pi и программа управления джойстиком

На рис. 6.12 показана схема оснащения джойстиком платформы Raspberry Pi. Подключите все показанное на ней оборудование, а затем выполните программный код, который приведен в листинге 6.8.

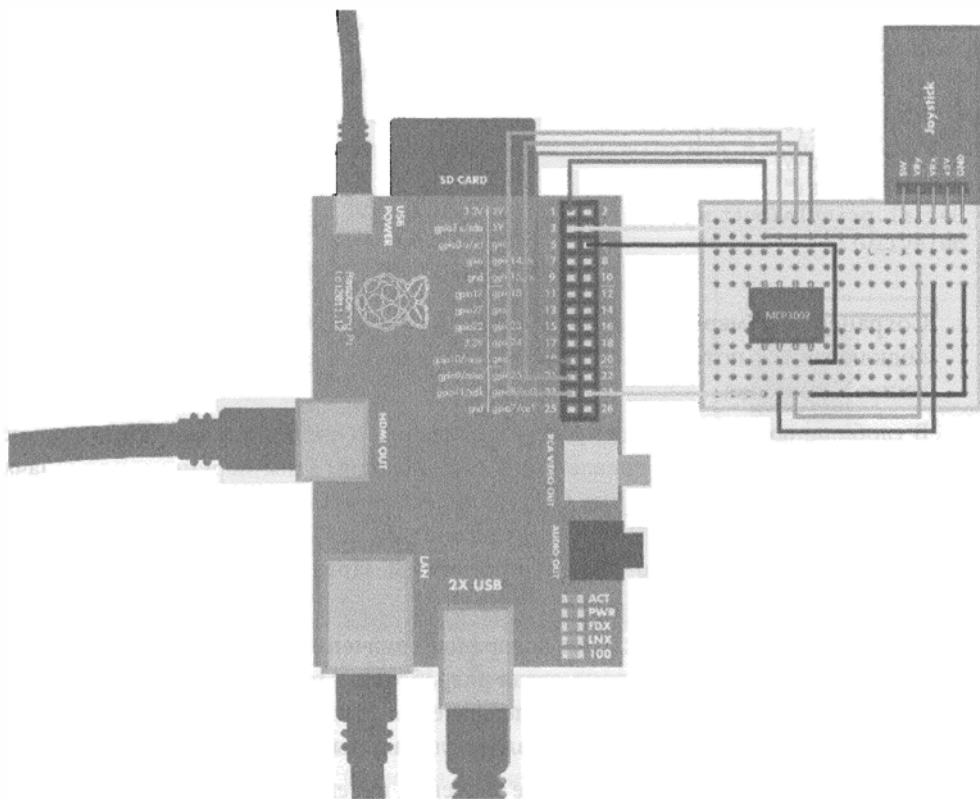


Рис. 6.12. Схема подключения джойстика к Raspberry Pi (см. цветную вклейку)

Листинг 6.8. `xu_joystick.py`

```
# xu_joystick.py - вывод состояния джойстика KY 023 и кнопки на
# монитор последовательного порта
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import botbook_mcp3002 as mcp # ❶
import botbook_gpio as gpio # ❷

def readX(): # ❸
    return mcp.readAnalog(0, 0)

def readY():
    return mcp.readAnalog(0, 1) # ❹

def readButton():
    buttonPin = 25
    gpio.mode(buttonPin, "in")
    return gpio.read(buttonPin)

def main():
    while True: # ❺
```

```

xAxel = readX() # ❶
yAxel = readY()
button = readButton()
print("X: %i, Y: %i, Кнопка: %r" % (xAxel, yAxel, button)) # ❷
time.sleep(0.5)

```

```

if __name__ == "__main__":
    main()

```

- ❶ Подключите библиотеку, обеспечивающую поддержку аналого-цифрового преобразователя MCP3002. Файл библиотеки `botbook_mcp3002.py` должен находиться в том же каталоге, что и файл текущей программы, `xy_joystick.py`. Кроме того, вам нужно подключить библиотеку `spidev`, применяемую в библиотеке `botbook_mcp3002`. Детальнее об этом см. в главе 3.
- ❷ Постарайтесь использовать максимально информативное пространство имен, чтобы упростить понимание программного кода. Оператор `as` позволяет в дальнейшем применять название `gpio.read()` вместо `botbook_gpio.read()`.
- ❸ Давайте названия функциям согласно их назначению. Например, функция `readX()` предназначена для считывания положения рычага джойстика вдоль оси X. В дополнительных описаниях она не нуждается.
- ❹ Измерение напряжения на втором канале (вывод номер 1).
- ❺ Выполнение продолжается бесконечно или до нажатия пользователем комбинации клавиш `<Ctrl+C>` для завершения программы (или прекращения подачи питания на плату Raspberry Pi).
- ❻ Считывание наклона рычага вдоль оси X и сохранение состояния в новой переменной `xAxel`.
- ❼ При выводе значений нескольких переменных в виде строки они организуются в кортеж (набор значений, разделенных запятыми). Можете представить себе кортеж `(1, 2, 3)` как список `[1, 2, 3]`, недоступный для изменения.

Эксперимент в окружающей среде: вторая жизнь старого игрового контроллера

Если у вас на полке завалялась старая игровая приставка (Xbox, PlayStation и т.п.), то можете разобрать ее игровой контроллер и извлечь из него пару двунаправленных джойстиков, которые можно легко приспособить для использования в проектах Arduino или Raspberry Pi (рис. 6.13). Вскрытие корпуса игрового контроллера не представляет особых трудностей, а вот для извлечения электронных компонентов вам придется воспользоваться паяльником.

При демонтаже оборудования с игрового контроллера вам пригодятся такие инструменты/материалы: паяльник, отсос для припоя, припой (олово) и канифоль.

Многие контроллеры также оснащаются системой обратной связи. У нее очень простая функция: реагировать на определенные действия в игре. Например, при нанесении урона в бою система обратной связи начинает производить вибрации. Эта функция реализуется с помощью вибродвигателей — электродвигателей постоянного тока с эксцентричными кулачками на валу, которые при вращении создают вибрации. Они вам обязательно пригодятся в собственных проектах, поэтому извлеките их тоже (рис. 6.14).

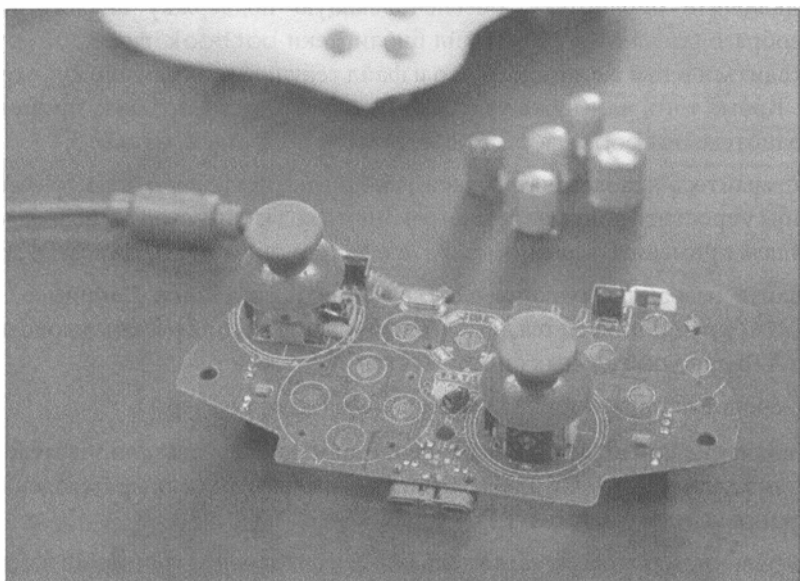


Рис. 6.13. Джойстики под большие пальцы в старом игровом контроллере

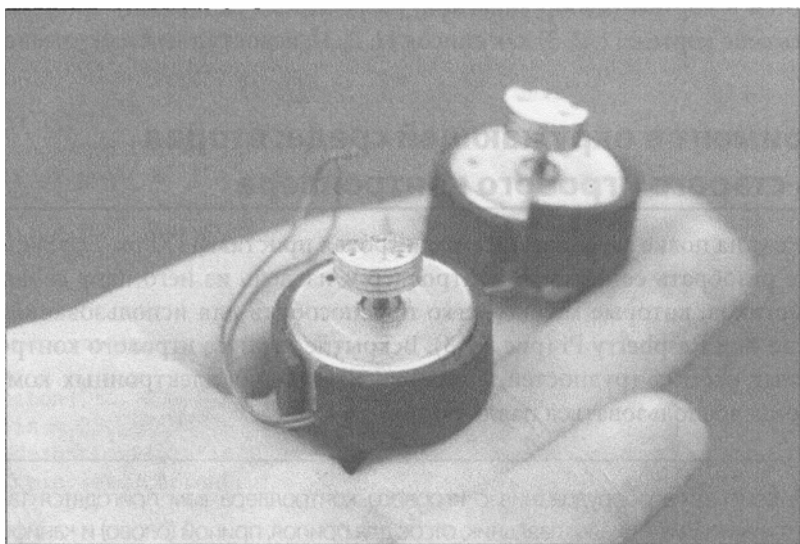


Рис. 6.14. Все еще работающие вибродвигатели

Эксперимент: охранная сигнализация (пассивный инфракрасный датчик движения)

В охранных сигнализациях для обнаружения злоумышленников чаще всего применяется пассивный инфракрасный датчик движения (Passive Infrared Sensor — PIR). Как вы знаете, все теплые объекты излучают невидимые глазу инфракрасные лучи. Инфракрасный датчик движения реагирует на изменение потока инфракрасного излучения, поступающего в его приемник. Зачастую изменение инфракрасного фона вызывается перемещением поблизости датчика достаточно крупного объекта — человека.

Перед использованием пассивный инфракрасный датчик движения необходимо адаптировать к окружающей среде. Поскольку такие датчики не измеряют абсолютный уровень инфракрасного излучения, а только регистрируют его изменение в окружающем пространстве, то сначала нужно их “ознакомить” с текущим уровнем инфракрасного фона в помещении. После подачи напряжения пассивному датчику необходимо около 30 секунд на адаптацию к окружающей среде. В течение этого времени в помещении не должно наблюдаться никаких температурных изменений, а также в него не должны входить и выходить люди.

Для ограничения области, которую диагностирует датчик, используйте обычную картонную коробку. Этот прием часто применяется для тестирования пассивных инфракрасных датчиков. Положите датчик в коробку, расположив его приемником вверх, и оставьте коробку открытой только сверху. Теперь датчик может спокойно адаптироваться к помещению, а вы, находясь в одном с ним помещении, спокойно работать над другими задачами проекта, например, написанием программного кода. Чтобы проверить, насколько удачно датчик адаптировался в новой среде, проведите рукой над открытой коробкой.

Компания Parallax снабжает свои пассивные инфракрасные датчики движения двумя рабочими режимами:

- H — соответствует выводу сигнала высокого уровня на протяжении всего времени обнаружения движения;
- L — сбрасывание сигнала в положение низкого уровня после срабатывания.

В режиме L датчик отправляет сигнал в виде всего одного импульса даже в том случае, когда движение продолжается после его первой регистрации.

Изменение рабочего режима производится специальной небольшой перемычкой, прикрытой пластиковой крышечкой. В нашем проекте перемычка установлена в положение H (режим High).

Подключение к Arduino и программа управления охранной сигнализацией

На рис. 6.15 показана схема подключения пассивного инфракрасного датчика движения к Arduino. После создания соответствующей электрической цепи выполните программный код, приведенный в листинге 6.9.

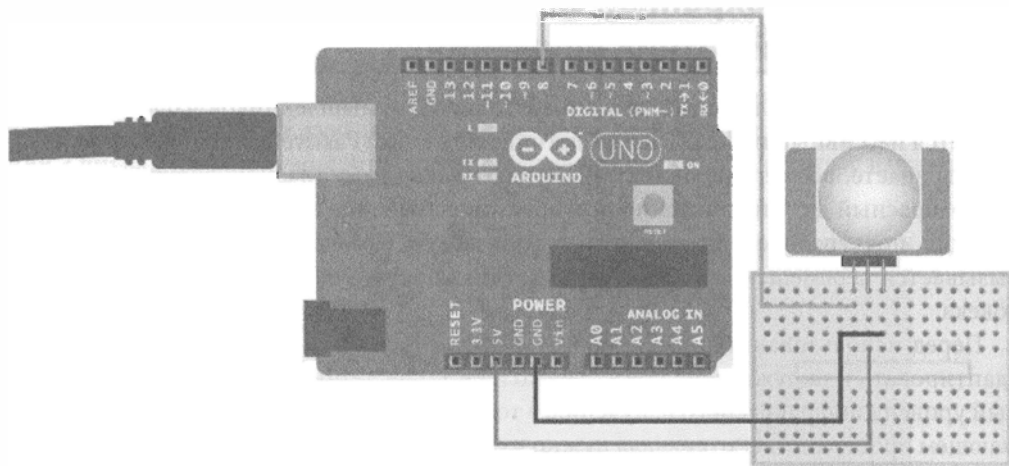


Рис. 6.15. Схема подключения пассивного инфракрасного датчика движения (модель А) производства Parallax к Arduino

Листинг 6.9. `parallax_pir_reva.ino`

```
// parallax_pir_reva.ino - регистрация движения
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 8;
const int ledPin = 13;
const int learningPeriod = 30*1000; // мс // ❶
int learningCompleted = 0;

void setup() {
    Serial.begin(115200);
    pinMode(sensorPin, INPUT);
    Serial.println("Изучение исходной среды..."); // ❷
    pinMode(ledPin, OUTPUT);
}

void loop() {
    if(millis() < learningPeriod) { // ❸
        delay(20); // мс // ❹
        return; // ❺
    }
    if(learningCompleted == 0) { // ❻
        learningCompleted = 1;
        Serial.println("Изучение среды завершено");
    }

    if(digitalRead(sensorPin) == HIGH) { // ❼
        Serial.println("Обнаружено движение");
        digitalWrite(ledPin,HIGH);
    }
}
```



```

} else {
    Serial.println("Движение отсутствует");
    digitalWrite(ledPin, LOW);
}
delay(100);
}

```

- ❶ Датчик необходимо адаптировать к окружающему пространству, исключив движение в нем. В нашем случае адаптация занимает 30 000 мс (30 с). Время адаптации представлено глобальной константой. Избегайте повторного комментирования назначения этой константы. Лучше укажите в комментарии единицы измерения (в данном случае — мс), чтобы избежать неправильного толкования размерных величин. В подобном случае при изменении числового значения вам не придется корректировать комментарий. Если числовое значение константы получено в результате вычисления (например, количество секунд, умноженное на 1000), то лучше представить их в коде ($30 \cdot 1000$), а не ограничиваться одним лишь результатом.
- ❷ Вывод на монитор последовательного порта извещения. Для отображения на экране этого монитора в среде разработки выполните команду Tools⇒Serial Monitor (Среда⇒Монитор порта). Не забудьте выбрать одинаковую скорость передачи данных в программном коде и настройках монитора (боды или бит/с).
- ❸ Стандартный способ измерения времени в Arduino. Функция `millis()` возвращает количество миллисекунд, прошедших с момента выполнения текущей программы.
- ❹ Короткая задержка предотвращает безостановочное повторение функции `loop()`, что приводит к полной загрузке микропроцессора.
- ❺ Оператор `return` завершает выполнение функции `loop()`. Как всегда, функция `loop()` сразу после завершения автоматически вызывается еще раз.
- ❻ Переменная `learningCompleted` применяется для однократного вывода сообщения “Изучение среды завершено”. Здесь значения 1 и 0 используются вместо булевых `true` (истина) или `false` (ложь).
- ❼ При обнаружении датчиком перемещения теплого объекта переменная `sensorPin` устанавливается в значение `HIGH`.

Подключение к Raspberry Pi и программа управления охранной сигнализацией

На рис. 6.16 показана цепь подключения пассивного инфракрасного датчика движения к Raspberry Pi. После ее создания выполните программу из листинга 6.10.

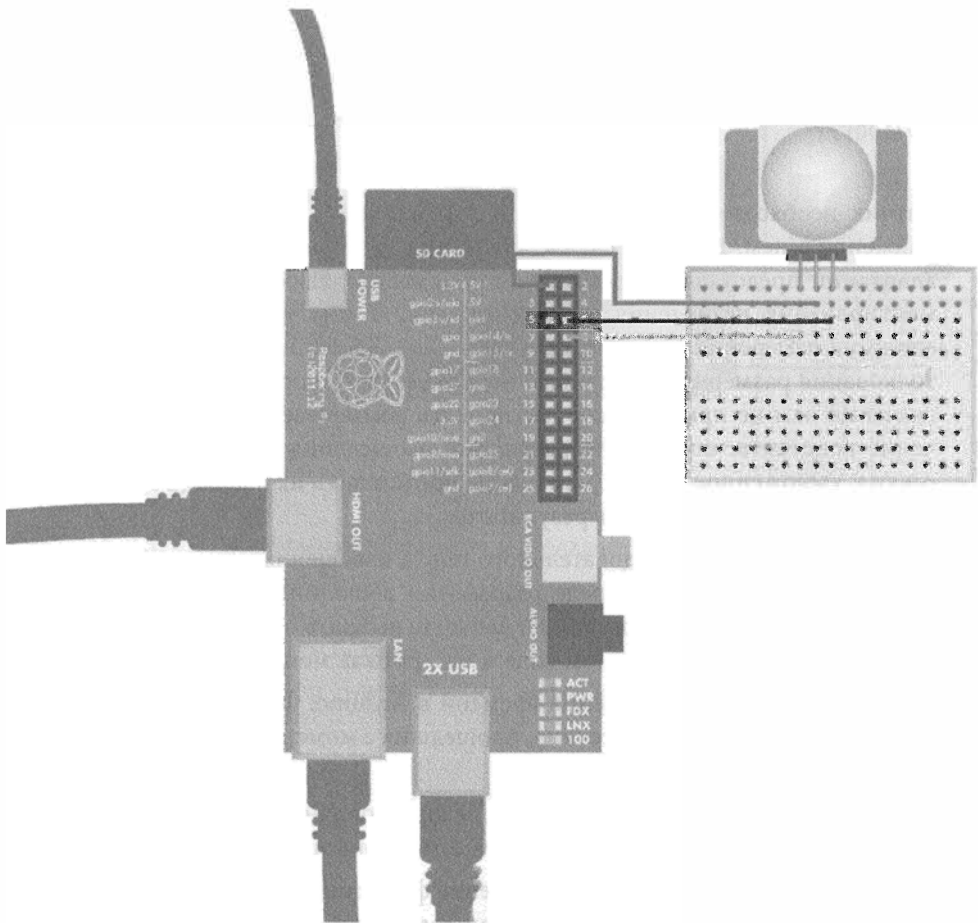


Рис. 6.16. Схема подключения пассивного инфракрасного датчика движения (модель А) производства Parallax к Raspberry Pi

Листинг 6.10. `parallax_pir_reva.py`

```
# parallax_pir_reva.py - вывод извещения о регистрации движения
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio
learningPeriod = 60

def main():
    pirPin = 14
    gpio.mode(pirPin, "in")
    # Адаптация
    time.sleep(learningPeriod) # ❶
    while (True):
        movement = gpio.read(pirPin) # ❷
        if(movement == gpio.HIGH):
```

```

        print "Обнаружено движение"
    else:
        print "Нет движения"
        time.sleep(0.3)

if __name__ == "__main__":
    main()

```

- ❶ Датчику нужно время для адаптации к помещению, для чего в нем исключаются любые перемещения. В нашем случае время адаптации составляет 60 секунд, но можно использовать другое значение.
- ❷ При обнаружении движения на выход датчика подается сигнал высокого уровня.

Эксперимент в окружающей среде: взлом охранной сигнализации

В задачу сертифицированной службы взлома сигнализаций (или технических экспертов, занимающихся поиском уязвимостей в охранном оборудовании) входит незамеченное проникновение в охраняемое помещение. В нашем текущем проекте вы попробуете себя в роли взломщика сигнализации, правда, у себя дома. Это упростит задачу, поскольку от вас не потребуется скрываться от соседей (в случае профессионального взлома), а от конечного результата пострадает только ваше самолюбие, но не карьера (как правило, заканчивающаяся в местах не столь отдаленных).

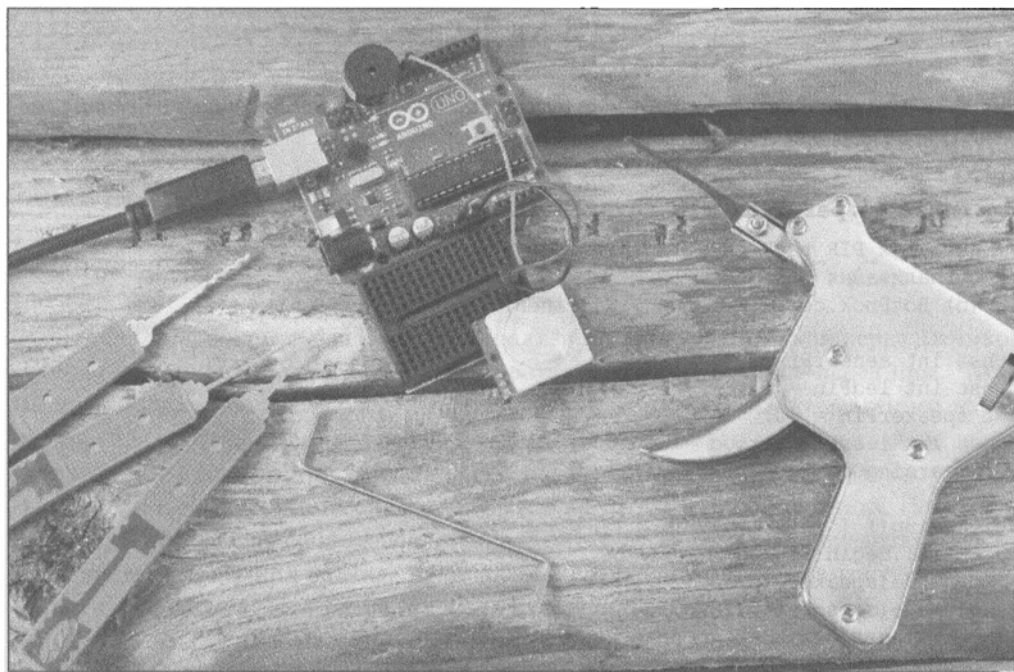


Рис. 6.17. Иногда игольчатый тестер позволяет быстро взломать сигнализацию

Вы можете попробовать собственными силами обмануть пассивный инфракрасный датчик движения, но, как вы вскоре сможете заметить, это не так просто, как показывают в голливудских боевиках. В текущем проекте мы воспользуемся предыдущей схемой, добавив в нее пьезоэлектрический зуммер. Он быстро даст знать об обнаружении движения. Подключите пьезозуммер так, как показано на рис 6.18, а затем выполните программный код из листинга 6.11.

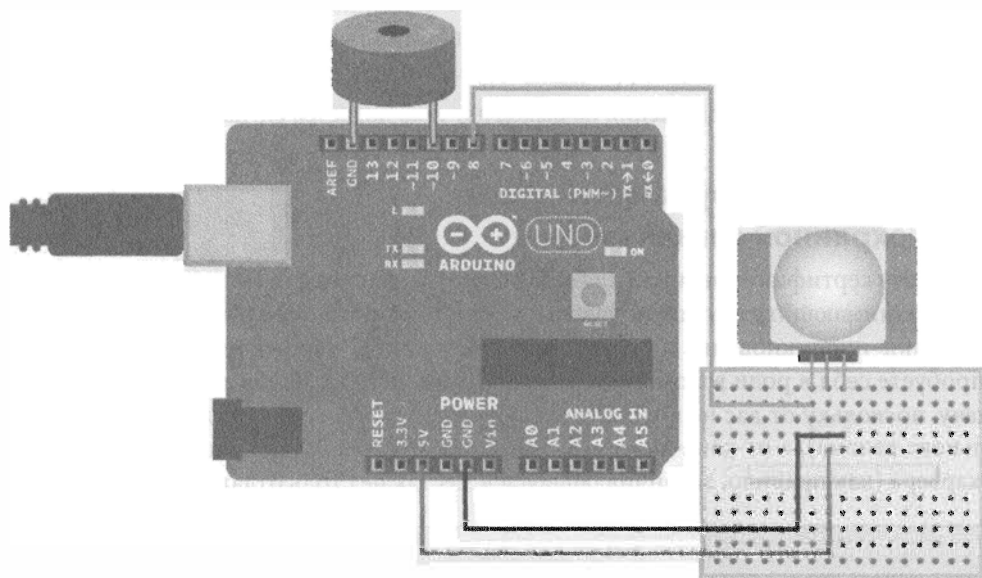


Рис. 6.18. Схема подключения к Arduino пассивного инфракрасного датчика движения (модель А) производства Parallax, светодиода и зуммера

Листинг 6.11. `parallax_PIR_revA_cheating_pir.ino`

```
// parallax_PIR_revA_cheating_pir.ino - световая и звуковая
// сигнализация
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 8;
const int ledPin = 13;
int speakerPin = 10;
const int learningPeriod = 30*1000; // 30 секунд на адаптацию
int learningCompleted = 0;

void setup() {
    Serial.begin(115200);
    pinMode(speakerPin, OUTPUT);
    pinMode(sensorPin, INPUT);
    Serial.println("Начало адаптации в течение 30 секунд.");
    pinMode(ledPin, OUTPUT);
}

void alarm()
```

```

{
    wave(speakerPin, 440, 40);
    delay(25);
    wave(speakerPin, 300, 20);
    wave(speakerPin, 540, 40);
    delay(25);
}

void wave(int pin, float frequency, int duration)
{
    float period=1/frequency*1000*1000; // мс
    long int startTime=millis();
    while(millis()-startTime < duration) {
        digitalWrite(pin, HIGH);
        delayMicroseconds(period/2);
        digitalWrite(pin, LOW);
        delayMicroseconds(period/2);
    }
}

void loop() {
    if(millis() < learningPeriod) {
        return; // Сенсор еще не адаптировался в среде
    }
    if(learningCompleted == 0) {
        learningCompleted = 1;
        Serial.println("Адаптация завершена.");
    }
    if(digitalRead(sensorPin) == HIGH) {
        Serial.println("Обнаружено движение");
        alarm();
        digitalWrite(ledPin, HIGH);
    } else {
        Serial.println("Нет движения");
        digitalWrite(ledPin, LOW);
    }
}
delay(100);
}

```

Теперь вы должны слышать звук сигнализации при взмахе рукой перед датчиком. Попробуйте отойти на несколько метров от датчика и пройти перед ним. Чтобы не сработала сигнализация, вам необходимо двигаться как можно медленнее, как бы трудно это ни было. Накрывшись простыней или пледом, вы упростите свою задачу. Замотайтесь с головой (не забывайте про отверстия для глаз), став похожим на привидение, и начинайте потихоньку двигаться по направлению к датчику. Покрывало экранирует инфракрасное излучение, исходящее от вашего тела, предотвращая его попадание в приемник датчика. В подобном случае вы сможете подойти к датчику вплотную и коснуться его перед тем, как будете обнаружены.

В реальных охранных устройствах пассивные инфракрасные датчики движения применяются совместно с другими системами обнаружения, например, ультразвуковыми датчиками расстояния. И, конечно же, повсеместные камеры наблюдения не дадут незаметно пройти мимо охраняемого объекта.

Ранее мы уже рассказывали о том, как можно обмануть ультразвуковой датчик расстояния (см. главу 3). Несмотря на то что камеры слежения представляют собой труднопреодолимое препятствие, при должной подготовке вы сможете проскользнуть и мимо их недремлющего ока. Для съемки в темное время суток многие камеры оснащаются активными инфракрасными лампами подсветки. О том, как обнаружить инфракрасный свет, см. в главе 3.

Пилотный проект: электронная игра

Задача обнаружения движения становится намного интереснее, если ее можно использовать в практических целях. В следующем проекте вы узнаете о том, как с помощью датчиков заставить объекты перемещаться по экрану. Чтобы упростить управление программой, в качестве устройства ввода мы будем использовать джойстик. Рассмотренные в этом проекте приемы вы сможете легко использовать в других проектах далеко не игрового характера. В качестве устройства ввода можно применять любой другой датчик, а область применения, возможности конечной программы и происходящего на экране ограничиваются только вашим воображением.



Рис. 6.19. Сыграем?

В качестве примера попробуем воссоздать ставшую классической игру Pong (Теннис), выпущенную компанией Atari в далеком 1972 году. В этой игре вам как игроку нужно успеть отражать мяч с помощью перемещаемой вверх и вниз ракетки. Чтобы победить в игре, нельзя позволить мячу вылететь за границы экрана.

Воссоздавая эту простую игру, вы познакомитесь с библиотекой `pyGame`, одной из старейших библиотек, используемых при программировании игр. Кроме того, вы собственноручно создадите игровой контроллер и научитесь управлять объектами на экране с помощью датчиков. Чтобы усилить эффект, воспроизводимый игрой, в качестве устройства вывода используйте проектор.

Данный проект проще реализовать с помощью платформы `Raspberry Pi`, поскольку к ней легко подключается внешнее видеоустройство, для чего на плату добавлен разъем `HDMI`. Подключение телевизора или проектора к `Arduino` — задача не столь простая.

Нельзя сказать, что перед нами стоит легко решаемая задача. Игра `Pong` разрабатывалась во времена низкопроизводительных процессоров и скромных объемов оперативной памяти. Рабочие характеристики настольных систем были тогда значительно хуже, чем в `Arduino`, — изображение на мониторах формировалось в результате построчной развертки, а вычислительной мощности едва хватало, чтобы поспевать за событиями на экране. Если вы серьезно намерены посвятить себя разработке старых игр на `Arduino`, то обратите внимание на плату расширения `Video Game Shield` от компании `Wayne and Layne`.



Рис. 6.20. Игровое поле

Получаемые навыки

В проекте игры `Pong` вы научитесь следующему:

- использовать данные с датчика для управления объектами на экране;
- выводить изображение с `Raspberry Pi` на мониторы высокой четкости;
- запускать `Raspberry Pi` в виде отдельной компьютерной системы, а не из среды разработки, чтобы увеличить производительность ее работы;
- создавать простые программы на `Python` с использованием библиотеки `pyGame`;
- автоматически запускать программы при загрузке `Raspberry Pi`.

Подключение контроллеров

На рис. 6.21 показана схема электрической цепи для проекта игры Pong. Подключите все необходимое оборудование и выполните программу, приведенную в листинге 6.12. Убедитесь, что библиотека `botbook_gpio.py` находится в том же каталоге, что и программа `pong.py`.

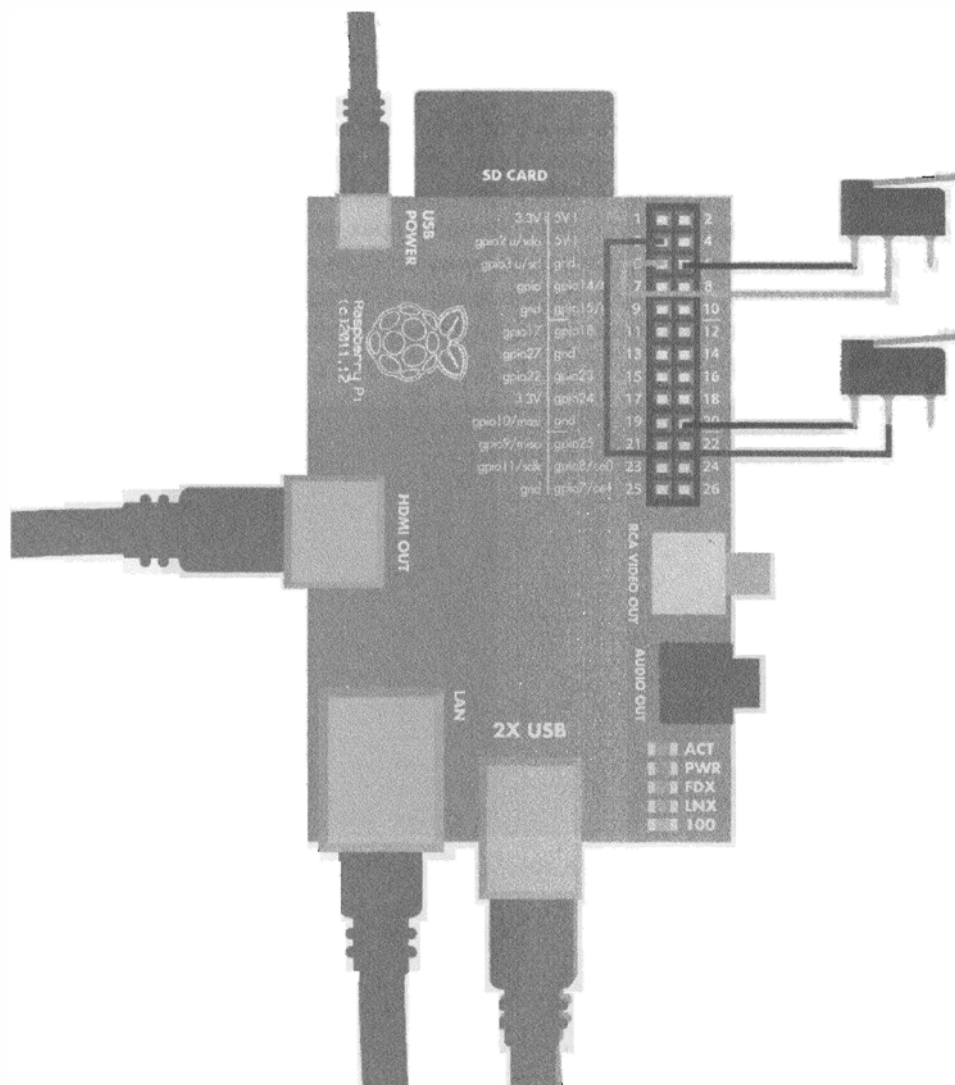


Рис. 6.21. Подключение оборудования к Raspberry Pi для получения игрового контроллера

Листинг 6.12. `pong.py`

```
# pong.py - классический теннис, управляемый джойстиком  
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```



```

import time
import sys
import pygame
import botbook_gpio as gpio
from pygame.locals import *

print "Загрузка игры Pong от BotBook.com..."
pygame.init() # ❶

width = pygame.display.Info().current_w # ❷
height = pygame.display.Info().current_h

size = width, height # ❸
background = 0, 0, 0 # ❹
screen = pygame.display.set_mode(size, pygame.FULLSCREEN) # ❺
normalSpeed = 512
ballrect = Rect(width/2, height/2, 16, 16) # ❻
computerrect = Rect(width-20, 0, 20, 120) # ❼
playerrect = Rect(0, 0, 20, 120) # ❽
# Разделение движения вдоль осей X и Y. Мячик отражается под
# углом 45 градусов
speed = [normalSpeed, normalSpeed] # ❾
clock = pygame.time.Clock() # ❿
pygame.mouse.set_visible(False)
mainloop = True

uppin = 2
downpin = 3
gpio.mode(uppin, "in")
gpio.mode(downpin, "in")

while mainloop: # ❶
    seconds = clock.tick(30) / 1000.0 # секунд со времени
                                         # отображения последнего кадра # ❶

    # Входные данные
    for event in pygame.event.get(): # ❶
        if event.type == pygame.QUIT: mainloop = False # ❶
        if (event.type == KEYUP) or (event.type == KEYDOWN):
            if event.key == K_ESCAPE: mainloop = False

    # Перемещение и столкновения
    playerspeed = 0
    if gpio.read(uppin) == gpio.LOW:
        playerspeed = -normalSpeed
    if gpio.read(downpin) == gpio.LOW:
        playerspeed = normalSpeed
    ballrect.x += speed[0] * seconds # ❶
    ballrect.y += speed[1] * seconds
    if ballrect.left < 0 or ballrect.right > width: # ❶
        ballrect.x = width/2;
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]

```

```

computerrect.y = round(ballrect.y) # ⑦
playerrect.y += playerspeed * seconds # ⑧
if playerrect.top < 0: playerrect.top = 0 # ⑨
if playerrect.bottom > height: playerrect.bottom = height # ⑩
if computerrect.colliderect(ballrect): # ⑪
    speed[0] = -normalSpeed

if playerrect.colliderect(ballrect):
    speed[0] = normalSpeed

# Отображение кадра
screen.fill(background)
pygame.draw.circle(screen, (255, 255, 255),
    ⑤ (int(round(ballrect.x+8)), int(round(ballrect.y+8))), 10) # ⑫
pygame.draw.rect(screen, (255, 255, 255), computerrect) # ⑬
pygame.draw.rect(screen, (255, 255, 255), playerrect)
pygame.display.update() # ⑭

```

- ① Перед использованием библиотеку `pygame` нужно сначала подключить.
- ② В `pygame` задается рабочее поле с заранее определенными размерами. Это позволяет работать в точном координатном пространстве, без необходимости применения относительных величин. Переменные `width` и `height` определяют ширину и высоту игрового поля соответственно.
- ③ Размеры игрового поля представляется кортежем: `(width, height)`.
- ④ В библиотеке `pygame` применяется цветовая палитра RGB (Red, Green, Blue — красный, зеленый, синий). Вы должны быть знакомы с такой палитрой, если хотя бы немного занимались разработкой веб-страниц. Значение `(0, 0, 0)` соответствует черному цвету.
- ⑤ Объект `screen` представляет собой область, в которой происходят действия игры. Он применяется в коде ближе к концу основного цикла программы.
- ⑥ Создание ограничивающих прямоугольных рамок для объектов на экране: `Rect(x, y, width, height)`. Мячик начинает свое движение с левого верхнего угла (`y=0, x=0`). Ограничивающая рамка для мячика имеет размер `16×16` пикселей.
- ⑦ Ракетка компьютерного игрока располагается возле правой границы игрового поля. Ее ширина составляет `20` пикселей (`px`), длина — `120` пикселей.
- ⑧ Ракетка пользователя начинает свое движение на поле с координаты `(0, 0)`. Ее размеры также составляют `20×120` пикселей.
- ⑨ Определение вектора скорости мячика. Каждую секунду мячик перемещается на расстояние `64` пикселя вдоль оси `X` и на `64` пикселя вдоль оси `Y`.
- ⑩ Создание объекта `Clock` и сохранение его в новой переменной `clock`. Он необходим для поддержания стабильной скорости перемещения мячика по игровому полю даже при увеличении производительности микроконтроллера Raspberry Pi.

- 11 Библиотека `pyGame` имеет программную структуру с основным циклом, как и все программы для `Raspberry Pi`. Этот прием программирования очень распространен при разработке игр. Подобно основному циклу `loop()` в `Arduino` и циклической структуре `while (True)`, в программах, написанных на `Python`, программный код библиотеки выполняется циклически бесконечное количество итераций. В каждой итерации нашей игры в основном цикле сначала обрабатываются входные данные (представляющие действия игрока), перемещаются все необходимые объекты, проверяются такие события, как столкновения, и, наконец, прорисовывается один кадр видеоизображения.
- 12 Функция `clock.tick()` возвращает количество времени, прошедшего с момента ее последнего запуска. При однократном вызове в каждой итерации основного цикла (для каждого кадра видеоизображения) вы получите время отображения на экране предыдущего кадра. Параметр 30 представляет максимальную частоту смены кадров — 30 Гц, или 30 кадров в секунду. Чтобы предотвратить отображение кадров игры с большей скоростью, применяется функция `tick()`, определяющая соответствующую задержку. Для лучшего понимания программного кода миллисекунды (1/1000 с), возвращаемые функцией `tick()`, преобразованы в секунды.
- 13 Вводимые с клавиатуры данные обрабатываются в объекте `pygame.event`. Метод `pygame.event.get()` возвращает объект, содержащий очередь доступных для обработки событий. Данный цикл можно условно представить как последовательный обработчик событий, просматривающий в каждой итерации отдельное событие из очереди. Первым обрабатывается первое событие из очереди событий, вторым — второе и так далее до последнего элемента в очереди событий.
- 14 Сравнение атрибутов каждого из событий с заранее заданными в `pyGame` значениями; при совпадении выполняются действия, соответствующие событию. Например, если пользователь совершает на экране действие, которое должно приводить к завершению программы, то обрабатывается событие `QUIT`, и программа завершает свою работу.
- 15 Перемещение мячика. Поскольку время счета определяется при прорисовке каждого кадра, то мячик всегда будет перемещаться на стандартные 64 пикселей в секунду, даже несмотря на изменение частоты смены кадров.
- 16 Если мячик достигает границы игровой области, то засчитывается очко!
- 17 Перемещение ракетки компьютерного игрока на высоту расположения мячика. У вас очень серьезный соперник!
- 18 Перемещение ракетки обычного игрока в вертикальном направлении в соответствии с ускорением, заданным клавишами на клавиатуре. Выражение `a+=2` обозначает математическое уравнение `a=a+2`.
- 19 Если в результате перемещения ракетка игрока должна выйти за верхнюю границу игровой области, то необходимо зафиксировать ее у верхнего края.

- 20 Если в результате перемещения ракетка игрока должна выйти за нижнюю границу игровой области, то необходимо зафиксировать ее у нижнего края.
- 21 Проверка столкновения мячика с любой из ракеток. Если столкновение произошло, то направление движения мячика изменяется на противоположное.
- 22 Прорисовка мячика в рассчитанном месте игрового поля.
- 23 Ракетка компьютерного игрока отображается в ранее рассчитанном месте. Поскольку она имеет прямоугольную форму, то ее ограничительная рамка совпадает с самой ракеткой.
- 24 Одновременная прорисовка всех объектов текущего кадра.

Корпус игрового контроллера

В качестве прочного и долговечного корпуса для нашего игрового контроллера мы использовали штампованную в заводских условиях коробку, показанную на рис. 6.22. Ее утилитарный вид несколько не мешает устройству свои функции, хотя она и не выглядит столь привлекательно, как большинство малопрочных пластиковых изделий.

В задней части корпуса мы проделали отверстие под кабели питания и HDMI (рис. 6.23). Чтобы получить продолговатое отверстие, мы просверлили рядом два небольших отверстия, а затем удалили получившуюся перемычку с помощью напильника.

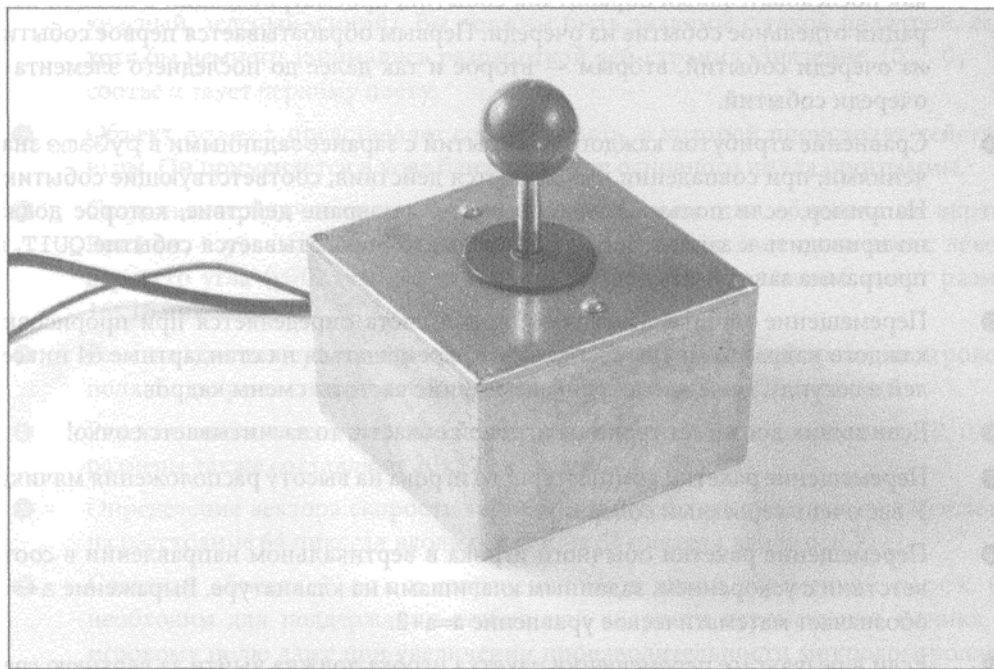


Рис. 6.22. Игровой контроллер в корпусе

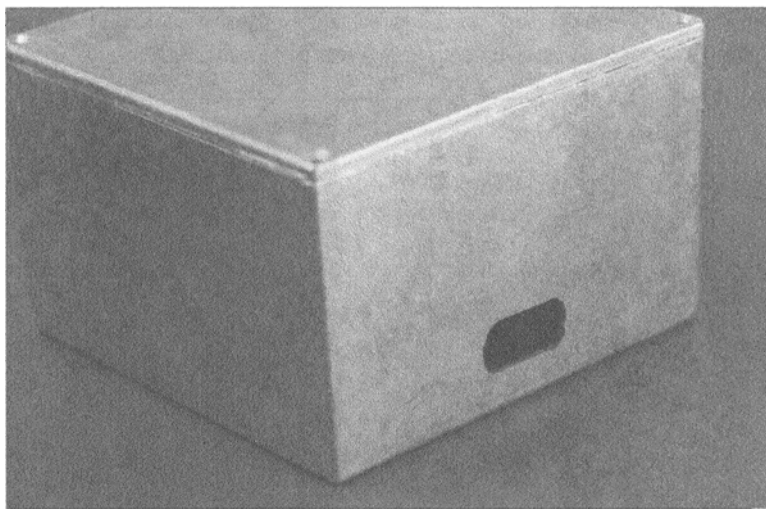


Рис. 6.23. *Отверстие для кабелей питания и HDMI*

Традиционный джойстик для игровых контроллеров должен надежно фиксироваться в корпусе, чтобы исключить люфт в любом из рабочих направлений (рис. 6.24). Для долговечного крепления джойстика нам нужно будет просверлить с каждой стороны по три отверстия: два отверстия диаметром 5 мм — для фиксации накладной рамки, а еще одно, диаметром 10 мм, — для крепления рамки к алюминиевому корпусу (рис. 6.25).

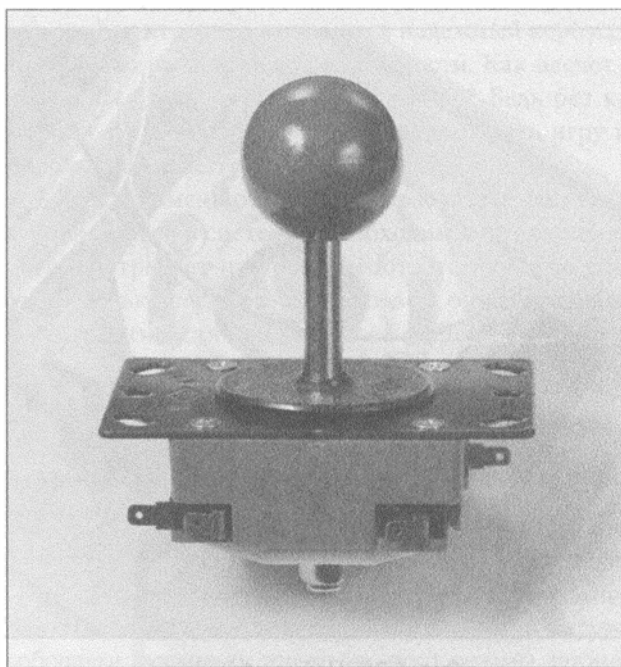


Рис. 5.24. *Игровой джойстик*

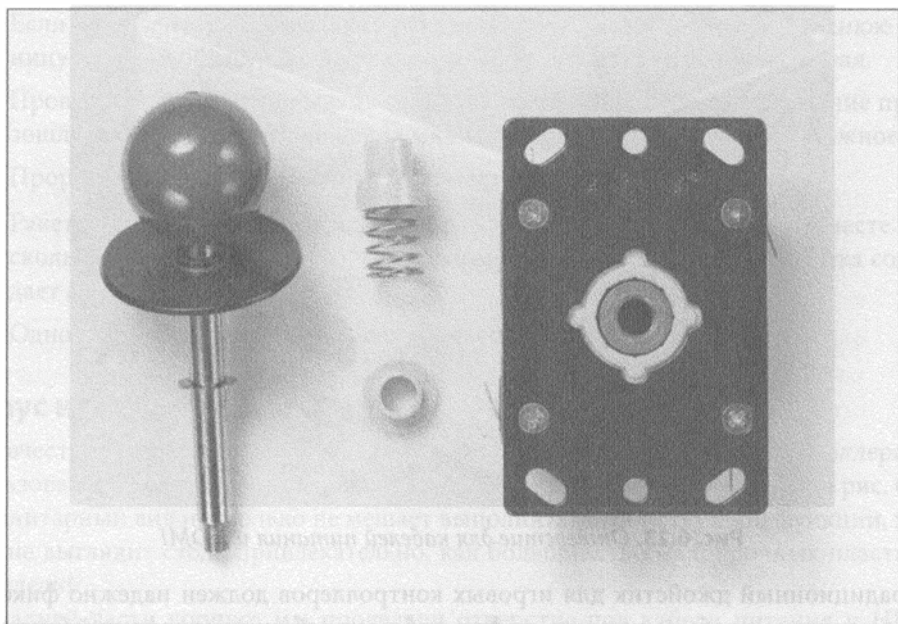


Рис. 6.25. Демонтаж рычага джойстика

Поскольку мы будем управлять ракеткой только в вертикальном направлении, то припаивать провода нужно только к двум выводам джойстика, как показано на рис. 6.26.

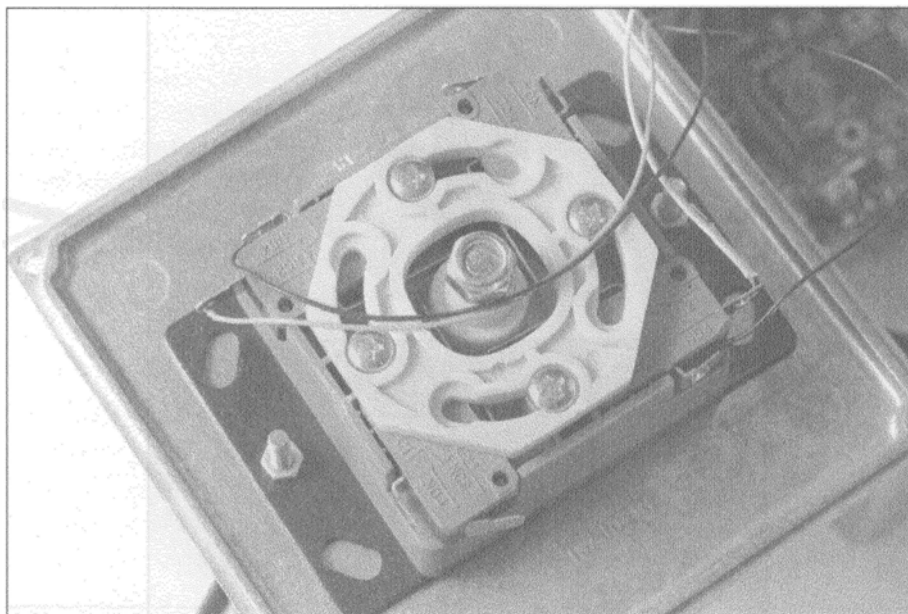


Рис. 6.26. Пайка проводов

Закрепление платы Raspberry Pi выполняется уже привычным вам образом — с помощью термоклея или клея к нижней внутренней части корпуса (рис. 6.27).

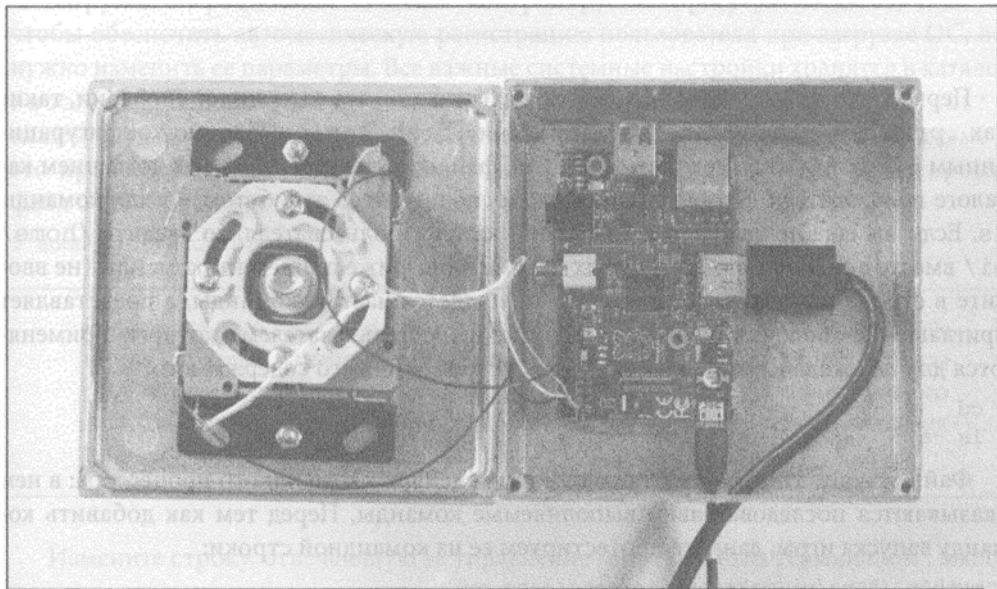


Рис. 6.27. Все компоненты в корпусе

Автоматический запуск игры при загрузке Raspberry Pi

Теперь, когда игровой контроллер заключен в надежный корпус, попробуем улучшить нашу игру, несколько расширив ее возможности. Как насчет того, чтобы игра автоматически запускалась при загрузке Raspberry Pi? Ведь без клавиатуры очень сложно ввести команду `python pong.py`, чтобы загрузить игру из операционной системы, установленной в Raspberry Pi.

Чтобы запустить игру от имени обычного пользователя, вам сначала нужно автоматически зарегистрироваться в системе с необходимыми привилегиями. Автоматический запуск программ требует настройки соответствующего сценария для конкретного пользователя. Только после выполнения всех описанных выше действий игра начнет запускаться автоматически при загрузке Raspberry Pi.

Запуск игры при регистрации

При успешной регистрации в системе вы попадаете в Bash (Bourne again shell) — это ваша командная оболочка: она интерпретирует в операционной системе команды, вводимые в текстовом виде после символа приглашения (\$).

Если Raspberry Pi автоматически загрузилась до графического интерфейса (это вариант по умолчанию), то вам нужно перенастроить ее на загрузку в режиме командной оболочки, поскольку только из этого режима автоматически запускаются сторонние программы. Запустите LXTerminal и выполните команду

sudo raspi-config. Далее выберите в меню вариант **Enable Boot To Desktop/Scratch** и укажите загружать ОС в режиме командной оболочки, а не с графическим интерфейсом.

Первое, что делает оболочка Bash при загрузке, — это выполняет сценарии, такие как `.profile`, `.login`, `.bash_profile` и `.bash_login`. Подобно конфигурационным файлам для каждого пользователя, файлы сценариев скрыты в домашнем каталоге пользователя. Чтобы увидеть их, воспользуйтесь атрибутом `-a` для команды `ls`. Если вы еще не перешли в домашний каталог пользователя, то введите `/home/pi/` вместе с `cd` без дополнительных параметров. Ниже приведены команды (не вводите в строке символ `$`, поскольку он не является частью команды, а представляет приглашение оболочки на обмен информацией с пользователем), которые применяются для перехода в домашний каталог и отображения его содержимого.

```
$ cd
$ ls -a
```

Файл `.bash_login` представляет собой сценарий командного процессора: в нем указываются последовательно выполняемые команды. Перед тем как добавить команду запуска игры, давайте протестируем ее из командной строки:

```
$ python /home/pi/makesensors/pong/pong.py
```

Замените `makesensors` на путь к каталогу, в котором в вашем случае хранится файл `pong.py`. Нажмите клавишу `<Esc>` для выхода из игры. Если программа игры выполняется, то откройте файл `.bash_login` в редакторе `nano`:

```
$ nano .bash_login
```

Добавьте в файл `.bash_login` строку `python /home/pi/makesensors/pong/pong.py`. Если в файле уже имеются какие-то строки, то смело удалите их. В листинге 6.13 показан пример того, как должен выглядеть файл `.bash_login`. Сохраните измененный файл сценария, нажав комбинацию клавиш `<Ctrl+X>`, а затем подтвердите выполнение операции нажатием клавиши `<Y>`. Оставьте имя файла без изменений, для чего нажмите `<Enter>` или `<Return>`.

Листинг 6.13. `.bash_login`

```
# /home/pi/.bash_login - автоматический запуск игры Pong при
# регистрации в системе
python /home/pi/makesensors/pong/pong.py
```

Выйдите из системы с помощью следующей простой команды:

```
$ exit
```

Войдите в систему повторно; игра запустится автоматически.

Автоматический вход

Нам осталось только обеспечить автоматическую регистрацию в Raspberry Pi при загрузке операционной системы. Поскольку игра автоматически запускается после регистрации пользователя в системе, то вам останется только подать питание на

плату, чтобы запустить игру. Если вы все еще находитесь в игре, то нажмите клавишу <Esc>, чтобы выйти из нее.

Загрузка операционной системы контролируется программой `init`, поэтому, чтобы обеспечить автоматическую регистрацию пользователя при загрузке ОС, вам нужно изменить ее параметры. Все важные системные настройки хранятся в каталоге `/etc/`, а название конфигурационного файла обычно начинается с `/etc/init*` (в нашем случае `/etc/inittab`). Поскольку регистрация пользователей в системе — это процесс, требующий максимальных привилегий, то вам нужно редактировать файл от имени суперпользователя с помощью команды `sudoedit`:

```
$ sudoedit /etc/inittab
```

Редактирование текстовых файлов от имени суперпользователя лучше выполнять с помощью команды `sudoedit`, а не `nano` с префиксом `sudo`. В таком случае при последующем применении команды `nano` от имени обычного пользователя вы не получите сообщение об ошибке неправильного владения файлом истории этой программы.

Измените строку, отвечающую за управление “виртуальным терминалом”, выполняющим автоматическую регистрацию пользователя (ни в коем случае не добавляйте еще одну такую строку, а всего лишь измените содержимое уже имеющейся строки, которая начинается со значения `1:2345:respawn:/sbin/getty`):

```
1:2345:respawn:/sbin/getty --noclear 38400 tty1 --autologin pi
```

Сохраните файл, нажав комбинацию клавиш <Ctrl+X>, а на запрос сохранения файла нажмите клавишу <Y>, после чего подтвердите операцию сохранения нажатием клавиши <Enter> или <Return>. Готовый к использованию файл `/etc/inittab` показан в листинге 6.14.

Завершите работу Raspberry Pi такой командой:

```
$ sudo shutdown -P now
```

Отключите и снова подайте питание через разъем USB.

Просто наблюдайте за происходящим на экране, так как регистрация пользователя будет проводиться автоматически. Игра также запустится автоматически, и вам останется всего лишь ухватиться за собственноручно созданный контроллер и отправиться ставить новые рекорды. Чемпионат по электронному теннису считается открытым!

Листинг 6.14. `inittab`

```
# /etc/inittab - автоматическая регистрация в Raspberry Pi
# (сопровождается отключением монитора последовательного порта)
```

```
id:2:initdefault:
si::sysinit:/etc/init.d/rcS
~~:S:wait:/sbin/sulogin
10:0:wait:/etc/init.d/rc 0
11:1:wait:/etc/init.d/rc 1
```

```

12:2:wait:/etc/init.d/rc 2
13:3:wait:/etc/init.d/rc 3
14:4:wait:/etc/init.d/rc 4
15:5:wait:/etc/init.d/rc 5
16:6:wait:/etc/init.d/rc 6
z6:6:respawn:/sbin/sulogin
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
# изменения:
1:2345:respawn:/sbin/getty --noclear 38400 tty1 --autologin pi
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
# освобождение последовательного порта для датчиков
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

```

Старайтесь всегда запускать программы от имени обычного пользователя, если это возможно. Удел суперпользователя — это администрирование операционной системы (такие команды, как `apt-get`, `raspi-config`). Особенно важно запускать от имени обычного пользователя игры, поскольку они требуют прямого доступа к встроенному оборудованию, а программы, получающие полный доступ к системным ресурсам, могут нанести в случае критической ошибки в коде непоправимый вред. Именно поэтому игры лучше запускать после автоматической регистрации пользователя (как описано выше), а не просто путем добавления соответствующей команды в загрузочный сценарий, например `rc.local`, который всегда выполняется от имени суперпользователя.

Чтобы прервать игру и вернуться к командной оболочке, нажмите клавишу <Esc>.

В этой главе вы научились распознавать движение несколькими способами, заключающимися в обработке сигналов, поступающих с самых разных устройств, даже таких примитивных, как кнопки и потенциометры. Как ни странно, но они также относятся к резистивным датчикам. Кнопка — это простейший цифровой (состояние “Вкл./Выкл.” или нулевое/бесконечное сопротивление) резистивный датчик, а потенциометр — классический аналоговый резистивный датчик. В дальнейшем вы сможете использовать программный код, описанный в примерах этой главы, в проектах, в которых задействованы другие, более сложные датчики.

Вы также узнали, как правильно управлять некоторыми узкоспециализированными датчиками, такими как датчик прикосновения, который способен распознать касание даже через деревянную доску. Вы научились измерять нажим, что находит практическое применение при определении того, сидит ли человек на стуле или лежит на кровати, а также в простых силомерах.

В следующей главе мы перейдем к изучению менее осязаемой формы материи — света.

Все мы уже давно привыкли к робототехнике на крупном производстве. Вы наверняка видели, как в автосборочных цехах роботы развозят комплектующие вдоль специально проложенных дорожек и делают это без видимых задержек и трудностей. При внимательном изучении несложно заметить четкую черную линию, проложенную вдоль их маршрута. Становится понятно, что роботы умеют отслеживать линии с помощью специальных датчиков. Такие же, но несколько упрощенные датчики встречаются повсеместно; например, на улицах они включают наружное освещение при наступлении темноты и выключают его рано утром.

Поскольку цвет — это отраженный свет, то датчику света специального типа не составит большого труда определить оттенки окружающих его объектов. Поместив датчик света в цилиндрический футляр, вы ограничите его рабочую область и сможете определить расположение источника света. Конструируя робот для пожарного депо, вам точно не обойтись без датчика пламени.

Хотите научиться отслеживать перемещения людей с помощью инфракрасных датчиков? Обратитесь к предыдущей главе. Вам нужно узнать, располагается ли что-то на удалении в темном помещении? Тогда обратитесь к проектам, описанным в главе 3.

Эксперимент: обнаружение пламени (датчик пламени)

Пламя излучает инфракрасный свет, спектрально сильно отличающийся от инфракрасной составляющей дневного света. Эта особенность применяется в датчике пламени KY-026, который отслеживает интенсивность инфракрасного излучения пламени, представляя выходные данные в виде изменяющегося сопротивления (рис. 7.1).

Программный код управления датчиком пламени ничем не отличается от такового для аналогового резистивного датчика: считывание сигнала с него выполняется с помощью функции `analogRead()`.

Датчик пламени KY-026 поддерживает два режима вывода данных: цифровой (сигнал считывается функцией `digitalRead()`) и аналоговый (считывание выполняется функцией `analogRead()`). В программном коде данного примера реализованы оба варианта, но в собственных проектах вы сможете выбрать только один из них.

Цифровой режим работы датчика особенно хорош при подключении его к Raspberry Pi, поскольку эта платформа не оснащается встроенным аналого-цифровым преобразователем.

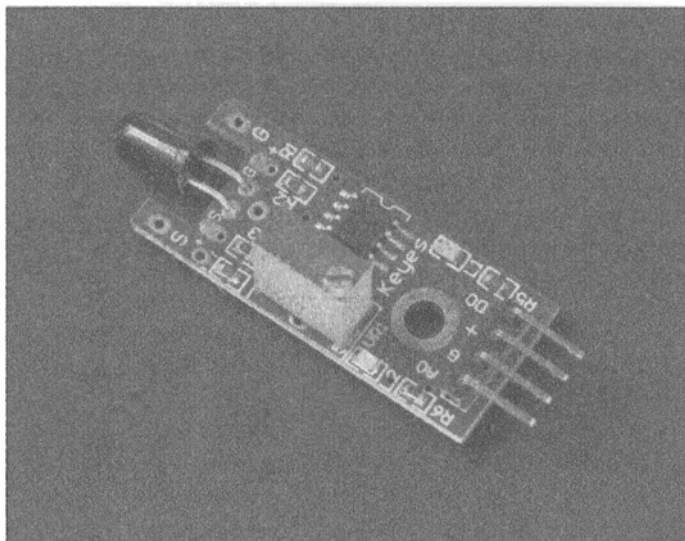


Рис. 7.1. Датчик пламени



Рис. 7.2. Прототип робота, следующего за источником пламени (мастерская робототехники в Австрии)

Подключение к Arduino и программа управления датчиком пламени

На рис. 7.3 показана схема подключения датчика пламени к Arduino. После соединения показанного оборудования в единую цепь выполните программный код из листинга 7.1.

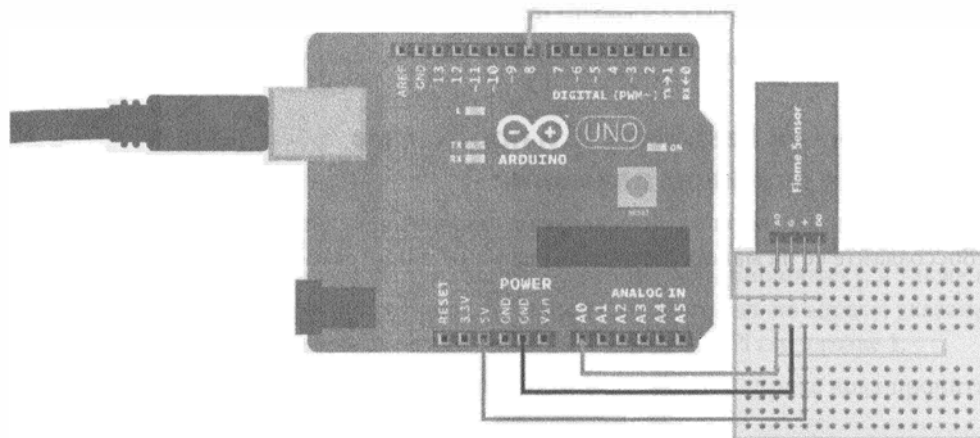


Рис. 7.3. Схема подключения датчика пламени к Arduino

Листинг 7.1. `ky_026_flame.ino`

```
// ky_026_flame.ino - вывод интенсивности ИК-излучения пламени
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const int analogPin = A0;
const int digitalPin = 8;
const int ledPin = 13;

void setup() {
    Serial.begin(115200);
    pinMode(digitalPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    int threshold = -1; // HIGH или LOW
    int value = -1; // 0..1023
    value = analogRead(analogPin); // ❶
    threshold = digitalRead(digitalPin); // ❷
    Serial.print("Аналоговый сигнал: ");
    Serial.print(value);
    Serial.print("Цифровой сигнал: ");
    Serial.println(threshold);
    delay(10);
    if(threshold==HIGH) { // ❸
        digitalWrite(ledPin, HIGH);
    }
}
```

```

    } else {
        digitalWrite(ledPin, LOW);
    }
}

```

- ❶ Считывание абсолютного значения напряжения с вывода A0. Напряжение представляется числовыми значениями в диапазоне от 0 (0 В) до 1023 (+5 В).
- ❷ Считывание сигнала на выводе D8: LOW (0 В) или HIGH (+5 В).
- ❸ Включение встроенного светодиода (D13) при обнаружении пламени.

Подключение к Raspberry Pi и программа управления датчиком пламени

На рис. 7.4 показана электрическая цепь подключения датчика пламени к Raspberry Pi. Воссоздайте ее на макетной плате и выполните программу из листинга 7.2.

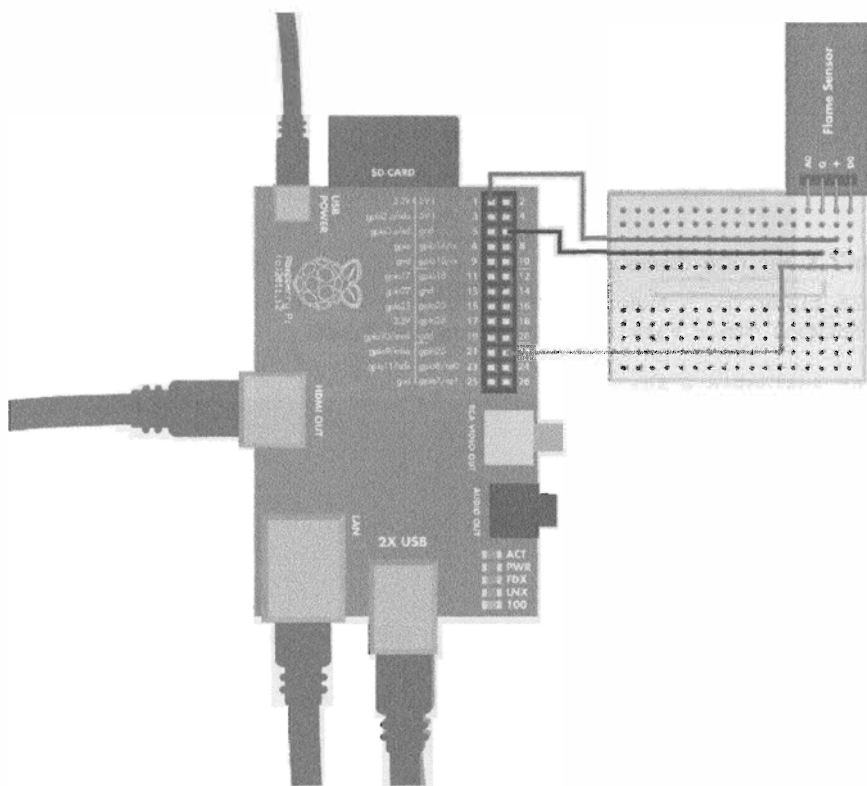


Рис. 7.4. Схема подключения датчика пламени к Raspberry Pi

Листинг 7.2. `ky_026_flame.py`

```

# ky_026_flame.py - распознавание ИК-излучения пламени
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

```

```
import time
import botbook_gpio as gpio # ❶

def main():
    triggerPin = 25
    gpio.mode(triggerPin, "in") # ❷
    flame = gpio.read(triggerPin) # ❸
    if(flame == gpio.HIGH): # ❹
        print "Горим!"
    else:
        print "Нет пламени"
    time.sleep(0.5)

if __name__ == "__main__":
    main()
```

- ❶ Убедитесь, что копия библиотеки `botbook_gpio.py` сохранена в том же каталоге, что и текущая программа. Можете загрузить эту библиотеку, а также все файлы примеров, рассматриваемых в книге, обратившись по адресу, указанному во введении. О настройке портов GPIO в Raspberry Pi см. в главе 1.
- ❷ Вывод переводится в режим входа, чтобы в следующей строке кода прочитать сигнал с датчика.
- ❸ Получение статуса вывода 22. Значение `True` соответствует сигналу высокого уровня (+3,3 В); `False` — сигналу низкого уровня (0 В).
- ❹ Булево значение (`True` или `False`) позволяет избежать применения оператора сравнения. Таким образом, можно использовать упрощенную конструкцию `if (b)` вместо `if (b==True)`.

Эксперимент в окружающей среде: ярче пламя!

Встроенный фоторезистор датчика пламени обладает высокими рабочими характеристиками. Главная его особенность заключается в том, что он не реагирует на инфракрасное излучение, присутствующее в спектре дневного освещения. Более того, его всегда можно настроить на восприятие узкого спектра ИК-излучения, характерного для горения только определенных материалов.

Для начала подключите дополнительный светодиод к выводам GND и 13 (рис. 7.6). Внешний светодиод светится ярче, и за ним проще наблюдать при срабатывании датчика пламени. Загрузите программный код управления датчиком пламени (см. листинг 7.1) в Arduino. Прокрутите регулятор потенциометра на датчике максимально вправо, чтобы убедиться в работоспособности светодиода, а затем поверните регулятор до упора влево, добившись прекращения тока через светодиод. Неплохо при этом запахнуть на окнах шторы, чтобы солнечный свет не мог превзойти по интенсивности остальные источники освещения.

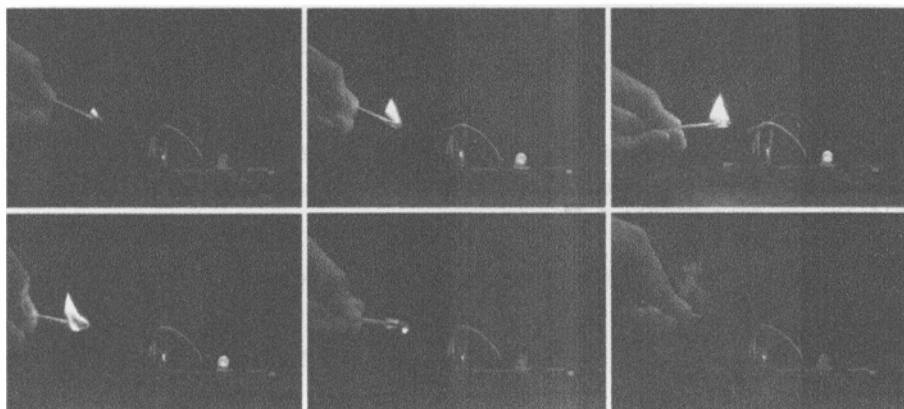


Рис. 7.5. Проверка и настройка чувствительности датчика пламени

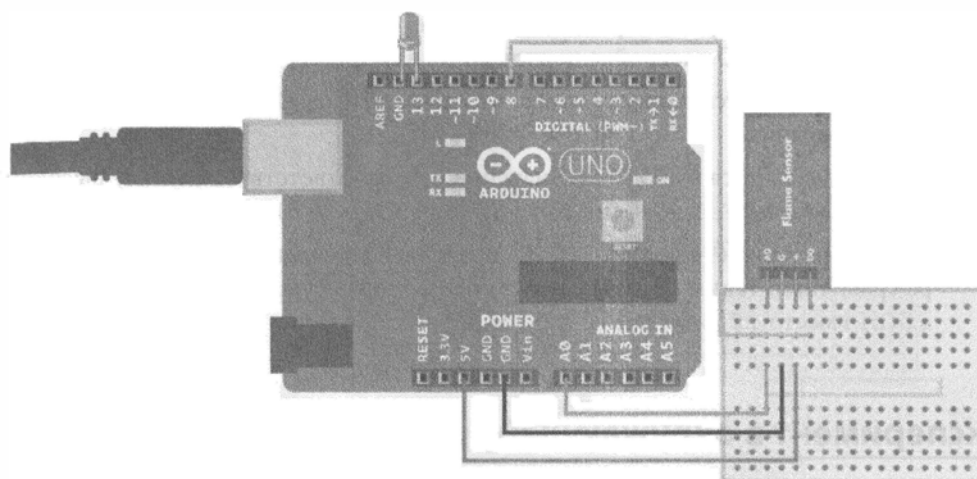


Рис. 7.6. Включение сигнального светодиода в цепь датчика пламени

Теперь зажгите спичку. Светодиод должен загореться снова. Подстройте чувствительность фоторезистора так, чтобы датчик срабатывал только на полноценное пламя. И не забывайте, что у вас в руках горящая спичка. Потушите ее до того, как она обожжет вам кончики пальцев!

Эксперимент: увидеть свет (фоторезистор)

Фоторезистор (Light-Dependent Resistor — LDR) изменяет свое сопротивление в зависимости от интенсивности падающего на него света видимого диапазона спектра. Чем выше уровень освещенности, тем меньше сопротивление фоторезистора (рис. 7.7). Фоторезисторы иногда путают с фотодиодами.

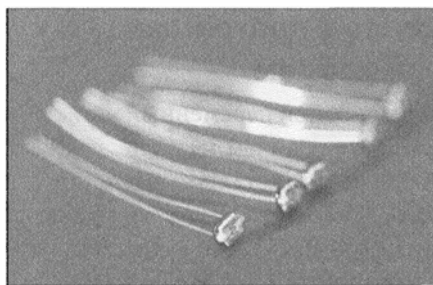


Рис. 7.7. Фоторезистор

Фоторезисторы нашли широкое применение в технике. С их помощью включается наружное освещение при наступлении ночи, они применяются в системах управления роботами, а также в простых сигнализациях, реагирующих на внешнее освещение. При должной сноровке вы сможете использовать их для создания датчиков, определяющих направление распространения света в помещении. При тестировании фоторезистора, чтобы полностью исключить попадание на него света, просто закройте его пальцем. Для максимального освещения фоторезистора направьте на него яркий фонарик.

Тестирование робота, движущегося на свет, в светлое время суток нужно проводить под плотным покрывалом, как показано на рис. 7.8. Конечно, лучше всего дожидаться ночи или проводить эксперименты в помещении без окон за плотно закрытыми дверями.

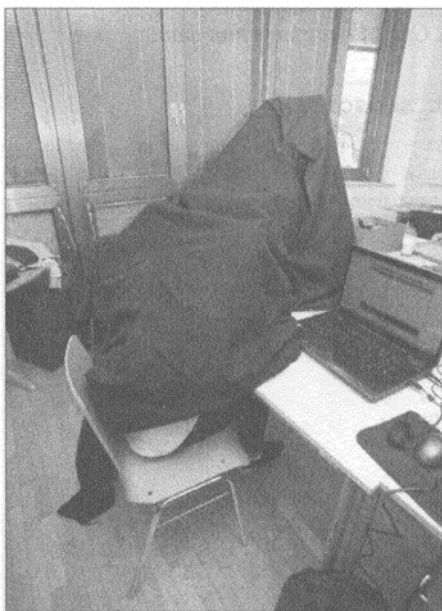


Рис. 7.8. Импровизированный экран, не пропускающий внутрь естественное освещение (мастерская робототехники в Австрии)

Подключение к Arduino и программа управления фоторезистором

Фоторезистор представляет собой переменный резистор с двумя выводами. С точки зрения платформы Arduino он ничем не отличается от потенциометра или любого другого делителя напряжения (детально о принципе работы и управлении потенциометром см. в главе 5). Считывание показаний фоторезистора осуществляется стандартной функцией `analogRead()`, применяемой в проектах с другими типами аналоговых резистивных датчиков.

Подключите фоторезистор к Arduino, как показано на рис. 7.9, и выполните программный код из листинга 7.3.

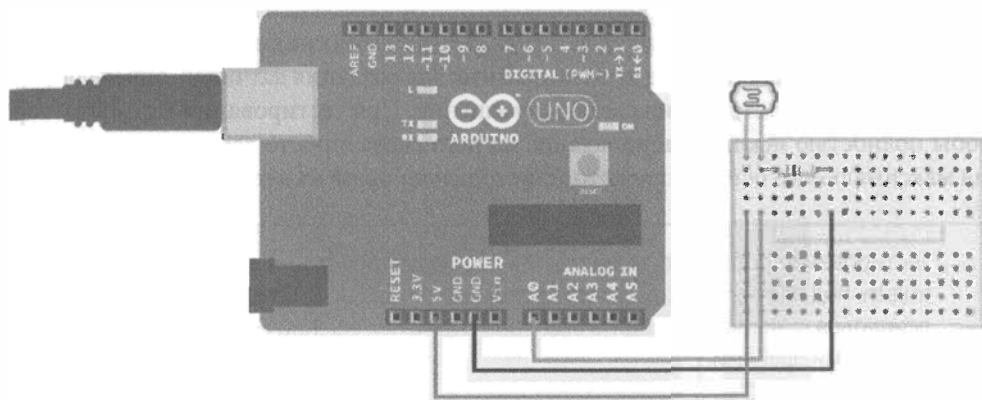


Рис. 7.9. Схема подключения фоторезистора к Arduino

Листинг 7.3. `ldr_light_sensor.ino`

```
// ldr_light_sensor.ino - определение высокого уровня
// освещенности с помощью фоторезистора
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

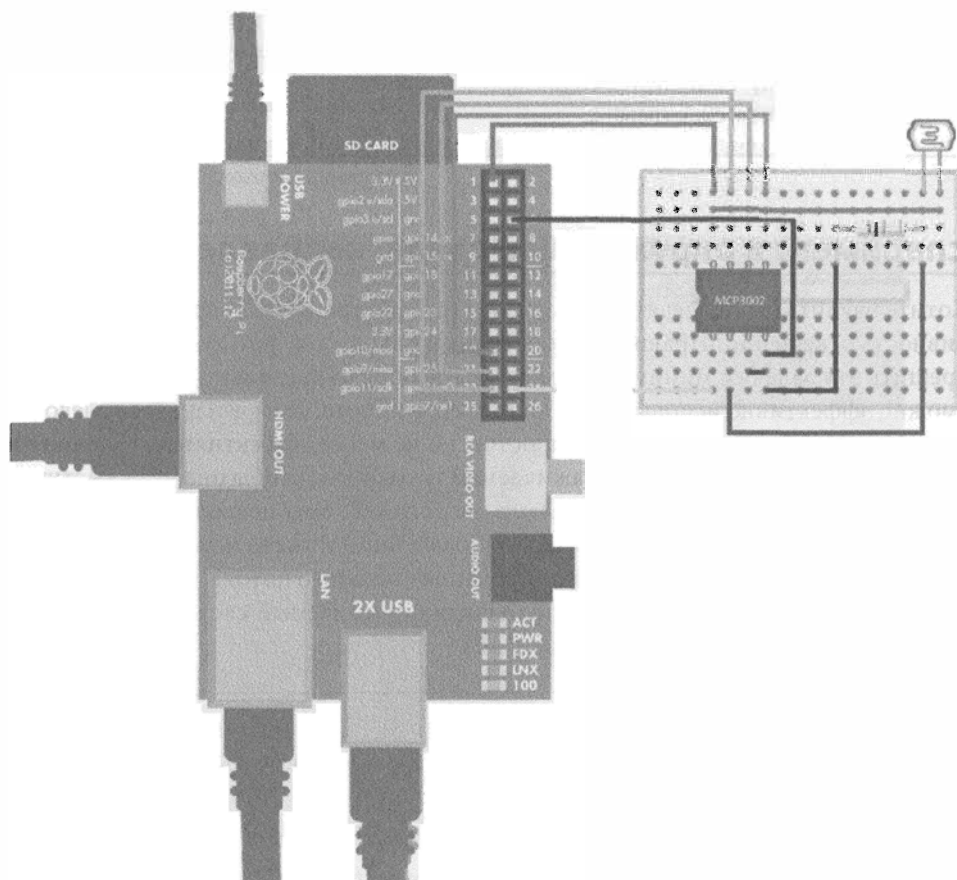
const int lightPin = A0;
const int ledPin = 13;
int lightLevel = -1;

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  lightLevel = analogRead(lightPin); // ❶
  Serial.println(lightLevel);
  if(lightLevel < 400) { // ❷
    digitalWrite(ledPin, HIGH);
  } else {
```

```
    digitalWrite(ledPin, LOW);
}
delay(10);
}
```

Подключение к Raspberry Pi и программа управления фоторезистором



Листинг 7.4. ldr.py

```
# ldr.py - измерение и вывод на экран уровня освещенности
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp

def main():
    while True:
        lightLevel = mcp.readAnalog()
        print("Уровень освещенности: %i " % lightLevel)
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Файл библиотеки `botbook _ mcp3002.py` должен располагаться в том же каталоге, что и файл текущей программы, `ldr.py`. Кроме того, вам нужно подключить библиотеку `spidev`, применяемую в библиотеке `botbook _ mcp3002`. Подробнее об этом см. в главе 3.

Эксперимент в окружающей среде: направленный свет

Хотите научиться определять направление распространения света, а не только интенсивность освещения? Правильно подобранный экран для фоторезистора поможет вам решить эту задачу с минимальными затратами на оборудование. Подобный датчик можно эффективно использовать, например, в роботе, который должен следовать за источником света. Решение очень простое, но не менее эффективное. Наденьте на датчик с выводами отрезок термоусаживаемой трубки так, чтобы она чуть выступала за край приемника датчика, как показано на рис. 7.11. Это позволит экранировать свет, поступающий на датчик с боковых направлений. Можете использовать другие материалы и приспособления для экранирования светового потока, идущего к датчику. Главное — не перестараться и не уменьшить принимаемый световой поток ниже уровня, распознаваемого датчиком (рис. 7.12).

При использовании сразу нескольких фоторезисторов можно заключать в термоусаживаемую трубку не только датчик, но и все его выводы, а также подтягивающий резистор (рис. 7.13).

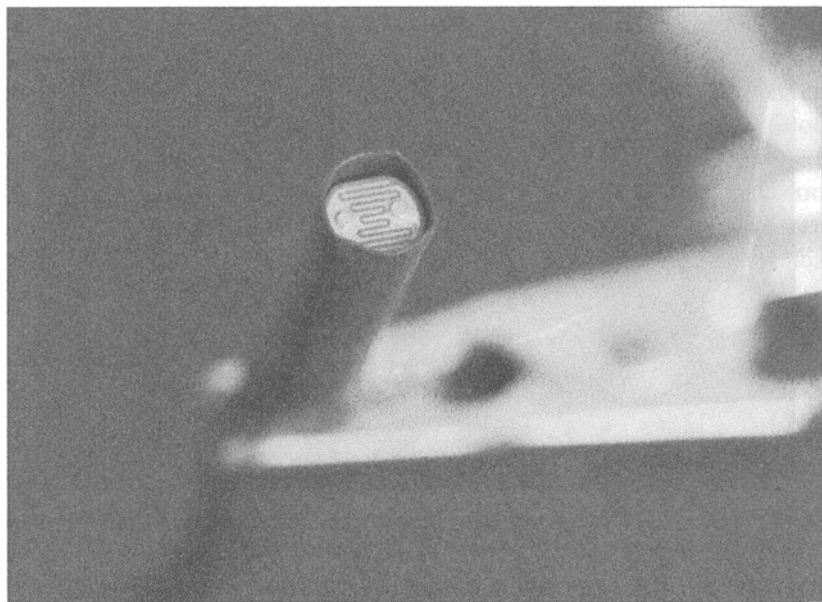


Рис. 7.11. Такая конструкция позволяет свету попадать на датчик только с одного направления



Рис. 7.12. Фоторезисторы закреплены внутри одноразовых пластиковых стаканчиков (мастерская робототехники в Австрии)

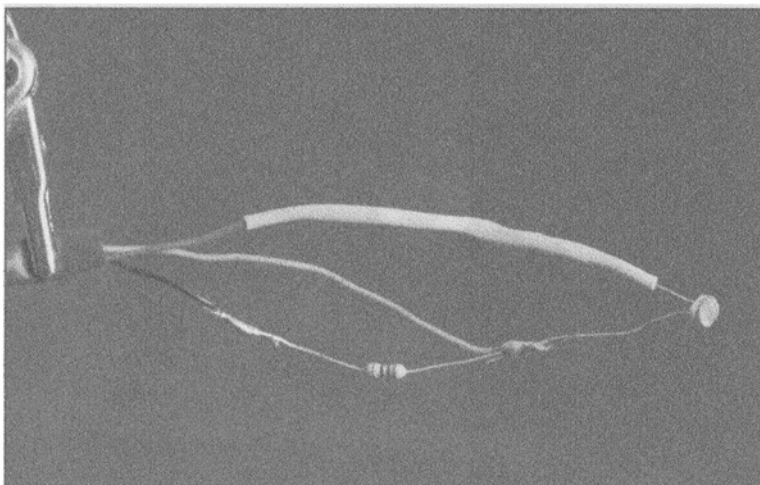


Рис. 7.13. Термоусаживаемые трубки значительно упрощают упаковку проводов

Эксперимент: следи за линией (детектор линий)

На многих машиностроительных производствах комплектующие развозятся автоматически с помощью роботов, которые следуют по заранее проложенным маршрутам. Самый простой способ обозначить маршрут для такого робота — нарисовать на напольном покрытии черную полосу. Ничего удивительного в поведении роботов нет, они прекрасно справляются с распознаением линий на полу, поскольку снабжены специальными детекторами, подобными показанным на рис. 7.14.

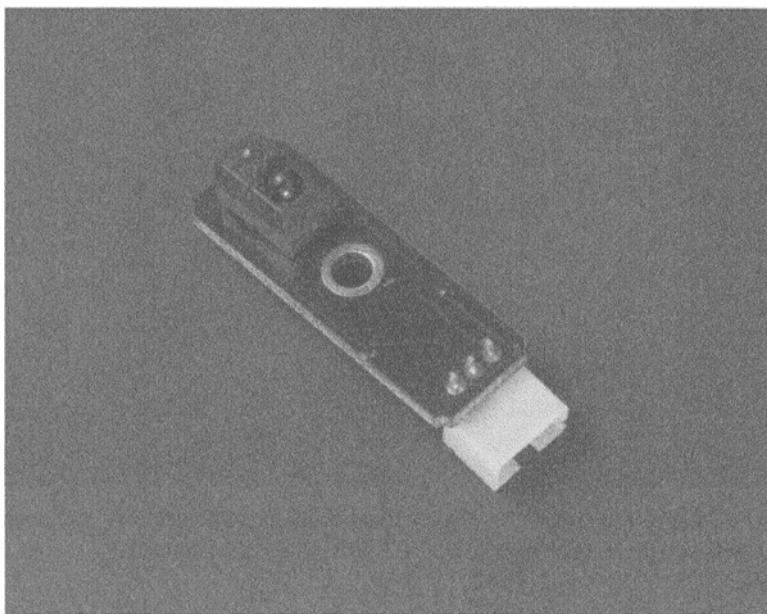


Рис. 7.14. Детектор линий

Обычно детекторы линий освещают поверхность, расположенную перед ними, с помощью инфракрасного источника света. При этом поверхность представляется детектору белой, если от нее отражается большая часть излученного ими света. Для определения направления смещения в сторону от начерченной полосы в роботах используется сразу по несколько детекторов линий, закрепляемых с каждой стороны. Алгоритм работы робота прост: если центральный детектор распознает линию, а два боковых видят только белое пространство, то это означает, что робот движется в правильном направлении. В продаже можно найти готовое устройство распознавания линий, совмещающее в одном модуле несколько детекторов линий.

Подключение к Arduino и программа управления детектором линий

В нашем примере мы используем трехвыводной детектор линий, поэтому дополнительно снабжать его подтягивающим резистором не нужно. На один из выводов подается питание, средний вывод используется как сигнальный, а третий вывод заземляется. Плата Arduino снабжена всем необходимым для обеспечения корректной работы датчика.

Создайте электрическую цепь согласно схеме, показанной на рис. 7.15, и загрузите в Arduino программный код, приведенный в листинге 7.5.

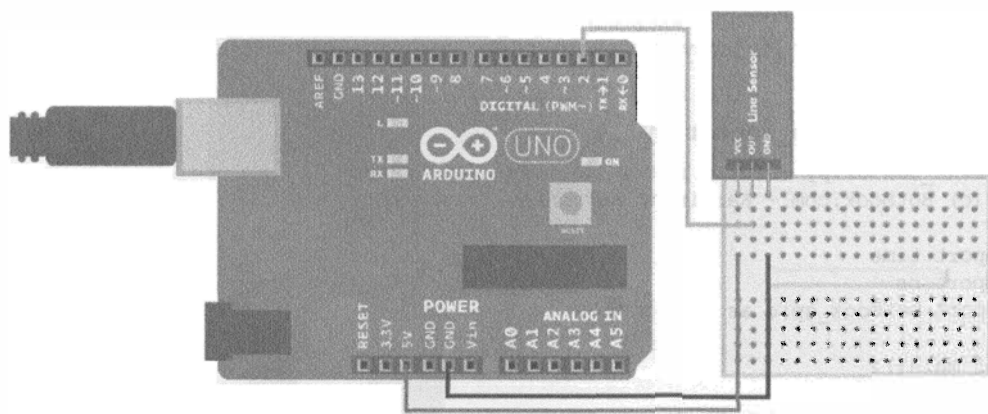


Рис. 7.15. Схема подключения детектора линий к Arduino

Листинг 7.5. line_sensor.ino

```
// line_sensor.ino - вывод данных о слежении за линией
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;

void setup() {
    Serial.begin(115200);
    pinMode(sensorPin, INPUT);
```

```

    pinMode(ledPin, OUTPUT);
}

void loop() {
    lineFound = digitalRead(sensorPin); // ❶
    if(lineFound == HIGH) {
        Serial.println("На маршруте");
        digitalWrite(ledPin, HIGH);
    } else {
        Serial.println("Маршрут потерян");
        digitalWrite(ledPin, LOW);
    }
    delay(10);
}

```

- ❶ Функция `digitalRead(8)` возвращает значение `HIGH` (сигнал высокого уровня), если датчик продолжает распознавать линию.

Подключение к Raspberry Pi и программа управления детектором линий

Поскольку детектор линий — это цифровой датчик, то его подключение к Raspberry Pi мало чем отличается от подключения к Arduino. На рис. 7.16 приведена схема подключения, следуя которой вы получите электрическую цепь управления датчиком линий в Raspberry Pi. Загрузите и выполните программный код, приведенный в листинге 7.6.

Листинг 7.6. `line_sensor.py`

`line_sensor.py` - вывод данных о слежении за линией
 # (c) BotBook.com - Karvinen, Karvinen, Valtokari

```

import time
import os
import botbook_gpio as gpio # ❶

def main():
    linePin = 23
    gpio.mode(linePin, "in")
    while True:
        lineState = gpio.read(linePin) # ❷
        if( lineState == gpio.HIGH ):
            print "На маршруте"
        else:
            print "Сбился с маршрута"
            time.sleep(0.5)

if __name__ == "__main__":
    main()

```

- ❶ Убедитесь, что копия файла библиотеки `botbook_gpio.py` сохранена в том же каталоге, что и файл текущей программы. Можете загрузить библиотеку, а

также все файлы примеров, рассматриваемых в книге, по адресу, указанному во введении. В главе 1 детально рассказывалось о настройке портов GPIO в Raspberry Pi.

- ② Сигнал считывается так же, как и в случае любого другого цифрового датчика.

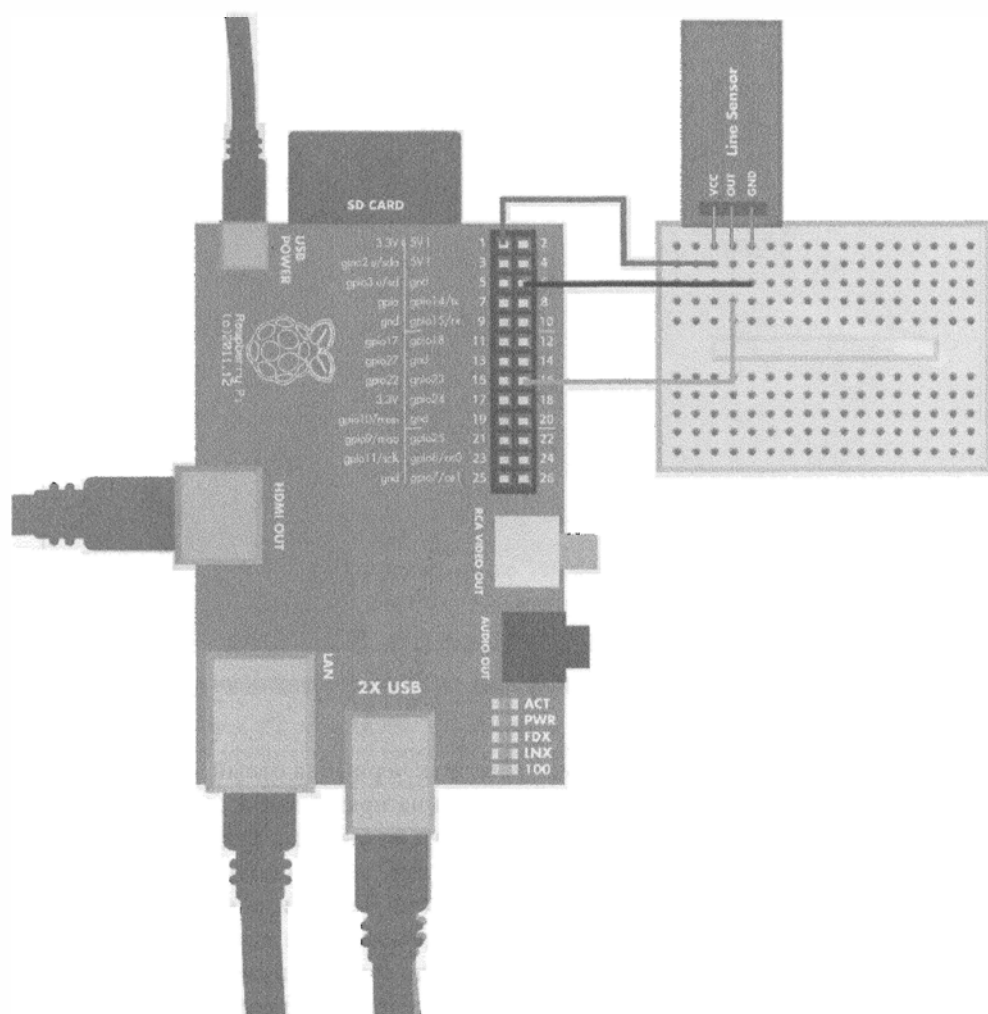


Рис. 7.16. Схема подключения детектора линий к Raspberry

Эксперимент в окружающей среде: черное или белое?

Скорее всего, вы знаете, что инфракрасные датчики воспринимают окружающее их пространство совсем не так, как мы с вами. Все материалы отражают свет, особенно инфракрасного диапазона, по-своему, и очень часто темные для нас объекты

воспринимаются датчиком как светлые, и наоборот. Если ваш робот,двигающийся вдоль черной полосы на полу, начинает перемещаться по странной траектории, то не спешите винить в этом программу или использованное оборудование. Вероятно, причина кроется в неоднородной текстуре поверхности, которая вводит и вас, и робота в заблуждение.

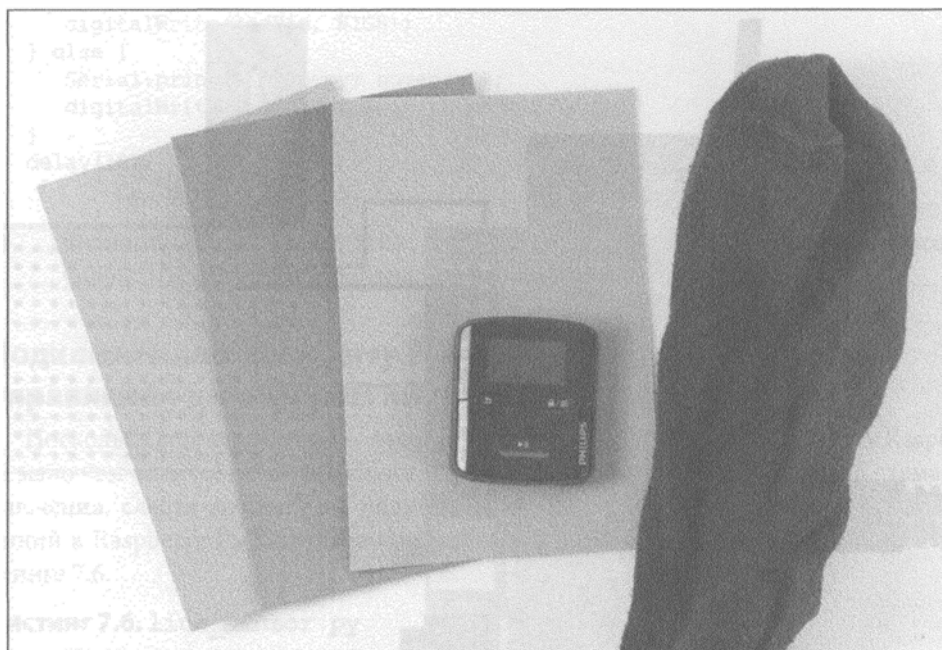


Рис. 7.17. Хотите — верьте, хотите — нет, но датчик воспримет все эти объекты как белые (см. цветную вклейку)

Как правило, черный цвет воспринимается как черный, а белый — как белый, но случаются и досадные недоразумения. Однажды на презентации своего робота, которая проходила на выставке Maker Faire, нам нужно было соорудить из подручных средств подиум. Мы не придумали ничего лучшего, как воспользоваться большой картонной коробкой, проклеенной черной изолянтной лентой. Черная изолянта клеилась только по краям импровизированной платформы. Идея состояла в том, что робот будет поворачивать назад, как только распознает черную границу платформы, тем самым никогда не выезжая за ее пределы. Как выяснилось позже, детектор линий распознавал картонную поверхность и черную изолянтную ленту совсем не так, как предполагалось, поскольку гладкая изолянта сильно отражает свет, а потому кажется светлой. При этом картон с его шероховатой поверхностью прекрасно поглощает ИК-излучение, поэтому для детектора линий выглядит темным.

Чувствительность детектора линий регулируется с помощью встроенного потенциометра, как показано на рис. 7.18.

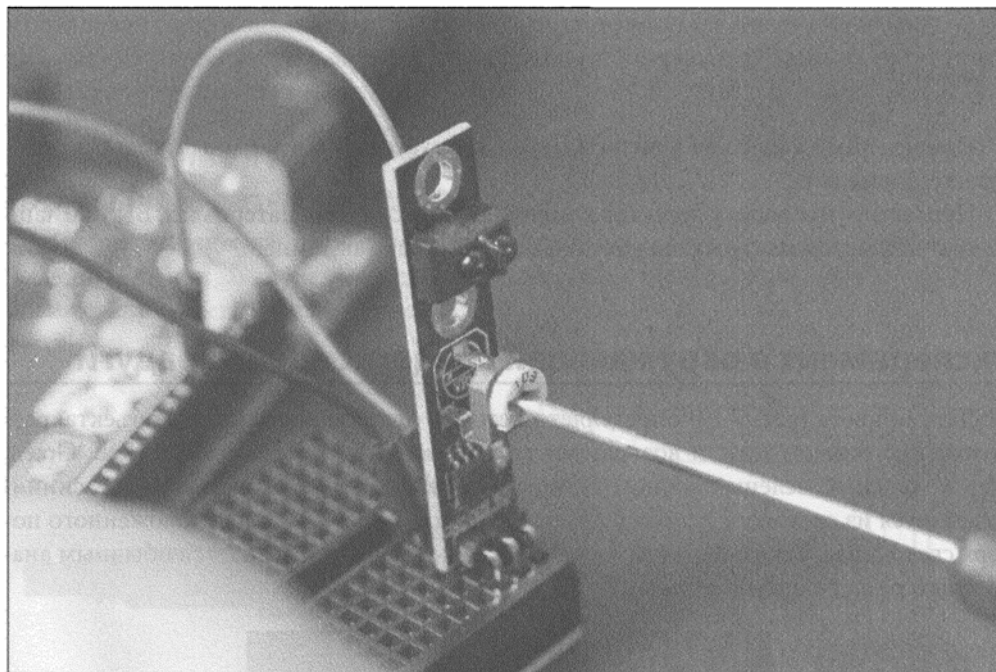


Рис. 7.18. Настройка чувствительности детектора линий

Загрузите код, приведенный в листинге 7.7, в Arduino. Мы немного изменили предыдущий вариант для того, чтобы на монитор последовательного порта выводились значения BLACK и WHITE в зависимости от исследуемой поверхности.

Листинг 7.7. line_sensor_black_or_white.ino

```
// line_sensor_black_or_white.ino - детектор линий.  
// Сигнал низкого уровня соответствует темной поверхности.  
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const int sensorPin = 2;  
const int ledPin = 13;  
int lineFound = -1;  
  
void setup() {  
  Serial.begin(115200);  
  pinMode(sensorPin, INPUT);  
  // В подтягивающем резисторе нет смысла  
  pinMode(ledPin, OUTPUT);  
}  
  
void loop() {  
  lineFound = digitalRead(sensorPin);  
  if(lineFound == 1) {  
    Serial.println("BLACK");  
    digitalWrite(ledPin, HIGH);  
  } else {  
    Serial.println("WHITE");  
  }  
}
```

```

    digitalWrite(ledPin, LOW);
}
delay(50);
}

```

Приведенный выше код в полной мере согласуется с программным кодом предыдущего примера.

Протестируйте работу детектора линий на самых разных материалах. Вы уже нашли светлый материал, который датчик воспринимает как темный?

Эксперимент в окружающей среде: все цвета радуги

Датчик цвета (рис. 7.19) определяет цветовой оттенок поверхности объекта, расположенного перед ним, и возвращает его компоненты в модели RGB (Red, Green, Blue — красный, зеленый, синий). Каждый базовый цвет (красный, зеленый, синий) выделяется из светового потока с помощью цветového фильтра, расположенного поверх светодиода. Таким образом, каждый основной цвет распознается обычным аналоговым резистивным датчиком.

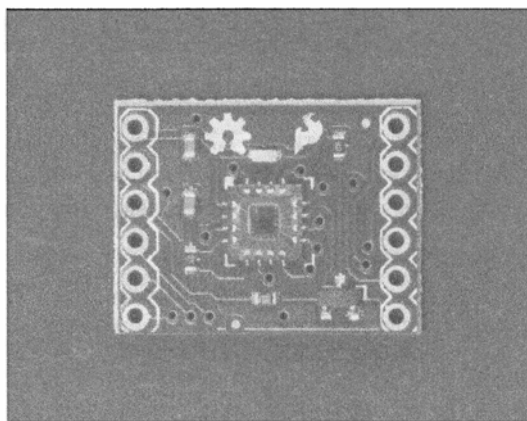


Рис. 7.19. Датчик цвета

В следующем примере мы научимся выводить на монитор значения цветовых компонентов модели RGB, распознаваемые датчиком цвета, а впоследствии воспользуемся полученными наработками для создания цветового купола.

Каждый цвет имеет кроме названия еще и числовое обозначение, соответствующее определенной длине волны электромагнитного излучения. Например, электромагнитная волна длиной 555 нм (нанометров) или частотой 540 ТГц (терагерц) соответствует зеленому цвету. Далеко не все животные распознают все цвета одинаково, более того, некоторые не распознают привычные нам оттенки видимой части спектра, но зато прекрасно различают инфракрасное и ультрафиолетовое излучение.

Основные цвета модели RGB в полной мере соответствуют цветовому восприятию окружающего мира человеческим глазом. Наши глаза различают всего три основных цвета: красный, зеленый и синий, смешение которых в разных пропорциях

и приводит к распознаванию всевозможных оттенков. Изображение формируется на сетчатке глаза, состоящей из трех типов колбочек, отвечающих за восприятие основных цветов видимого спектра.

Подключение к Arduino и программа управления датчиком цвета

На рис. 7.20 показана схема подключения датчика цвета к Arduino. Соедините показанные на нем компоненты в единую электрическую цепь и выполните программный код, приведенный в листинге 7.8.

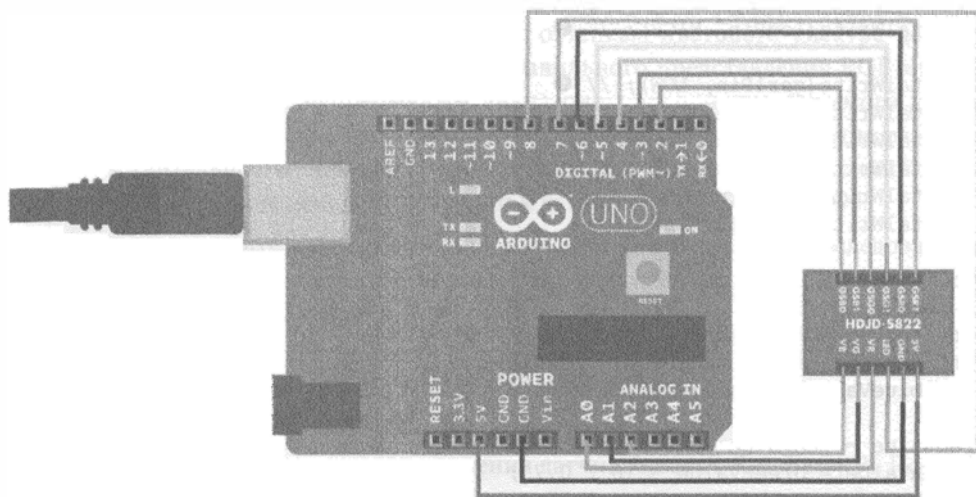


Рис. 7.20. Схема подключения датчика цвета к Arduino (см. цветную вклейку)

Листинг 7.8. color_sensor.ino

```
// color_sensor.ino - распознавание цветов датчиком
// HDJD-S822-QR999 и вывод их RGB-значений
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsrlPin = 7; // ❶
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;

const int ledPin = 8; // ❷

const int redPin = A0; // ❸
const int greenPin = A1;
const int bluePin = A2;
```

```

int red = -1; // ❶
int green = -1;
int blue = -1;

void setup() {
  Serial.begin(115200);
  pinMode(gsr1Pin, OUTPUT);
  pinMode(gsr0Pin, OUTPUT);
  pinMode(gsg1Pin, OUTPUT);
  pinMode(gsg0Pin, OUTPUT);
  pinMode(gsb1Pin, OUTPUT);
  pinMode(gsb0Pin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  digitalWrite(ledPin, HIGH); // ❷

  digitalWrite(gsr1Pin, LOW); // ❸
  digitalWrite(gsr0Pin, LOW);
  digitalWrite(gsg1Pin, LOW);
  digitalWrite(gsg0Pin, LOW);
  digitalWrite(gsb1Pin, LOW);
  digitalWrite(gsb0Pin, LOW);
}

void loop() {
  int redValue = analogRead(redPin); // ❹
  int greenValue = analogRead(greenPin);
  int blueValue = analogRead(bluePin);

  redValue = redValue * 10 / 1.0; // ❺
  greenValue = greenValue * 10 / 0.75;
  blueValue = blueValue * 10 / 0.55;

  Serial.print(redValue); Serial.print(" "); // ❻
  Serial.print(greenValue); Serial.print(" ");
  Serial.print(blueValue); Serial.println(" ");
  delay(100);
}

```

- ❶ Определение выводов, позволяющих настроить чувствительность датчика к отдельным цветам. В случае необходимости проводится калибровка отдельных цветов, заключающаяся в изменении чувствительности датчика, например, к красному (gsr1Pin, gsr0Pin), зеленому (gsg1Pin, gsg0Pin) или синему (gsb1Pin, gsb0Pin) оттенку. В результате вы подкорректируете уровень восприятия указанных цветовых значений.
- ❷ Светодиод освещает анализируемую поверхность, и его работоспособность критически важна для правильной работы датчика. Светодиод смонтирован на плате датчика цветов.
- ❸ Выводы, с которых считываются интенсивности красной, зеленой и синей составляющих цвета.

- ④ Переменные, в которых сохраняются значения, считываемые с выводов датчика. Исходно переменным назначены невозможные значения, что упрощает дальнейшую отладку программного кода (если в процессе выполнения программы вы видите эти значения на мониторе, значит, она работает неправильно).
- ⑤ Включение встроенного светодиода для освещения диагностируемой поверхности.
- ⑥ Отключение режима калибровки датчика для основных цветов. Чувствительность к каждому основному цвету регулируется двумя битами (0 и 1) или представляется четырьмя уровнями (включая отсутствие калибровки).
- ⑦ Считывание значений производится стандартной функцией `analogRead()`.
- ⑧ Чувствительность датчика к разным оттенкам цветового спектра далеко не одинакова. С другой стороны, для правильного представления RGB-цвета необходим датчик с непрерывным диапазоном восприятия видимого диапазона светового спектра. Коэффициенты усиления, компенсирующие различную чувствительность к цветам (10 — красный, 14 — зеленый, 17 — синий), указываются в технической документации к датчику HDJD-S822-QR999.
- ⑨ Вывод каждого цветового RGB-значения на монитор последовательного порта (отображается на экране с помощью команды `Tools⇒Serial Monitor` (Сервис⇒Монитор порта)).

Подключение к Raspberry Pi и программа управления датчиком цвета

На рис. 7.21 показана схема подключения датчика цвета к Raspberry Pi. После создания соответствующей электрической цепи выполните программный код из листинга 7.9.

Распутываем провода

Взглянув на схему подключения датчика цвета к Raspberry Pi, вы хватаетесь за голову? Если неправильно подключить или не подключить хотя бы одну проволочную перемычку, то серьезных проблем не избежать.

Существуют два способа уменьшения количества соединений на монтажной плате: по возможности упростить схему или скомбинировать Arduino и Raspberry Pi в один рабочий модуль. Конечно, совмещение обеих платформ не только выглядит конструктивнее,

но и вызывает неподдельный интерес самой возможностью такой операции.

В подобной схеме датчик цвета подключается к Arduino, а затем считанные ею RGB-значения передаются в Raspberry Pi через последовательное USB-соединение. Получение цветовых оттенков в Arduino (рис. 7.20) выполняется стандартной функцией `analogRead()`, не поддерживаемой в Raspberry Pi, поскольку последняя не оснащается встроенным аналого-цифровым преобразователем.

В Raspberry Pi получение данных через последовательное USB-соединение реализуется с использованием библиотеки pySerial.

Существует еще один, противоречивый способ уменьшения количества проволочных соединений на макетной плате: использование единственного аналого-цифрового преобразователя с большим количеством каналов, например MCP3008. Но чтобы воспользо-

зоваться им в текущем проекте, вам придется изменить программный код библиотеки botbook_mcp3002.

Если вы ни при каких обстоятельствах не планируете калибровать датчик цвета, то просто заземлите соответствующие выводы платы (таким образом, на них всегда будет подаваться сигнал низкого уровня). Количество соединений несколько уменьшится.

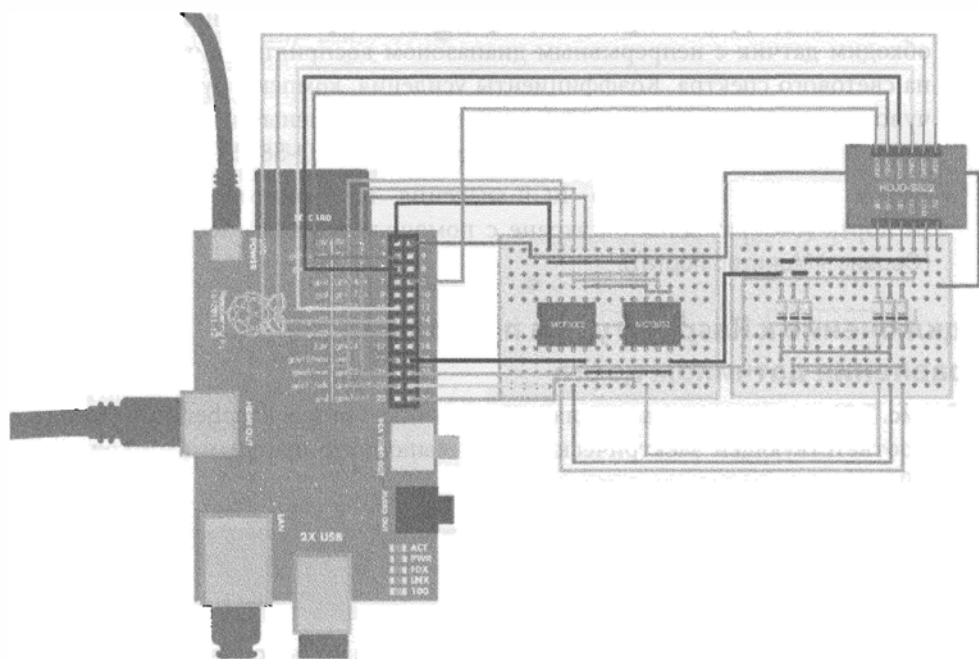


Рис. 7.21. Схема подключения датчика цвета к Raspberry Pi (см. цветную вклейку)

Листинг 7.9. color_sensor.py

color_sensor.py - распознавание и вывод RGB-оттенков
(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_mcp3002 as mcp # ❶
import botbook_gpio as gpio

def initializeColorSensor():
    ledPin = 25
    gpio.mode(2, "out") # ❶
```



```

gpio.mode(3,"out")
gpio.mode(14,"out")
gpio.mode(17,"out")
gpio.mode(22,"out")
gpio.mode(27,"out")

gpio.write(2,gpio.LOW)
gpio.write(3,gpio.LOW)
gpio.write(14,gpio.LOW)
gpio.write(17,gpio.LOW)
gpio.write(22,gpio.LOW)
gpio.write(27,gpio.LOW)

gpio.mode(ledPin,"out")
gpio.write(ledPin, gpio.HIGH) # ❶

def main():
    initializeColorSensor()
    while True: # ❷
        redValue = mcp.readAnalog(0, 0)
        greenValue = mcp.readAnalog(0, 1)
        blueValue = mcp.readAnalog(1, 0) # ❸

        redValue = redValue * 10 / 1.0; # ❹
        greenValue = greenValue * 10 / 0.75;
        blueValue = blueValue * 10 / 0.55;

        print("R: %d, G: %d, B: %d" %
❷ (redValue,greenValue,blueValue)) # ❶
        time.sleep(0.1) # c

if __name__ == "__main__":
    main()

```

- ❶ Обе библиотеки, `botbook_mcp3002.py` и `botbook_gpio.py`, необходимо разместить в том же каталоге, что и файл программы (`color_sensor.py`). Кроме них в каталог конечной программы нужно также включить библиотеку `spidev`, подключаемую в библиотеке `botbook_mcp3002`. Детально об использовании указанных библиотек см. в комментариях к файлу `botbook_mcp3002.py` или в главе 3. Загрузить эти библиотеки, а также файлы примеров, рассматриваемые в книге, можно по адресу, указанному во введении. В главе 1 детально рассказывается о настройке портов GPIO в Raspberry Pi.
- ❷ Отключение калибровки цветов подачей на соответствующие выводы (gs*) сигнала низкого уровня (LOW).
- ❸ Включение осветительного светодиода, встроенного в датчик цвета. Без подсветки датчик не сможет правильно оценить цветовой оттенок анализируемой поверхности.
- ❹ Программа продолжает выполняться до нажатия комбинации клавиш <Ctrl+C>.

- ❹ Считывание `readAnalog(device=1, channel=0)` сигнала с первого канала второго аналого-цифрового преобразователя модуля MCP3002 (счет в обоих случаях ведется со значения 0). В предыдущих командах указываются другие комбинации устройства и канала.
- ❺ Согласование уровней цветовых значений в соответствии с техническими характеристиками датчика цветов HDJD-S822-QR999. В результате все основные цвета приводятся к одному уровню яркости.
- ❻ Создание выводимого сообщения с помощью формирующей строки, в которой задействуется только один параметр, поэтому выводимые данные представлены в виде кортежа значений.

Пилотный проект: цветовой купол

В последнем проекте этой главы мы создадим купол, приобретающий цвет поверхности, на которой он располагается. Мы воспользуемся программным кодом управления датчиком цвета для указания цветового оттенка свечения RGB-светодиода. Заклучив работающее оборудование в сферический корпус, вы получите невероятно эффектный ночной светильник.

Получаемые навыки

В проекте цветového купола вы научитесь следующему:

- создавать устройства, изменяющие свой цвет подобно хамелеону;
- изменять цвет свечения RGB-светодиода;
- применять скользящее среднее для исключения шумов;
- использовать степенную функцию для масштабирования входных и выходных значений.

RGB-светодиод

В RGB-светодиоде (рис. 7.22) скомбинированы три монохромных светодиода, хотя и выглядит он как обычный светодиод с большим количеством “ножек”. Изменяя интенсивности свечения красной, зеленой и синей составляющих, вы сможете добиться свечения RGB-светодиода любым цветовым оттенком.

Человеческий глаз также имеет рецепторы для восприятия трех основных цветов светового спектра: красного, зеленого и синего. Именно поэтому данные цвета чаще остальных применяются для передачи других цветовых оттенков в самых разных технических устройствах. Особенность (или недостаток) цветовосприятия человека заключается в распознавании комбинации двух световых волн разной частоты как световой волны третьей частоты. В реальной жизни это означает, что комбинацию красного и зеленого цвета мы воспринимаем как желтый цвет.

Стандартный RGB-светодиод имеет четыре вывода: по одному для каждого из основных RGB-цветов и один общий для всех.

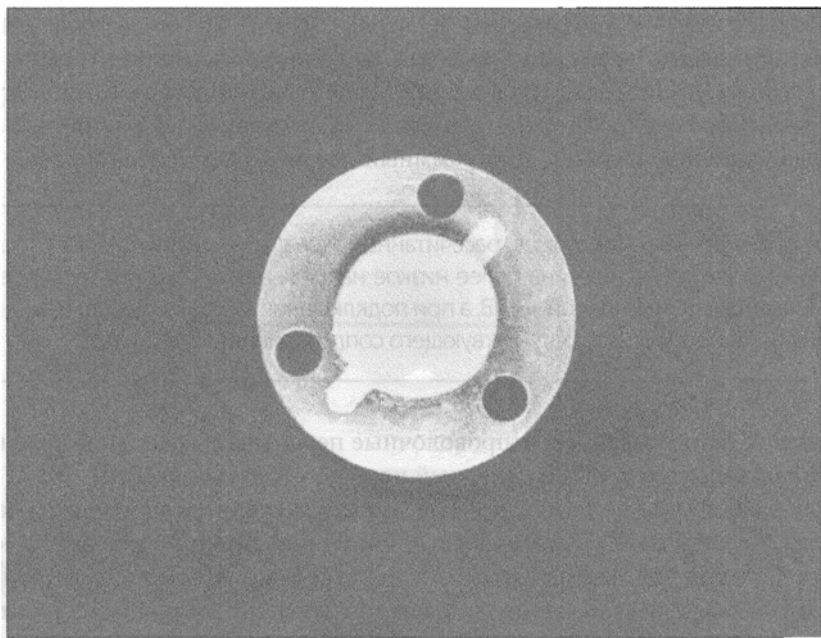


Рис. 7.22. RGB-светодиод

Несмотря на ваши ожидания, в большинстве RGB-светодиодов общий вывод положительный. Обычно положительный вывод в радиотехнических устройствах называют анодом. Так как в RGB-светодиоде общий вывод положительный, то для свечения каждым цветом на выводы для красного, зеленого и синего цветов нужно подать напряжение низкого уровня (0 В или GND).

Анод или катод?

Если бы RGB-светодиод имел общий катод (отрицательный вывод), то вам пришлось бы изменить не только схему подключения, но и программный код управления им. В электрической цепи вам нужно было подключить общий катод не к питанию +5 В, а заземлить. В программе вам пришлось бы отказаться от инвертирования числового

представления цветов, передаваемых в функцию `analogWrite()`. Например, вместо `255-red` для указания интенсивности свечения красного цвета вам пришлось бы использовать аргумент `red`. Функция подачи сигнала на светодиод имела бы в таком случае вид `analogWrite(redPin, red)`.

Но как узнать, какой из выводов является общим анодом, а на какой подаются сигналы для определения интенсивности свечения основных цветов?

Выводы RGB-светодиода

Можно определить назначение выводов светодиода экспериментально. Подайте питание на плату Arduino, подключив ее к USB-порту компьютера. Поскольку к плате не подключено больше ничего, кроме кабеля с напряжением +5 В и линией заземления, то выполняемый в Arduino программный код не играет большой роли.

Мы рассматриваем светодиод, рассчитанный на напряжение питания 5 В. Если ваш светодиод рассчитан на более низкое напряжение, то при его тестировании воспользуйтесь шиной +3,3 В, а при подключении к Arduino добавьте в цепь нагрузочный резистор соответствующего сопротивления.

На плате Arduino подключите проволочные перемычки так, чтобы красный провод соединялся с разъемом +5 В, а черный провод — с разъемом GND.

Подключите другие концы перемычек к расположенным рядом выводам RGB-светодиода. Перебирайте всевозможные комбинации соседних выводов, пока светодиод не засветится. Последующим перебором добейтесь раздельного свечения RGB-светодиода красным, зеленым и синим цветами. Обратите внимание на то, что в каждом из трех случаев один общий вывод всегда будет подключаться к шине питания +5 В с помощью красного провода. Этот вывод и будет общим анодом, в то время как остальные три вывода будут подключаться к Arduino только с помощью черного провода (к выводу GND). Пометьте общий вывод для каждого из компонентных цветов светодиода.

Вы можете заметить, что вывод общего анода несколько длиннее остальных. Примите это к сведению, чтобы исключить путаницу в дальнейшем. Вы всегда можете пометить общий анод самоклеющейся полоской с однозначной маркировкой (например, А — анод, или просто +).

Если в вашем случае общим для трех компонентных светодиодов является не вывод питания +5 В, а “земля” (питание подается на остальные три вывода), то вы держите в руках светодиод с общим катодом. Детально о том, чем отличаются схемы с общим катодом и общим анодом, см. во врезке “Анод или катод?”

Также не забудьте обозначить остальные три вывода RGB-светодиода, чтобы указать соответствующий цвет свечения (например, R, G и B).

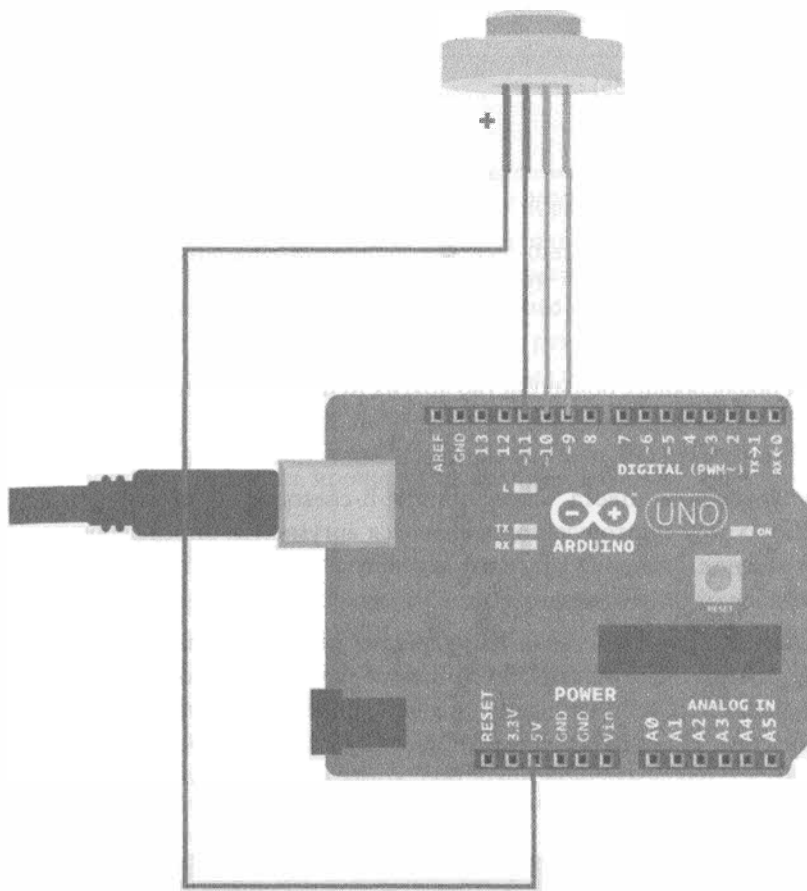


Рис. 7.23. Схема подключения RGB-светодиода к Arduino

Листинг 7.10. hellorgb.ino

```
// hellorgb.ino - произвольный цвет свечения RGB-светодиода  
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const int redPin=11; // ❶  
const int greenPin=10;  
const int bluePin=9;  
  
void setup()  
{  
  pinMode(redPin, OUTPUT);  
  pinMode(greenPin, OUTPUT);  
  pinMode(bluePin, OUTPUT);  
}
```

```

void loop()
{
    setColor(255, 0, 0); // ❶
    delay(1000);

    setColor(255, 255, 255);
    delay(1000);
}

void setColor(int red, int green, int blue)
{
    analogWrite(redPin, 255-red); // ❷
    analogWrite(greenPin, 255-green);
    analogWrite(bluePin, 255-blue);
}

```

- ❶ Как и подавляющее большинство других RGB-светодиодов, наш светодиод имеет один общий вывод (анод). Это означает, что управление свечением осуществляется через остальные три вывода, называемых отрицательными (LOW, 0 В).
- ❷ Для установки цвета свечения RGB-светодиода применяется функция `setColor()`, в которую подставляются аргументы, определяющие необходимый оттенок. Значения красного, зеленого и синего компонентов изменяются в диапазоне от 0 (нет свечения) до 255 (максимальная интенсивность).
- ❸ Для увеличения яркости свечения напряжение на сигнальном выводе понижается. Поэтому нам нужно обеспечить обратный порядок учета уровней яркости. Например, если на вывод `redPin` подается сигнал HIGH (значение 255), то красный светодиод перестает светиться. И наоборот, красный светодиод светится с максимальной яркостью, когда на вывод `redPin` подается сигнал низкого уровня (значение 0).



Рис. 7.24. Прототип робота, умеющего изменять свой цвет, созданный студентами на скорую руку всего за пару дней

Усреднение цветового значения

Цветовой купол выглядит привлекательнее, если изменение цвета свечения происходит плавно. Резкая смена цветовых оттенков порядком раздражает, что сильно портит общее впечатление о прототипе. Но ведь одна из ключевых особенностей светодиодов заключается в быстром изменении своего состояния. Изменения цвета свечения RGB-светодиода в результате подачи другого управляющего сигнала происходит за доли секунды, что совсем не играет нам на руку.

Случайный шум — это еще одна важная проблема при измерении рабочих характеристик оборудования, и в этом смысле она далеко не исключение из правила.

Для примера рассмотрим процесс измерения высоты небольшого дерева. Чтобы уменьшить вероятность ошибки, проведем несколько замеров. Предположим, мы получили такие значения высоты:

102 см, 100 см, 180 см, 103 см, 105 см.

Некоторые из вас примут значение 180 см как среднее для данного случая. К счастью (или к сожалению), компьютерная логика часто отличается от общепринятой.

Чтобы разобраться в правильности предположения, необходимо представить полученные величины в виде массива, а затем в каждой итерации вычислить для массива среднее значение. Несмотря на то что такой алгоритм решения задачи позволяет добиться правильного результата, он достаточно громоздок для столь простой задачи, ведь вам придется использовать большое количество программных конструкций: массив, указатель, формулу вычисления среднего и циклическую структуру.

Скользящее среднее — вот наш спаситель! Для сглаживания характеристики можете вычислять среднее для двух значений: предыдущего и текущего. Чтобы получить большее количество точек для характеристики, не прибегая к использованию массива, рассчитайте взвешенное скользящее среднее.

Во взвешенном скользящем среднем значимость нового (текущего) значения принимается равной 70%, а значимость старого значения (предыдущего) составляет всего 30% (100% минус 70%):

текущее = $0,7 \times \text{текущее} + 0,3 \times \text{предыдущее}$;

предыдущее = *текущее*.

Поскольку предыдущее значение рассчитывается таким же образом, исходя из еще более раннего набора значений, то необходимость в использовании массива при вычислении скользящего среднего отпадает.

Отображение произвольного цветового значения

Для получения различных цветовых оттенков достаточно скомбинировать три основных цвета — красный, зеленый и синий — с правильно подобранными уровнями яркости. Если совместить три основных цвета с максимальной яркостью свечения,

то получится белый цвет. Если яркость свечения всех трех цветов опустить до минимума, то светодиод перестанет излучать свет вообще.

В Arduino не предусмотрена возможность плавного изменения яркости светодиода, поскольку сам светодиод не обеспечивает плавность изменения рабочих характеристик. При понижении напряжения на выводах яркость свечения светодиода действительно падает, но по достижении некоего, далеко не нулевого, значения светодиод просто выключается.

Чтобы обойти технические ограничения в рабочих характеристиках светодиода, в Arduino применяется широтно-импульсная модуляция (ШИМ). Чтобы создать впечатление понижения яркости светодиода до 10% от максимального, на него подается ШИМ-сигнал с коэффициентом заполнения 10%. Это означает, что светодиод будет гореть на максимальной яркости в течение 10% рабочего времени, а остальные 90% рабочего времени будет выключен. Поскольку включение и выключение светодиода в Arduino выполняется очень часто (каждый цикл занимает всего 2 мс), то вы не заметите мерцания в воспроизводимом свечении. При коэффициенте заполнения 10% светодиод в каждом цикле будет гореть в течение 200 мс, а в выключенном состоянии проведет 1800 мс.

В программном коде вы не будете учитывать такие тонкости процесса свечения. Вам нужно всего лишь правильно рассчитать напряжение, подаваемое на светодиод (в диапазоне от 0 до 255), чтобы получить соответствующий уровень яркости. Поскольку в RGB-светодиоде на общий вывод всегда подается сигнал высокого уровня (HIGH), то для свечения основных цветов с максимальной яркостью достаточно подать на остальные три вывода сигнал LOW. Максимальный уровень яркости красного светодиода, например, устанавливается функцией `analogWrite(redPin, 0)`.

Остальные уровни яркости, большие минимального и меньшие максимального, определяются такой же функцией, но в нее в качестве второго аргумента подставляется требуемый оттенок, вычитенный из 255. Например, для задания интенсивности свечения произвольного оттенка красного цвета, `r`, используется следующая функция:

```
analogWrite(redPin, 255-r);
```

Уровни сигналов для всех базовых цветов представлены в табл. 7.1.

Таблица 7.1. Цвета свечения RGB-светодиода и уровни подаваемого сигнала

Цвет	Цвет (RGB)	Сигнал	Описание
Нет	(0,0,0)	(255, 255, 255)	Выкл.
Красный	(255,0,0)	(0, 255, 255)	
Зеленый	(0,255,0)	(255, 0, 255)	
Синий	(0,0,255)	(255, 255, 0)	
Белый	(255,255,255)	(0, 0, 0)	Максимальная яркость
Произвольный	(r,g,b)	(255-r, 255-g, 255-b)	

Как видите, получить все цвета радуги с помощью RGB-светодиода не так уж и сложно. Достаточно поэкспериментировать с интенсивностью красного, синего и зеленого цветов!

Масштабирование входных и выходных значений

Входные и выходные данные представляются разными диапазонами значений. Вам следует научиться преобразовывать одни значения в другие.

В самом простом случае выходное значение можно получить, если умножить входное значение на некий коэффициент. Все сводится к простой операции умножения.

Например, функция `analogRead()` в Arduino принимает в качестве аргумента значения в диапазоне от 0 до 1023, а функция `analogWrite()` — значения из диапазона от 0 до 255. Чтобы правильно масштабировать один диапазон значений к другому, сначала нужно определить соотношение (p) входного значения (in) к максимальному значению этого диапазона:

$$p = in/1023$$

Масштабированное выходное значение (out) представляется как полученное выше соотношение p , умноженное на максимальное значение выходного диапазона:

$$out = p*255$$

Во встроенной в Arduino программной библиотеке описанные выше вычисления проводятся с помощью специальной функции:

```
out = map(in, 0, 1023, 0, 255)
```

Пример линейного преобразования значений двух диапазонов приведен в табл. 7.2.

Таблица 7.2. Линейное масштабирование значений входного и выходного диапазонов с помощью функции `map()`

Входное значение, <code>analogRead()</code>	Пересчетный коэффициент, %	Выходное значение, <code>analogWrite()</code>
0	0	0
234	23	58
511	50	127
1023	100	255

Тем не менее при проведении расчетов далеко не всегда выходные значения представляются целыми числами. В таком случае выполняется округление дробных значений. В данном проекте правильно определить яркость свечения RGB-светодиода без округления масштабированного значения просто невозможно.

В подавляющем большинстве RGB-светодиодов произвольные цветовые оттенки лучше проявляются при низких уровнях яркости основных цветов. При высоких выходных значениях яркости свечения основных цветов конечный оттенок становится настолько близок к белому цвету, что распознать его становится очень сложно.

Как видите, линейное масштабирование диапазонов входных и выходных значений малоэффективно. Технические характеристики RGB-светодиодов таковы, что масштабирование лучше всего проводить согласно экспоненциальной или степенной зависимости. В противном случае больше половины цветовых оттенков (например, оранжевый и фиолетовый) будет представляться ярко-белым свечением.

Сначала определим соотношение входного значения к максимальному в диапазоне:
 $p = in/1023$

Осталось определить выходные значения, рассчитанные согласно некой нелинейной функции. Граничные значения диапазона остаются неизменными — 0 и 255, поскольку представляют максимально и минимально допустимые уровни сигнала. В проекте цветового купола для масштабирования диапазонов входных и выходных значений применяется такая степенная функция:

$$out = 255 * p^4$$

Поскольку максимальное значение соотношения p равно 1 (100%), то диапазон значений выражения p^4 будет находиться в пределах от 0 (0%) до 1 (100%). Графически данная степенная функция представляется классической гиперболой. Примеры масштабированных согласно степенной функции значений приведены в табл. 7.3.

Таблица 7.3. Масштабирование по степенной функции

p	p^4 , %	Выходное значение, %	Описание
0%	0	0	Мин.
	20	0,2	0
	40	2,6	6
	50	6,2	15
Половина	60	13	33
	80	41	104
	90	65,6	167
	100	100	255

Подобные преобразования часто проводятся в анимации. Если нужно остановить (или ускорить) движущийся (неподвижный) в кадре объект, то скорость остановки (ускорения) обычно рассчитывается по степенной функции.

Объединение программного кода

В проекте цветового купола RGB-светодиод применяется совместно с рассмотренным ранее датчиком цвета.

Подключите датчик цвета и RGB-светодиод к Arduino так, как показано на рис. 7.27. Вам на помощь придут соединительные кабели компании DuPont, собранные в шлейфы (рис. 7.25), и плата расширения ScrewShield. Не правда ли, несмотря на большое количество соединений, конечное устройство выглядит очень аккуратно (рис. 7.26)?

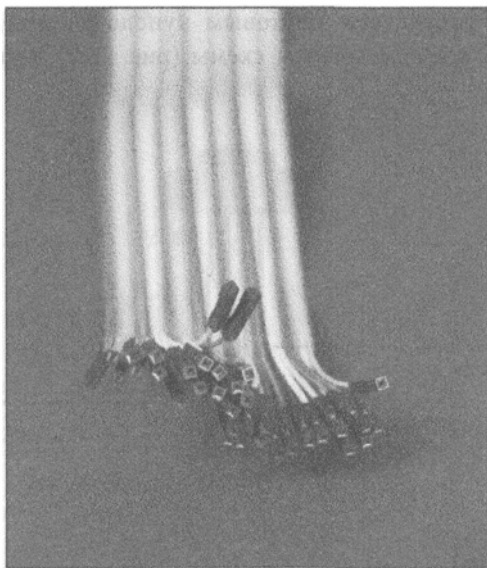


Рис.7.25. Многожильный кабель от DuPont упростит подключение к Arduino большого количества проводов

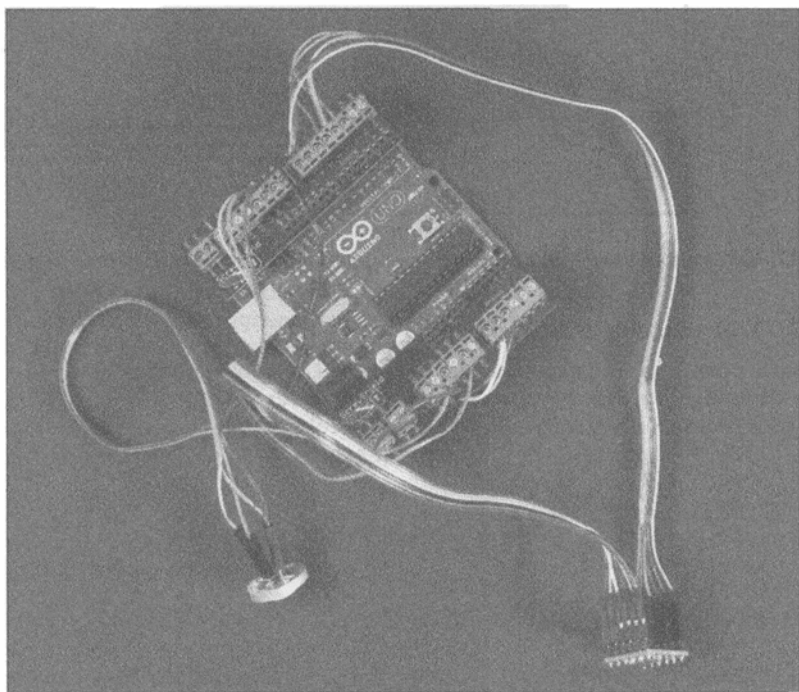


Рис. 7.26. Датчик цвета и RGB-светодиод совмещены в одном устройстве

Программный код управления цветowym куполом приведен в листинге 7.11. Правильно подключив все компоненты схемы (рис. 7.27), выполните программу в Arduino.

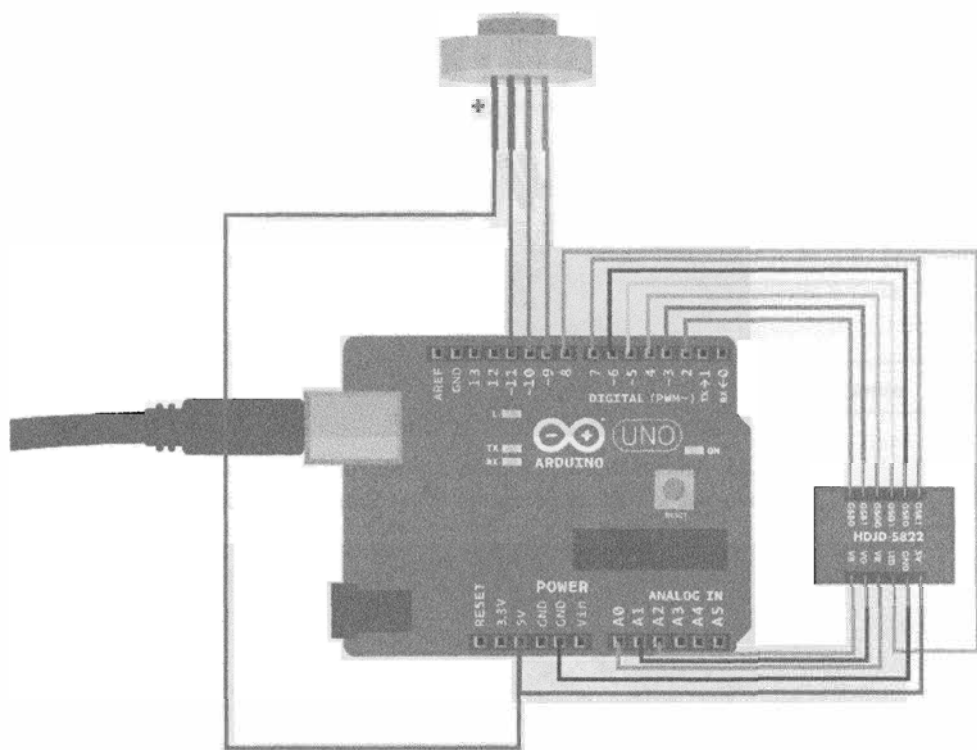


Рис. 7.27. Схема подключения компонентов цветовой сферы (см. цветную вклейку)

Листинг 7.11. `chameleon_cube.ino`

```
// chameleon_dome.ino - изменение цвета купола в соответствии с
// цветом поверхности, на которой он располагается
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const int gsr1Pin = 7; // ❶
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;

const int ledPin = 8; // ❷

const int redInput = A0; // ❸
const int greenInput = A1;
const int blueInput = A2;
```

```

const int redOutput = 11; // ❹
const int greenOutput = 10;
const int blueOutput = 9;

int red = -1; // ❺
int green = -1;
int blue = -1;

const float newWeight = 0.7; // ❻

void setup() {
  Serial.begin(115200);
  pinMode(gsr1Pin, OUTPUT);
  pinMode(gsr0Pin, OUTPUT);
  pinMode(gsg1Pin, OUTPUT);
  pinMode(gsg0Pin, OUTPUT);
  pinMode(gsb1Pin, OUTPUT);
  pinMode(gsb0Pin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(redOutput, OUTPUT);
  pinMode(greenOutput, OUTPUT);
  pinMode(blueOutput, OUTPUT);

  digitalWrite(ledPin, HIGH); // ❼

  digitalWrite(gsr1Pin, LOW);
  digitalWrite(gsr0Pin, LOW);
  digitalWrite(gsg1Pin, LOW);
  digitalWrite(gsg0Pin, LOW);
  digitalWrite(gsb1Pin, LOW);
  digitalWrite(gsb0Pin, LOW);
}

void loop() {
  int redValue = analogRead(redInput); // ❸
  int greenValue = analogRead(greenInput);
  int blueValue = analogRead(blueInput);

  redValue = redValue * 10 / 1.0;
  greenValue = greenValue * 10 / 0.75;
  blueValue = blueValue * 10 / 0.55;

  redValue = map(redValue, 0, 1023, 0, 255); // ❶
  greenValue = map(greenValue, 0, 1023, 0, 255);
  blueValue = map(blueValue, 0, 1023, 0, 255);

  if(redValue > 255) redValue = 255; // ❷
  if(greenValue > 255) greenValue = 255;
  if(blueValue > 255) blueValue = 255;

  red = runningAverage(redValue, red); // ❸
  green = runningAverage(greenValue, green);
  blue = runningAverage(blueValue, blue);
}

```

```

Serial.print(red); Serial.print(" ");
Serial.print(green); Serial.print(" ");
Serial.print(blue); Serial.println(" ");
if(red < 200 || green < 180 || blue < 180) {
    green = green - red * 0.3; // ⑦
    blue = blue - red * 0.3;
}

red = easing(red); // ⑧
green = easing(green);
blue = easing(blue);

setColor(red,green,blue); // ④

delay(100);
}

int runningAverage(int input, int old) {
    return newWeight*input + (1-newWeight)*old; // ⑤
}

int easing(int input) { // ⑩
    float percent = input / 255.0f;
    return 255.0f * percent * percent * percent * percent;
}

int setColor(int r, int g, int b) { // ⑪
    analogWrite(redOutput, 255-r); // ⑤
    analogWrite(greenOutput, 255-g);
    analogWrite(blueOutput, 255-b);
}

```

- ① Определение выводов для калибровки основных цветов. В данной программе калибровка не выполняется, поэтому на указанные выводы подается сигнал низкого уровня. Подробнее об использовании этих выводов см. в описании листинга 7.8.
- ② Назначение вывода для управления осветительным светодиодом датчика цвета.
- ③ На эти выводы передаются сигналы, соответствующие цветовым значениям, которые были прочитаны датчиком.
- ④ Выводы для подключения RGB-светодиода.
- ⑤ Объявление глобальных переменных, в которых хранятся обрабатываемые цветовые значения. Исходно им назначаются невозможные значения, чтобы упростить дальнейшую отладку программы (если впоследствии эти значения выводятся на монитор, то вы точно будете знать, что программа работает некорректно).
- ⑥ Взвешенное значение, используемое при расчете скользящего среднего (см. предыдущий раздел).
- ⑦ Освещение поверхности перед распознаванием ее цвета.

- 8 Считывание насыщенности красного цвета. Функция `analogRead()` возвращает соответствующее целочисленное значение из диапазона 0–1023.
- 9 Масштабирование значения (0–1023), возвращенного функцией `analogRead()`, до значения (0–255), которое можно передать функции `analogWrite()`.
- 10 11 Поскольку цветовые значения уравниваются (например, значение красного оттенка умножается на 10), то конечное значение может выходить за пределы диапазона, принимающего участие в масштабировании. В результате, например, яркость красного цвета может оказаться больше 255. Впоследствии при использовании такого значения в функции `analogWrite()` для управления свечением RGB-светодиода оно ограничивается максимальным уровнем 255.
- 12 Художественная коррекция цветового оттенка, повышающая его эстетическую оценку.
- 13 Замедление процесса изменения цветового оттенка купола, обеспечивающая плавность работы устройства.
- 14 Установка конечного цвета RGB-светодиода. Это цвет, которым после проведения всех необходимых вычислений светится купол.
- 15 Расчет скользящего среднего позволяет сгладить влияние случайных значений на конечный цветовой оттенок (см. главу 6).
- 16 Термин *easing* (плавность) придуман и применяется аниматорами. При настройке движения в анимации Flash или JavaScript ускорение или остановка объектов в кадре выполняется согласно специальной функции плавности. В применяемой нами функции плавности уменьшаются значения, меньшие 255 (100%). Чем меньше цветовые значения, тем сильнее они уменьшаются. В результате изменение оттенка свечения RGB-светодиода производится плавно.
- 17 Изменение цвета свечения RGB-светодиода до указанного значения. Функция принимает целочисленные значения из диапазона 0–255.
- 18 Для светодиода с общим анодом (положительный вывод) на выводы цветовых компонентов подаются инверсные значения. См. раздел “Отображение произвольного цветового значения”.

Корпус в виде полусферы

Корпус для проекта цветовой сферы мы случайно нашли в IKEA. Если не считать мелких доработок, светильник Solvinden идеально подошел для наших целей. Вам не обязательно искать магазин IKEA, чтобы собрать готовый прототип. Воспользуйтесь полупрозрачным корпусом любой подходящей формы, имеющимся под рукой. Сферическую форму имеют плафоны большинства старых ламп, кроме того, вы всегда можете приспособить под свои нужды пластиковый контейнер из-под замороженных продуктов.

Вначале мы разобрали светильник Solvinden (рис. 7.28) и сняли защитную крышку с внутренней стороны подставки (рис. 7.29). Нам нужно удалить все расположенные под ней электронные компоненты, чтобы освободить место для своих целей (рис. 7.30).

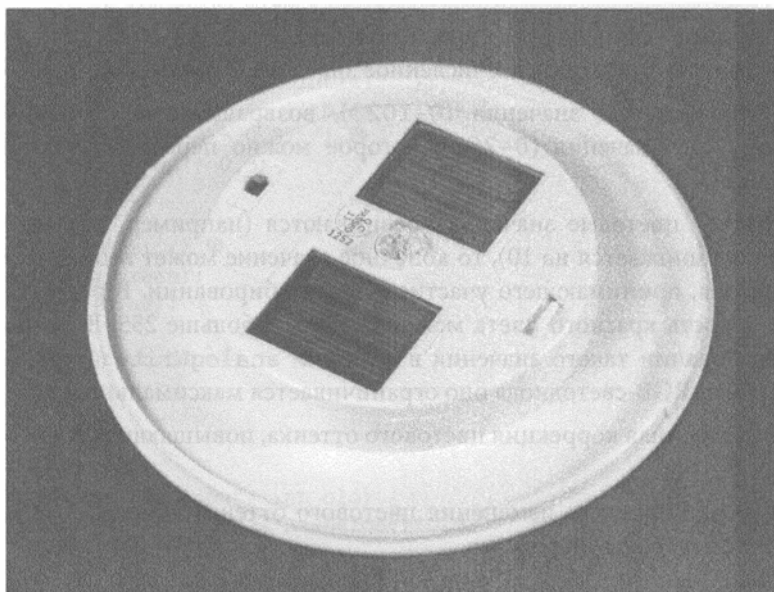


Рис. 7.28. Снимите плафон

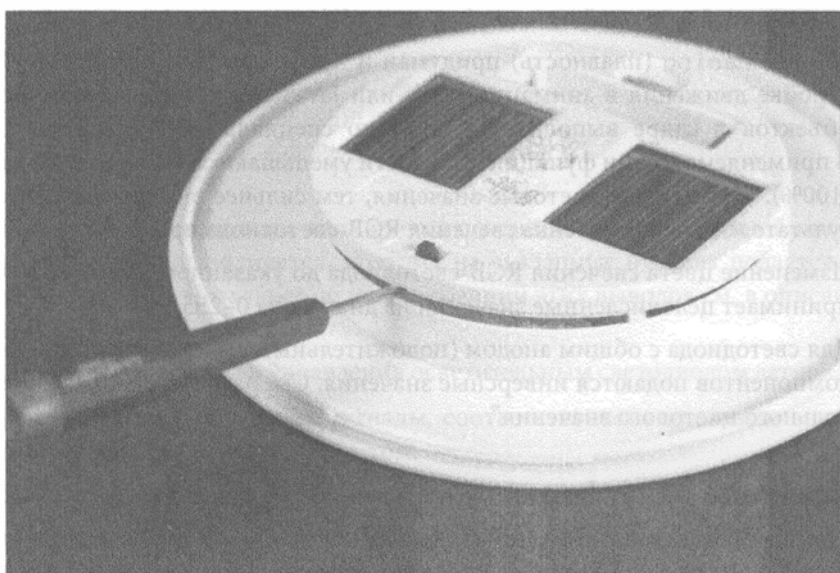


Рис. 7.29. Вскройте контейнер с электроникой

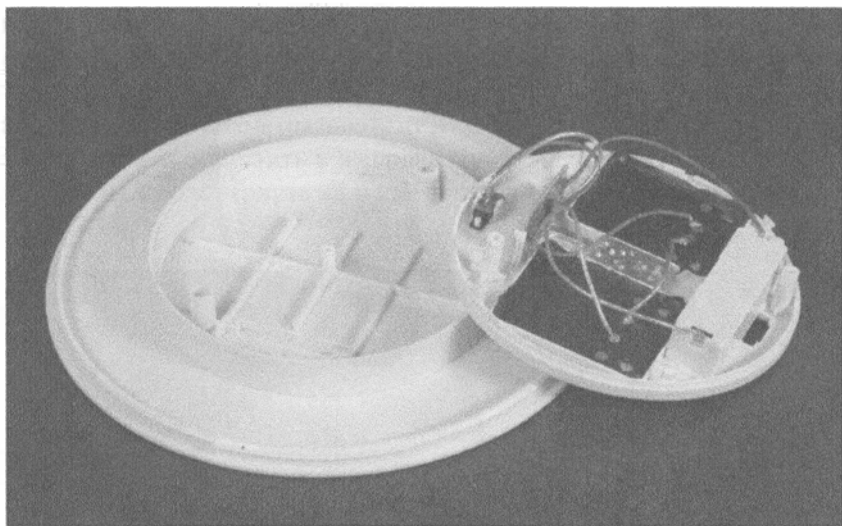


Рис. 7.30. *Выбросьте все заводские компоненты*

В центре контейнера, закрываемого крышкой, располагается пластиковый штырь — срежем его (рис. 7.31). Теперь нужно проделать в подставке два отверстия: одно, диаметром 19 мм, под датчик, располагаемый приемником вниз, а второе, диаметром 3 мм, для крепления платы Arduino и RGB-светодиода (рис. 7.32).

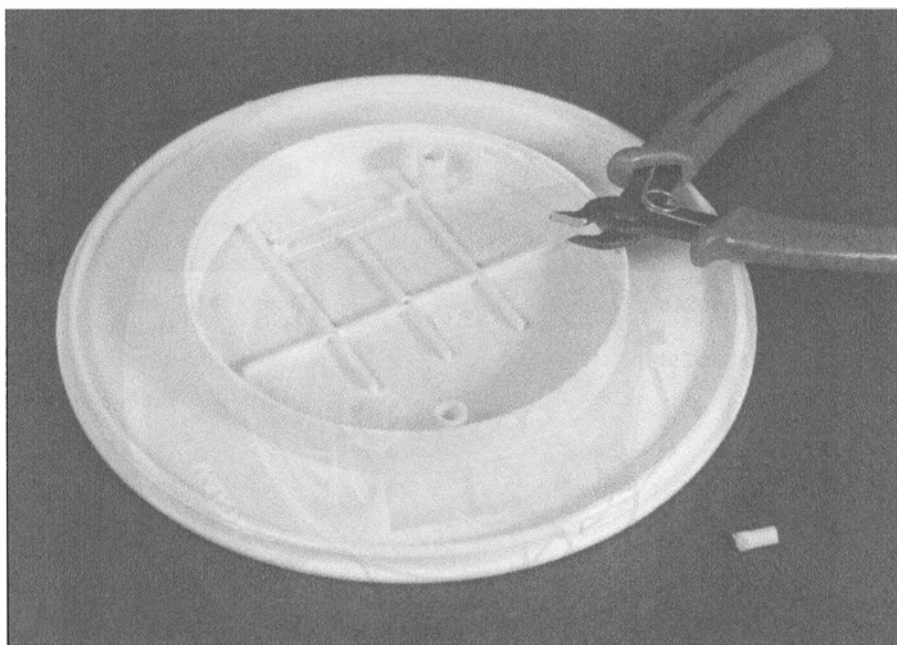


Рис. 7.31. *Срезаем центральный упор*

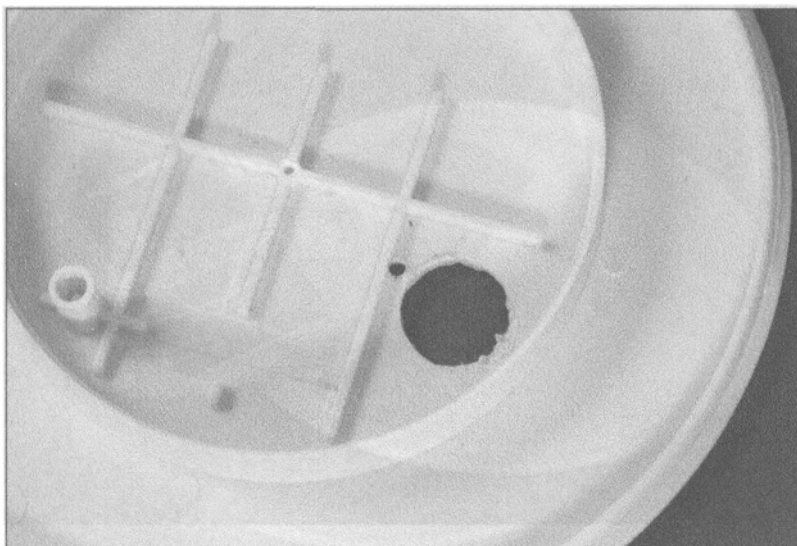


Рис. 7.32. Мы просверлили большое ($\varnothing 19$ мм) отверстие под датчик и еще одно — маленькое ($\varnothing 3$ мм), предназначенное для крепления платы Arduino и светодиода

Датчик закрепляется над отверстием с помощью термоклей; он должен располагаться строго по центру отверстия, как показано на рис. 7.33.

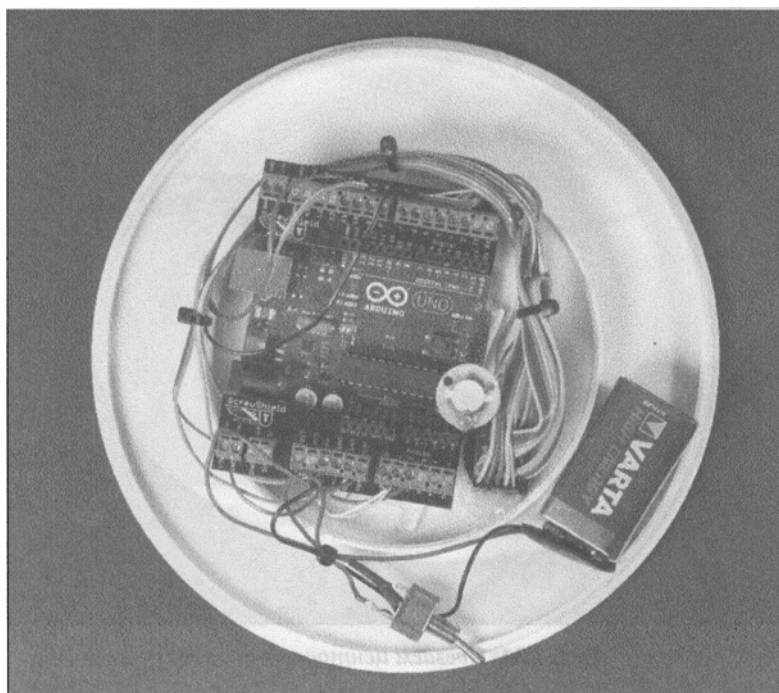


Рис. 7.33. Основание цветовой полусферы со всем необходимым оборудованием

Плата Arduino крепится к подложке, представленной подставкой светильника, с помощью самореза диаметром 3 мм. RGB-светодиод фиксируется тем же саморезом, что и плата Arduino. В корпусе светодиода уже предусмотрены отверстия для крепления, но они меньшего диаметра, поэтому нужно несколько расширить одно из них.

Осталось только поместить в импровизированный корпус батарею питания, подать питание на плату, вернуть на место плафон и насладиться полученным результатом (рис. 7.34).

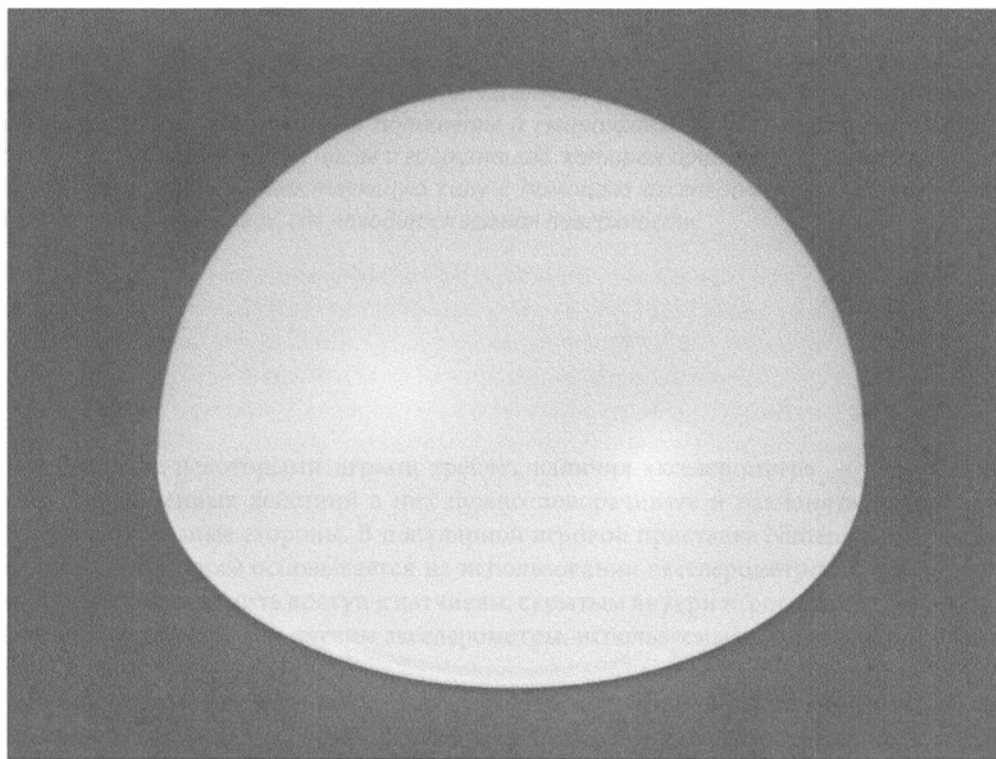


Рис. 7.34. *Купол надежно скрывает электронные компоненты*

Вот вы и научились создавать устройства, распознающие базовые характеристики света: его наличие, направление и даже цвет.

Ускорение



Поворачивая в руках планшет или смартфон, вы заметите, что ориентация изображения на его экране изменяется с книжной на альбомную и наоборот. Это очень удобно, но как это осуществляется? В планшеты и смартфоны встроены акселерометры, которые различают в том числе и гравитацию, которая притягивает объекты вниз. Определив постоянно действующую силу с помощью акселерометра, планшет или смартфон всегда "знает", где находится земная поверхность.

Управление некоторыми играми требует наличия акселерометра — для выполнения определенных действий в них нужно поворачивать и наклонять мобильное устройство в разные стороны. В популярной игровой приставке Nintendo Wii система управления также основывается на использовании акселерометров. В этой главе вы узнаете, как получить доступ к датчикам, скрытым внутри игровых контроллеров. Кроме того, мы детально изучим акселерометры, используемые в некоторых мобильных устройствах.

Жесткие диски, на которых хранятся данные ноутбуков и настольных компьютеров, не любят больших перегрузок. Многие из них выдерживают удары силой не более 150 г (в выключенном состоянии), а ведь удар такой силы при определенных условиях может нанести человеку тяжелые увечья. Чтобы предотвратить повреждение, многие жесткие диски оснащены встроенной системой защиты, которая автоматически отключает питание и убирает магнитные головки с поверхности дисков (эта операция называется парковкой головок жесткого диска), как только акселерометр определяет, что устройство находится в состоянии свободного падения.

Вы когда-нибудь задумывались о создании самобалансирующихся устройств, подобных Segway или Solowheel? Несмотря на неуверенный старт, вам может показаться чудом, что такое неустойчивое средство передвижения сохраняет вертикальное положение даже при движении по пересеченной местности.

Но ничего мистического в самобалансирующихся устройствах нет. Как только система управления определяет критический наклон вертикальной оси устройства вперед, на двигатель подается сигнал увеличения крутящего момента, чтобы ускорить вращение колес и тем самым восстановить вертикальное положение устройства. В самобалансирующих средствах передвижения угловая скорость определяется гироскопом, поскольку акселерометр подвержен быстрому накоплению ошибок, что не

позволяет применять его для поддержания устройства в вертикальном положении в течение длительного промежутка времени.

Ускорение и угловая скорость

Ускорение — это мера увеличения скорости перемещения объекта в пространстве. Оно возникает, например, при наборе скорости автомобилем или торможении (отрицательное ускорение). Угловая скорость характеризует скорость вращения объекта вокруг некой оси. В зависимости от проекта вам может понадобиться измерить только угловую скорость, только ускорение или обе эти величины.

Ускорение обычно измеряется в единицах g , представляющих собой ускорение свободного падения, вызванного земной гравитацией. Метрическая единица измерения ускорения — это метр в секунду в квадрате (м/с^2). Ускорение свободного падения (g) приравнивается к $9,81 \text{ м/с}^2$.

Почему ускорение определяется такой странной величиной, обратно пропорциональной времени в квадрате? Потому что ускорение характеризует скорость изменения скорости. Если скорость определяется как расстояние, деленное на время (м/с), то ускорение — это скорость, деленная на время (мера изменения скорости), или м/с^2 .

Гироскопы, в свою очередь, измеряют угловую скорость, или скорость их поворота относительно некой оси. Например, гироскоп может определить, что скорость вращения составляет 10 градусов в секунду. Без гироскопов невозможно представить себе самобалансирующиеся устройства и гироскопасы (системы курсовой устойчивости).

Таблица 8.1. Акселерометр и гороскоп

Датчик	Измеряемая величина	Описание	Единицы измерения	Гравитация
Акселерометр	Ускорение	Скорость набора или потери скорости объекта	м/с^2	Определяет, в единицах g
Гироскоп	Угловая скорость	Скорость изменения угла поворота или вращения	рад/с (СИ), чаще градус/с или оборот/мин	Не учитывается

Эксперимент: определение ускорения датчиком MX2125

Микросхема MX2125 (рис. 8.1) представляет собой обычный двунаправленный датчик ускорения. Длительность выходного сигнала этого датчика пропорциональна измеряемому им ускорению, что сильно упрощает управление им в программном коде.

Реальное физическое пространство, если не учитывать время, — трехмерное. Объекты в нем имеют три степени свободы: вверх-вниз (ось Y), влево-вправо (ось X) и вперед-назад (ось Z). Двунаправленный акселерометр позволяет измерять ускорение только в двух направлениях из трех.

Датчик MX2125 позволяет измерять ускорение, не превышающее по модулю $3g$ в каждом из допустимых направлений. Существуют также датчики, предназначенные для измерения сверхнагрузок. Например, максимально измеренное с помощью датчика ADXL377 ускорение составляет $200g$, которого более чем достаточно, чтобы убить любого человека. Это делает его применимым в космических аппаратах и военных самолетах. Их скорость определения ускорения настолько велика, что превышает скорость выстрела современного огнестрельного оружия. В ранних прототипах созданного нами спутника Aalto-1, предназначенного для изучения солнечного излучения, применение такого датчика было нецелесообразным, поскольку ускорений такого уровня он вряд ли достиг бы.

Вам вряд ли придется в ближайшем будущем столкнуться с проблемой измерения таких сильных нагрузок, по крайней мере, вы точно не будете применять для управления датчиком ADXL377 микроконтроллерные платформы Arduino и Raspberry Pi (подобные нагрузки гарантированно приведут к поломке компонентов платы!). Как ни странно, но стоимость такого датчика невелика, зато и точность измерения не самая высокая. Примите к сведению, что чем больше рабочий диапазон датчика ускорения, тем ниже точность определяемого им значения.

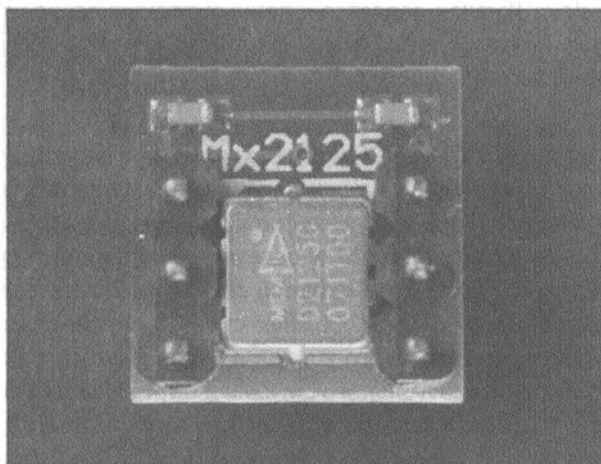


Рис. 8.1. Датчик MX2125

Определение длительности импульса датчика MX2125

Чаще всего пересчетная формула, используемая для вычисления ускорения соответственно длительности выходного сигнала акселерометра, приводится в технической документации к устройству. Наш случай далеко не так однозначен и требует отдельного поиска технических характеристик устройства.

Поиск технических характеристик датчика может занять довольно много времени. Чтобы сузить область поиска, в точности указывайте модель датчика и правильное название искоемых параметров. Иногда технические характеристики устройств приводятся на сайте интернет-магазина, торгующего электротехническим оборудованием (обычно они выносятся на отдельную вкладку или страницу).

Например, при поиске технических характеристик датчика MX2125 вы получите множество ссылок на базовую документацию, предоставленную компанией Parallax, которая не содержит необходимых нам данных. Микросхему для датчика MX2125 производит компания Memsic, а Parallax является лишь сборщиком, поэтому детальные сведения о параметрах выходного сиг-

нала в базовой документации, предоставленной Parallax, отсутствуют.

Тем не менее, если включить в поиск слово "Memsic", то легко выяснить, что формула преобразования длительности импульса в измеренное ускорение описана в документе "Parallax Memsic 2125 Accelerometer Demo Kit".

Если вернуться на сайт Parallax, то трудно заметить, что требуемые технические характеристики можно найти и на нем, достаточно провести поиск по модели микросхемы, а не датчика. Проведите поиск названия "MXD2125," а именно так маркируется микросхема для датчика ускорения MX2125. Документация к микросхеме содержит множество детальных сведений, не включенных в технические характеристики устройств, основанных на ней.

Принцип работы датчика MX2125 весьма хитроумен: внутри него нагреваются крохотные пузырьки газа, а при изменении ориентации устройства они начинают двигаться — по их перемещению и определяется направление смещения.

При подаче питания датчик MX2125 определяет ускорение вдоль каждой оси, генерируя импульсы с частотой 100 Гц. Выходной сигнал имеет прямоугольную форму с двумя уровнями напряжения, HIGH и LOW. Чем сильнее определяемое ускорение, тем длительнее в сигнале импульс высокого уровня и короче импульс низкого уровня. По длительности импульсов прямоугольного сигнала и определяется ускорение датчика.

В каждом периоде сигнала (T) присутствует как импульс высокого уровня, так и низкого. Как видите, период равен сумме длительностей импульсов высокого и низкого уровней. Пусть длительность импульсов высокого уровня определяется переменной t_{HIGH} .

Коэффициент заполнения ($dutyCycle$) указывает, какую часть времени в периоде сигнала занимает сигнал высокого уровня. Коэффициент заполнения измеряется в процентах, например 50% (0,5) или 80% (0,8):

$$dutyCycle = t_{HIGH}/T$$

В соответствии с техническими характеристиками датчика период его прямоугольного сигнала (T) по умолчанию составляет 10 мс:

$$dutyCycle = t_{HIGH}/10 \text{ мс}$$

Согласно документации, формула вычисления ускорения имеет следующий вид:

$$A = (t_{HIGH}/T - 0,5) / 20\%$$

Если заменить t_{HIGH}/T коэффициентом заполнения ($dutyCycle$), а вместо 20% использовать 0,2, то итоговая формула примет такой вид:

$$A = (dutyCycle - 0,5) / 0,2$$

Проведя простые математические преобразования ($x/0,2$ всегда равно $5 \cdot x$), получим такое равенство:

$$A = 5 \cdot (dutyCycle - 0,5)$$

Или такое:

$$A = 5 \cdot (t_{HIGH}/T - 0,5)$$

Данное уравнение показывает, что при отсутствии ускорения (0) коэффициент заполнения составляет 50%:

$$0 = 5 \cdot (dutyCycle - 0,5)$$

$$0/5 = dutyCycle - 0,5$$

$$0 = dutyCycle - 0,5$$

$$0,5 = dutyCycle$$

Нам удалось самостоятельно выяснить, что документации к датчику, поставляемые Parallax и Memsic, указывают несколько разные пересчетные коэффициенты: в технических данных Memsic используется значение 1/20% (5), а в формулах, предоставленных Parallax, — 1/12,5% (8).

Согласно нашим экспериментам, значение 1/12,5% (8) точнее, поскольку обеспечивает высокую точность определения ускорения акселерометром, смонтированным на коммутационных платах Parallax. На самом деле даже детальное экспериментальное изучение рабочих характеристик MXD2125 показывает, что данные, приведенные на сайте Parallax, точные. Поэтому будьте предельно внимательны при получении технических характеристик из ненадежных источников — всегда проверяйте все, что только возможно, экспериментально. Итак, с множителем мы определились — это значение 8:

$$A = 8 \cdot (t_{HIGH}/T - 0,5)$$

Так как функция `pulseIn()` возвращает в Arduino длительность импульса в микросекундах ($1 \text{ мкс} = 0,001 \text{ мс} = 1 \cdot 10^{-6} \text{ с}$), указанная выше формула приводится к следующему виду:

$$A = 8 \cdot (t_{HIGH} / (10 \cdot 1000) - 0,5)$$

В наших расчетах мы будем измерять величину A в единицах g , которая составляет $9,81 \text{ м/с}^2$.

Например, если t_{HIGH} составляет 5 000 мкс (5 мс), то коэффициент заполнения вычисляется так:

$$dutyCycle = 5 \text{ мс} / 10 \text{ мс} = 0,5 = 50\%$$

Это соответствует 0g:

$$A = 8 \cdot (0,5 - 0,5) = 8 \cdot 0 = 0 \quad // \text{ g}$$

В качестве следующего примера рассчитаем ускорение, представленное импульсом длительностью $t_{HIGH} = 6250 \text{ мкс}$ (6,25 мс).

$$A = 8 * (6250 / 10000 - 0,5) = 8 * (0,625 - 0,5) = 8 * 0,125 = 1 \quad // \text{ g}$$

Таким образом, импульс высокого уровня длительностью 6,25 мс соответствует ускорению 1g.

Подключение к Arduino и программа управления акселерометром

На рис. 8.2 показана схема подключения акселерометра к Arduino. После подключения всех компонентов цепи выполните программный код из листинга 8.1.

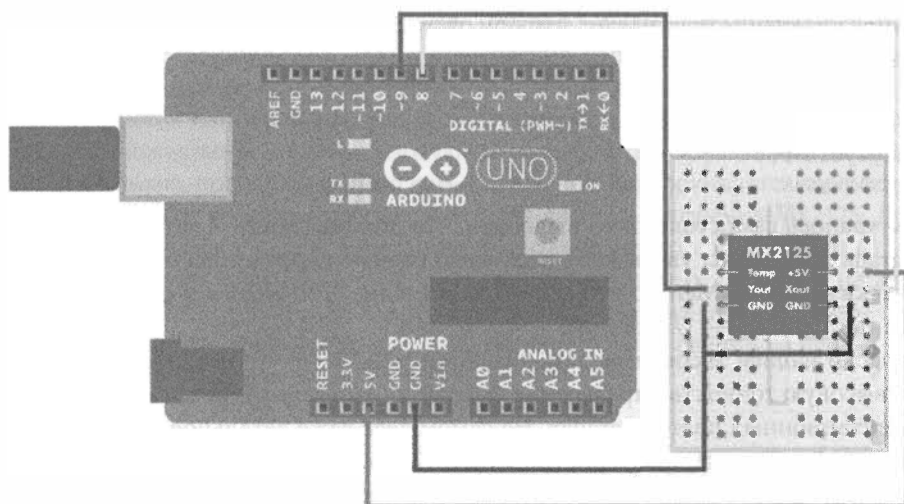


Рис. 8.2. Схема подключения акселерометра MX2125 к Arduino

Листинг 8.1. `mx2125.ino`

```
// mx2125.ino - измерение ускорения в двух направлениях
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const int xPin = 8;
const int yPin = 9;
```

```
void setup() {
  Serial.begin(115200);
  pinMode(xPin, INPUT);
  pinMode(yPin, INPUT);
}
```

```
void loop() {
  int x = pulseIn(xPin, HIGH); // ❶
  int y = pulseIn(yPin, HIGH);
  int x_mg = ((x / 10) - 500) * 8; // ❷
  int y_mg = ((y / 10) - 500) * 8;
```

```

Serial.print("Ускорение вдоль x: ");
Serial.print(x_mg);
Serial.print("Ускорение вдоль y: ");
Serial.println(y_mg);
delay(10);
}

```

- ❶ Величина ускорения определяется длительностью импульса высокого уровня. Функция `pulseIn()` возвращает длительность импульса, выраженную в микросекундах (мкс): $1 \text{ мкс} = 0,001 \text{ мс} = 0,000001 \text{ с} = 1 \cdot 10^{-6} \text{ с}$.
- ❷ Преобразование конечного значения в единицы *millig* (0,001g — тысячные доли ускорения свободного падения *g*).

Подключение к Raspberry Pi и программа управления акселерометром

На рис. 8.3 показана электрическая цепь подключения датчика MX2125 к Raspberry Pi. После ее создания выполните программный код из листинга 8.2.

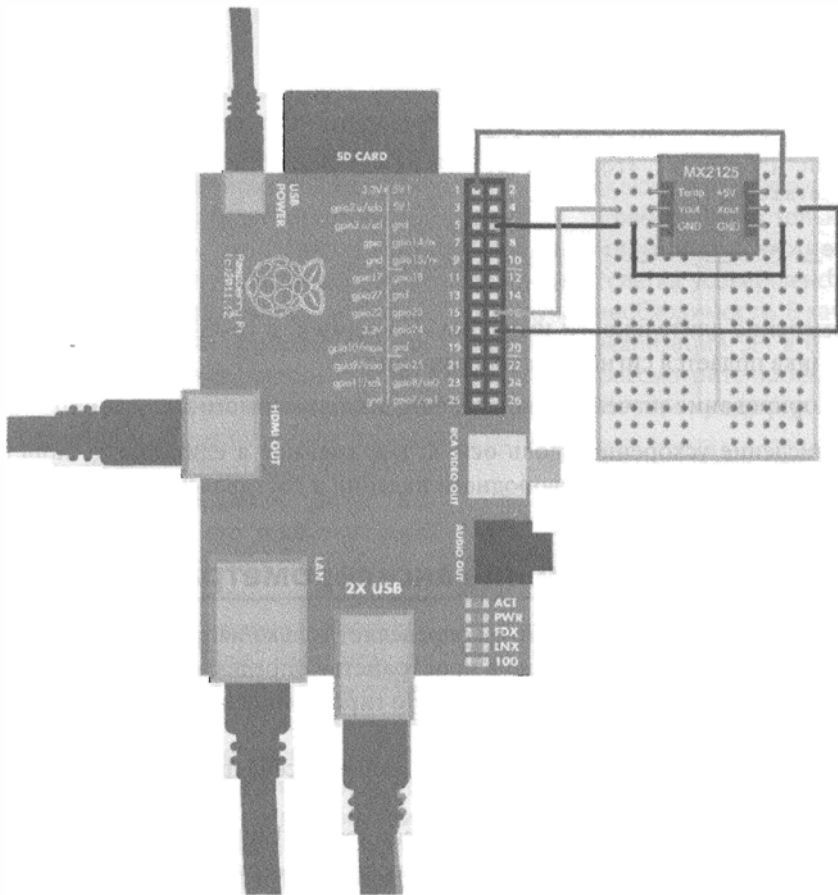


Рис. 8.3. Схема подключения акселерометра MX2125 к Raspberry Pi

Листинг 8.2. mx2125.py

```
# mx2125.py - вывод значений ускорения для двух направлений
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import botbook_gpio as gpio

xPin = 24
yPin = 23

def readAxel(pin):
    gpio.mode(pin, "in")
    gpio.interruptMode(pin, "both")
    return gpio.pulseInHigh(pin) # ❶

def main():
    x_g = 0
    y_g = 0
    while True:
        x = readAxel(xPin) * 1000
        y = readAxel(yPin) * 1000
        if(x < 10): # ❷
            x_g = ((x / 10) - 0.5) * 8 # ❸
        if(y < 10):
            y_g = ((y / 10) - 0.5) * 8
        print ("Вдоль X: %fg, и Y: %fg" % (x_g, y_g))
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

- ❶ Измерение длительности импульса — времени, в течение которого на вывод датчика подается сигнал высокого уровня.
- ❷ Игнорирование значений, выпадающих из допустимого диапазона.
- ❸ Вычисление ускорения вдоль оси X, выраженное в единицах g. Константа g представляет ускорение свободного падения и составляет 9,81 м/с².

Эксперимент: совмещение акселерометра и гироскопа

Если акселерометр не двигать, то он определяет только направление действия гравитации. Относительное положение в пространстве определяет гироскоп, даже если его вращать в произвольном направлении. Но гироскоп не измеряет гравитацию, более того, он ее попросту игнорирует.

А можно ли создать устройство, объединяющее в себе функции акселерометра и гироскопа? Конечно, да!

Блок инерциальных измерений (Inertial Measurement Unit — IMU) совмещает в себе несколько разнофункциональных датчиков, а также логические структуры (дополнительно), обеспечивающие высокую точность и надежность проведения измерений. В данном проекте мы воспользуемся базовыми функциями подобного устройства — MPU 6050.

В общем случае блоки инерциальных измерений намного дороже, но и существенно точнее обычных гироскопов и акселерометров. К тому же они подключаются к микроконтроллерам и другому оборудованию через стандартизированные промышленные интерфейсы, например I²C.

Блок MPU 6050, показанный на рис. 8.4, совмещает в себе акселерометр, гироскоп и микроконтроллер. Несмотря на то что размер электронных компонентов на этапе прототипирования не играет ключевой роли, приятно осознавать, что все функции MPU 6050 реализованы в виде единого устройства поверхностного монтажа. Во многих случаях размер все же имеет значение. Например, ранние версии нашего датчика солнечного излучения едва помещались в корпусе размером 10×10×10 см. Конечная же задача состояла в том, чтобы создать устройство поверхностного монтажа с площадью 5×5 мм.

Подключение устройства MPU 6050 осуществляется через интерфейс I²C. Благодаря библиотеке `python-smbus` программный код для Raspberry Pi значительно проще и эффективнее, чем программа с аналогичной функциональностью, написанная для Arduino. Как правило, Raspberry Pi лучше справляется со сложными интерфейсами, чем Arduino.

Стандартизированные интерфейсы

Большая часть компонентов, с которыми вам доведется работать, поддерживает один или несколько стандартизированных интерфейсов, поэтому вам не придется изобретать собственный.

Интерфейс I²C — это один из простейших стандартизированных производственных протоколов обмена данными. Поскольку передача данных в нем строго регламентирована, а информация кодируется и декодируется по однозначно определенным правилам, он по праву считается простейшим для использования. Далее вы найдете примеры его практического использования в готовых прототипах.

Еще один часто применяемый интерфейс для подключения оборудования — это SPI (Serial Peripheral Interface — последовательный периферийный интерфейс). Поскольку в SPI необходимо детально определять все правила установки соединения и обмена данными, то его использование для подключения нового оборудования сопряжено со многими трудностями. С другой стороны, если у вас есть

пример программного кода, предназначенного для управления SPI-соединением, или детальные инструкции по подключению оборудования через SPI, то вам остается только скопировать в свой проект чужие наработки, лишь слегка усовершенствовав их в случае необходимости. В качестве эталонной реализации SPI в сложных прототипах рассмотрите проект, представленный по такому адресу:

<http://botbook.com/satellite>

Последовательное соединение предстает перед нами во многих ипостасях: последовательное соединение USB, последовательное соединение Bluetooth, последовательное соединение через отдельные провода. И только монитор последовательного порта в Arduino на удивление стабилен в своих возможностях по выводу данных, поступающих через последовательное соединение. Но сам интерфейс только предоставляет возможность передачи данных, а способ кодирования и декодирования информации определяете вы.

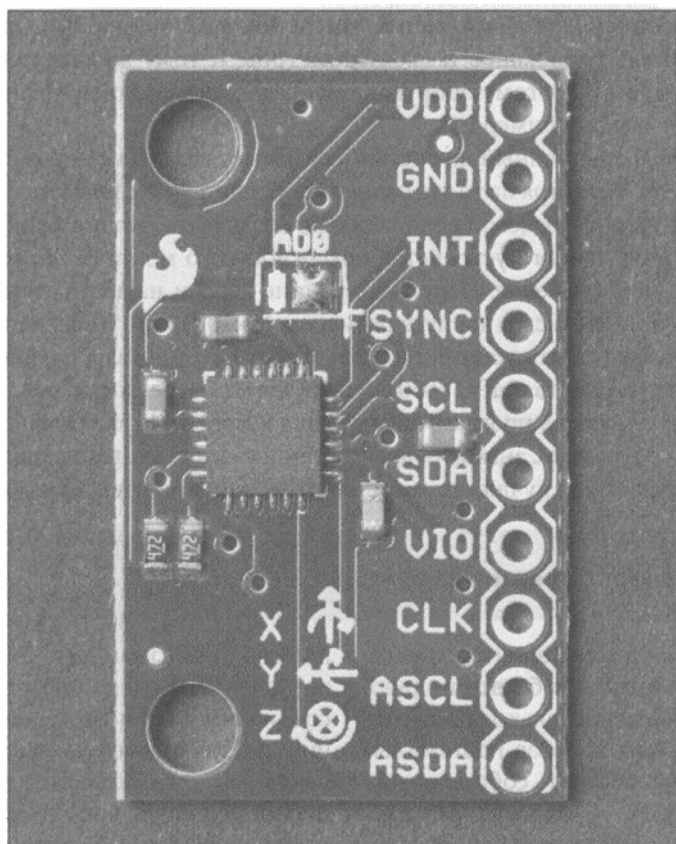


Рис. 8.4. Модуль MPU 6050

Подключение к Arduino и программа управления устройством MPU 6050

На рис 8.5 показана схема подключения блока MPU 6050 к Arduino. После ее реализации выполните программный код из листинга 8.3.

Ниже приведен сложный программный код! При написании программы управления MPU 6050 используются более сложные принципы программирования, чем те, что применялись в предыдущих проектах. Если вы ничего не хотите знать о порядке следования байтов, побитовых операциях, а также структурах (struct), то не останавливайтесь на детальном описании программы и переходите непосредственно к ее выполнению. Чтобы выполнить приведенный ниже программный код, совсем не обязательно понимать его структуру.

В подразделах, приведенных после программного кода, вы найдете подробное описание выполняемых в нем операций.

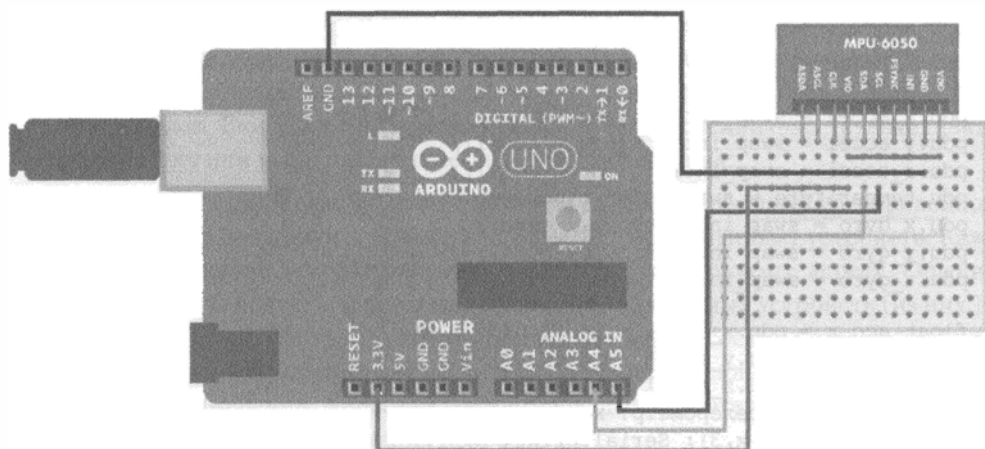


Рис. 8.5. Схема подключения модуля MPU 6050 к Arduino (см. цветную вклейку)

Листинг 8.3. `mpu_6050.ino`

```
// mpu_6050.ino - вывод на монитор ускорения (м/с2) и угловой
// скорости (градусы/с)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h> // ❶

const char i2c_address = 0x68; // ❷

const unsigned char sleep_mgmt = 0x6B; // ❸
const unsigned char accel_x_out = 0x3B;

struct data_pdu // ❹
{
    int16_t x_accel; // ❺
    int16_t y_accel;
    int16_t z_accel;
    int16_t temperature; // ❻
    int16_t x_gyro; // ❼
    int16_t y_gyro;
    int16_t z_gyro;
};

void setup() {
    Serial.begin(115200);
    Wire.begin(); // ❽
    write_i2c(sleep_mgmt, 0x00); // ❾
}

int16_t swap_int16_t(int16_t value) // ❿
{
    int16_t left = value << 8; // ⓫
    int16_t right = value >> 8; // ⓬
    right = right & 0xFF; // ⓭
    return left | right; // ⓮
}
```

```

void loop() {
    data_pdu pdu; // 15
    read_i2c(accel_x_out, (uint8_t *)&pdu, sizeof(data_pdu)); // 15

    pdu.x_accel = swap_int16_t(pdu.x_accel); // 15
    pdu.y_accel = swap_int16_t(pdu.y_accel);
    pdu.z_accel = swap_int16_t(pdu.z_accel);
    pdu.temperature = swap_int16_t(pdu.temperature); // 15
    pdu.x_gyro = swap_int16_t(pdu.x_gyro);
    pdu.y_gyro = swap_int16_t(pdu.y_gyro);
    pdu.z_gyro = swap_int16_t(pdu.z_gyro);

    float acc_x = pdu.x_accel / 16384.0f; // 15
    float acc_y = pdu.y_accel / 16384.0f;
    float acc_z = pdu.z_accel / 16384.0f;
    Serial.print("Акселерометр: x,y,z ");
    Serial.print(acc_x,3); Serial.print("g, "); // 20
    Serial.print(acc_y,3); Serial.print("g, ");
    Serial.print(acc_z,3); Serial.println("g");

    int zero_point = -512 - (340 * 35); // 20
    double temperature = (pdu.temperature - zero_point) / 340.0; // 20
    Serial.print("Температура (C): ");
    Serial.println(temperature,2);

    Serial.print("Гироскоп: x,y,z ");
    Serial.print(pdu.x_gyro / 131.0f); Serial.print(" градус/с,
    "); // 20
    Serial.print(pdu.y_gyro / 131.0f); Serial.print(" градус/с, ");
    Serial.print(pdu.z_gyro / 131.0f); Serial.println(" градус/с");
    delay(1000);
}

void read_i2c(unsigned char reg, uint8_t *buffer, int size) // 20
{
    Wire.beginTransmission(i2c_address); // 20
    Wire.write(reg); // 20
    Wire.endTransmission(false); // 20
    Wire.requestFrom(i2c_address,size,true); // 20
    int i = 0; // 20
    while(Wire.available() && i < size) { // 20
        buffer[i] = Wire.read(); // 20
        i++;
    }
    if(i != size) { // 20
        Serial.println("Ошибка получения данных через I2C");
    }
}

void write_i2c(unsigned char reg, const uint8_t data) // 20
{
    Wire.beginTransmission(i2c_address); // 20
    Wire.write(reg);
    Wire.write(data);
    Wire.endTransmission(true);
}

```


- ❶ Библиотека `Wire.h`, встроенная в Arduino, применяется для поддержки интерфейса I²C. Она поставляется вместе со средой разработки Arduino, поэтому вам достаточно подключить ее в программном коде, не беспокоясь о расположении файлов. Вам также не нужно предварительно специально устанавливать библиотеку или выполнять какие-либо другие действия.
- ❷ Адресное пространство I²C для устройства MPU 6050. К одной трехпроводной шине можно подключить множество ведомых (подчиненных) устройств. Каждое подчиненное устройство распознается по уникальному адресу. Обычно одна шина I²C поддерживает подключение до 128 ведомых устройств (из них 16 адресов зарезервировано). Длина кабелей, которыми соединяются устройства через шину I²C, не превышает нескольких метров, что сильно ограничивает область ее применения. Адреса указываются в шестнадцатеричной системе счисления, описанной в следующих подразделах.
- ❸ Регистры команд, взятые из документации к MPU 6050. Полный список поддерживаемых команд вы найдете, проведя поиск по ключевым фразам “Документация MPU 6050” и “MPU 6050 карта регистров”. Значения указываются в шестнадцатеричной системе счисления, но при необходимости могут представляться десятичными числами.
- ❹ Структура (`struct`), предназначенная для обработки ответа, поступающего с устройства. Структура позволяет комбинировать в единое целое множество значений. В языке C структура может содержать только данные, но не функции. Это отличает структуры от объектов и классов, с которыми вы можете быть знакомы по другим языкам программирования. Конструкция `struct data_pdu` объявляет новый тип данных, который впоследствии применяется при объявлении соответствующих переменных. В нашем случае структура содержит переменные, имеющие такой же размер, как и поля данных, определенные в протоколе обмена данными с датчиком. Впоследствии считанные с датчика байты будут передаваться непосредственно в текущую структуру. Для того чтобы получить необходимое значение, нам достаточно будет последовательно просмотреть переменные, включенные в структуру. Несколько необычный прием!
- ❺ Переменная для хранения величины ускорения вдоль оси X. Тип данных `int16_t` — это целочисленный тип специального размера, определенный в библиотеке `avr-libc` (библиотека C, используемая компилятором Arduino). В данном случае применяются 16-битовые (двухбайтовые) целочисленные значения со знаком. Поскольку в структуру с датчика поступают неструктурированные данные, необходимость применения типа данных строго заданного размера вполне оправдана.
- ❻ Наш блок инерциальных измерений умеет определять температуру окружающей среды. Даже если вы не собираетесь использовать эту функцию в своем проекте, для хранения значения температуры нужно предусмотреть отдельную переменную, чтобы обеспечить структуре `data_pdu` правильный размер.

- 7 Угловая скорость при вращении вокруг оси X, определяемая гироскопом, включенным в MPU 6050.
- 8 Инициализация соединения I²C средствами библиотеки `Wire.h`.
- 9 Перевод датчика в активное (рабочее) состояние передачей команды 0 в соответствующий регистр 0x6B. Этот этап очень важный, поскольку по умолчанию модуль MPU 6050 начинает свою работу со спящего режима.
- 10 Перестановка двух байтов в значении параметра. В MPU 6050 поддерживается порядок следования байтов от старшего к младшему, а в Arduino, как и в большинстве процессоров, — от младшего к старшему. При обмене данными между платформами порядок следования байтов нужно изменить на противоположный. Подробно об этом рассказывается в разделе “Порядок следования байтов — от младшего к старшему”.
- 11 Эта новая двухбайтовая (16-битовая) переменная `left` представляет самый правый байт параметра `value`. После сдвига одного байта (8 бит) влево (`<<`) левый байт переменной `value` опускается, поскольку не помещается в двухбайтовой переменной `left`. Правый байт переменной `left` при выполнении операции битового сдвига заполняется нулями.
- 12 Эта новая двухбайтовая (16-битовая) переменная `right` представляет самый левый байт параметра `value`. Левый байт переменной `right` при битовом сдвиге заполняется нулями.
- 13 Сбрасывание в нуль левого байта переменной `right`, чтобы удостовериться в отсутствии в нем данных. Детально о необходимости этой операции рассказывается далее в этой главе, при описании побитовых операций.
- 14 Совмещение левого и правого байтов. Как вы знаете, переменная `left` содержит два байта (16 бит), правый из которых заполнен нулями.
- 15 Создание новой переменной `pdu` с типом данных `data_pdu`, представляющей структуру, созданную в коде ранее.
- 16 Заполнение структуры `pdu` данными, передаваемыми датчиком. Первый параметр функции указывает провести считывание (`accel_x_out`, регистр 0x3B). В качестве второго параметра используется указатель на структуру `pdu`. Такой подход позволяет функции изменять структуру без возвращения значения. Последний параметр указывает количество считываемых байтов. Для удобства вы можете сделать количество считываемых байтов равным размеру структуры.
- 17 Преобразование значений, поступающих с датчика, из порядка от старшего к младшему в обратный порядок — от младшего к старшему.
- 18 Ссылаться на переменные структуры проще всего с помощью синтаксиса `имя_структуры.имя_переменной`, например `pdu.temperature`.
- 19 Необработанный результат ускорения преобразуется в единицы *g*. Ускорение свободного падения, *g*, равно 9,81 м/с². Коэффициент преобразования взят из технической документации к датчику. Чтобы получить результат в виде значения с плавающей запятой (в виде десятичной дроби), делитель должен также быть представлен числом с плавающей запятой.

- 23 Вывод ускорения на монитор последовательного порта. Значение представлено в единицах g (ускорение свободного падения, $g = 9,81 \text{ м/с}^2$).
- 24 Вычисление точки отсчета для преобразования необработанного значения в температуру, представленную в градусах Цельсия ($^{\circ}\text{C}$). Формула пересчета создавалась согласно техническим характеристикам устройства. Температура измеряется в диапазоне от -40°C до $+85^{\circ}\text{C}$. Необработанное значение -512 соответствует температуре 35°C . Начиная с этой точки, изменение на 1°C представляется изменением необработанного значения на 340 единиц. Таким образом, чтобы определить необработанное значение для 0°C , из -512 нужно вычесть результат умножения 340 на 35 (уменьшение температуры на 35°C , рассчитанное в необработанных единицах, — 340 единиц на каждый градус Цельсия). Несложные математические вычисления — $-512 - (340 \cdot 35)$ — приводят к результату -12412 . Но все же вместо расчетного значения -12412 лучше использовать соответствующий показатель из технической документации к датчику.
- 25 Представление необработанного значения температуры в значение, выраженное в градусах Цельсия, рассчитанное согласно формуле, представленной в документации к устройству.
- 26 Вывод угловой скорости, определенной гироскопом. Преобразование необработанного значения в стандартизированные единицы измерения (градусы/с) проводится согласно коэффициенту $1/131$, который указывается в технических характеристиках датчика. Чтобы получить результат в виде значения с плавающей запятой (в виде десятичной дроби), делитель должен также быть представлен числом с плавающей запятой.
- 27 Функция получения количества байтов `size` из регистра. Результат сохраняется в структуре (`struct`), на которую ссылается указатель `*buffer`. Указатель позволяет изменять значение структуры, вместо того, чтобы возвращать его.
- 28 Отправка команды I^2C в устройство (в модуль MPU 6050 по адресу `0x69`).
- 29 Определение адреса в регистре для считывания. В нашей программе функция `read_i2c()` используется для считывания данных только по адресу `accel_x_out` (`0x3B`).
- 30 Соединение не закрывается, поскольку считывание необходимых данных еще не выполнено.
- 31 Запрос с датчика данных, количество байт которых определяется переменной `size`. Адрес в регистре, с которого начинается считывание, определен выше — `accel_x_out` (`0x3B`). Параметр `true` (третий аргумент, который в документации к Arduino называется `stop`) указывает закрыть соединение после считывания, что приводит к освобождению шины I^2C для других нужд.
- 32 Объявление новой переменной, используемой в следующей циклической структуре `while`. Счетчик цикла `i` указывает количество уже считанных байт. Кроме того, переменная `i` соответствует количеству проведенных итераций.

- ❶ Выход из цикла осуществляется только в случае, когда считаны не все запрашиваемые данные и ожидаются байты для получения. Принимая во внимание вид функции `read_i2c()` в нашей программе, значение переменной `size` всегда будет равняться длине структуры `data_pdu`.
- ❷ Считывание байта данных (8 бит) и сохранение его в буфере. Учитывая синтаксис функции `read_i2c()`, вызываемой в нашем программном коде, рассмотрим несколько первых итераций. Указатель `*buffer` ссылается на первый байт структуры `pdu`, имеющей тип данных `data_pdu`. В первой итерации `i` равно 0, поэтому `buffer[i]` указывает на первый байт `pdu`. Поскольку в функцию передается указатель на `pdu`, то в результате ее выполнения содержимое самой переменной `pdu` в основной программе переписывается. При этом ничего не возвращается, а функция `read_i2c()` имеет тип `void`. Во второй итерации `buffer[1]` указывает на второй байт `pdu`. Выполнение циклической структуры продолжается до просмотра всего буфера (`pdu`). Как только выполнится условие `i == size`, операторы цикла `while` игнорируются, а выполнение программы продолжается с операторов, следующих после циклической структуры `while`.
- ❸ Если максимальное значение счетчика цикла `i` меньше `size`, то это означает, что считано недостаточное количество байт данных. Так как переменная `i` объявлена вне цикла, то она остается доступной во внешнем программном коде.
- ❹ Запись одного байта из переменной `data` в регистр `reg` датчика.
- ❺ Адрес для сохранения данных в регистре датчика взят из глобальной переменной `i2c_address`.

Подключение к Raspberry Pi и программа управления устройством MPU 6050

На рис. 8.6 показана схема подключения блока MPU 6050 к Raspberry Pi. После создания указанной электрической цепи выполните программный код из листинга 8.4.

Листинг 8.4. `mpu_6050.py`

```
# mpu_6050.py - вывод ускорения (м/с²) и угловой
# скорости (градус/с)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import smbus # sudo apt-get -y install python-smbus # ❶
import struct

i2c_address = 0x68 # ❷
sleep_mgmt = 0x6B # ❸
accel_x_out = 0x3B # ❹

bus = None # ❺
acc_x = 0
acc_y = 0
acc_z = 0
```

```

temp = 0
gyro_x = 0
gyro_y = 0
gyro_z = 0

def initmpu():
    global bus # ❶
    bus = smbus.SMBus(1) # ❷
    bus.write_byte_data(i2c_address, sleep_mgmt, 0x00) # ❸

def get_data():
    global acc_x, acc_y, acc_z, temp, gyro_x, gyro_y, gyro_z
    bus.write_byte(i2c_address, accel_x_out) # ❹
    rawData = ""
    for i in range(14): # ❺
        rawData += chr(bus.read_byte_data(i2c_address, accel_x_out+i)) # ❻
    data = struct.unpack('>hhhhh', rawData) # ❼

    acc_x = data[0] / 16384.0 # ❶
    acc_y = data[1] / 16384.0
    acc_z = data[2] / 16384.0
    zero_point = -512 - (340 * 35) # ❷
    temp = (data[3] - zero_point) / 340.0 # ❸

    gyro_x = data[4] / 131.0 # ❹
    gyro_y = data[5] / 131.0
    gyro_z = data[6] / 131.0

def main():
    initmpu()
    while True: # ❶
        get_data() # ❷
        print("Данные:")
        print("Ускорение (%.3f,%.3f,%.3f) g, " % (acc_x, acc_y,
↳ acc_z)) # ❸
        print("Темп. %.1f C, " % temp)
        print("Угловая скорость (%.3f,%.3f,%.3f) градусы/с" %
↳ (gyro_x, gyro_y, gyro_z))
        time.sleep(0.5) # s # ❹

if __name__ == "__main__":
    main()

```

- ❶ В библиотеке SMBus реализована поддержка стандарта I²C для Raspberry Pi. Эта библиотека позволяет сделать программу управления устройствами I²C в Raspberry Pi намного короче, чем ее аналог в Arduino. Чтобы получить доступ к этой библиотеке, необходимо установить в Raspberry Pi программный пакет `python-smbus` (детально об этом рассказывается в разделе “SMBus и поддержка I²C”).
- ❷ Адрес MPU 6050 для I²C указывается в документации к устройству. Числа представляются в шестнадцатеричной системе счисления, которая детально описана в разделе “Шестнадцатеричная, двоичная и другие системы счисления”.

- 3 Адрес регистра команд. Карту регистров вы легко найдете в Интернете, проведя поиск по фразе “MPU 6050 карта регистров”.
- 4 Адрес регистра, хранящий ускорение вдоль оси X, — это первый из набора адресов определяемых параметров: ускорение, температура и угловая скорость.
- 5 Объявление глобальной переменной `bus`, доступной всем функциям.
- 6 Чтобы иметь возможность изменять значение глобальной переменной в функции, необходимо указать, что она глобальная, в начале самой функции.
- 7 Инициализация класса `SMBus` (шина I²C). Сохранение нового объекта класса `SMBus` в глобальной переменной `bus`.
- 8 Модуль MPU 6050 начинает работу со спящего режима. Перед обменом данными с ним выведите его из спящего режима. Команды блоку инерциальных измерений передаются через шину I²C (объект `SMBus`) с указанием адреса устройства, регистра и записываемого в регистр значения.
- 9 Запрос данных, начиная с ускорения вдоль оси X.
- 10 Операция повторяется 14 раз, каждый раз для нового значения счетчика `i` (от 0 до 13).
- 11 Получение текущего байта данных, преобразование его в формат ASCII и добавление в строку `rawData`.
- 12 Конвертация строки `rawData` в кортеж Python. Операторы форматирующей строки указывают представлять данные в виде короткого (2 байта, 16 бит) целочисленного значения со знаком в представлении от младшего к старшему (<).
- 13 Преобразование необработанного значения ускорения в единицы g (ускорение свободного падения, $9,81 \text{ м/с}^2$).
- 14 Вычисление точки отсчета (нуля) температурной шкалы. В Raspberry Pi эта задача выполняется очень просто, а включение вычислений в программный код делает его понятнее при дальнейшей отладке или применении в других проектах.
- 15 Преобразование необработанного значения температуры в градусы Цельсия. Чтобы вывести результат как дробное значение (с плавающей точкой), делитель также нужно представить в этом формате. Соответствующие формулы приведены в технической документации к устройству.
- 16 Преобразование необработанного значения угловой скорости в стандартизированное значение, представленное в градусах в секунду.
- 17 Выполнение программы длится до нажатия комбинации клавиш <Ctrl+C>.
- 18 Функция `get_data()` обновляет глобальные переменные, не возвращая никакого значения.
- 19 Вывод значения ускорения с использованием форматирующей строки. Оператор `% .3f` определяет дробный формат вывода данных с тремя знаками после десятичной запятой.
- 20 Задержка позволяет вам не спеша ознакомиться с выводимыми значениями и предотвращает постоянную полную загрузку процессора.

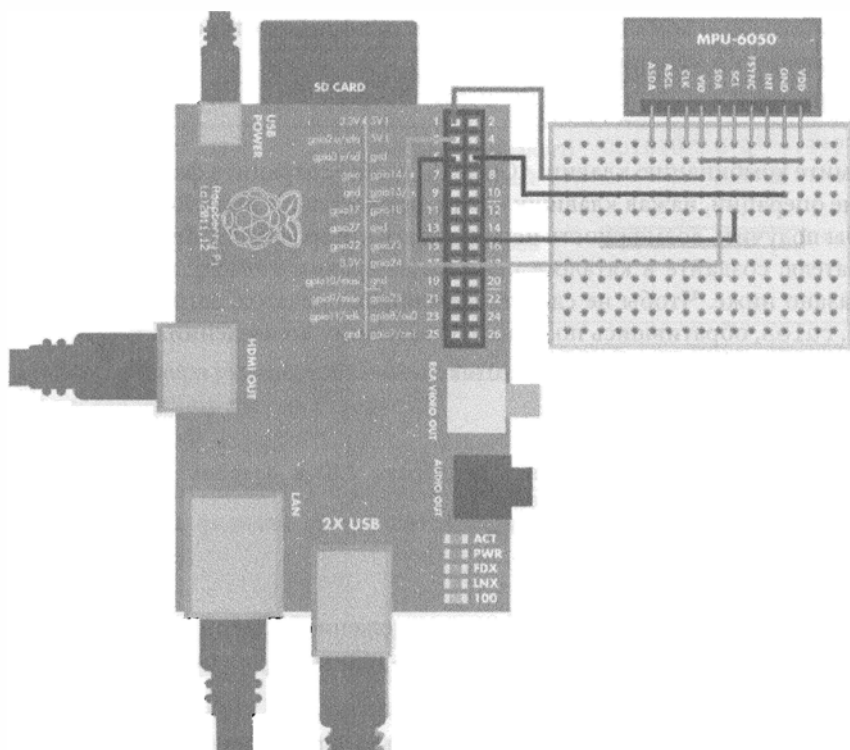


Рис. 8.6. Схема подключения модуля MPU 6050 к Raspberry Pi (см. цветную вклейку)

Отследите, как изменяются показания на мониторе последовательного порта при изменении положения датчика в пространстве и его вращении в любом направлении. А что будет, если его несильно потрясти?

SMBus и поддержка I²C

В приведенном выше программном коде, выполняемом в Raspberry Pi, применяется библиотека SMBus, которая обеспечивает поддержку в Python шины I²C. К счастью, установка программного обеспечения в Linux не вызывает особых затруднений. В Raspbian установка новых программных средств выполняется так же, как и в Debian, Ubuntu и Mint. Для начала дважды щелкните на значке LXTerminal, расположенном в левой части рабочего стола Raspbian. Далее введите такие команды.

```
$ sudo apt-get update
$ sudo apt-get install python-smbus
```

Чтобы обеспечить поддержку I²C, необходимо подключить ее программные модули. Сначала убедитесь, что они еще не подключены. Откройте файл /etc/modprobe.d/raspi-blacklist.conf, воспользовавшись командой `sudoedit/etc/modprobe.d/raspi-blacklist.conf`, и удалите в нем следующую строку:

```
blacklist i2c-bcm2708
```

Сохраните файл, нажав комбинацию клавиш <Ctrl+X>, далее нажмите клавишу <Y>, а затем <Enter> или <Return>.

Затем с помощью команды `sudoedit /etc/modules` откройте соответствующий ресурс и добавьте в него две строки.

```
i2c-bcm2708
i2c-dev
```

Нажмите комбинацию клавиш `<Ctrl+X>`, чтобы сохранить файл. Подтвердите выполнение операции, нажав клавишу `<Y>`, а затем `<Enter>` или `<Return>`.

Чтобы получить возможность использовать интерфейс I²C без привилегий суперпользователя, создайте в каталоге `udev` файл правил `99-i2c.rules` (листинг 8.5), как показано ниже. Чтобы не сделать случайных опечаток, загрузите готовый файл `99-i2c.rules`, обратившись по адресу, указанному во введении.

```
$ sudo cp 99-i2c.rules /etc/udev/rules.d/99-i2c.rules
```

Листинг 8.5. 99-i2c.rules

```
# /etc/udev/rules.d/99-i2c.rules - поддержка I2C в Raspberry Pi
# без привилегий суперпользователя
# Copyright 2013 http://BotBook.com
```

```
SUBSYSTEM=="i2c-dev", MODE="0666"
```

Перезагрузите Raspberry Pi, запустите приложение LXTerminal и выполните следующую команду, чтобы проверить доступность шины I²C для обычного пользователя:

```
$ ls -l /dev/i2c*
```

В результате на экране должен отобразиться список из двух файлов, имеющих атрибуты `crw-rw-rwT`. Если у вас другой результат, то выполните описанные выше действия еще раз.

Шестнадцатеричная, двоичная и другие системы счисления

Одно и то же число можно представить несколькими способами. Например, десятичное число 65 в шестнадцатеричной системе счисления записывается как `0x41`, а в двоичной — как `0b1000001`. Нам привычнее всего работать в десятичной системе, в которой $5+5 = 10$.

Чтобы различать системы счисления, в которых представляется число, перед самим числом добавляется специальный префикс. В десятичной системе счисления, как общепринятой во всех сферах жизнедеятельности, такой префикс отсутствует. Шестнадцатеричные числа начинаются с префикса `0x`, двоичные — с префикса `0b`, восьмеричные — с `0`.

Представление о наиболее используемых системах счисления можно получить в табл. 8.2.

Таблица 8.2. Представление чисел в разных системах счисления

Префикс	Система счисления	Основание	Применение	Пример	Пример вычислений
0x	Десятичная	10	Программный код C и C++, техническая документация	10	$0 \cdot 10^0 + 1 \cdot 10^1$
	Шестнадцатеричная	16		0xA	$10 \cdot 16^0$

Префикс	Система счисления	Основание	Применение	Пример	Пример вычислений
0b	Двоичная	2	Протоколы низкого уровня, битовые операции	0b1010	$0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3$
0	Восьмеричная	8	Изменение прав доступа командой Chmod в Linux	012	$2 \cdot 8^0 + 1 \cdot 8^1$

В качестве примера рассмотрим число 42. Ниже показано, как оно выглядит в каждой из представленных систем счисления.

$$42 = 0x2a = 0b101010 = 052$$

В привычной вам десятичной системе счисления основание равно 10. Начиная с последней цифры, сначала вычисляется количество единиц, а затем количество десятков.

$$2 \cdot 1 + 4 \cdot 10 = 42$$

В случае больших значений, например 1917, число не ограничивается вторым разрядом. После единиц подсчитывается количество десятков (10), затем сотен (10*10) и в конце — тысяч (10*10*10). Вычисления проводить очень просто, если возводить основу в степень соответствующего разряда:

$$\begin{aligned} 10 \cdot 10 &= 10^2 \\ 10 \cdot 10 \cdot 10 &= 10^3 \end{aligned}$$

А как же единицы? Как известно, любое число в нулевой степени равняется единице:

$$10^0 = 1$$

Таким образом, число 42 представляется так:

$$42 = 4 \cdot 10^1 + 2 \cdot 10^0$$

В шестнадцатеричной системе счисления число 42 выражается как 0x2A. В этой системе числа, большие 9, обозначаются буквами: A=10, B=11... F=15. Начиная с крайнего правого разряда и помня, что A представляет число 10, легко провести следующие преобразования:

$$0x2A = 10 \cdot 1 + 2 \cdot 16 = 10 + 32 = 42$$

Если возводить основу каждого разряда в соответствующую степень, то результат должен получиться таким же:

$$10 \cdot 16^0 + 2 \cdot 16^1 = 0x2A$$

Попробуйте преобразовать другие числа в произвольные системы счисления. Для проверки проведенных вычислений воспользуйтесь консолью Python (подробно о ней рассказано во врезке “Консоль Python”) или сверьтесь с табл. 8.3. Попробуйте самостоятельно преобразовать произвольное число в двоичную систему счисления.

Вы можете попрактиковаться с числовыми преобразованиями в консоли Python. Формат представления числовых значений (1 == 0x1 == 0b1) одинаков в Python, C и C++. В консоли Python выполняются любые программные команды.

```
>>> print("Botbook.com")
Botbook.com
>>> 2+2
4
```

Любые числовые значения автоматически преобразовываются в десятичный вид.

```
>>> 0x42
66
>>> 66
66
>>> 0b1000010
66
```

Ниже приведены функции преобразования любых чисел в двоичную, шестнадцатеричную, восьмеричную систему, а также символы ASCII.

```
>>> bin(3)
'0b11'
>>> hex(10)
'0xa'
>>> oct(8)
'010'
>>> chr(0x42)
'B'
```

Таблица 8.3. Примеры чисел, представленных в разных системах счисления

Десятичная	Шестнадцатеричная	Двоичная	Восьмеричная	ASCII
0	0x0	0b0	0	\0 — NUL, завершение строки
1	0x1	0b1	01	
2	0x2	0b10	02	
3	0x3	0b11	03	
4	0x4	0b100	04	EOT, конец текста, <Ctrl+D> в Linux
5	0x5	0b101	05	
6	0x6	0b110	06	
7	0x7	0b111	07	
8	0x8	0b1000	010	\n — новая строка
9	0x9	0b1001	011	
10	0xA	0b1010	012	
11	0xB	0b1011	013	
16	0x10	0b10000	020	
17	0x11	0b10001	021	

Десятичная	Шестнадцатеричная	Двоичная	Восьмеричная	ASCII
32	0x20	0b100000	040	' ' — пробел, первый выводимый символ
48	0x30	0b110000	060	0 — ноль ASCII (не числовой ноль)
65	0x41	0b1000001	0101	A
97	0x61	0b1100001	0141	a
126	0x7e	0b1111110	0176	~ — тильда, последний выводимый символ ASCII

Введите в терминале команду `man ascii`, чтобы вывести в Linux или OS X таблицу символов ASCII. В ней, например, показано, что символу A в ASCII соответствует десятичное числовое значение 65, шестнадцатеричное — 0x41 и восьмеричное — 0101. Руководство снабжено примитивной системой управления постраничным просмотром. Для перехода к следующей странице нажмите <Пробел>; чтобы перейти на страницу назад, нажмите , а для выхода из руководства нажмите <Q>.

Консоль Python

Консоль Python запускается при введении в терминале команды `python`. В Linux терминал (он же командная оболочка или `bash`) запускается с помощью команды **Accessories** ⇒ **Terminal** (**Стандартные** ⇒ **Терминал**). Вы также можете провести поиск команды **Terminal** (**Терминал**) в главном меню **Dash**.

В Macintosh терминал запускается из каталога `/Applications/Utilities/Terminal`. В Windows можно запустить терминал с помощью команды **Applications** ⇒ **Accessories** ⇒ **Command Prompt** (**Все программы** ⇒ **Стандартные** ⇒ **Командная строка**) или запустить приложение IDLE из меню **Start** (**Пуск**) (если, конечно, установлена среда разработки Python).

Приглашение Python `>>>` говорит о том, что все вводимые далее команды будут интерпретированы как программный код Python. Чтобы завершить сеанс работы в терминале Python, введите команду `exit()`.

Если вам нужна полноценная консоль Python, поддерживающая автоматическое дополнение и снабженная интерактивным справочным руководством, то воспользуйтесь оболочкой `ipython`.

Побитовые операции

Отдельные датчики передают данные в виде потока битов, организованных в байты, например 01010000. Для обработки таких данных, представленных в двоичной системе счисления, применяются поразрядные, или побитовые, операции, которые позволяют управлять сразу целыми последовательностями битов.

Бит представляет собой единицу измерения количества информации, равную одному разряду в двоичной системе счисления, 1 или 0. Бит также может применяться для хранения булева значения: 0 — ложь или 1 — истина.

Байт — это единица хранения информации, состоящая из восьми битов, например 0b1010100. Байт представляет один символ в кодировке ASCII, например T, 3 или x. Кроме того, байт может представлять числовое значение, такое как 256 или -127.

Побитовые операции относятся к низкоуровневым командам (они чаще всего применяются для прямого управления памятью), а представляющий их программный код сложен для понимания, поэтому к ним обращаются только в случае крайней необходимости. С другой стороны, существуют датчики, передающие необработанные данные, обработать которые можно только с помощью двоичной арифметики. Для низкоуровневого управления таким устройствами, а к ним относится и блок интегральных измерений MPU 6050, вам придется овладеть побитовыми операциями в совершенстве.

Чаще всего к побитовым относят все логические операции и битовый сдвиг.

Чтобы лучше понять, к каким результатам приводит выполнение таких операций, попрактикуйтесь работе с ними в консоли Python. Побитовые операции работают также в C и C++ (Arduino), но в этих средах программирования не предусмотрена консоль, поэтому экспериментировать в Python гораздо проще.

Вы, скорее всего, знакомы с булевой алгеброй. Оператор И (AND) указывает на истинность (True) результата только в случае истинности обоих условий. Оператор ИЛИ (OR) определяет истинность результата в случае истинности хотя бы одного из условий. Второе условие при этом может быть ложным (False).

```
>>> if (True): print "Yes"
...
Yes
>>> if (False): print "Yes"
... # ничего не выводится
>>> if (True and True): print "Yes"
...
Yes
```

В Python значения True и False обязательно вводятся с прописной буквы.

Логические значения могут применяться отдельно от конструкции `if`.

```
>>> True
True
>>> False
False
>>> True and True
True
```

В большинстве языков программирования 1 соответствует значению `True`, 0 — значению `False`.

```
>>> 1 and 1
1
>>> 1 and 0
0
```

В побитовых операциях вы можете выполнять подобные булевы операции над каждой единицей и нулем байта. Но в поразрядных операциях Python не применяются операторы `AND` и `OR`. Вместо них используются операторы `&` (побитовое И) и `|` (побитовое ИЛИ).

```
>>> 0b0101 & 0b0110
4
```

Иногда в двоичном виде результат выглядит проще для понимания.

```
>>> bin(0b101 & 0b110)
'0b100'
```

Действие побитового И (`&`) заключается в применении логического И к каждой паре битов, которые стоят на одинаковых позициях двух байтов.

```
0b0101
0b0110
=====
& 0b0100
```

Начиная с крайнего левого бита: 0-ложь и 0-ложь, результат 0-ложь; 1-истина и 1-истина, результат 1-истина; 0-ложь и 1-истина, результат 0-ложь; 1-истина и 0-ложь, результат 0-ложь.

Битовое маскирование с помощью побитового И (AND)

Битовое маскирование позволяет получить отдельные биты или поля из нескольких битов двоичного значения. Например, вы можете получить значение четвертого бита (отсчитывая слева) числа `0b 1010 1010`, выполнив такую операцию:

```
>>> bin(0b10101010 & 0b11110000)
'0b10100000'
```

Ниже показано, как выполняется маскирование.

```
0b 1010 1010 # число
0b 1111 0000 # маска
=====
0b 1010 0000 # & (битовое И или конъюнкция)
```

Побитовое И возвращает результат True, если истинны оба входных значения (табл. 8.4).

Таблица 8.4. Таблица истинности для операции побитового И

a	b	a И b
0	0	0
1	0	0
0	1	0
1	1	1

Побитовое ИЛИ

Что если левая и правая часть одного байта создаются отдельно, но их нужно вывести в виде единого результата? В Python для этих целей применяется операция побитового ИЛИ (`|`).

```
>>> left=0b10100000
>>> right=0b1111
>>> bin(left|right)
'0b10101111'
```

Детально это выглядит так:

```
0b 1010 0000 # левая часть дополняется справа нулями
0b 1111 # дополнять правую часть нулями слева бессмысленно
=====
0b 1010 1111 # побитовое ИЛИ (|)
```

Побитовое ИЛИ — это далеко не то же самое, что операция сложения, представляемая оператором `+`. Чтобы убедиться в этом, сложите два числа: `0b1 + 0b1`.

Битовый сдвиг

В процессе битового сдвига биты смещаются в поле влево или вправо. В Python это выглядит так.

```
>>> bin(0b0011<<1)
'0b0110'
>>> bin(0b0011<<2)
'0b1100'
```

При сдвиге битов за правый край числа лишние биты опускаются.

```
>>> bin(0b0011>>2)
'0b00'
```

Прямой и обратный порядок следования байтов

В большинстве аппаратных устройств используется прямой порядок следования байтов. Этим представление чисел в компьютерах отличается от использования чисел в нашей повседневной жизни. Мы не задумываемся над этим, но используемые нами числовые значения также соответствуют определенному правилу следования разрядов. Например, в числе 1996 слева направо указываются тысячи, затем сотни, потом десятки и только в конце — единицы. Таким образом, самый значимый разряд указывается первым, что соответствует обратному порядку следования байтов.

Чаще всего при операциях с данными в компьютерах используется прямой порядок следования байтов, когда самый младший байт числа располагается по младшему адресу. Настольные системы и ноутбуки имеют архитектуру x86, amd64 или x86-64, поэтому при передаче и обработке информации в них также используется прямой порядок следования байтов. Платформа Arduino базируется на микроконтроллерах Atmel AVR, поэтому обрабатывает данные по такому же принципу. Как Raspberry Pi, так и Android (Linux) — на данный момент популярнейшая платформа для мобильных устройств — основаны на архитектуре ARM, в которой также используется прямой порядок следования байтов.

Ниже приведен пример представления десятичного числа 135 в двоичном формате при использовании прямого и обратного порядка следования битов с точки зрения системы передачи данных. Представьте себе, что данные передаются по последовательному каналу связи слева направо, т.е. самым первым будет передаваться крайний правый бит числа. Поэтому для приемной системы порядок следования битов приобретает важное значение.

```
0b 1000 0111 # прямой, более распространенный
               # формат: Arduino, Raspberry Pi, рабочие станции
0b 0111 1000 # обратный порядок следования байтов, принятый в MPU 6050
```

Несмотря на популярность прямого порядка следования байтов, ряд устройств поддерживают другой формат передачи информации. Блок инерциальных измерений MPU 6050 относится именно к таким устройствам, поскольку самый значимый байт располагается в крайней правой позиции и будет передаваться первым. Устанавливая соединение между Arduino и MPU 6050, вам нужно учитывать это и научиться правильно конвертировать один формат в другой.

Порядок следования байтов учитывается только при выполнении низкоуровневых операций, заключающихся в обработке данных на уровне байтов и битов. В высокоуровневом программировании вам достаточно назначить переменной значение в формате с плавающей запятой и в дальнейшем оперировать только ею. Среда разработки выполнит все необходимые действия по согласованию порядка обработки байтов автоматически.

Эксперимент в окружающей среде: подключение контроллера Wii Nunchuk к порту I²C

Вам нужен контроллер, совмещающий в себе функции акселерометра, джойстика и переключателя, — и все это в удобном корпусе? Можете не утруждать себя поисками,

поскольку контроллер Nunchuk, разработанный для игровых консолей Wii, включает все необходимое. Если вам кажется, что его стоимость неоправданно высока, то поищите его более дешевые копии — они существуют и прекрасно справляются с возложенными на них задачами.

Взяв в руки Wii Nunchuk, вы надолго усвоите простую истину: разработчики и инженеры тоже люди. И как большинство из нас, обычных людей, инженеры не спешат утруждать себя разработкой новых технологий до тех пор, пока прекрасно работают старые. Разработчикам намного комфортнее работать с проверенными и стандартизированными интерфейсами, снабженными детальными библиотеками и разносторонней технической поддержкой. Контроллер Wii Nunchuk подключается через специальный (запатентованный!) разъем, но комплектуется скромной и малоинформативной документацией. Но под эффектным корпусом скрывается стандартное оборудование, подключаемое через шину I²C. Как вы знаете из врезки “Стандартизированные интерфейсы”, стандарт I²C — это один из популярнейших промышленных интерфейсов, предназначенный для подключения оборудования через соединения малого радиуса действия.

Компания Nintendo производит Wii Nunchuk просто гигантскими партиями, что предопределяет их небольшую стоимость и высокое качество сборки. Широкая распространенность Wii Nunchuk также означает, что вы можете найти большое количество программных решений для использования этого контроллера в сторонних проектах, не говоря уже о подробном описании технических характеристик. Вам даже не придется срезать запатентованный разъем с кабеля, достаточно приобрести адаптер WiiChuck, чтобы подключить контроллер к Arduino или макетной плате (рис. 8.7).

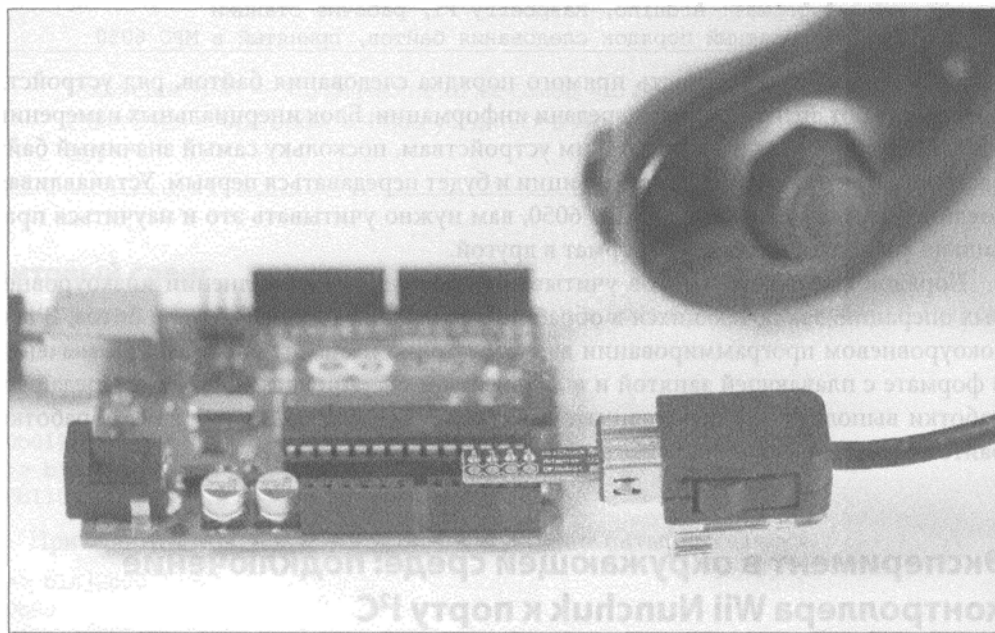


Рис. 8.7. Подключение контроллера Wii Nunchuk к Arduino с помощью специального адаптера

Соединение с Wii Nunchuk устанавливается с помощью протокола SMBus. Это упрощенный протокол, основанный на стандартизированном протоколе I²C, что предопределяет способ его управления в программном коде.

Подключение к Arduino и программа управления контроллером Wii Nunchuk

На рис. 8.8 показана схема подключения Wii Nunchuk к Arduino. Подключите согласно ей контроллер к плате и выполните программный код, приведенный в листинге 8.6.

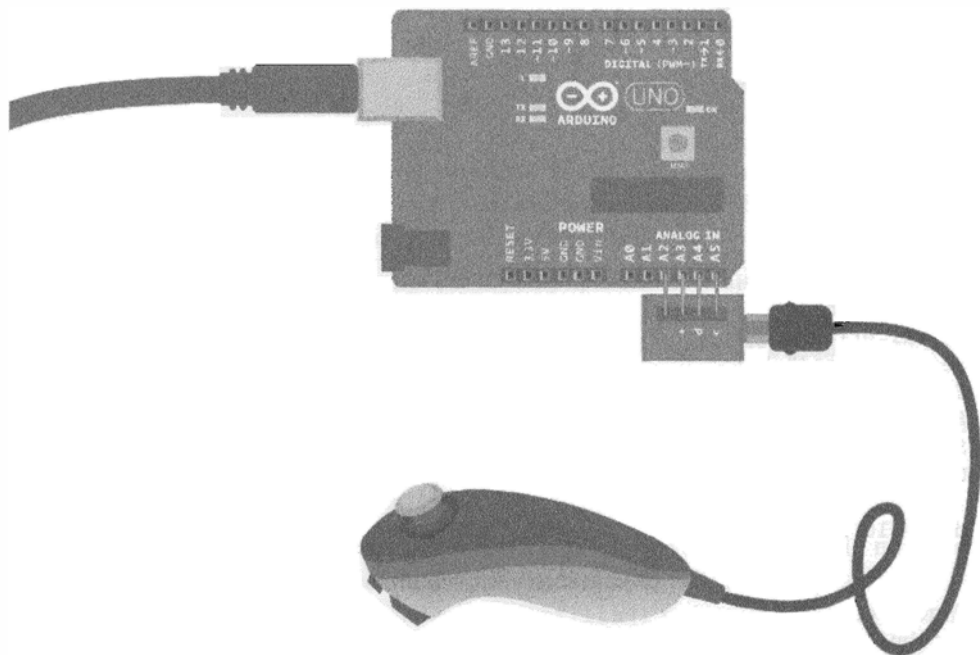


Рис. 8.8. Схема подключения адаптера и контроллера Wii Nunchuk к Arduino

Листинг 8.6. `wiichuck_adapter.ino`

```
// wiichuck_adapter.ino - вывод сигналов, передаваемых с
// джойстика, акселерометра и кнопки
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>

const char i2c_address = 0x52;

unsigned long lastGet=0; // мс
int jx = 0, jy = 0, accX = 0, accY = 0, accZ = 0, buttonZ = 0,
buttonC = 0; // ❶

void setup() {
```

```

Serial.begin(115200);
Wire.begin();
pinMode(A2, OUTPUT);
pinMode(A3, OUTPUT);
digitalWrite(A2, LOW); // ❶
digitalWrite(A3, HIGH); // ❷
delay(100);
initNunchuck(); // ❸
}

void loop() {
  if(millis() - lastGet > 100) { // ❹
    get_data(); // ❺
    lastGet = millis(); // ❻
  }
  Serial.print("Кнопка Z: ");
  Serial.print(buttonZ); // ❼
  Serial.print(" Кнопка C: ");
  Serial.print(buttonC);
  Serial.print(" Джойстик (x,y): (");
  Serial.print(jx); // ❽
  Serial.print(",");
  Serial.print(jy);
  Serial.print(" Ускорение (x,y,z): (");
  Serial.print(accX); // ❾
  Serial.print(",");
  Serial.print(accY);
  Serial.print(",");
  Serial.print(accZ);
  Serial.println(")");

  delay(10); // мс
}

void get_data() {
  int buffer[6]; // ❶
  Wire.requestFrom(i2c_address, 6); // ❷
  int i = 0; // ❸
  while(Wire.available()) { // ❹
    buffer[i] = Wire.read(); // ❺
    buffer[i] ^= 0x17; // ❻
    buffer[i] += 0x17; // ❼
    i++;
  }
  if(i != 6) { // ❽
    Serial.println("Ошибка доступа к I2C");
  }
  write_i2c_zero(); // ❾

  buttonZ = buffer[5] & 0x01; // ❿
  buttonC = (buffer[5] >> 1) & 0x01; // ⓫
  jx = buffer[0]; // ⓬
  jy = buffer[1];
  accX = buffer[2];

```

```

    accY = buffer[3];
    accZ = buffer[4];
}

void write_i2c_zero() {
    Wire.beginTransmission(i2c_address);
    Wire.write(0x00);
    Wire.endTransmission();
}

void initNunchuck()
{
    Wire.beginTransmission(i2c_address);
    Wire.write(0x40);
    Wire.write(0x00);
    Wire.endTransmission();
}

```

- ❶ Объявление глобальных переменных. Поскольку функции Arduino не поддерживают возвращение сразу нескольких значений, данные передаются из функции в функцию через глобальные переменные.
- ❷ В качестве шины заземления (GND, 0 В) применяется аналоговый вывод A2. Таким образом, адаптер WiiChuck можно будет надеть непосредственно на штыревой разъем Arduino, не прибегая к использованию макетной платы и проводочных перемычек.
- ❸ На вывод A3 подается питание +5 В (обеспечивающее сигнал высокого уровня).
- ❹ Инициализация Wii Nunchuk в результате вызова функции `initNunchuck()`, передающей команды 0x40 с кодом 0x00 через I²C.
- ❺ Получение сигналов каждые 100 мс. Демонстрирует стандартный прием в программировании, обеспечивающий выполнение действий через строго заданные временные интервалы. Функция `millis()` возвращает время, прошедшее с момента подачи питания на плату Arduino (продолжительность работы платы), выраженное в миллисекундах. С помощью переменной `lastGet` определяется время, которое прошло с момента вызова функции `get_data()`, также отсчитанное от начала подачи на плату питания и выраженное в миллисекундах. (В первой итерации `lastGet` равно 0.) Разница между возвращенным `millis()` значением и значением `lastGet` определяет время последнего вызова функции `get_data()`. Если с момента последнего вызова функции `get_data()` прошло больше 100 мс, то выполняется программный блок, приведенный ниже.
- ❻ Так как функция `get_data()` обновляет значения глобальных переменных, то ей нет необходимости возвращать значение.
- ❼ Обновление времени последнего вызова функции `get_data()`.
- ❸ Кнопка имеет два состояния: 0 — нажата, 1 — отжата.
- ❹ Наклон джойстика в обоих направлениях (jx, jy) представляется необработанными значениями, изменяющимися в диапазоне от 30 до 220.

- 10 Ускорение, измеряемое акселерометром (`accX`, `accY`, `accZ`), представляется не-обработанными значениями в диапазоне от 80 до 190 единиц.
- 11 Объявление нового массива, содержащего шесть целочисленных значений.
- 12 Запрос шести байтов данных у контроллера Wii Nunchuk.
- 13 Счетчик цикла `i` указывает номер обрабатываемого байта.
- 14 Вход в цикл осуществляется только в случае существования полученных байтов.
- 15 Получение байта и сохранение его в текущем элементе массива `buffer[]`. В первой итерации обрабатывается элемент `buffer[0]`, в последней — `buffer[5]`.
- 16 Операция исключающего ИЛИ (`XOR`) для текущего значения и `0x17` с заменой текущего значения полученным результатом (операция с заменой). Эта булева операция подобна операции ИЛИ с тем лишь отличием, что в ней истинность выражения достигается при истинности одного из условий, но не при истинности их обоих.
- 17 Добавление `0x17` к текущему значению.
- 18 Если количество полученных байтов отлично от шести, то выводится предупреждающее сообщение.
- 19 Запрос новых данных отправкой кода `0x00` через шину I²C.
- 20 Получение последнего бита последнего байта. Последний байт хранится в элементе массива `buffer[5]`. Последний бит извлекается в результате битового маскирования — операция побитового И выполняется для байта и маски `0b 0000 0000 0000 0001`. Детально о битовом маскировании см. в разделе “Битовое маскирование с помощью побитового И (`AND`)”.
- 21 Получение предпоследнего бита последнего байта. Предыдущий бит смещается на последнюю позицию с помощью операции `>> 1`. Далее над этим битом выполняются действия, описанные в предыдущем пункте. Операция битового сдвига описана в разделе “Битовый сдвиг”.
- 22 Положение джойстика `jx` вдоль оси X представляется одним байтом. Положение джойстика вдоль остальных осей и величина ускорения в трехмерном пространстве (вдоль трех осей) рассчитываются подобным образом.

Таблица 8.5. Байты данных, передаваемые контроллером Wii Nunchuk

Байт	Использование
0	Джойстик, ось X
1	Джойстик, ось Y
2	Акселерометр, ось X
3	Акселерометр, ось Y
4	Акселерометр, ось Z
5	<code>ZCxxuyzz</code> : кнопки "Z" и "C", точность акселерометра

Подключение к Raspberry Pi и программа управления контроллером Wii Nunchuk

На рис. 8.9 показана схема подключения контроллера Wii Nunchuk к Raspberry Pi. Подключив контроллер через адаптер с помощью макетной платы и проводочных перемычек, выполните программный код, приведенный в листинге 8.7.

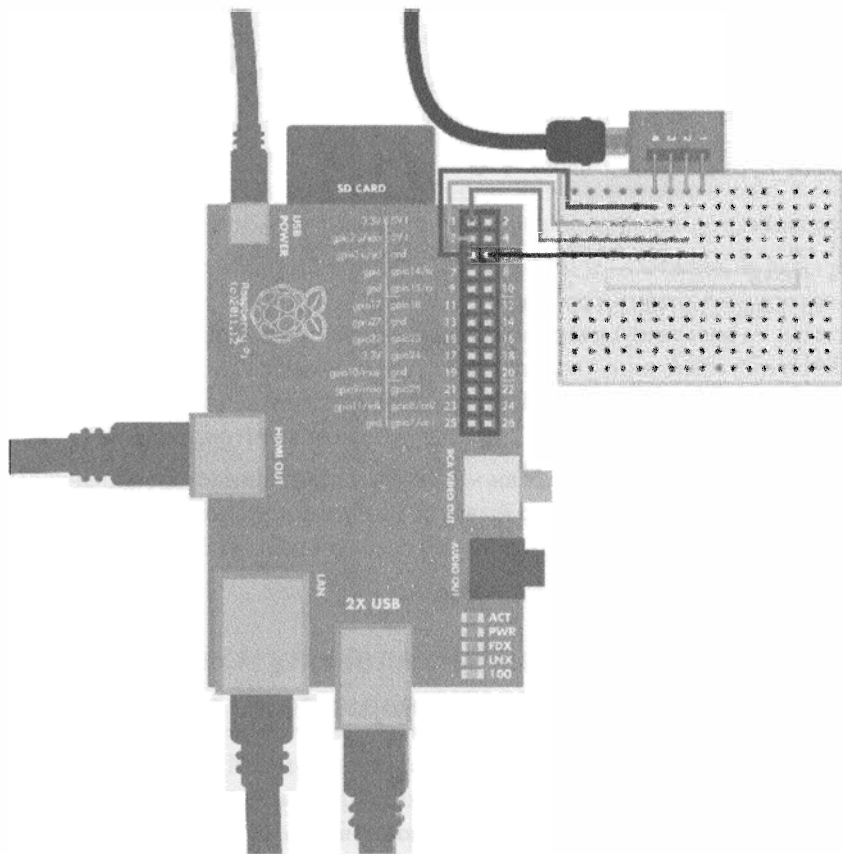


Рис. 8.9. Схема подключения адаптера и контроллера Wii Nunchuk к Raspberry Pi

Листинг 8.7. `wiichuck_adapter.py`

```
# wiichuck_adapter.py - вывод ускорения и положения джойстика в  
# Wii Nunchuck  
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time  
import smbus # sudo apt-get -y install python-smbus # ❶  
  
bus = None  
address = 0x52 # ❷  
  
z = 0 # ❸
```

```

c = 0
joystick_x = 0
joystick_y = 0
ax_x = 0
ax_y = 0
ax_z = 0

def initNunchuck():
    global bus
    bus = smbus.SMBus(1) # ❶
    bus.write_byte_data(address, 0x40, 0x00) # ❷

def send_request():
    bus.write_byte(address, 0x00) # ❸

def get_data():
    global bus, z, c, joystick_x, joystick_y, ax_x, ax_y, ax_z
    data = [0]*6
    for i in range(len(data)): # ❹
        data[i] = bus.read_byte(address)
        data[i] ^= 0x17
        data[i] += 0x17

    z = data[5] & 0x01 # ❺
    c = (data[5] >> 1) & 0x01 # ❻

    joystick_x = data[0]
    joystick_y = data[1]
    ax_x = data[2]
    ax_y = data[3]
    ax_z = data[4]
    send_request()

def main():
    initNunchuck()
    while True:
        get_data()
        print("Кнопка Z: %d Кнопка C: %d Джойстик (x,y) (%d,%d) \
Ускорение (x,y,z) (%d,%d,%d)" \ %
(z,c,joystick_x,joystick_y,ax_x, ax_y, ax_z))
        time.sleep(0.1)

if __name__ == "__main__":
    main()

```

- ❶ Для реализации нашего подключения необходимо установить в Raspberry Pi библиотеку `smbus` (см. выше).
- ❷ Адрес устройства Wii Nunchuk, указанный в виде шестнадцатеричного значения (о различных системах счисления и форматах представления числовых значений в них см. в разделе “Шестнадцатеричная, двоичная и другие системы счисления”).
- ❸ Глобальная переменная для одной из кнопок.

- 4 Создание нового объекта класса `SMBus` и сохранение его в новой переменной `bus`. Конструктор `SMBus()` принимает только один аргумент — номер устройства. Число 1 обозначает файл `/dev/i2c-1`. Данный номер устройства характерен для последних версий плат Raspberry Pi. Если у вас первая модель Raspberry Pi, то воспользуйтесь номером 0 для указания файла `/dev/i2c-0`.
- 5 Инициализация Wii Nunchuk проводится с помощью встроенных команд I²C. Команда `bus.write_byte_data(addr=0x52, cmd=0x40, val=0x00)` передает контроллеру команду 0x40 с кодом 0x00.
- 6 Запрос у Wii Nunchuk следующего набора значений с помощью кода 0x00, который равен \0 или просто 0.
- 7 Получение шести байтов данных. (Их описание приведено в табл. 8.5.) Следующие строки программного кода отвечают за декодирование данных.
- 8 Получение состояния кнопки “Z”: 1 — нажата; 0 — отжата. В качестве маски применяется один бит, 0b1. При использовании в операции побитового И (&) это значение маскирует крайний правый бит, а все, что слева от него, заполняется нулями. Таким образом, в результате выполнения операции побитового И с маской 0b1 возвращается крайний правый бит. Подробно операция маскирования была рассмотрена ранее.
- 9 Получение состояния кнопки “C”: 1 — нажата; 0 — отжата. Чтобы получить предпоследний бит, применяется битовый сдвиг вправо (`x >> 1`). Для получения последнего бита выполняется битовое маскирование.

Пилотный проект: управление механическим манипулятором с помощью Wii Nunchuk

Контроллер Wii Nunchuk можно приспособить для решения серьезных задач, а не только для управления игровой приставкой, например, для взаимодействия с роботом. Так как получение данных об ускорении и положении джойстика с этого контроллера не является для нас большой проблемой, полученные данные легко использовать для управления сервоприводами. Немного доработав его механическую часть, вы сможете получить манипулятор типа роботозахвата, управляемый Wii Nunchuk.

Получаемые навыки

В проекте создания манипулятора, управляемого контроллером Wii Nunchuk, вы научитесь следующему:

- получать данные с акселерометра и механического джойстика;
- совместно использовать несколько сервоприводов для управления механическим устройством.

Вы также закрепите навыки по управлению сервоприводами (см. главу 5) и фильтрации случайных значений в результате вычисления скользящего среднего (см. главу 7).

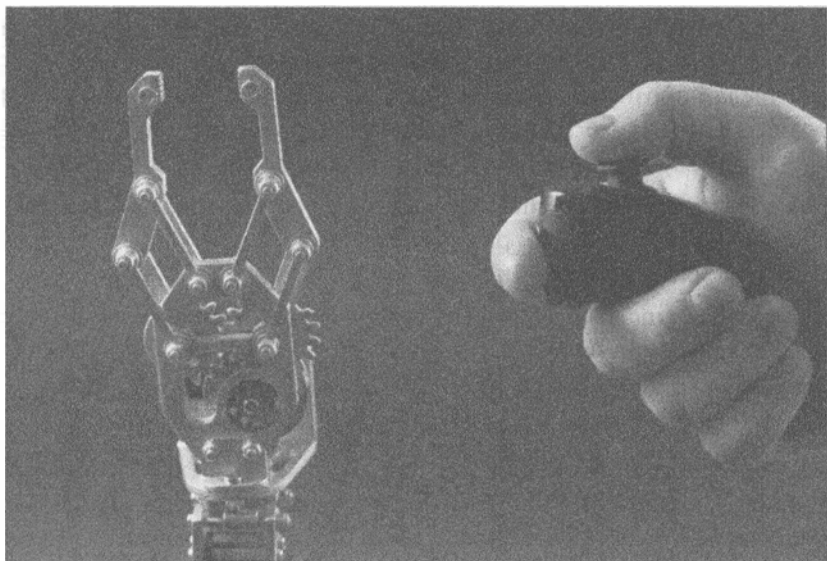


Рис. 8.10. Роботозахват, управляемый Wii Nunchuk

Итак, начнем с подключения сервоприводов (рис. 8.11). Как только вы добьетесь уверенного управления ими, смело приступайте к созданию механической части манипулятора.

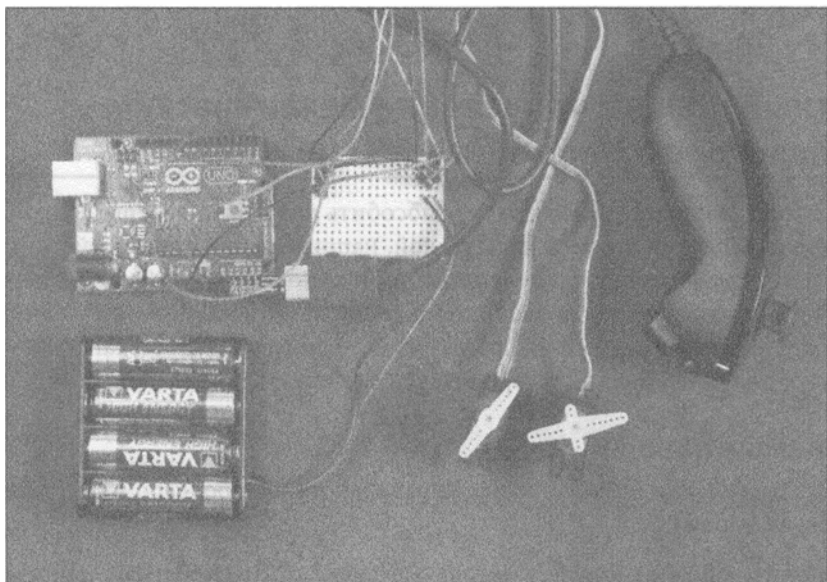


Рис. 8.11. Контроллер Wii Nunchuk управляет с помощью Arduino двумя сервоприводами

На рис. 8.12 показана схема подключения сервоприводов и контроллера к Arduino. Тщательно все соедините, стараясь не допустить ошибок, и выполните код, приведенный в листинге 8.8.

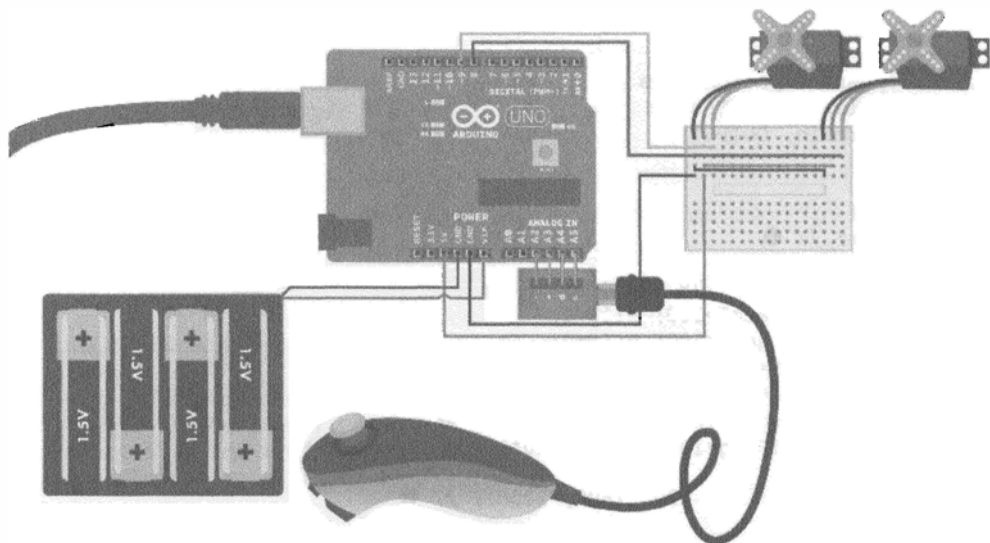


Рис. 8.12. Схема подключения компонентов системы управления манипулятором (см. цветную вклейку)

Детально программный код управления контроллером Wii Nunchuk рассмотрен в разделе “Подключение к Arduino и программа управления контроллером Wii Nunchuk”.

Листинг 8.8. `wiichuck_adapter_claw.ino`

```
// wiichuck_adapter_claw.ino - управление роботозахватом
// контроллером Wii Nunchuk
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>

const int clawPin = 8;
const int armPin = 9;
int armPos=0, clawPos=0;
float wiiP = 0.0; // ❶
float wiiPAvg = 0.0; // ❷
int lastarmPos = 350;

const char i2c_address = 0x52;
int jx = 0, jy = 0, accX = 0, accY = 0, accZ = 0, buttonZ = 0,
    buttonC = 0;

void setup() {
    Serial.begin(115200);

    // Wii Nunchuck
    Wire.begin();
    pinMode(A2, OUTPUT);
    pinMode(A3, OUTPUT);
    digitalWrite(A2, LOW);
    digitalWrite(A3, HIGH);
```

```

    delay(100);
    initNunchuck();

    // Настройка сервоприводов
    pinMode(clawPin, OUTPUT);
    pinMode(armPin, OUTPUT);
}

void loop() {
    get_data(); // ❶

    wiiP = (accZ-70.0)/(178.0-70.0); // ❷
    if (accY>120 && accZ>100) wiiP=1;
    if (accY>120 && accZ<100) wiiP=0;
    if (wiiP>1) wiiP=1;
    if (wiiP<0) wiiP=0;
    wiiPAvg = runningAvg(wiiP, wiiPAvg); // ❸
    armPos = map(wiiPAvg*10*1000, 0, 10*1000, 2200, 350);

    clawPos = map(jy, 30, 220, 1600, 2250); // ❹

    pulseServo(armPin, armPos); // ❺
    pulseServo(clawPin, clawPos);

    printDebug();
}

float runningAvg(float current, float old) {
    float newWeight=0.3;
    return newWeight*current + (1-newWeight)*old; // ❻
}
// Управление сервоприводами
void pulseServo(int servoPin, int pulseLenUs) // ❼
{
    digitalWrite(servoPin, HIGH);
    delayMicroseconds(pulseLenUs);
    digitalWrite(servoPin, LOW);
    delay(15);
}

/// I²C
void get_data() {
    int buffer[6];
    Wire.requestFrom(i2c_address,6);
    int i = 0;
    while(Wire.available()) {
        buffer[i] = Wire.read();
        buffer[i] ^= 0x17;
        buffer[i] += 0x17;
        i++;
    }
    if(i != 6) {
        Serial.println("Ошибка получения данных через i2c");
    }
    write_i2c_zero();

    buttonZ = buffer[5] & 0x01;
    buttonC = (buffer[5] >> 1) & 0x01;
}

```

```

    jx = buffer[0];
    jy = buffer[1];
    accX = buffer[2];
    accY = buffer[3];
    accZ = buffer[4];
}

void write_i2c_zero() {
    Wire.beginTransaction(i2c_address);
    Wire.write((byte)0x00);
    Wire.endTransmission();
}

void initNunchuck()
{
    Wire.beginTransaction(i2c_address);
    Wire.write((byte)0x40);
    Wire.write((byte)0x00);
    Wire.endTransmission();
}

// Отладка
void printDebug()
{
    Serial.print("Ускорение Z:");
    Serial.print(accZ);
    Serial.print(" Наклон джойстика:");
    Serial.print(wiiP);
    Serial.print(" Средний наклон джойстика:");
    Serial.print(wiiPAvg);
    Serial.print(" Положение:");
    Serial.print(jy);
    Serial.print(" Положение захвата:");
    Serial.println(clawPos);
}

```

- ❶ Переменная `wiiP` указывает наклон рычага Wii Nunchuk, представленный в процентах. Полный наклон назад соответствует значению 0,0, а наклон вперед — значению 1,0.
- ❷ Скользящее среднее для `wiiP`.
- ❸ Получение данных от Wii Nunchuk через шину I²C требует задержки в 20 мс между запросами. Эта задержка реализована при двойном вызове функции `pulseServo()` далее в конструкции `loop()`.
- ❹ Вычисление процентных значений для необработанных данных, полученных от Wii Nunchuk. Применяемая формула подобна используемой во встроенной функции `map()`.
- ❺ Для исключения случайных показаний вычисляется среднее двух последних выборок для значения ускорения вдоль оси Z.
- ❻ Получение необработанного значения положения джойстика (в диапазоне от 30 до 220) и преобразование его в управляющие сервоприводом импульсы соответствующей длительности (в диапазоне от 1500 до 2400). Например,

необработанное значение положения джойстика 30 соответствует управляющему импульсу сервопривода длительностью 1500 мкс.

- ⑦ Отправка на сервопривод импульса управления манипулятором. Чтобы сервопривод мог удерживать положение после поворота вала, необходимо подавать на его вход непрерывный импульсный сигнал указанной длительности.
- ⑧ Чтобы получить скользящее среднее для множества значений, применяется взвешенное среднее. При таком подходе требуется хранить в памяти только одно, предыдущее значение, хотя остальные ранние значения несомненно влияют на получаемое среднее.
- ⑨ Отправка на сервопривод одного импульса. Для четкого управления сервоприводом необходимо добиться вызова функции 50 раз в секунду. Детально о тонкостях управления сервоприводами см. в главе 5.

Устройство механического манипулятора

Теперь вы знаете, как с помощью Wii Nunchuk управлять сервоприводами. Полученные навыки при должном старании можно применить для производства коммерческих манипуляторов и роботозахватов. К счастью, на рынке вы найдете большое количество всевозможных механических манипуляторов, а имея за плечами инженерное образование, вы сможете смастерить собственную механическую руку. Мы приобрели свой манипулятор в одном из интернет-магазинов (<http://dx.com>).

Какой бы манипулятор вы ни использовали, всегда неплохо закрепить его на массивной подставке в вертикальном положении, чтобы предотвратить возможное опрокидывание набора. Мы закрепили наш манипулятор на прочном деревянном брусе при помощи винтов (рис. 8.13).

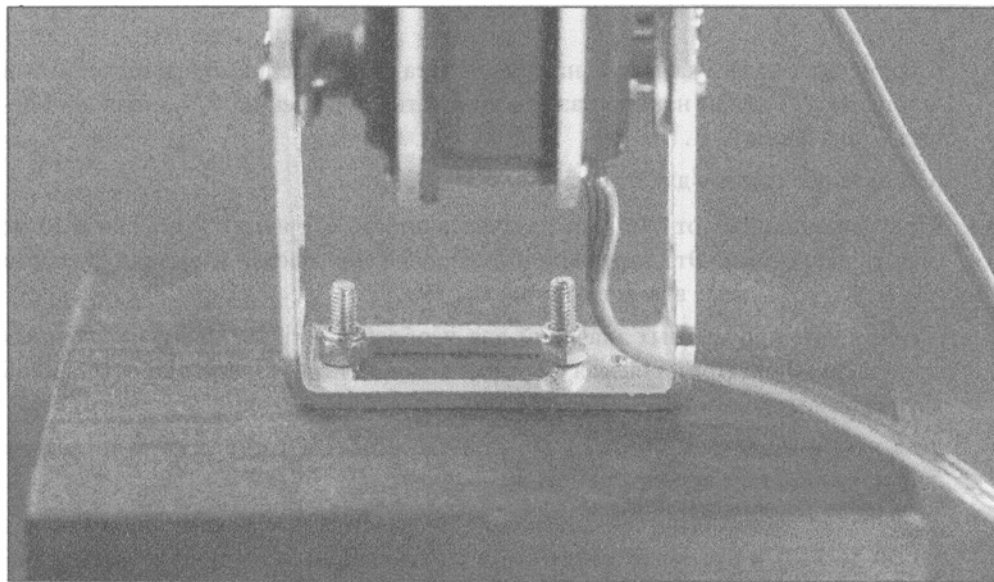


Рис. 8.13. Манипулятор надежно прикреплен нижней частью к прочному основанию

Программный код управления устройством у нас уже отлажен (см. листинг 8.8). Присоедините сервоприводы к манипулятору так, чтобы управлять зажимом с помощью джойстика контроллера Wii Nunchuk, а движением подъемной лапы — с помощью акселерометра. На рис. 8.14 показан конечный результат.

Теперь можете приступить к определению требуемых ускорения и угловой скорости. Вы знаете, как расположен ваш манипулятор и в какую сторону поворачивается, поэтому вам не составит большого труда правильно задать необходимые величины. В зависимости от типа манипулятора, вы сможете обойтись только стандартными настройками или, воспользовавшись блоком инерциальных измерений, получить большее количество рабочих характеристик.

Снабжая свои устройства всевозможными датчиками, вы можете использовать примеры программ, рассмотренные в настоящей книге, как справочное пособие. Но лучше продолжить самообразование и поближе познакомиться с побитовыми операциями, чтобы в полной мере овладеть навыками низкоуровневого управления оборудованием. Полученные знания вы сможете применить для управления другими, сложными датчиками, программы управления которыми найти очень сложно.

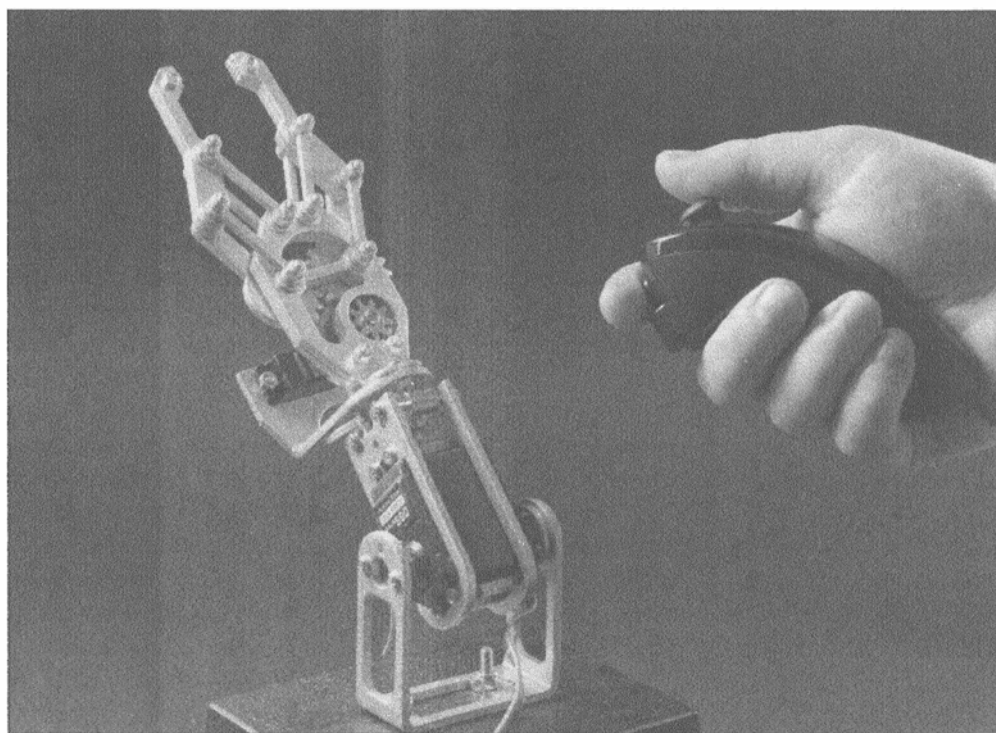


Рис. 8.14. Манипулятор готов к работе

Освоив механическую руку, мы перейдем к еще одному увлекательнейшему занятию — идентификации людей и устройств. В следующей главе вы познакомитесь с принципами считывания отпечатков пальцев и радиочастотной идентификации.

Идентификация



Охранным устройствам для поддержания высокого уровня защищенности данных всегда важно знать, с кем или с чем они имеют дело. Для опознавания людей и предметов применяются специальные датчики идентификации. В этой главе мы поговорим о способах радиочастотной идентификации объектов и распознавания людей по отпечаткам пальцев и порлям.

Метод радиочастотной идентификации (Radio Frequency Identification — RFID) в свое время считался невероятно прогрессивной технологией опознавания объектов. Постепенно он стал неотъемлемой частью нашей жизни. Радиочастотная идентификация была призвана заменить штрих-коды в оптовой торговле, хотя последние все еще используются в розничных торговых сетях. В отличие от штрихового кодирования, радиочастотная идентификация производится на расстоянии и позволяет обмениваться большим количеством данных. Например, штрих-код обычно содержит информацию о таких параметрах товара: название товара, единица измерения, производитель, страна. Радиочастотная идентификация поддерживает гораздо больше полей данных, применяемых для детального описания товара. Она, например, позволяет указать характеристики материалов, из которых производится товар, и даже сорт картона, из которого изготовлена упаковка.

В настоящее время все большее распространение приобретает биометрическая идентификация. Большая часть современных ноутбуков оснащается дактилоскопическим считывателем, блокирующим сторонним пользователям доступ к данным и заменяющим аутентификацию с помощью пароля. Во многих странах уже давно реализуются программы по созданию баз данных отпечатков пальцев всех граждан, которые включаются в биометрические паспорта. В цифровых паспортах нового типа кроме отпечатков пальцев содержится изображение радужной оболочки глаза его владельца, хотя она нигде и не используется. Справедливости ради стоит заметить, что ДНК-маркеры, также представленные в цифровом паспорте, применяются при идентификации преступников.

К преимуществам биометрической идентификации относится ее постоянная доступность, — все необходимое у вас всегда при себе. Основной же недостаток заключается в невозможности изменения данных при случайном включении в паспорт

неверных сведений или умышленном изменении их киберпреступниками. Например, вы не сможете самостоятельно изменить отпечатки пальцев, уже единожды добавленные в паспорт. На самом деле ценность таких паспортов несколько преувеличена, поскольку многие биометрические датчики легко ввести в заблуждение. Подтверждение тому криминальная статистика, которая утверждает, что даже проверка отпечатков пальцев профессионалами вручную не столь надежна, как исходно предполагалось.

На сегодня самым привычным способом проверки личности является введение регистрационных данных с помощью клавиатуры. Попробуйте себе только представить, сколько раз в неделю вы вводите всевозможные пароли для получения доступа к различным ресурсам и устройствам.

Цифровая клавиатура

Цифровая клавиатура очень удобна при вводе числовых значений. Если у вашего телефона есть кнопки, то вы знаете, что цифровая клавиатура очень удобна для введения номеров (и текста!) одной рукой, а при должной сноровке совсем не обязательно контролировать введенную информацию на экране телефона. Цифровыми клавиатурами оснащаются также пульты управления телевизорами и микроволновыми печами. Очень часто цифровая клавиатура применяется для ввода паролей. Вы точно встретите ее в банковских терминалах, дверных замках, оснащенных домофонами, и в охранных сигнализациях.

В нашем проекте применяется недорогая цифровая клавиатура, приобретенная в местном интернет-магазине (рис. 9.1). Принципы подключения клавиатуры к микроконтроллерной плате одинаковые для моделей, предлагаемых большинством производителей.

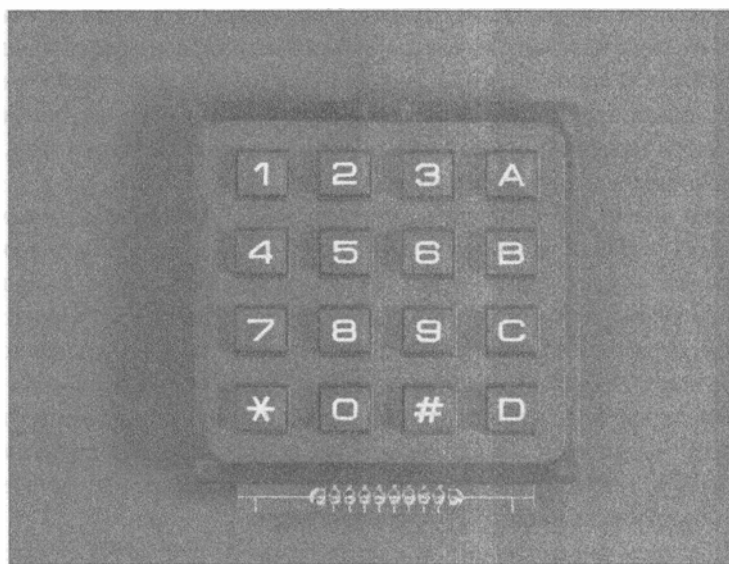


Рис. 9.1. Шестнадцатиклавишная цифровая клавиатура

К плате Raspberry Pi можно подключить обычную клавиатуру, воспользовавшись разъемом USB. Существуют также отдельные цифровые клавиатуры, подключаемые через USB-порт.

Под каждой клавишей на клавиатуре проходят две пересекающиеся под прямым углом (вертикальная и горизонтальная) проводящие дорожки. При нажатии клавиши контакты в точке пересечения замыкаются, и клавиша считается нажатой.

Исходно на все горизонтальные дорожки подается сигнал высокого уровня. Устанавливая по очереди на каждой из вертикальных дорожек сигнал низкого уровня, мы будем опрашивать состояние горизонтальных дорожек. Если ни одна клавиша не нажата, то напряжение на всех горизонтальных линиях будет оставаться на высоком уровне.

Если какая-либо клавиша нажата, то соответствующая вертикальная и горизонтальная дорожки окажутся замкнутыми. Как только на указанной вертикальной дорожке будет установлен сигнал низкого уровня, на сопряженной с ней горизонтальной дорожке также установится сигнал низкого уровня. После нахождения первой нажатой клавиши программа должна возвращаться к обработке сигнала первой вертикальной дорожки, чтобы отследить клавишу, нажимаемую далее.

Подключение к Arduino и программа управления цифровой клавиатурой

На рис. 9.2 показана схема подключения цифровой клавиатуры к Arduino. После соединения всех выводов с помощью макетной платы и проволочных перемычек выполните программный код из листинга 9.1.

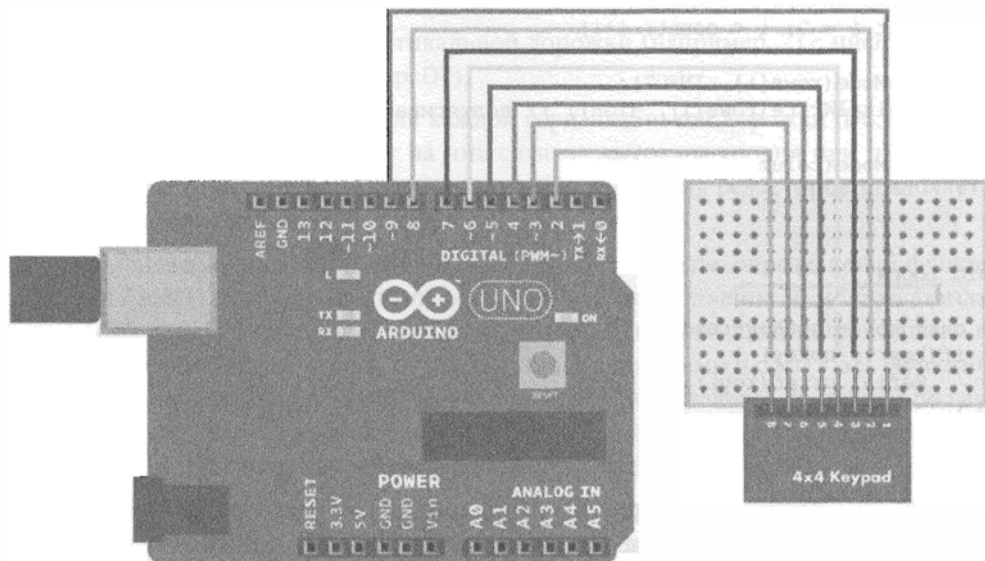


Рис. 9.2. Схема подключения цифровой клавиатуры к Arduino (см. цветную вклейку)

Плата Arduino оснащается встроенными подтягивающими резисторами. Если установить цифровой вывод в режим получения входного сигнала, то при выполнении функции `digitalWrite(pin, HIGH)` напряжение +5 В на вывод подается через сопротивление 20 кОм.

Листинг 9.1. keypad.ino

```
// keypad.ino - распознавание нажатий клавиш 16-клавишной
// цифровой клавиатуры
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int count = 4; // ❶
char keymap[count][count] = { // ❷
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};
const char noKey = 'n';
byte columns[count] = {9, 8, 7, 6}; // ❸
byte rows[count] = {5, 4, 3, 2};
unsigned int lastReadTime;
unsigned int bounceTime = 30; // мс

void setup()
{
    Serial.begin(115200);
    lastReadTime = millis();

    for(int i = 0; i < count; i++)
    {
        pinMode(rows[i], INPUT);
        digitalWrite(rows[i], HIGH); // подтягивающий резистор // ❹

        pinMode(columns[i], INPUT);
    }
}

void loop()
{
    char key = getKeyPress();
    if(key != noKey) {
        Serial.print(key);
    }
    delay(100);
}

// Многократные нажатия клавиши не поддерживаются;
// обрабатывается только первое нажатие
char getKeyPress()
{
    char foundKey = noKey;
```

```

if((millis() - lastReadTime) > bounceTime) { // ❶
    // Подача сигнала на вертикальные дорожки и опрос
    // горизонтальных дорожек
    for(int c = 0; c < count; c++) {
        // Подача сигнала
        pinMode(columns[c], OUTPUT); // ❷
        digitalWrite(columns[c], LOW);
        // Опрос горизонтальных дорожек
        for(int r = 0; r < count; r++) {
            if(digitalRead(rows[r]) == LOW) { // ❸
                // Определение нажатой клавиши
                foundKey = keypad[r][c]; // ❹
            }
        }
        digitalWrite(columns[c], HIGH);
        pinMode(columns[c], INPUT); // ❺
        if(foundKey != noKey)
        {
            break;
        }
    }
    lastReadTime = millis();
}
return foundKey;
}

```

- ❶ Счетчик дорожек (клавиш), расположенных вертикально и горизонтально. В нашем случае клавиатура содержит 4×4 клавиши.
- ❷ Преобразование положения клавиши (например, столбец 0, строка 1) в номер клавиши (например, 4). Вся раскладка клавиатуры представляется двухмерным массивом.
- ❸ Согласование номера вертикальной дорожки (например, 2) с цифровыми выводами Arduino (например, D7).
- ❹ “Подтягивание” вывода для каждой горизонтальной дорожки (D6, D7, D8, D9), чтобы обеспечить подачу на них сигнала высокого уровня при разомкнутом контакте. Применение встроенных подтягивающих резисторов позволяет отказаться от подключения внешних резисторов.
- ❺ Вычисление разницы между временем работы платы, определяемой функцией `millis()`, и временем последнего считывания, большим 30 мс. Это стандартный прием в программировании, применяемый для оценки прошедшего времени. В данном случае в основном цикле программы `loop()` эта функция не вызывается чаще одного раза в 100 мс, что определяется функцией `delay()`. Такая проверка востребована функцией `getKeyPress()` и пригодится вам при импортировании текущего программного кода в другие проекты.
- ❻ Подача на текущую вертикальную дорожку сигнала низкого уровня. На остальные вертикальные дорожки подается сигнал высокого уровня.
- ❼ Если на горизонтальной дорожке наблюдается сигнал низкого уровня...

- 8 ...то нажатая клавиша находится на пересечении с вертикальной дорожкой, на которую подается сигнал низкого уровня.
- 9 В нашем опросе при переводе выводов вертикальных дорожек в режим OUTPUT сигнал высокого уровня накладывается на выходной сигнал. Поэтому выводы вертикальных дорожек переводятся в состояние INPUT или высокого импеданса (выкл.), чтобы обеспечить получение корректных данных.

Подключение к Raspberry Pi и программа управления цифровой клавиатурой

На рис. 9.3 показана схема подключения цифровой клавиатуры к Raspberry Pi. После ее реализации выполните программный код из листинга 9.2.

Попробуйте подключить полноценную или отдельную цифровую клавиатуру через разъем USB, которым оснащена плата Raspberry Pi. Как и любая другая клавиатура, цифровая клавиатура, подсоединяемая к USB-порту, подключается автоматически после включения. Нажатия клавиш на ней регистрируются функцией `raw_input()` или средствами библиотеки `pyGame`, как это делается в любом стандартном ноутбуке или настольном компьютере.

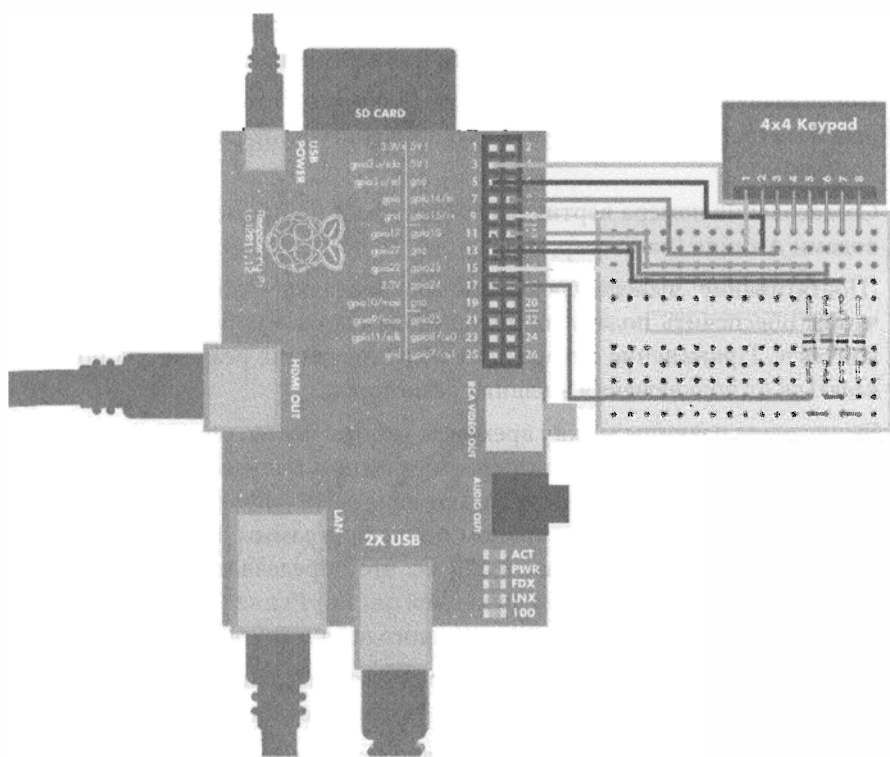


Рис. 9.3. Схема подключения цифровой клавиатуры к Raspberry Pi (см. цветную вклейку)

Листинг 9.2. keypad.py

```
# keypad.py - идентификация клавиш, нажатых на 16-клавишной
# клавиатуре
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio # ❶

keymap = []
keymap.append(['1', '2', '3', 'A']) # ❷
keymap.append(['4', '5', '6', 'B'])
keymap.append(['7', '8', '9', 'C'])
keymap.append(['*', '0', '#', 'D'])

columns = [2, 3, 14, 15] # ❸
rows = [18, 17, 27, 22]
lastReadTime = None
bounceTime = 0.03 # c

def initializeKeyPad():
    for x in range(len(rows)):
        gpio.mode(rows[x], 'in')
        gpio.mode(columns[x], 'in')

def getKeyPress():
    global lastReadTime
    foundKey = None
    if (time.time() - lastReadTime) > bounceTime: # ❶
        # Подача сигнала и опрос дорожек
        for c in range(len(columns)):
            gpio.mode(columns[c], 'out')
            gpio.write(columns[c], gpio.LOW) # ❷

            for r in range(len(rows)):
                if gpio.read(rows[r]) == gpio.LOW: # ❸
                    foundKey = keymap[r][c] # ❹

            gpio.write(columns[c], gpio.HIGH)
            gpio.mode(columns[c], 'in') # ❺
            if not foundKey == None:
                break # ❶
        lastReadTime = time.time()
    return foundKey

def main():
    global lastReadTime
    initializeKeyPad()
    lastReadTime = time.time()
    while True:
        key = getKeyPress()
        if not key == None:
            print(key)
            time.sleep(0.1) # c

if __name__ == "__main__":
    main()
```

- ❶ Убедитесь, что копия файла библиотеки `botbook_gpio.py` располагается в том же каталоге, что и файл текущей программы. Указанная библиотека, а также все файлы проектов настоящей книги доступны на сайте, указанном во введении. Детально о настройке портов GPIO в Raspberry Pi см. в главе 1.
- ❷ Сопоставление физической позиции клавиши (например, строка 1, столбец 0) с ее числом (например, 4).
- ❸ Сопоставление вертикальной дорожки (например, 2) с выводом GPIO (например, `gpio14`).
- ❹ Проверка частоты опроса клавиатуры (не чаще 30 мс). Подобная проверка пригодится вам при использовании функции `getKeyPress()` в других программах. В данном примере функция `main()` вызывает функцию `getKeyPress()` настолько редко, что указанная проверка необязательна. Функция `time.time()` возвращает интервал в секундах, прошедший с начала отсчета времени в Unix, например 1376839738.068395. Это значение сравнивается со временем получения сигнала от последней опрошенной клавиши (также выражается в секундах и определяется от начала отсчета времени в Unix). За начало отсчета в Unix принимается временная точка 00:00:00, 1 января 1970 года, согласно всемирному координированному времени (UTC).
- ❺ Подайте на одну из вертикальных дорожек сигнал низкого уровня. На остальных оставьте сигнал высокого уровня.
- ❻ Как только на одной из горизонтальных дорожек образуется сигнал низкого уровня, с помощью подтягивающих резисторов на остальные горизонтальные дорожки подается сигнал высокого уровня (рис. 9.3).
- ❼ Нажатая клавиша определяется по пересечению горизонтальной и вертикальной дорожек, на которых поддерживается сигнал низкого уровня.
- ❽ Перевод дорожки обратно в режим `INPUT`, чтобы обеспечить ей состояние высокого импеданса, предотвращающее искажение измерений.
- ❾ Выход из цикла `for` (нажатая клавиша определена).

Эксперимент в окружающей среде: снимаем отпечатки пальцев

Разве нельзя взломать пароль, применяемый для подтверждения личности?

Конечно можно! Самый простой способ — это перебрать все возможные комбинации, но хватит ли у вас времени и сил? Как несложно подсчитать, количество трехсимвольных комбинаций, вводимых с 16-клавишной клавиатуры, составляет несколько тысяч:

$$16 \times 16 \times 16 = 16^3 = 4096$$

Но всегда можно определить символ(ы), используемый(е) на клавиатуре чаще других. Если клавиатура используется регулярно, то наиболее часто нажимаемые клавиши сильно загрязнены или заметно потертые.

Скорее всего, вам достанется новая, только что из магазина, цифровая клавиатура. Протестируйте ее, многократно введя один и тот же код. При детальном рассмотрении на часто нажимаемой клавише вы увидите отблески, возникающие при попадании яркого света на жирные пятна, оставленные пальцами (рис. 9.4).

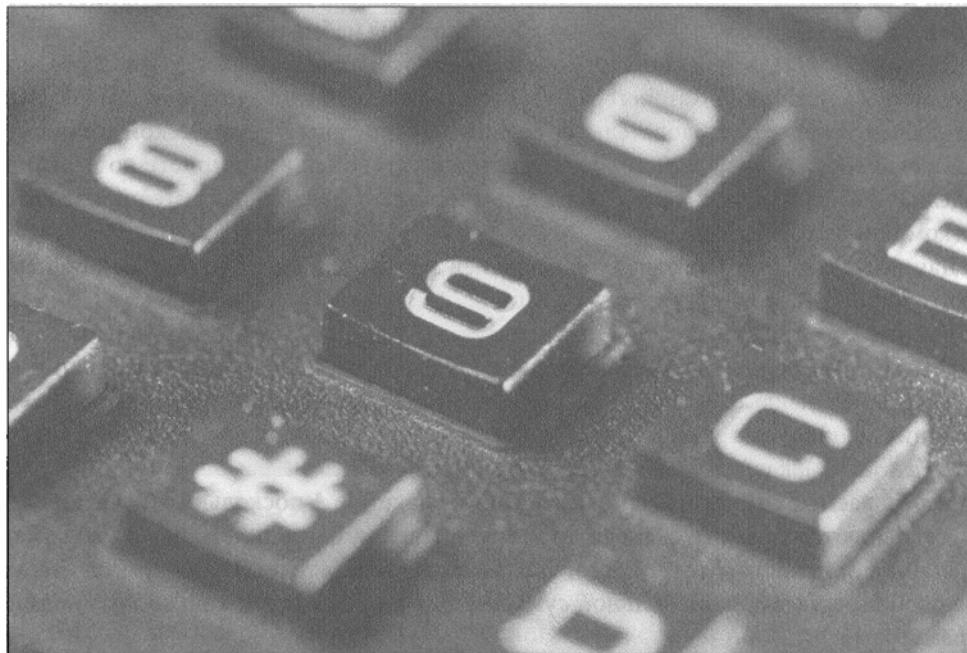


Рис. 9.4. При хорошем освещении на клавишах становятся заметны жирные разводы

Что если взять немного графитного порошка из простого карандаша или другого твердого красителя и нанести на клавиши? Конечно, краситель прилипнет только к “жирным” клавишам. Может, стоит в качестве предосторожности перед каждым вводом пароля хорошенько вымывать руки? А кто вообще придумал, что жирные разводы появляются от пальцев?

Скрытые отпечатки пальцев проявятся, если нанести на содержащую их поверхность цианоакрилатную клейкую пасту. Хотя она позволяет снимать отпечатки пальцев с таких поверхностей, как стекло и пластик, испарения цианоакрила вредны для здоровья, а потому паста требует от работающего с ней специалиста особой подготовки. К тому же клей, входящий в состав пасты, очень липкий и норовит прилепиться к чему угодно.

Вы уже нашли три наиболее часто нажимаемые клавиши? Если да, то вам осталось определить только порядок их нажатия. Количество возможных вариантов сильно уменьшается:

$$3 \times 3 \times 3 = 3^3 = 27$$

Дактилоскопический сканер GT-511C3

Дактилоскопический сканер — это более технологичное устройство идентификации пользователя, чем клавиатура. Он применяется для автоматического распознавания и не требует от вас запоминания учетных данных. Вам также не придется беспокоиться, что кто-то запомнит ваш пароль, пока вы набираете его на клавиатуре.

Датчик GT-511C3 представляет собой дактилоскопический сканер (рис. 9.5), подключаемый к Arduino и Raspberry Pi через последовательный интерфейс. Параметры подключения через последовательное соединение описаны в технической документации к сканеру, которую легко найти, если провести поиск в Интернете по названию модели, GT-511C3, или зайти в раздел технической поддержки на сайте производителя (компания SparkFun).

Данные о сканируемых отпечатках пальцев датчик сохраняет во встроенной памяти, что упрощает его управление из Arduino.

Данные со сканера GT-511C3 передаются на скорости 9600 бит/с через последовательное соединение. Передача данных со сканера замедляется при ожидании отклика принимающего устройства, поэтому при настройке последовательного соединения необходимо установить дополнительную задержку (с помощью функции `setTimeout()`), как показано в следующем примере.

Передаваемые со сканера данные состоят из пакетов. А поскольку отпечатки пальцев сохраняются в памяти сканера, то мы будем оперировать только пакетами команд (табл. 9.1). Заголовок и устройство, указываемые в пакете, всегда одни и те же, как и способ вычисления контрольной суммы. Отличаются пакеты только командами и параметрами.

Таблица 9.1. Пакет данных, передаваемых с дактилоскопического сканера GT-511C3

Название	Пример значений	Описание
Заголовок	0x55 0xAA	Одинаковый для всех пакетов
Идентификатор устройства	1	Одинаковый для одного и того же устройства
Параметр	0	Выключение
Команда	0x12	Управление светодиодом (CMD_LED)
Контрольная сумма		Сумма байтов

После установки соединения микроконтроллер (ведущее устройство) посылает пакет команд в сканер (подчиненное устройство). Среди доступных команд следующие:

- включение светодиода;
- удаление всех отпечатков пальцев из памяти сканера;
- сканирование и сохранение нового отпечатка пальца в памяти;
- сканирование и сравнение отпечатка пальца с отпечатками, сохраненными в памяти ранее.

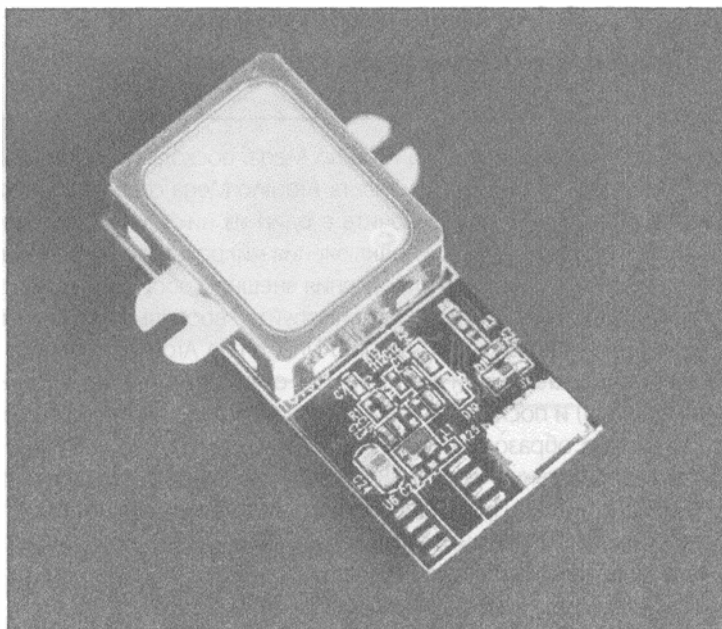


Рис. 9.5. Дактилоскопический сканер GT-511C3

Сканер отвечает на запрос пакетом команд. В ответе содержится идентификатор подтверждения приема — ACK (acknowledge) или идентификатор ошибки — NACK (negative acknowledge). Ответный пакет данных также может содержать следующие данные:

- идентификатор распознанного отпечатка пальца;
- идентификатор отсутствия отпечатка пальца в базе данных;
- код ошибки.

Чаще всего встречаются коды ошибок 100F (соединение уже открыто) и 100A (память пуста).

Дактилоскопические сканеры легко обмануть, используя фальшивые отпечатки пальцев. Пищевой желатин, подобный применяемому при производстве мармелада, пластилин, воск и даже клейкая лента — это те материалы, которые позволяют легко снять отпечатки пальцев с ладони. Впоследствии эти фальшивые отпечатки можно прикладывать к дактилоскопическому сканеру вместо пальца. Выбор используемого материала зависит от способа считывания отпечатка пальцев датчиком.

На примере программного кода, приведенного в следующем примере, вы ознакомитесь с базовыми принципами управления дактилоскопическим сканером. Если хотите больше узнать о тонкостях программного управления датчиком GT-511C3, то обратитесь к технической документации, доступной на сайте производителя.

Подключение к Arduino Mega и программа управления дактилоскопическим сканером

В нашем случае применяется плата Arduino Mega, поскольку она оснащается лучшими средствами коммуникации. Плата Arduino Mega содержит несколько USB-разъемов, и вы сможете подключить в один из них дактилоскопический сканер, а второй использовать для подключения микроконтроллера к компьютеру, чтобы загружать программы управления внешним оборудованием.

В качестве дешевого варианта попробуйте воспользоваться Arduino Leonardo. Несмотря на схожесть с Arduino Uno, плата Arduino Leonardo оснащена двумя последовательными портами. Но, в отличие от Arduino Uno, выводы **RX/TX** (номер **0** и **1**) и последовательный порт в Arduino Leonardo не связаны между собой. Таким образом, в Arduino Leonardo переменная `Serial` представляет последовательный порт USB, а переменная `Serial1` соответствует выводам **0 (RX)** и **1 (TX)**. Применяя Arduino Leonardo вместо Arduino Mega, вам придется несколько видоизменить приведенный ниже программный код и схему подключения дактилоскопического сканера, чтобы воспользоваться выводами **0** и **1 (Serial1)** вместо выводов **15** и **14 (Serial3)**.

На рис. 9.6 показана схема подключения дактилоскопического сканера к Arduino. После подключения оборудования выполните программный код, показанный в листинге 9.3.

Внимание! Ниже приведен сложный для понимания программный код! Он гораздо сложнее программных кодов, приведенных ранее, — в нем для преобразования структуры (`struct`) в буфер байтов применяются указатели. Вам не обязательно вникать в тонкости структуры программы. Достаточно правильно подключить датчик к плате, загрузить код в микроконтроллер, и можно приступать к тестированию сканера. Если вам все же интересно, как работает программа управления дактилоскопическим датчиком, то попробуйте разобраться с назначением каждой программной конструкции.

В Arduino сигнал высокого уровня представляется напряжением 5 В, а дактилоскопический датчик рассчитан на напряжение 3,3 В. Повышенное рабочее напряжение приводит к повреждению датчика, сокращает срок его службы и вызывает частые ошибки сканирования. В текущем проекте для понижения напряжения на входе сканера применяется делитель напряжения, подключенный к выводу платы Arduino. Он чрезвычайно прост, поскольку состоит из двух обычных резисторов.

Вам не следует подключать делитель напряжения к выходу датчика, поскольку он рассчитан на максимальное напряжение 3,3 В — меньше, чем то, на которое рассчитан цифровой вход Arduino. Сигнал с напряжением 3,3 В будет прекрасно распознаваться Arduino, поскольку он составляет больше половины от 5 В.

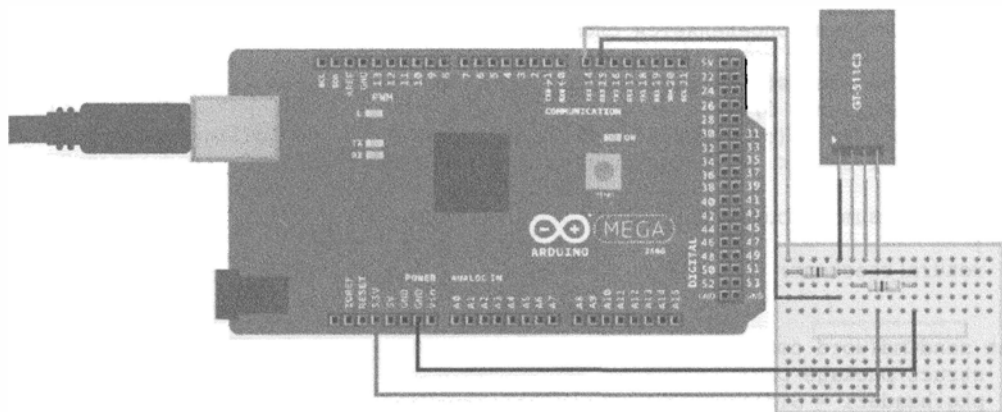


Рис. 9.6. Схема подключения дактилоскопического сканера к Arduino

Листинг 9.3. `fingerprint_scanner.ino`

```
// fingerprint_scanner.ino - получение отпечатков пальцев со
// сканера GT-511C3
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
// Требуется Arduino Mega с большим количеством USB-портов
```

```
const byte STX1 = 0x55; // ❶
const byte STX2 = 0xAA;

const word CMD_OPEN = 0x01; // ❷
const word CMD_CLOSE = 0x02;
const word CMD_LED = 0x12;
const word CMD_GET_ENROLL_COUNT = 0x20;
const word CMD_ENROLL_START = 0x22;
const word CMD_ENROLL_1 = 0x23;
const word CMD_ENROLL_2 = 0x24;
const word CMD_ENROLL_3 = 0x25;
const word CMD_IS_FINGER_PRESSED = 0x26;
const word CMD_DELETE_ALL = 0x41;
const word CMD_IDENTIFY = 0x51;
const word CMD_CAPTURE_FINGER = 0x60;
```

```
const word ACK = 0x30; // ❸
const word NACK = 0x31; // Ошибка
```

```
struct package { // ❹
    byte header1;
    byte header2;
    word deviceID;
    unsigned long param;
    word cmd;
    word checksum;
};
```

```
const int SIZE_OF_PACKAGE = 12;
```

```

/*
Контрольная сумма определяется сложением всех байтов
*/
word calcChecksum(struct package *pkg) { // 5
    word checksum = 0;
    byte *buffer = (byte*)pkg; // 6
    for(int i=0; i < (sizeof(struct package) - sizeof(word)); i++)
    {
        checksum += buffer[i];
    }
    return checksum;
}

int sendCmd(word cmd, int param) { // 7
    struct package pkg;
    pkg.header1 = STX1;
    pkg.header2 = STX2;
    pkg.deviceID = 1; // 8
    pkg.param = param;
    pkg.cmd = cmd;
    pkg.checksum = calcChecksum(&pkg);
    //Serial.println("Отправка команды");
    byte *buffer = (byte*)&pkg; // 9

    int bytesSent = Serial3.write(buffer, sizeof(struct package));

    if(bytesSent != sizeof(struct package)) {
        Serial.println("Ошибка соединения");
        return -1;
    }

    int bytesReceived = 0;
    char recvBuffer[SIZE_OF_PACKAGE]; // 10
    struct package *recvPkg = (struct package*) recvBuffer; // 11

    bytesReceived = Serial3.readBytes(recvBuffer, sizeof(struct
package)); // 12
    if(bytesReceived != SIZE_OF_PACKAGE) {
        Serial.println("Ошибка соединения");
        return -1;
    }

    if( recvPkg->header1 != STX1 || recvPkg->header2 != STX2) { // 13
        Serial.println("Ошибка заголовка!");
        return -1;
    }

    if(recvPkg->checksum != calcChecksum(recvPkg)) {
        Serial.println("Несовпадение контрольной суммы!");
        return -1;
    }
    if(recvPkg->cmd == NACK) {
        Serial.println("Ошибка получения команды!");
        Serial.print("Ошибка: ");
        Serial.println(recvPkg->param, HEX);
        return -1;
    }
}

```

```

    return recvPkg->param;
}

// Дальнейшие настройки определяются используемым интерфейсом
void setup() {
    Serial.begin(115200); // 14
    Serial3.begin(9600); // 15
    Serial3.setTimeout(10*1000); // мс // 16
}

void flashLed(int time) {
    sendCmd(CMD_LED, 1);
    delay(time);
    sendCmd(CMD_LED, 0);
}

void loop() {
    Serial.println("Отправка команды открытия соединения");
    sendCmd(CMD_OPEN, 0);
    // Исходное удаление отпечатков, хранящихся в памяти
    if(sendCmd(CMD_DELETE_ALL, 0) >= 0) {
        // Мигание светодиодом 3 раза в случае готовности
        flashLed(500);
        delay(500);
        flashLed(500);
        delay(500);
        flashLed(500);
    }

    Serial.println("Начало захвата");

    int id = 0;
    id = sendCmd(CMD_GET_ENROLL_COUNT, 0); // 17
    sendCmd(CMD_LED, 1);
    sendCmd(CMD_ENROLL_START, id);
    Serial.println("Приложите палец");
    int ret = 0;
    WaitForFinger(false);
    Serial.println("Сканирование отпечатка");
    ret = sendCmd(CMD_CAPTURE_FINGER, 1); // 18
    if(ret < 0) {
        EnrollFail();
        return;
    }
    Serial.println("Уберите палец");

    sendCmd(CMD_ENROLL_1, 0);
    WaitForFinger(true);
    Serial.println("Приложите палец еще раз");

    WaitForFinger(false);
    ret = sendCmd(CMD_CAPTURE_FINGER, 1);
    if(ret < 0) {
        EnrollFail();
        return;
    }
    Serial.println("Уберите палец");
}

```

```

sendCmd(CMD_ENROLL_2, 0);
WaitForFinger(true);
Serial.println("Приложите палец еще раз");

WaitForFinger(false);
ret = sendCmd(CMD_CAPTURE_FINGER, 1);
if(ret < 0) {
    EnrollFail();
    return;
}
Serial.println("Уберите палец");

ret = sendCmd(CMD_ENROLL_3, 0);
if(ret != 0) {
    EnrollFail();
    return;
}
WaitForFinger(true);
flashLed(500);
delay(500);
flashLed(500);
delay(500);
Serial.println("Сканирование завершено");
Serial.println("Приложите палец для идентификации");
sendCmd(CMD_LED, 1);

// Идентификация
WaitForFinger(false);
ret = sendCmd(CMD_CAPTURE_FINGER, 1); // 19
if(ret < 0) {
    IdentFail();
    return;
}
ret = sendCmd(CMD_IDENTIFY, 0); // 20
if(ret >= 0 && ret < 200) {
    Serial.print("Отпечаток найден: ");
    Serial.println(ret);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
} else {
    Serial.println("Отпечаток не найден");
}
sendCmd(CMD_CLOSE, 0);
delay(100000);
}

```

```

void WaitForFinger(bool bePressed) {
    delay(500);
    if(!bePressed) {
        while(sendCmd(CMD_IS_FINGER_PRESSED, 0) > 0) {
            delay(200);
        }
    } else {
        while(sendCmd(CMD_IS_FINGER_PRESSED, 0) == 0) {
            delay(200);
        }
    }
}

// Мигание светодиода 3 раза при ошибке идентификации
// и прекращение операции
void IdentFail() {
    Serial.println("Идентификация невозможна!");
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    sendCmd(CMD_CLOSE, 0);
}

// Мигание светодиода 4 раза при ошибке сканирования
// и прекращение операции
void EnrollFail() {
    Serial.println("Ошибка сканирования!");
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    delay(500);
    flashLed(500);
    sendCmd(CMD_CLOSE, 0);
}

```

- ❶ Заголовок пакета команды. Поскольку в настоящем коде изображения отпечатков пальцев не передаются, пакеты данных не используются.
- ❷ Команды, определенные в технической документации к сканеру. Префикс 0x обозначает шестнадцатеричное число (детально о шестнадцатеричных числах см. в главе 8).
- ❸ Возможные возвращаемые значения: ACK — подтверждение приема; NACK — сбой.
- ❹ Структура пакета команды. Точные длины переменных определяются далее при использовании структуры для декодирования полученных данных (см. табл. 9.1).
- ❺ Контрольная сумма — это сумма всех байтов.

- ❖ Для обработки всех байтов структуру необходимо преобразовать в буфер `byte`.
- ❖ Команда `sendCmd()` отправляет команду и возвращает параметр, извлеченный из ответа. В случае ошибки возвращается значение `-1`.
- ❖ Идентификатор устройства задается самим устройством. Он запрашивается при получении доступа к устройству, поэтому в структуру включено специальное значение, представляющее его.
- ❖ Преобразование структуры, представляющей пакет команды, в передаваемый набор байтов.
- ❖ Приемный буфер имеет такую же длину, как и структура, представляющая пакет команды.
- ❖ Указатель `*recvPkg` ссылается на приемный буфер как на структуру. Впоследствии вы сможете использовать этот указатель для получения доступа к переменным, например `recvPkg->cmd`.
- ❖ Заполнение приемного буфера байтами данных без дальнейшей обработки.
- ❖ Применяя указатель на структуру, представляющую пакет команды, вы получаете доступ к отдельным значениям структуры. Это невозможно при использовании буфера байтов напрямую.
- ❖ Первый последовательный порт USB. Ознакомиться с передаваемыми через него данными можно с помощью команды `Tools` → `Serial Monitor` (Сервис → Монитор порта) в среде разработки Arduino. При выводе данных на монитор последовательного порта отладка кода заметно упрощается.
- ❖ Дактилоскопический сканер подключается к порту `Serial3`. Он доступен в `ArduinoMega`, но отсутствует в `Arduino Uno`.
- ❖ Работа сканера иногда замедляется, поэтому нужно добавить задержку в 10 секунд.
- ❖ Поиск первой незанятой ячейки.
- ❖ Параметр 1 соответствует медленному сканированию, а 0 — быстрому.
- ❖ Сканирование отпечатка, подлежащего идентификации.
- ❖ Идентификация последнего отсканированного отпечатка пальца. Значение, возвращаемое функцией `sendCmd()`, представляет идентификатор уже имеющегося отпечатка (1, 2 или 3); значение 0 представляет неидентифицированный отпечаток.

Подключение к Raspberry Pi и программа управления дактилоскопическим сканером

Чтобы получить доступ к последовательному порту в Raspberry Pi, вам сначала нужно освободить его, поскольку по умолчанию он занят интерпретатором входа в систему, о чем рассказывается в главе 11. Подключите сканер к Raspberry Pi так, как показано на рис. 9.7, и выполните программный код из листинга 9.4.

Для доступа к последовательному порту в Python требуется установить библиотеку PySerial. Эта задача выполняется такой командой:

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

Физическое подключение сканера к Raspberry Pi реализуется очень просто, значительно проще, чем к Arduino. В Raspberry Pi сигнал высокого уровня соответствует напряжению 3,3 В; такое же напряжение высокого уровня поддерживается и самим сканером. Таким образом, в Raspberry Pi вам не нужно прибегать к использованию делителей напряжения, как это было в случае с Arduino.

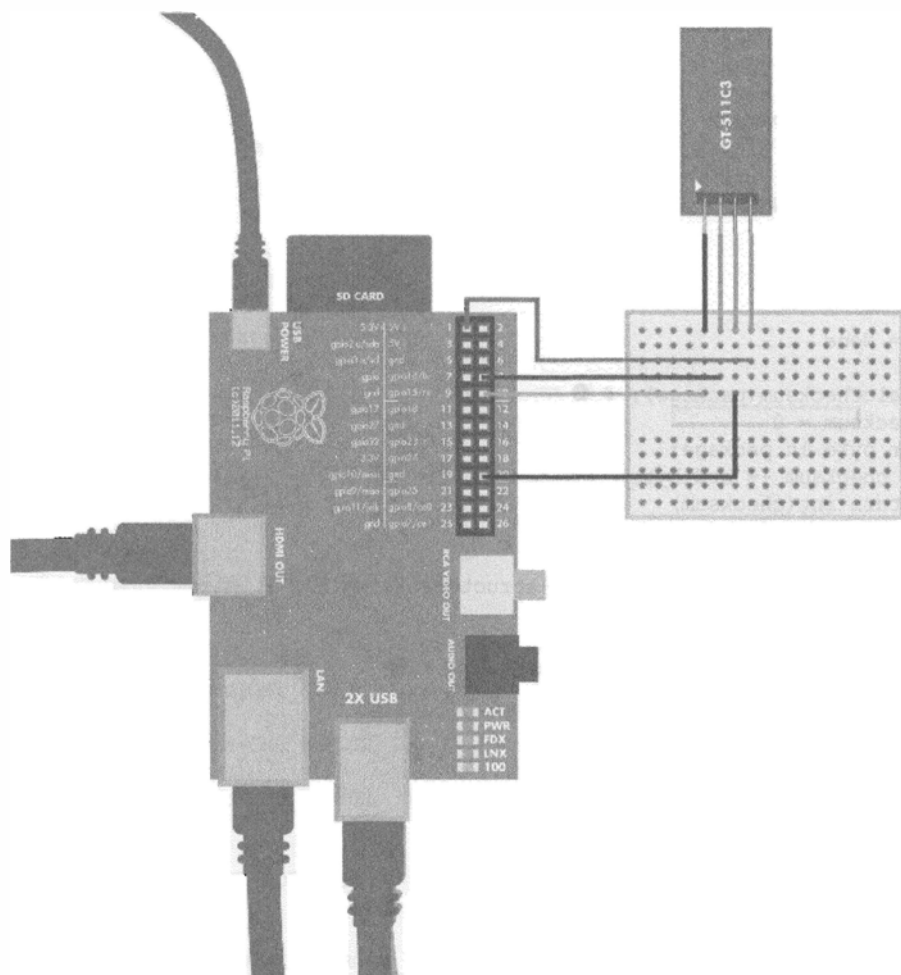


Рис. 9.7. Схема подключения датчикового сканера к Raspberry Pi

Листинг 9.4. `fingerprint_scanner.py`

```
# fingerprint_scanner.py - сканирование и распознавание  
# отпечатков пальцев сканером GT-511C3  
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```

import time
import serial
import struct

STX1 = 0x55 # ❶
STX2 = 0xAA

CMD_OPEN = 0x01 # ❷
CMD_CLOSE = 0x02
CMD_LED = 0x12
CMD_GET_ENROLL_COUNT = 0x20
CMD_ENROLL_START = 0x22
CMD_ENROLL_1 = 0x23
CMD_ENROLL_2 = 0x24
CMD_ENROLL_3 = 0x25
CMD_IS_FINGER_PRESSED = 0x26
CMD_DELETE_ALL = 0x41
CMD_IDENTIFY = 0x51
CMD_CAPTURE_FINGER = 0x60

ACK = 0x30 # ❸
NACK = 0x31

port = None

def calcChecksum(package): # ❹
    checksum = 0
    for byte in package:
        checksum += ord(byte)
    return int(checksum)

def sendCmd(cmd, param = 0): # ❺
    package = chr(STX1)+chr(STX2)+struct.pack('<hih', 1, param,
❻cmd) # ❻
    checksum = calcChecksum(package)
    package += struct.pack('<h',checksum) # ❼

    sent = port.write(package)

    if(sent != len(package)):
        print "Ошибка соединения"
        return -1

    recv = port.read(sent) # ❷
    recvPkg = struct.unpack('cchihh',recv) # ❸

    if recvPkg[4] == NACK:
        print("Ошибка: %s" % recvPkg[3])
        return -2
    time.sleep(1)
    return recvPkg[3]

def startScanner():
    print("Открытие соединения со сканером")
    sendCmd(CMD_OPEN)

```

```

def stopScanner():
    print("Закрытие соединения со сканером")
    sendCmd(CMD_CLOSE)

def led(status = True):
    if status:
        sendCmd(CMD_LED,1)
    else:
        sendCmd(CMD_LED,0)

def enrollFail():
    print("Ошибка сканирования")
    led(False)
    stopScanner()

def identFail():
    print("Ошибка идентификации")
    led(False)
    stopScanner()

def startEnroll(ident):
    sendCmd(CMD_ENROLL_START,ident)

def waitForFinger(state):
    if(state):
        while(sendCmd(CMD_IS_FINGER_PRESSED) == 0):
            time.sleep(0.1)
    else:
        while(sendCmd(CMD_IS_FINGER_PRESSED) > 0):
            time.sleep(0.1)

def captureFinger():
    return sendCmd(CMD_CAPTURE_FINGER)

def enroll(state):
    if state == 1:
        return sendCmd(CMD_ENROLL_1)
    if state == 2:
        return sendCmd(CMD_ENROLL_2)
    if state == 3:
        return sendCmd(CMD_ENROLL_3)

def identifyUser():
    return sendCmd(CMD_IDENTIFY)

def getEnrollCount():
    return sendCmd(CMD_GET_ENROLL_COUNT)

def removeAll():
    return sendCmd(CMD_DELETE_ALL)

def main():
    print("Удаление всех хранящихся отпечатков")
    startScanner()
    removeAll()

```

```

led()
print("Начало сканирования")
newID = getEnrollCount()
print(newID)

startEnroll(newID)
print("Приложите палец")
waitForFinger(False)
if captureFinger() < 0:
    enrollFail()
    return
enroll(1)
print("Уберите палец")
waitForFinger(True)

print("Приложите палец еще раз")
waitForFinger(False)
if captureFinger() < 0:
    enrollFail()
    return
enroll(2)
print("Уберите палец")
waitForFinger(True)

print("Приложите палец еще раз")
waitForFinger(False)
if captureFinger() < 0:
    enrollFail()
    return

if enroll(3) != 0:
    enrollFail()
    return

print("Уберите палец")
waitForFinger(True)

print("Приложите палец для идентификации")
waitForFinger(False)
if captureFinger() < 0: # ⑩
    identFail()
    return
ident = identifyUser()
if(ident >= 0 and ident < 200): # ⑪
    print("Отпечаток найден: %d" % ident)
else:
    print("Отпечаток не найден")
led(False)
stopScanner()

if __name__ == "__main__":
    try:
        if port == None:
            port = serial.Serial("/dev/ttyAMA0", baudrate=9600,
❧timeout=None) # ⑫

```

```

    main()
except Exception, e:
    print e
    port.close()
finally:
    port.close()

```

- ❶ Заголовок пакета команд (см. табл. 9.1).
- ❷ Команды, определенные в технической документации к сканеру. Префикс 0x обозначает шестнадцатеричное число (детально о шестнадцатеричных числах см. в главе 8).
- ❸ Возможные возвращаемые значения: ACK — подтверждение приема; NACK — сбой.
- ❹ В каждый пакет включена контрольная сумма, т.е. сумма всех байтов.
- ❺ Функция `sendCmd()` отправляет пакет команд и получает ответ. Она возвращает параметр, извлеченный из ответа, или отрицательное число в случае ошибки.
- ❻ Преобразование переменных Python в набор передаваемых байтов. Байты заголовка STX1 и STX2 связаны между собой. Значения `param` и `cmd` объединяются в пакет с помощью конструкции `struct.pack`: `h` — короткое двухбайтовое целое число со знаком с прямым порядком следования байтов (<); `i` — обычное четырехбайтовое целочисленное значение.
- ❼ Контрольная сумма добавляется в передаваемый набор данных. Функция `pack()` преобразовывает контрольную сумму в короткое двухбайтовое целое число со знаком с прямым порядком следования байтов (`h`).
- ❽ Получение ответа. Пакеты команд всегда имеют одинаковую длину, поэтому можете указать в переменной `sent` точное количество считываемых байтов.
- ❾ Распаковка необработанных данных в кортеж. Функция `struct.pack()` принимает такие параметры: однобайтовый символ (`c`); короткое целое число (`h`) и целое число (`i`).
- ❿ Выполнение команды `CMD_CAPTURE_FINGER (0x60)`. В случае возникновения ошибки сканер возвращает отрицательное значение. Идентификация в таком случае не выполняется. В случае успеха функция `captureFinger()` отправляет эту команду и возвращает результат ее выполнения.
- ⓫ Функция `identifyUser()` возвращает идентификатор распознанного отпечатка пальца (1, 2 или 3) или отрицательное значение при сканировании нового отпечатка.
- ⓬ Сканер передает данные на скорости 9600 бит/с. В Linux последовательные порты представляются файлами устройств, не выполняющими непосредственную обработку. Ответ со сканера может запаздывать, поэтому в коде нужно установить соответствующую задержку, задаваемую в секундах (в случае применения значения с плавающей точкой значение указывается в виде десятичной дроби).

Модуль радиочастотной идентификации ELB149C5M

Радиочастотная идентификация (Radio Frequency Identification — RFID) представляет собой простой и дешевый способ уникального опознавания объекта на расстоянии. На данный момент он применяется на крупных складах и в хранилищах товаров, а по мере удешевления технологии становится все доступнее для рядовых потребителей. Эту технологию можно встретить в самых разных устройствах, начиная с электронных ошейников для животных и кончая библиотечными каталогами, где она вытесняет ранее распространенное штриховое кодирование книг.

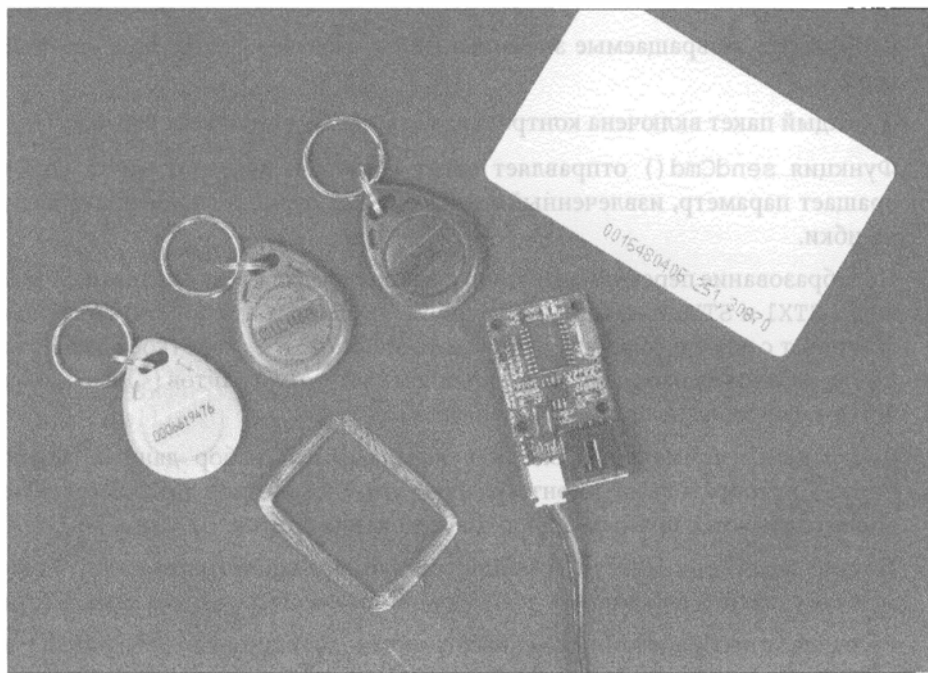


Рис. 9.8. Модуль радиочастотной идентификации ELB149C5M с брелоками

Вам доступно оборудование, основанное на нескольких конкурирующих стандартах RFID, отличающихся ценой, расстоянием действия, уровнем безопасности, объемом памяти и распространенностью. В текущем проекте мы изучим возможности электронного брелока ELB149C5M, который поддерживает считывание меток стандарта UEM4100, работающих на частоте 125 кГц.

После подключения RFID-устройства к плате и подачи на него питания приложите к датчику карточку для ее считывания. Распознавание карточки сопровождается миганием зеленого светодиода на устройстве считывания.

В ELB149C5M поддерживаются два интерфейса передачи данных — последовательное соединение и Wiegand. Поскольку последовательное соединение вам уже знакомо по другим проектам, то и в данном примере мы будем устанавливать соединение с RFID-устройством через него. Детальное описание параметров последовательного соединения приводится к технической документации к датчику ELB149C5M.

На момент написания книги модель ELB149C5M указывалась, как снятая с производства, а потому найти ее в продаже было проблематично. Тем не менее компания Seeedstudio выпустила обновленную модель Grove - 125KHz RFID Reader. Она оснащается штекером, унифицированным для подключения к устройствам, поддерживающим технологию Grove собственной разработки. Вы же сможете использовать его для подключения модуля к Arduino Mega, для чего вам не понадобятся проволочные перемычки: черный провод подключается к выводу **GND**; красный — к выводу +5 В; желтый — к выводу **RX3** (или **15**) платы Arduino Mega. Белый провод остается неподключенным.

Для перевода датчика в режим последовательного соединения установите перемычку **UW** в положение **U** (сокращенно от **UART**). Передача данных через последовательный порт осуществляется по стандарту 9600 N81 TTL: скорость соединения 9600 бит/с; без проверочного бита; 8 бит данных и один стоповый бит. Уровни сигналов в датчике стандартные для логики TTL/UART, поэтому для подключения его к Arduino используются обычные проволочные перемычки.

Устройства, поддерживающие TTL-логику, обычно подключаются к Arduino и Raspberry Pi напрямую. В технологии RS-232, поддерживаемой старыми компьютерами, несколько другие физические стандарты, поэтому устройства, совместимые с RS-232, напрямую к Raspberry Pi и Arduino не подключаются.

В TTL (Transistor-Transistor Logic — транзисторно-транзисторная логика) сигнал низкого уровня соответствует 0; сигнал высокого уровня — 1. Поэтому в TTL-устройствах сигнал низкого уровня представлен напряжением 0 В (земля); сигнал высокого уровня — напряжением +3,3 В или 5 В.

В последовательном соединении, работающем по стандарту RS-232, низкий и высокий уровни определяются напряжениями –25 В и +25 В соответственно. Чтобы подключить оборудование через такое RS-232-соединение к Arduino или Raspberry Pi, вам понадобится специальный преобразователь, такой как MAX 232.

Считывающие устройства RFID начинают обмен данными с отправки пакета данных. Он содержит статический идентификационный номер RFID-метки (табл. 9.2).

Отправка ASCII-символов RFID-считывателем осуществляется достаточно странным способом, в котором каждая пара символов представляется шестнадцатеричным числом. Например, двухбайтовая строка 3E представляется шестнадцатеричным кодом 0x3E (число 62).

Детально о шестнадцатеричной системе счисления см. в главе 8.

После проверки микроконтроллером контрольной суммы пакета номер карточки, поднесенной к считывателю, подтверждается. В настоящем эксперименте программа будет выводить номер карточки на монитор последовательного порта.

Таблица 9.2. Пакет данных считывающего устройства RFID

Назначение	Длина, символы ASCII	Длина, байты	Описание
Начало	1	Не определяется	0x02 — в ASCII начало текста
Номер карточки	10	5	1 байт — производитель, 4 байта — идентификатор
Контрольная сумма	2	1	Побитовое исключающее ИЛИ для байтов идентификатора
Завершение	1	Не определяется	0x03 — в ASCII конец текста

Подключение к Arduino Mega и программа управления модулем радиочастотной идентификации

Для упрощения отладки в этом проекте мы будем работать с платой Arduino Mega. Так как она оснащается несколькими последовательными портами, вы сможете использовать монитор последовательного порта (команда Tools⇒Serial Monitor (Сервис⇒Монитор порта) в среде разработки Arduino) одновременно с подключенным к плате RFID-считывателем. В Arduino Uno имеется только один разъем USB, что делает прототипирование и отладку непростым и трудозатратным занятием. В следующем примере программного кода устанавливаются два последовательных соединения, поэтому для выполнения в Arduino Uno его нужно несколько видоизменить. Подключите считыватель радиочастотного идентификатора к Arduino, как показано на рис. 9.9, а затем выполните программный код из листинга 9.5.

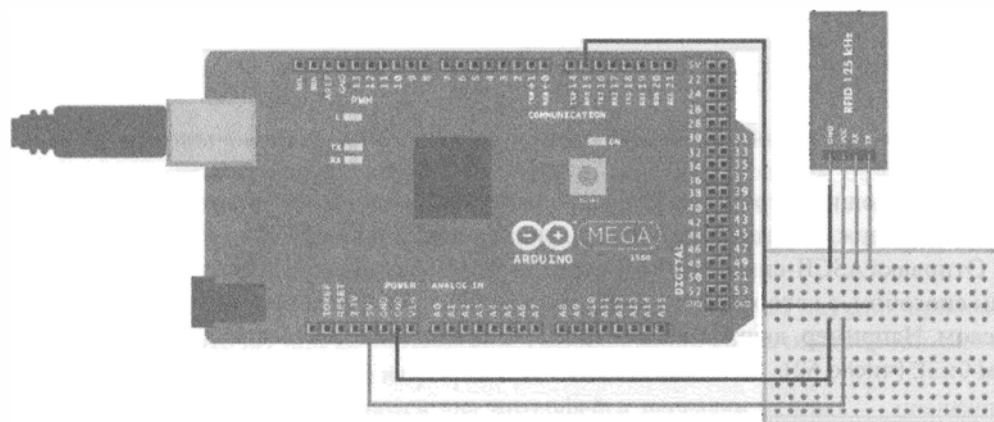


Рис. 9.9. Схема подключения модуля радиочастотной идентификации к Arduino Mega

Листинг 9.5. rfid_reader.ino

```
// rfid_reader.ino - считывание RFID-метки на частоте 125 КГц с
// помощью модуля EMB149C5M
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
// Требуется Arduino Mega с дополнительным последовательным портом

int bytesRead = 0; // ❶
char buffer[13]; // ❷

void setup() {
    Serial.begin(115200); // компьютер
    Serial3.begin(9600); // RFID-считыватель // ❸
}

void loop() {
    char recv;
    if(Serial3.available() > 0) { // ❹
        recv = Serial3.read();
        if(recv == 0x02) { // ❺
            bytesRead = 0;
            Serial.println("Считывание метки");
        } else if(bytesRead == 12 && recv == 0x03) { // ❻
            Serial.println();
            String data = buffer;
            byte checksum = 0;
            byte chk = toLong(data.substring(10, 12));
            long id = toLong(data.substring(4, 10));
            for(int i = 0; i < 10; i=i+2) {
                checksum ^= toLong(data.substring(i, i+2)); // ❼
            }
            Serial.print(id); // ❽
            if(checksum == chk) { // ❾
                Serial.println("Карточка принята");
            } else {
                Serial.println("Ошибка контрольной суммы!");
            }
        } else {
            buffer[bytesRead] = recv;
            bytesRead++;
            Serial.print(recv);
        }
    }
    delay(10);
}

long toLong(String data) {
    char buf[20];
    data = "0x"+data;
    data.toCharArray(buf, 19);
    return strtol(buf, NULL, 0);
}
```

- ❶ Переменная `bytesRead` определяет количество байтов, обрабатываемых в текущем пакете. Она используется при выборе нужного элемента в буфере (`buffer[]`).
- ❷ Инициализация 14-байтового буфера, хранящего пакет данных, который передается со считывателя. Отсчет начинается с нулевого элемента: `buffer[0]`, `buffer[1]`... `buffer[13]`.
- ❸ Наличие дополнительных последовательных портов в Arduino Mega делает программный код и его отладку гораздо проще.
- ❹ Считывание байтов, поступающих через последовательное соединение; проводится только при доступности данных.
- ❺ Значение `0x02` указывает на начало текста. При этом содержимое буфера игнорируется и считывание начинается с самого начала (см. табл. 9.2).
- ❻ Значение `0x03` определяет конец текста. Если прочитано 12 байтов...
- ❼ ...то вычисляется контрольная сумма. Контрольная сумма определяется в результате выполнения операции побитового исключающего ИЛИ для каждого байта в идентификаторе (номере карточки). В нашем случае исключающее ИЛИ представлено как $a \oplus b$ и означает $a = a \oplus b$, где \oplus — побитовое исключающее ИЛИ.
- ❽ Вывод идентификатора (номера карточки) на монитор последовательного порта.
- ❾ Сверка вычисленной контрольной суммы с переданной в пакете.

Подключение к Raspberry Pi и программа управления модулем радиочастотной идентификации

Чтобы воспользоваться последовательным портом в Raspberry Pi, сначала нужно освободить его от применения интерпретатором входа в систему (детально об этом рассказано в главе 11). На рис. 9.10 показана схема подключения RFID-считывателя к Raspberry Pi. После подключения устройства выполните программный код, представленный в листинге 9.6.

Листинг 9.6. `rfid_reader.py`

```
# rfid_reader.py - считывание RFID-метки на частоте 125 КГц с
# помощью модуля ELB149C5M
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import serial # ❶
import struct

port = None

def main():
    global port
    bytesRead = -1 # ❷
    buff = [0x00]*12 # ❸
```

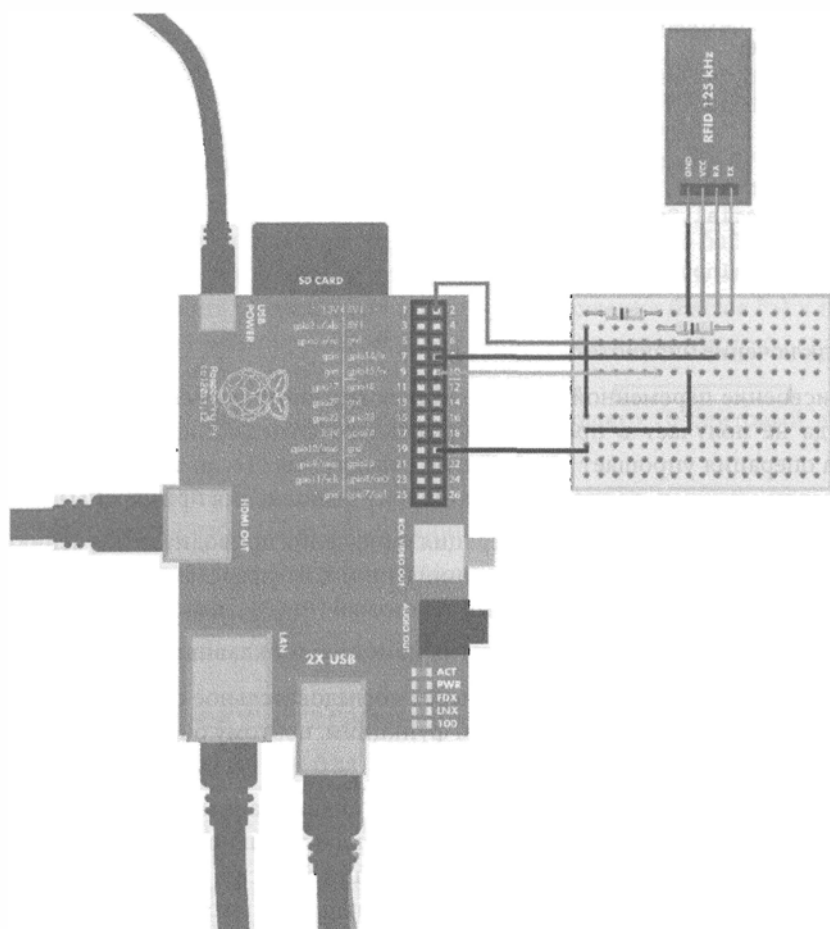


Рис. 9.10. Схема подключения модуля радиочастотной идентификации к Raspberry Pi

```
print("Готов к работе")
while True: # ❶
    recv = port.read() # ❷
    if(ord(recv) == 0x02): # ❸
        bytesRead = 0
        print("Считывание метки")
    elif(bytesRead == 12 and ord(recv) == 0x03): # ❹
        print("Проверка метки")
        data = "" # ❺
        checksum = 0x00
        for x in 0, 2, 4, 6, 8, 10:
            hexString = ''.join( buff[ x : x+2 ] ) # ❻
            translatedByte = int(hexString, 16)
            data += chr(translatedByte) # ❼
            checksum = checksum ^ translatedByte # ❽
        cardData = struct.unpack(">cic", data) # ❾
        if checksum != 0:
            print "Ошибка контрольной суммы"
```

```

        print cardData[1] # ❶
    else: # ❷
        buff[bytesRead] = recv # ❸
        bytesRead += 1

if __name__ == "__main__":
    if port == None:
        port = serial.Serial("/dev/ttyAMA0", baudrate=9600,
❸ timeout=None)
        port.flushInput()
    main()

```

- ❶ Подключение библиотеки pySerial.
- ❷ Присвоение переменной `bytesRead` невозможного значения, которое она никогда не получает в процессе нормального функционирования программы. Эта операция упрощает дальнейшую отладку кода (если такое значение выводится на монитор, то это означает ошибку выполнения программы).
- ❸ Заполнение буфера нулями. Операция умножения приводит к получению 12 элементов, заполненных вместо значений нулями. В буфере не сохраняются начальный текстовый (0x02) и конечный текстовый (0x03) символы (см. табл. 9.2).
- ❹ Программа выполняется до нажатия комбинации клавиш <Ctrl+C>.
- ❺ Считывание байта, полученного через последовательное соединение. Функция `read()` относится к блокирующим функциям, поэтому она ожидает получения считываемых данных.
- ❻ Значение 0x02 обозначает начало текста; любые данные, считанные ранее, не принимаются в расчет. Считываются исключительно данные нового пакета (см. табл. 9.2). Функция `ord()` возвращает целое число, соответствующее номеру символа в последовательности. Для 8-битовых символов оно представляется ASCII-значением.
- ❼ Значение 0x03 определяет конец текста. Вся метка считается прочитанной по достижении длины в 12 символов.
- ❽ Переменная `data` хранит необработанные байты данных.
- ❾ В текущий момент в буфере находятся ASCII-представления шестнадцатеричных кодов байтов. Например: `buff = ['3', 'E', '0', '0', 'F', 'B', '7', '8', '8', 'D', '3', '0']`. Да, такой подход несколько странный. Каждая пара ASCII-символов преобразуется в байт данных, т.е. строка 3E соответствует 0x3E (или 62).
- ❿ Добавление данных байта в переменную `data`. Это обычная техника программирования — представить пакет в виде переменной и вычислить контрольную сумму в одном цикле, чтобы избавиться от повторного использования циклических структур.
- ⓫ Контрольная сумма определяется в результате выполнения операции побитового исключающего ИЛИ. Существует специальный прием для сравнения вычисленной контрольной суммы с контрольной суммой, взятой из пакета данных.

Последняя операция исключающего ИЛИ выполняется над вычисленной контрольной суммой (т.е. $0x73$) и контрольной суммой, взятой из пакета данных (в случае успеха, тоже $0x73$). Но операция исключающего ИЛИ для одинаковых чисел возвращает нулевое значение ($0x73 \wedge 0x73 == 0x0$). В таком приеме программирования вам не придется повторно конвертировать “неправильный” ASCII-символ.

- 12 Извлечение значений в кортеж. Значения имеют разную байтовую длину, поэтому применяется функция `unpack()`. Параметр `>cis` в этой функции указывает выполнять операцию обратного расположения байтов (`>`) в последовательности: однобайтовый символ (`c`), 4-байтовое целое число со знаком (`i`) и конечный однобайтовый символ (`c`).
- 13 Вывод второго элемента кортежа, представляющего 10-байтовый номер карточки.
- 14 Если считываемый байт не относится к начальному и конечному, то, скорее всего, он относится к значимым данным.
- 15 Добавление только что считанного байта в буфер, который уже содержит переданные ранее ASCII-символы.

Пилотный проект: старинный сундук с современным замком

Давайте объединим дактилоскопический датчик с чем-то необычным, например, с сундуком с вымышленными сокровищами. Наша задача — заставить замок на сундуке открываться только после сканирования отпечатка пальца его владельца. Самой собой разумеется, что владельцем сундука будете вы.

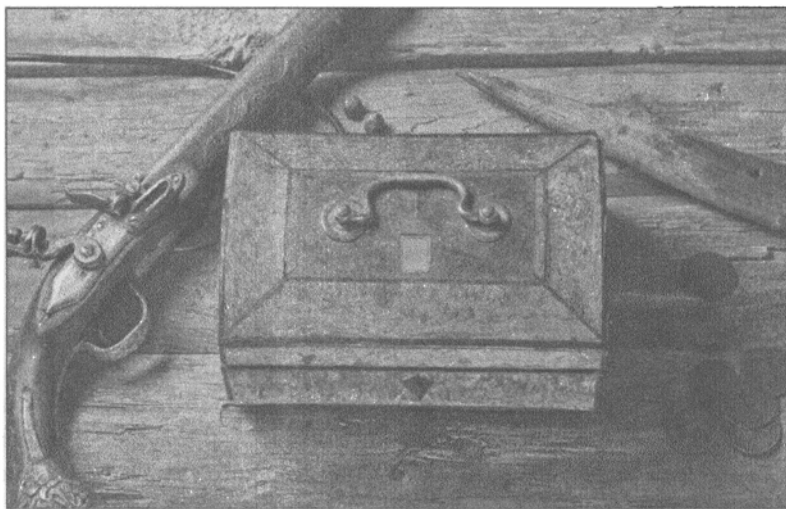


Рис. 9.11. Вот такой сундук с сокровищами на новый лад

Получаемые навыки

В проекте старинного сундука с дактилоскопическим датчиком вы научитесь следующему:

- управлять замком с помощью дактилоскопического датчика;
- разделить программный код на несколько файлов;
- создавать механизм замка, управляемый сервоприводом;
- стилизовать объекты под определенную эпоху и окружение.

На рис. 9.16 показана схема подключения оборудования данного проекта к Arduino.

Управление сундуком с сокровищами

Имеете ли вы право открывать сундук с сокровищами? Приложите палец к окошку, светящемуся синим цветом. Слабый сигнал и тихое жужжание механизма замка извещают о подтверждении вашей личности. Именно так мы представляем современные технологии на службе у кладоискателей.

Внутри сундука (помимо сокровищ) спрятаны две кнопки: добавления отпечатка пальца и сброса памяти. Они применяются для назначения владельца сундука. Распознавание правильного отпечатка пальца должно сопровождаться звуковым сигналом.

Кнопка сброса удаляет из памяти данные всех утвержденных ранее отпечатков пальцев. (Будьте внимательны, имеется в виду внешняя кнопка сброса дактилоскопического сканера, а не встроенная в Arduino кнопка сброса питания.)

Кнопка добавления отпечатка позволяет указать больше одного владельца сокровищ. Палец сканируется три раза. При сбое сканирования воспроизводятся пять коротких звуковых сигналов, извещающих о необходимости повторения операции назначения владельца. Первое удачное сканирование сопровождается одним коротким сигналом, второе — двумя, а третье — тремя. Назначив первого владельца, вы сможете продолжить добавлять другие отпечатки пальцев или ограничиться только одним. Чтобы добавить следующего владельца сокровищ, нажмите кнопку добавления отпечатка пальца еще раз.

Для того чтобы открыть замок в сундуке, необходимо авторизировать владельца по отпечатку пальца. Закройте крышку сундука и приложите свой палец к синему окошку дактилоскопического сканера. Короткий звуковой сигнал, слабое жужжание механизма — сундук заперт. Приложите палец к окошку датчика еще раз, и после звукового сигнала и жужжания замка можете открыть крышку.

Старинный сундук

Если у вас нет семейных драгоценностей или реликвий, хранящихся с начала XIX столетия, то вы вряд ли имели возможность детально изучить сундуки и шкапулки той эпохи. Для начала найдите или приобретите как можно более старый сундучок подходящего размера и прорежьте в нем отверстие под дактилоскопический датчик (рис. 9.12). Мы сначала карандашом обвели по контуру датчик, приложив его к крышке сундука, а затем просверлили небольшие отверстия по углам полученного прямоугольника и лобзиком прорезали боковые стороны будущего отверстия.



Рис. 9.12. *Отверстие в крышке под дактилоскопический сканер*

Механизм замка очень простой. Рычажок сервопривода при запираании заходит за специальную скобу, закрепленную на внутренней стороне верхней крышки (рис. 9.13). В качестве скобы можно использовать самые разные L-образные детали; мы, например, закрепили элемент детского конструктора Мессано (рис. 9.14).



Рис. 9.13. *Сервопривод замка*

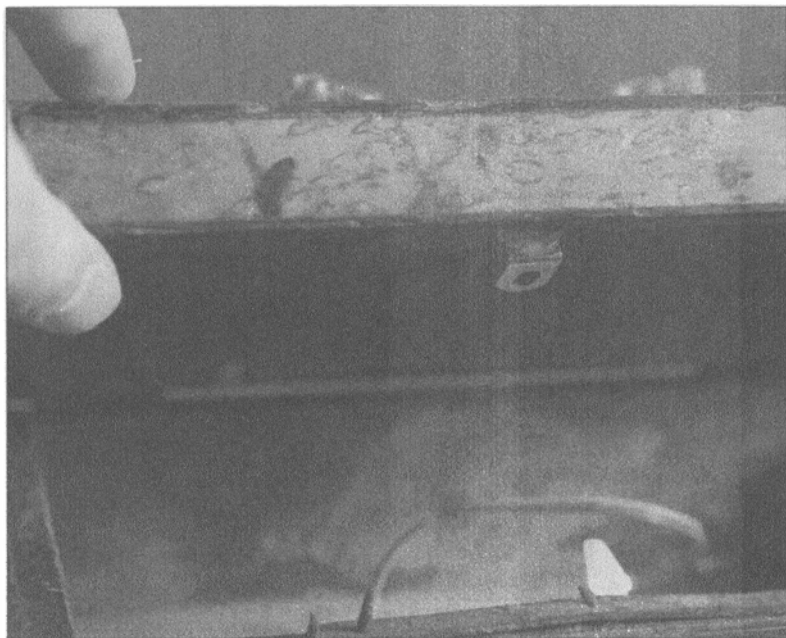


Рис. 9.14. *Замочное ушко*

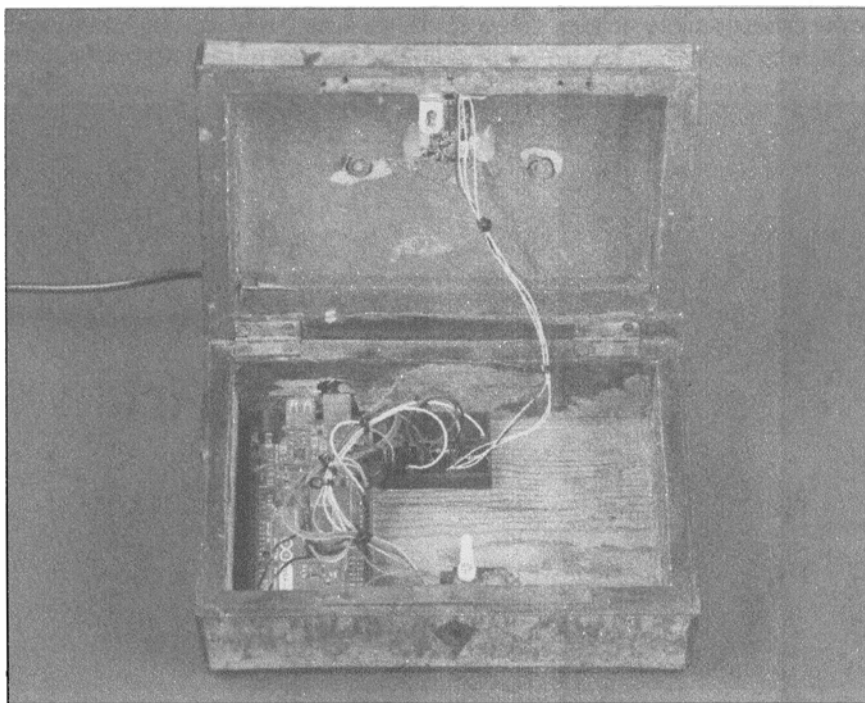


Рис. 9.15. *Система управления сундуком полностью укомплектована*

Подключение к Arduino и программа управления сундуком с сокровищами

На рис. 9.16 показана схема подключения оборудования текущего проекта к Arduino. После соединения всех компонентов в единую цепь выполните программный код из листинга 9.7.

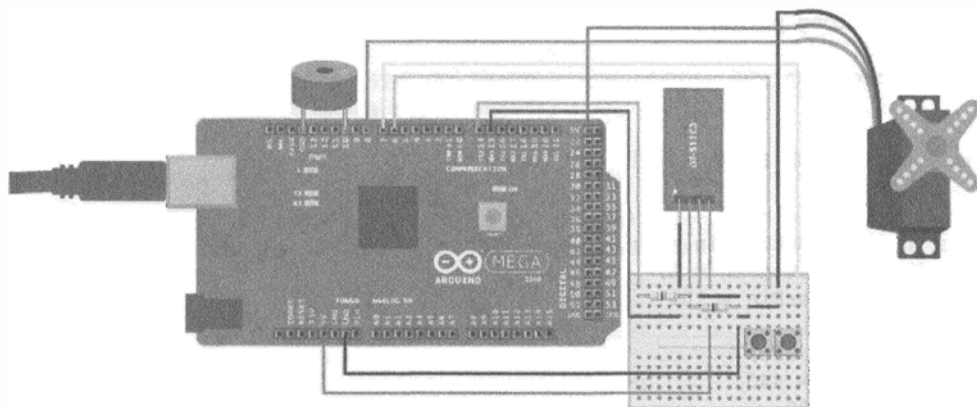


Рис. 9.16. Схема подключения компонентов системы управления сундуком к Arduino (см. цветную вклейку)

Все представленные в листинге 9.7 части программного кода уже рассматривались нами ранее: для управления дактилоскопическим датчиком применяется программа из листинга 9.3, а сервоприводы детально рассматривались в главе 5. В приведенных после листинга 9.7 сносках приведено описание новых, не рассматривавшихся ранее принципов программирования.

Листинг 9.7. `ancient_chest.ino`

```
// ancient_chest.ino - открывание замка после сканирования
// отпечатка пальца
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
const byte STX1 = 0x55;
const byte STX2 = 0xAA;

const word CMD_OPEN = 0x01;
const word CMD_CLOSE = 0x02;
const word CMD_LED = 0x12;
const word CMD_GET_ENROLL_COUNT = 0x20;
const word CMD_ENROLL_START = 0x22;
const word CMD_ENROLL_1 = 0x23;
const word CMD_ENROLL_2 = 0x24;
const word CMD_ENROLL_3 = 0x25;
const word CMD_IS_FINGER_PRESSED = 0x26;
const word CMD_DELETE_ALL = 0x41;
const word CMD_IDENTIFY = 0x51;
const word CMD_CAPTURE_FINGER = 0x60;
```

```

const word ACK = 0x30;
const word NACK = 0x31;

struct package {
    byte header1;
    byte header2;
    word deviceID;
    unsigned long param;
    word cmd;
    word checksum;
};

const int SIZE_OF_PACKAGE = 12;

const int lockPin = 8;
const int resetButtonPin = 7;
const int addButtonPin = 6;
const int speakerPin = 10;

float lowPeep = 220;
float highPeep = 440;
int closed = 2000;
int opened = 1000;
int state = 0;

word calcChecksum(struct package *pkg) {
    word checksum = 0;
    byte *buffer = (byte*)pkg;
    for(int i=0; i < (sizeof(struct package) - sizeof(word)); i++)
    {
        checksum += buffer[i];
    }
    return checksum;
}

int sendCmd(word cmd, int param) {
    struct package pkg;
    pkg.header1 = STX1;
    pkg.header2 = STX2;
    pkg.deviceID = 1;
    pkg.param = param;
    pkg.cmd = cmd;
    pkg.checksum = calcChecksum(&pkg);
    byte *buffer = (byte*)&pkg;

    int bytesSent = Serial3.write(buffer, sizeof(struct package));

    if(bytesSent != sizeof(struct package)) {
        Serial.println("Ошибка соединения");
        return -1;
    }

    int bytesReceived = 0;

```

```

char recvBuffer[SIZE_OF_PACKAGE];
struct package *recvPkg = (struct package*) recvBuffer;

bytesReceived = Serial3.readBytes(recvBuffer, sizeof(struct
package));
if (bytesReceived != SIZE_OF_PACKAGE) {
    Serial.println("Ошибка соединения");
    return -1;
}

if (recvPkg->header1 != STX1 || recvPkg->header2 != STX2) {
    Serial.println("Ошибка заголовка!");
    return -1;
}

if (recvPkg->checksum != calcChecksum(recvPkg)) {
    Serial.println("Неверная контрольная сумма!");
    return -1;
}

if (recvPkg->cmd == NACK) {
    Serial.println("Ошибка получения команды!");
    Serial.print("Ошибка: ");
    Serial.println(recvPkg->param, HEX);
    return -1;
}

return recvPkg->param;
}

void wave(int pin, float frequency, int duration)
{
    float period=1/frequency*1000*1000; // мкс
    long int startTime=millis();
    while(millis()-startTime < duration) {
        digitalWrite(pin, HIGH);
        delayMicroseconds(period/2);
        digitalWrite(pin, LOW);
        delayMicroseconds(period/2);
    }
}

void pulseServo(int servoPin, int pulseLenUs)
{
    digitalWrite(servoPin, HIGH);
    delayMicroseconds(pulseLenUs);
    digitalWrite(servoPin, LOW);
    delay(15);
}

void peep(int count, float frequency)
{
    for(int i = 0; i < count; i++) {
        wave(speakerPin, frequency, 400);
    }
}

```

```

        delay(400);
    }
}

void enrollFinger() {
    int id = 0;
    int ret = 0;
    id = sendCmd(CMD_GET_ENROLL_COUNT, 0);
    sendCmd(CMD_ENROLL_START, id);
    peep(1, lowPeep);
    WaitForFinger(false);
    ret = sendCmd(CMD_CAPTURE_FINGER, 1);
    if (ret < 0) {
        peep(5, highPeep);
        return;
    }
    sendCmd(CMD_ENROLL_1, 0);
    peep(1, highPeep);
    WaitForFinger(true);
    WaitForFinger(false);
    ret = sendCmd(CMD_CAPTURE_FINGER, 1);
    if (ret < 0) {
        peep(5, highPeep);
        return;
    }
    sendCmd(CMD_ENROLL_2, 0);
    peep(2, highPeep);
    WaitForFinger(true);

    WaitForFinger(false);
    ret = sendCmd(CMD_CAPTURE_FINGER, 1);
    if (ret < 0) {
        peep(5, highPeep);
        return;
    }
    sendCmd(CMD_ENROLL_3, 0);
    peep(3, highPeep);
    WaitForFinger(true);
}

void WaitForFinger(bool bePressed) {
    delay(500);
    if (!bePressed) {
        while (sendCmd(CMD_IS_FINGER_PRESSED, 0) > 0) {
            delay(200);
        }
    } else {
        while (sendCmd(CMD_IS_FINGER_PRESSED, 0) == 0) {
            delay(200);
        }
    }
}
}

```

```

void setup() {
    Serial.begin(115200);
    Serial3.begin(9600);
    Serial3.setTimeout(10*1000);
    delay(100);
    sendCmd(CMD_OPEN, 0);
    sendCmd(CMD_LED, 1);
    pinMode(resetButtonPin, INPUT);
    pinMode(addButtonPin, INPUT);
    pinMode(lockPin, OUTPUT);
    pinMode(speakerPin, OUTPUT);
    digitalWrite(resetButtonPin, HIGH);
    digitalWrite(addButtonPin, HIGH);
    for(int i = 0; i < 20; i++) {
        pulseServo(lockPin, closed);
    }
}

void loop() {
    if (digitalRead(resetButtonPin) == LOW) { // ❶
        if(sendCmd(CMD_DELETE_ALL, 0) >= 0) {
            peep(5,lowPeep);
        } else {
            peep(2,lowPeep); // Already empty
        }
    }
    if (digitalRead(addButtonPin) == LOW) { // ❷
        enrollFinger();
    }
    if(sendCmd(CMD_GET_ENROLL_COUNT, 0) == 0) // ❸
    {
        delay(100);
        return;
    }

    if(sendCmd(CMD_IS_FINGER_PRESSED, 0) == 0) {
        sendCmd(CMD_CAPTURE_FINGER, 1);
        int ret = sendCmd(CMD_IDENTIFY, 0);
        if(ret >= 0 && ret < 200) { // ❹
            if(state == 0) {
                peep(1,highPeep);
                for(int i = 0; i < 20; i++) {
                    pulseServo(lockPin, opened); // ❺
                }
                state = 1;
                Serial.println("Открыто");
            } else {
                Serial.println("Закрыто");
                peep(1,lowPeep);
                for(int i = 0; i < 20; i++) {
                    pulseServo(lockPin, closed);
                }
                state = 0;
            }
        }
    }
}

```

```

    } else {
        peep(5, lowPeep);
    }
    WaitForFinger(true);
}
}

```

- ❶ Нажатие кнопки сброса приводит к удалению всех сохраненных ранее отпечатков пальцев. Не путайте кнопку сброса памяти дактилоскопического датчика, размещенную в сундуке, с кнопкой сброса питания на плате Arduino.
- ❷ Кнопка добавления применяется для сохранения отпечатков пальцев всех владельцев сундука.
- ❸ Если в памяти сканера нет отпечатков пальцев владельцев, то сканирование и проверка их для открывания замка не проводятся.
- ❹ Авторизованный владелец сундучка найден!
- ❺ Открытие замка. Константа `opened` хранит значение 1000 мкс (1 мс), представляющее длительность импульса. При отправке повторяющегося импульсного сигнала сервопривод поворачивает рычажок на минимальный угол.

Кто или что ты?

Вы научились определять, что или кто взаимодействует с вашими устройствами: для идентификации объектов на них наклеиваются RFID-метки, опознание людей проводится по отпечаткам пальцев.

Сложив все ценности в надежный сундук с замком, управляемым дактилоскопическим сканером, переходите к изучению одного из самых важных разделов физики — электромагнетизма.

Электричество и магнетизм

10

Мы не ощущаем электромагнитное излучение, пронизывающее пространство вокруг. Наши органы чувств не в состоянии распознать электромагнитные поля, исходящие от линий электропередач и антенн сотовой связи, без которых немыслима современная жизнь. Электричество является источником питания для большинства бытовых устройств, в том числе и рассматриваемых в данной книге.

Датчики Холла предназначены для обнаружения магнитного поля. Они бывают самых разных видов: одни просто указывают на наличие магнитного поля, другие измеряют его напряженность, выражаемую в специальных единицах измерения, теслах (Тл). Датчики тока и напряжения, которыми оснащаются все мультиметры, измеряют электрический ток и напряженность электрического поля соответственно.

С помощью этих датчиков очень просто определить ток даже такой величины, что его подача на аналоговый вход микроконтроллерной платы приведет к поломке встроеного оборудования.

Электронный компас всегда указывает расположение северного магнитного полюса. Лучшие его модели дополняются акселерометром, позволяя указывать положение полюса даже в наклонном положении.

Северный и южный магнитные полюса на Солнце меняются местами каждые 11 лет. На момент написания книги смена полюсов на Солнце должна была произойти в течение недели. Но вам не стоит беспокоиться о том, что магнитные полюса Земли поменяются местами при вашей жизни. Последний раз это событие на нашей планете случилось 780 тысяч лет назад.

Эксперимент: определение напряжения и тока

В данном эксперименте мы воспользуемся прибором AttoPilot для измерения выходного напряжения блока батареек. Так вы сможете определить, насколько разряжены батарейки и как долго они еще будут вам служить. Этот прибор, оснащенный

аналоговым выходом, предназначен для измерения токов и напряжений электрического поля в очень широком диапазоне.

Мультиметр AttoPilot измеряет характеристики высокоомощных электрических приборов. Самая совершенная модель устройства предназначена для измерения напряжений до 50 В и токов до 180 А. Мощность тока (P) определяется как напряжение (U), умноженное на силу тока (I):

$$P = UI = 50 \text{ В} \times 180 \text{ А} = 9000 \text{ ВА} = 9000 \text{ Вт} = 9 \text{ кВт}$$

Модель AttoPilot, рассчитанная на измерение электрических характеристик с максимальными значениями 50 В и 180 А, надежно работает с источниками тока мощностью до 9 кВт. Чего не скажешь о нас с вами. Даже не пытайтесь самостоятельно ремонтировать оборудование высокой мощности, а ограничьтесь прототипированием слабомощных устройств, запитываемых от батареек или блоков питания USB.

Таблица 10.1. Пересчетные коэффициенты для токов и напряжений, измеряемых мультиметром AttoPilot

Параметры модели, В/А	Напряжение, на выходе/измеренное	Напряжение на выходе/сила тока измеренная	Описание
13,6/45	242,3 мВ/В	73,20 мВ/А	Применяется в текущем проекте
50/90	63,69 мВ/В	36,60 мВ/А	
50/180	63,69 мВ/В	18,30 мВ/А	9 кВт

Поскольку в проектах, реализуемых нами, высокие мощности недостижимы, то мы будем использовать самую младшую модель из представленных в табл. 10.1 — 13 В/45 А.

$$P = UI = 13 \text{ В} \times 45 \text{ А} = 585 \text{ Вт}$$

Применяемое нами устройство AttoPilot снабжено двумя аналоговыми выходами. Один представляет измеряемый ток, а второй — измеряемое напряжение. Максимальное напряжение на выходе прибора составляет 3,3 В, что гораздо больше максимально возможного входного, 50 В. Пересчетный коэффициент вычисляется как отношение максимальных значений измеряемых характеристик: 13,6 В/45 А. Используя этот подход, определим пересчетный коэффициент для измеряемого тока:

$$3,3 \text{ В (на выходе)}/45 \text{ А (измеренный)} = 73,3 \text{ мВ/А}$$

В программном коде мы будем оперировать обратной величиной:

$$45 \text{ А (измеренный)}/3,3 \text{ В (на выходе)} = 13,6363 \text{ А/В}$$

Таким образом, если необходимо вычислить измеренный ток, то умножьте напряжение на выходе прибора на пересчетный коэффициент 13,6363:

$0,05 \text{ В (на выходе)} \times 13,6363 \approx 0,6818 \text{ А} = 681,8 \text{ мА}$

Для напряжения применяется следующая пересчетная формула:

$3,3 \text{ В (на выходе)} / 13,6 \text{ В (измеренное)} = 242,6 \text{ мВ/В}$

Как и в случае тока, чтобы получить реальный пересчетный коэффициент, необходимо воспользоваться обратным значением:

$13,6 \text{ В (измеренное)} / 3,3 \text{ В (на выходе)} = 4,1212$

Для вычисления измеренного прибором напряжения умножьте напряжение на его выходе на полученный выше пересчетный коэффициент 4,1212:

$1,213 \text{ В (на выходе)} \times 4,1212 \approx 5 \text{ В}$

Подключение к Arduino и программа управления датчиком тока/напряжения AttoPilot

На рис. 10.1 показана схема подключения измерительного прибора к Arduino. После подключения всех показанных проволочных перемычек выполните программный код из листинга 10.1.

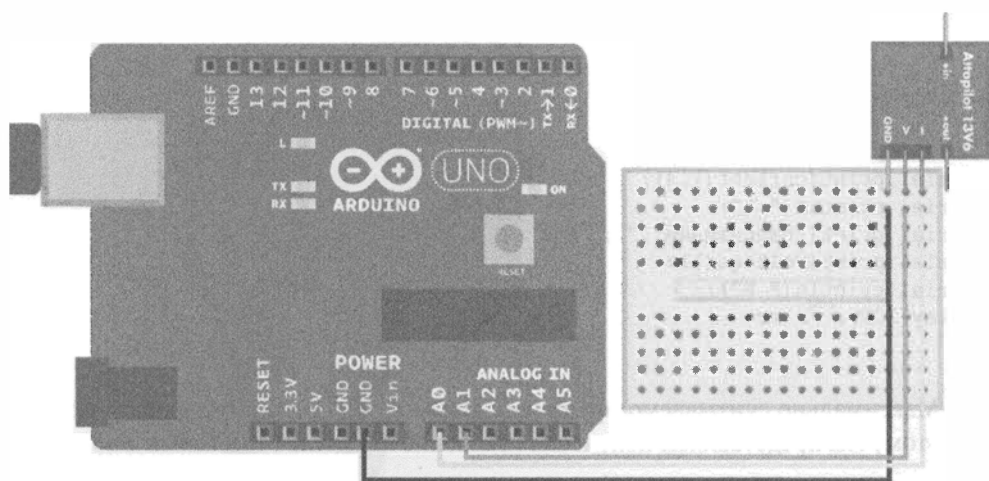


Рис. 10.1. Схема подключения датчика AttoPilot к Arduino (см. цветную вклейку)

Листинг 10.1. attopilot_voltage.ino

```
// attopilot_voltage.ino - измерение тока и напряжения блоком
// Attopilot 13,6 В/45 А
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int currentPin = A0;
int voltagePin = A1;
```

```

void setup()
{
    Serial.begin(115200);
    pinMode(currentPin, INPUT);
    pinMode(voltagePin, INPUT);
}

float current()
{
    float raw = analogRead(currentPin);
    Serial.println(raw);
    float percent = raw/1023.0; // ❶
    float volts = percent*5.0; // ❷
    float sensedCurrent = volts * 45 / 3.3; // A/B // ❸
    return sensedCurrent; // A // ❹
}

float voltage()
{
    float raw = analogRead(voltagePin);
    float percent = raw/1023.0;
    float volts = percent*5.0;
    float sensedVolts = volts * 13.6 / 3.3; // B/B // ❸
    return sensedVolts; // B
}

void loop()
{
    Serial.print("Ток: ");
    Serial.print(current(),4);
    Serial.println(" A");
    Serial.print("Напряжение: ");
    Serial.print(voltage());
    Serial.println(" B");
    delay(200); // мс
}

```

- ❶ Максимально считываемое функцией `analogRead()` значение равно 1023, поэтому мы представляем полученное значение как долю от максимального.
- ❷ Максимально получаемый функцией `analogRead()` сигнал соответствует напряжению 5 В. Таким образом, пять вольт сопоставляется с необработанным значением 1023.
- ❸ Коэффициент преобразования для тока составляет 45 А (измеренный)/3,3 В (на выходе) \approx 13,7 А/В (см. табл. 10.1).
- ❹ Возвращаемое значение представляется в амперах. Лучше всего указывать единицы измерения в комментариях (всегда!).
- ❺ Коэффициент преобразования для измеренного напряжения определяется как V (максимально измеренное напряжение)/ V (максимальное выходное напряжение).

Подключение к Raspberry Pi и программа управления датчиком тока/напряжения AttoPilot

На рис. 10.2 показано, как правильно подключить модуль датчиков тока и напряжения AttoPilot к плате Raspberry Pi. Убедившись в правильности соединения всех выводов, выполните программный код из листинга 10.2.

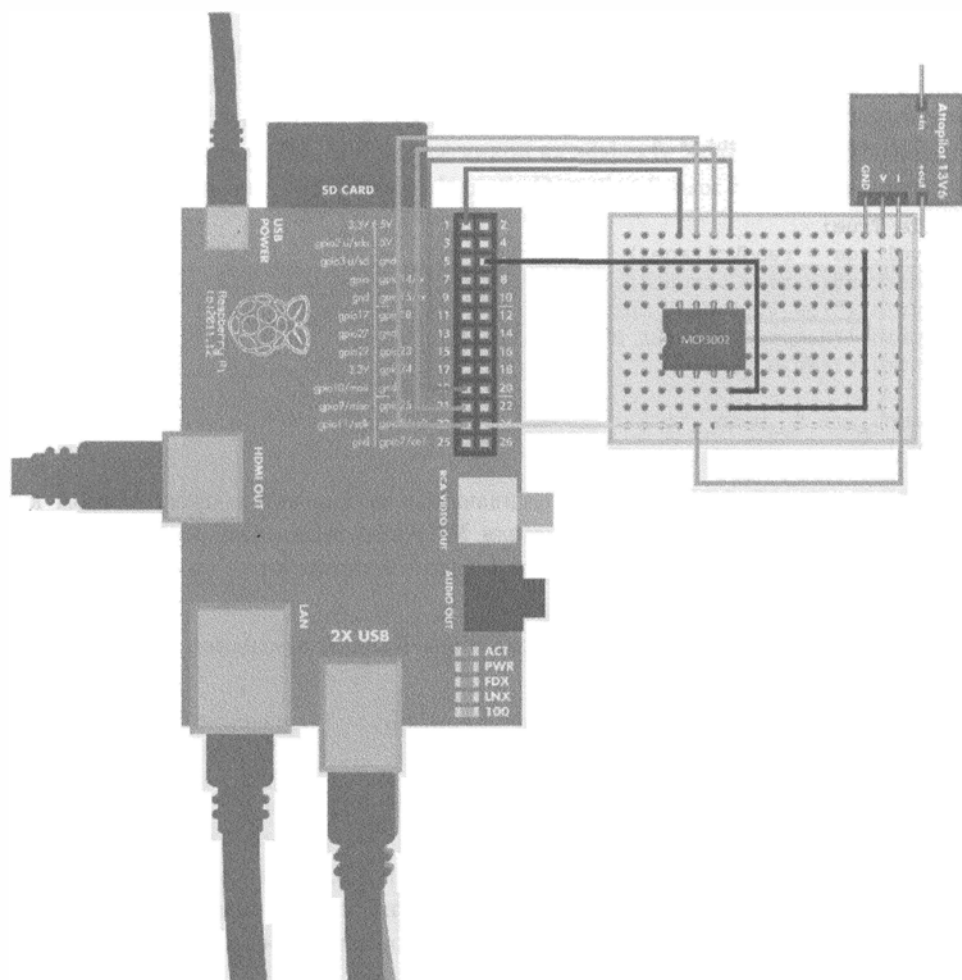


Рис. 10.2. Схема подключения датчика AttoPilot к Raspberry Pi (см. цветную вклейку)

Листинг 10.2. attopilot_voltage.py

```
# attopilot_voltage.py - измерение тока и напряжения блоком
# Attopilot 13,6 В/45 А
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import botbook_mcp3002 as mcp # ❶
```

```

def readVoltage():
    raw = mcp.readAnalog(0,1) # ❶
    percent = raw / 1023.0 # ❷
    volts = percent * 3.3 # ❸
    sensedVolts = volts * 13.6 / 3.3 # B/B # ❹
    return sensedVolts # B

def readCurrent():
    raw = mcp.readAnalog(0,0)
    percent = raw / 1023.0
    volts = percent * 3.3
    sensedCurrent = volts * 45.0 / 3.3 # A/B # ❺
    return sensedCurrent # A

def main():
    while True:
        voltage = readVoltage()
        current = readCurrent()
        print("Ток %.2f A" % current)
        print("Напряжение %.2f B" % voltage)
        time.sleep(0.5) # c

if __name__ == "__main__":
    main()

```

- ❶ Подключите библиотеку, обеспечивающую поддержку в программном коде аналого-цифрового преобразователя MCP3002. Файл библиотеки `botbook_mcp3002.py` должен располагаться в том же каталоге, что и файл текущей программы (`attopilot_voltage.py`).
- ❷ Считывание проводится со второго канала. Каналы измерения тока и напряжения в AttoPilot разделены.
- ❸ Максимально получаемое функцией `readAnalog()` значение равно 1023 и соответствует напряжению 3,3 В.
- ❹ Максимально допустимое напряжение на выводах GPIO платы Raspberry Pi составляет 3,3 В.
- ❺ Пересчетные коэффициенты вычисляются так же, как и в Arduino. Коэффициент преобразования для напряжения вычисляется по формуле V (максимально измеренное напряжение) / V (максимальное выходное напряжение).
- ❻ Коэффициент преобразования для тока составляет 45 А (измеренный) / 3,3 В (на выходе) $\approx 13,7$ А/В. (см. табл. 10.1).

Эксперимент: определение напряженности магнитного поля

Датчик, работающий на эффекте Холла, определяет напряженность магнитного поля. Подобные датчики применяются в огромном количестве устройств, например

в велосипедных спидометрах, где магнит, закрепленный на колесе, помогает определить количество его оборотов.

Суть эффекта Холла в том, что электроны под воздействием магнитного поля изменяют привычную траекторию своего движения, вызывая образование избыточной напряженности электрического поля.

Напряженность магнитного поля представляется датчиком в виде разных уровней электрического напряжения. Это напряжение считывается так же, как и с любого другого аналогового резистивного датчика, для чего применяются функции `analogRead()` и `botbook_mcp3002.readAnalog()`.

Данный способ измерения напряженности магнитного поля поддерживается большим количеством производителей датчиков на эффекте Холла. В своих начинаниях мы воспользуемся модулем KY-024 Magnetic Detecting Sensor Module, показанным на рис. 10.3.

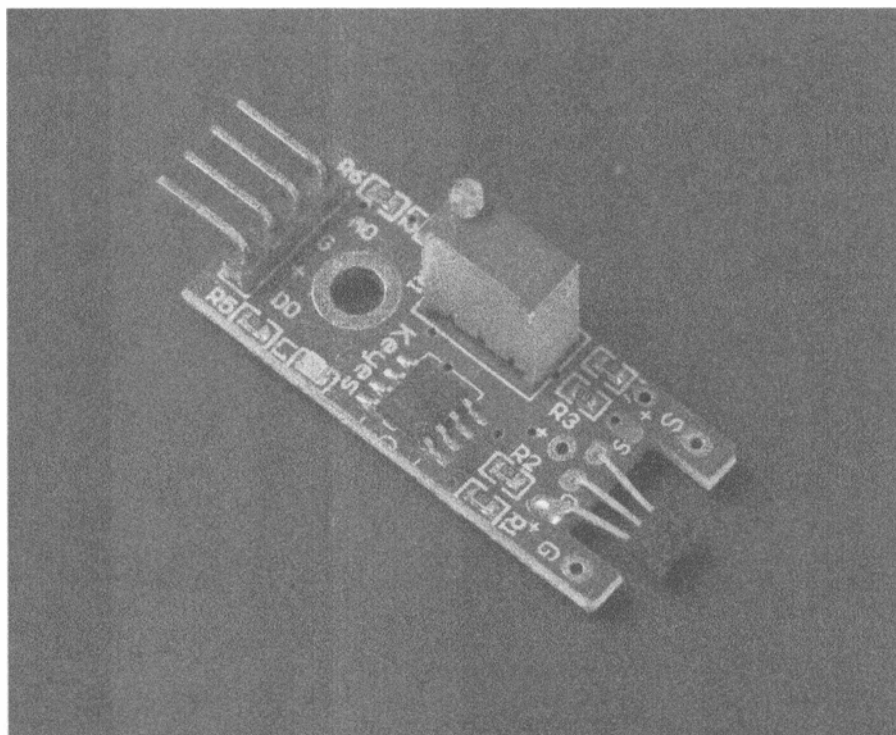


Рис. 10.3. Датчик KY-024

Подключение к Arduino и программа управления датчиком Холла

На рис. 10.4 показана схема подключения датчика Холла к Arduino. Подключите немногочисленные проводочные перемычки и выполните программный код из листинга 10.3.

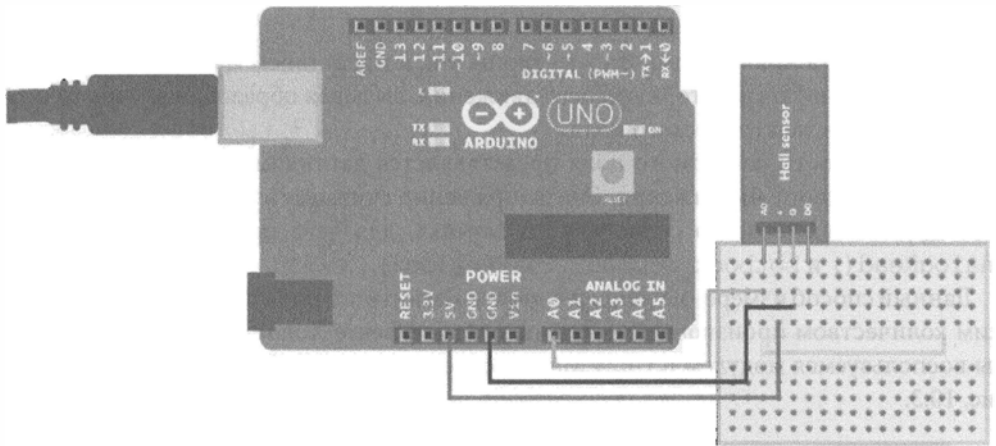


Рис. 10.4. Схема подключения датчика Холла к Arduino

Листинг 10.3. hall_sensor.ino

```
// hall_sensor.ino - вывод необработанного значения и
// определение полюсов магнита
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int hallPin = A0;
int rawMagneticStrength = -1; // ❶
int zeroLevel = 527; // ❷

void setup() {
    Serial.begin(115200);
    pinMode(hallPin, INPUT);
}

void loop() {
    rawMagneticStrength = analogRead(hallPin); // ❸
    Serial.print("Измеренная напряженность поля: ");
    Serial.println(rawMagneticStrength);
    int zeroedStrength = rawMagneticStrength - zeroLevel;
    // Если известен коэффициент пересчета напряжения в
    // напряженность магнитного поля для датчика, то абсолютное
    // значение равно zeroedStrength * conversion
    Serial.print("Уточненная напряженность поля: ");
    Serial.println(zeroedStrength);
    if(zeroedStrength > 0) {
        Serial.println("Южный полюс");
    } else if(zeroedStrength < 0) {
        Serial.println("Северный полюс");
    }
    delay(600); // мс
}
```

- ❶ Установка в значение, которое невозможно получить в результате нормального функционирования программы (датчика). Тем самым вы всегда сможете определить причину неправильного выполнения программы еще на этапе отладки.
- ❷ Необработанный значение, получаемое функцией `analogRead()`, при отсутствии магнитного поля. В нашем случае на выходе датчика при отсутствии магнитного поля регистрировалось значение 527. В технической документации к датчику указывается значение 500 единиц, предположительно для логического уровня с напряжением 5 В. Если в вашем случае при отсутствии магнитного поля на монитор последовательного порта Arduino выводится другое значение, то подкорректируйте значение переменной `zeroLevel`.
- ❸ Датчик Холла управляется микроконтроллером так же, как и любой другой аналоговый резистивный датчик, хотя по своей сути резистором не является.

Подключение к Raspberry Pi и программа управления датчиком Холла

Схема подключения датчика на эффекте Холла к Raspberry Pi показана на рис. 10.5. После его подключения выполните программный код, приведенный в листинге 10.4.

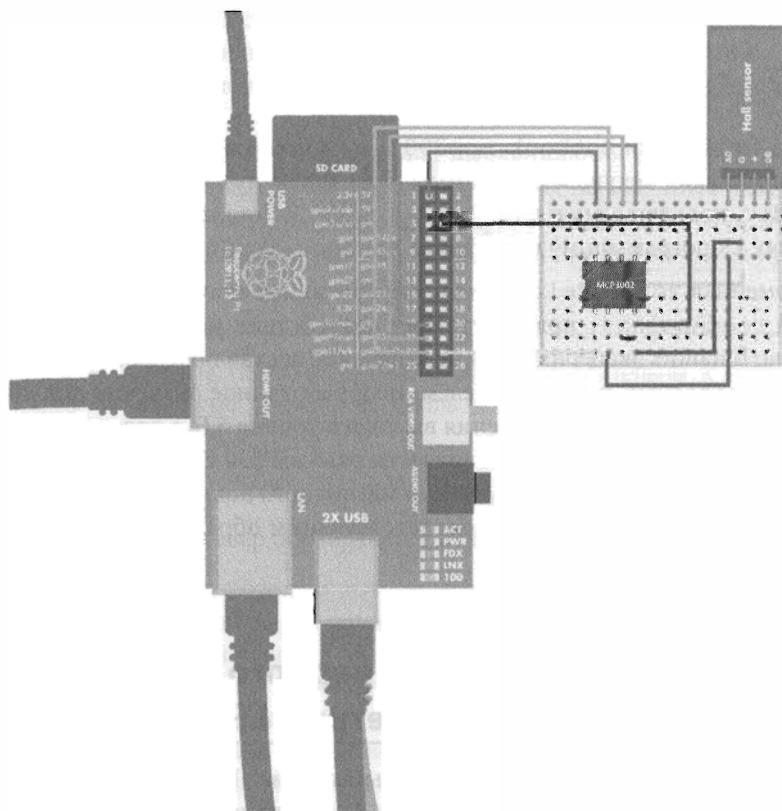


Рис. 10.5. Схема подключения датчика Холла к Raspberry Pi (см. цветную вклейку)

Листинг 10.4. hall_sensor.py

```
# hall_sensor.py - вывод необработанных значений и определение
# полюсов магнита
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp # ❶

zeroLevel = 388 # ❷

def main():
    while True:
        rawMagneticStrength = mcp.readAnalog()
        print("Измеренная напряженность поля: %i " %
rawMagneticStrength)
        zeroedStrength = rawMagneticStrength - zeroLevel
        print("Уточненная напряженность поля: %i " %
zeroedStrength)
        if(zeroedStrength > 0):
            print("Южный полюс")
        elif(zeroedStrength < 0):
            print("Северный полюс")
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

- ❶ Файл библиотеки `botbook_mcp3002.py` должен находиться в том же каталоге, что и файл текущей программы. Вам также нужно установить библиотеку `spidev`, подключаемую в `botbook_mcp3002`. Детально об этом см. в комментариях к программному коду файла `botbook_mcp3002.py` и в главе 3.
- ❷ Переменная `zeroLevel` хранит необработанное значение, считываемое функцией `readAnalog()` при отсутствии воздействия магнитного поля на датчик Холла. В наших экспериментах это значение составляло 388. В спецификациях, предоставляемых производителем, необработанное значение должно составлять 500 единиц для напряжения высокого уровня 5 В, а для напряжения высокого уровня 3,3 В оно устанавливается равным 330 единиц ($500 \times (3,3/5) = 330$). Если в вашем случае при отсутствии магнитного поля наблюдаются другие значения, то подкорректируйте соответствующим образом значение переменной `zeroLevel`.

Эксперимент: определение северного магнитного полюса компасом-акселерометром LSM303

Компас-акселерометр LSM303, показанный на рис. 10.6, указывает направление расположения северного магнитного полюса. Акселерометр применяется для коррекции показаний при изменении ориентации устройства в пространстве.

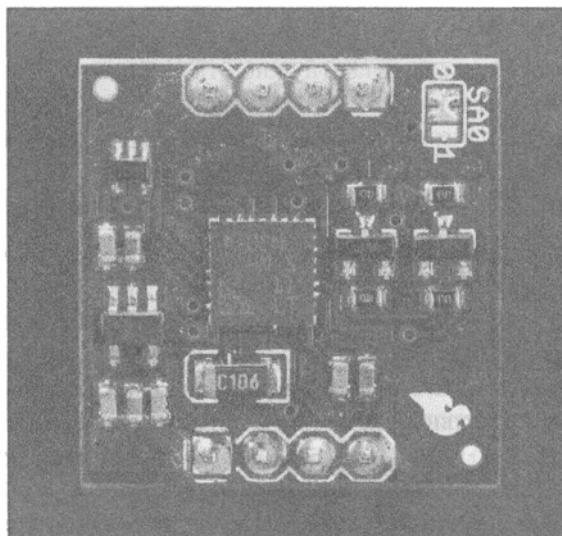


Рис. 10.6. Компас-акселерометр LSM303 от компании SparkFun

Если вам доводилось заниматься ориентированием на местности, то вы не испытаете затруднений с удержанием компаса в горизонтальной плоскости, выполняя другие действия (изучая карту или фотографируя медведя). Рассчитанный на новичков, этот прибор покажет расположение северного магнитного полюса даже в наклонном состоянии.

На плате датчика LSM303 нет маркера для выравнивания его по направлению к северному полюсу. Чтобы нанести маркировку вручную, переверните плату так, чтобы можно было прочитать надпись SA0 слева направо. Северный полюс можно обозначить стрелкой, указывающей на правый край платы при нормальной ориентации надписи SA0.

Электронные компасы, подобные LSM303, очень чувствительны к внешним электромагнитным полям. Держите их подальше от силовых кабелей и больших металлических предметов.

В выходных значениях север обозначается нулевыми показаниями компаса.

У вас нет компаса для проверки получаемых измерений? В северном полушарии определить стороны света в городских условиях очень просто. Большая часть спутниковых «тарелок» ориентирована на юг. За редким исключением, антенны спутниковой связи наподобие телевизионных неподвижны. Это означает, что они принимают сигнал со спутников,двигающихся по геостационарным орбитам, т.е.двигающихся над поверхностью со скоростью вращения Земли. А поскольку большинство геостационарных орбит проходит вдоль экватора, то спутниковые антенны ориентированы на юг.

Калибровка компаса

Для точной работы компас вначале нужно откалибровать. Конечно, можно понадеяться на удачу и не заниматься калибровкой прибора. Вы даже будете наблюдать на мониторе значения, выводимые им, но не будете уверены в них до тех пор, пока все же не откалибруете устройство.

Калибровка компаса заключается в выполнении следующих действий.

- Правильно подключите компас к выбранной платформе, Arduino или Raspberry Pi.
- Выполните программу и получите некоторые значения.
- Измените в программном коде значение константы `runningMode` на 0, чтобы перевести прибор в режим калибровки. В этом режиме программа выводит только минимальные и максимальные значения для каждого из направлений (осей).
- Установите максимальные значения равными нулю. В Raspberry Pi измените значения `magMax[]` и `magMin[]`. В Arduino необходимо обнулить каждую из шести переменных: `magMax_x`, `magMax_y`... `magMin_z`.
- Повращайте и понаклоняйте устройство (лучше всего делать движения, имитирующие рисование в воздухе цифры 8). Продолжайте движения до тех пор, пока значения на мониторе не перестанут изменяться. Запомните или запишите минимальные и максимальные значения, полученные в процессе калибровки.
- Строго задайте в коде полученные на предыдущем этапе значения. Установите их вместо обнуляемых выше значений, т.е. `magMax[]` и `magMin[]` или `magMax_x`, `magMax_y`... `magMin_z`.

Закончив с калибровкой, переведите компас в обычный режим, установив константу `runningMode` в исходное значение (1).

Выполните программный код и попробуйте откалибровать датчик описанным выше способом. Решившись на применение компаса в собственных проектах, вам не обойтись без детального изучения интерфейса, через который он подключается к микроконтроллерному устройству, о чем рассказывается далее.

Подключение к Arduino и программа управления компасом LSM303

Схема подключения компаса-акселерометра к Arduino показана на рис. 10.7. Подключите все перемычки согласно рисунку и выполните программный код из листинга 10.5.

Листинг 10.5. `lsm303.ino`

```
// lsm303.ino - калибровка и использование компаса-акселерометра
// LSM303DLH
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>

const char accelerometer_address = 0x30 >> 1; // ❶
const char magnetometer_address = 0x3C >> 1;

const int runningMode = 0; // ❷

float magMax_x = 0.1; float magMax_y = 0.1; float magMax_z = 0.1; // ❸
```

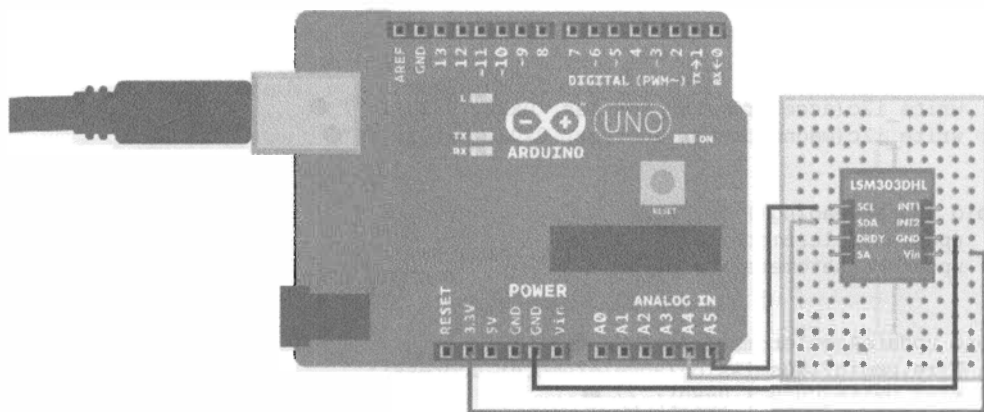


Рис. 10.7. Схема подключения датчика LSM303 к Arduino

```
float magMin_x = -0.1; float magMin_y = -0.1; float magMin_z = -0.1;

float acc_x = 0; float acc_y = 0; float acc_z = 0; // ❶
float mag_x = 0; float mag_y = 0; float mag_z = 0;

int heading = 0;

void setup() {
    Serial.begin(115200);
    Wire.begin(); // ❷
    Serial.println("Инициализация компаса");
    initializeism(); // ❸
    delay(100);
    Serial.println("Начало считывания");
}

void loop() {
    updateHeading(); // ❹
    if(runningMode == 0) { // ❺
        magMax_x = max(magMax_x, mag_x);
        magMax_y = max(magMax_y, mag_y);
        magMax_z = max(magMax_z, mag_z);
        magMin_x = min(magMin_x, mag_x);
        magMin_y = min(magMin_y, mag_y);
        magMin_z = min(magMin_z, mag_z);
        Serial.print("Максимально для осей X Y Z: ");
        Serial.print(magMax_x); Serial.print(" ");
        Serial.print(magMax_y); Serial.print(" ");
        Serial.print(magMax_z); Serial.print(" ");
        Serial.print("Минимально для осей X Y Z: ");
        Serial.print(magMin_x); Serial.print(" ");
        Serial.print(magMin_y); Serial.print(" ");
        Serial.print(magMin_z); Serial.println(" ");
    } else {
        calculateHeading();
        Serial.println(heading); // ❻
    }
}
```

```

    delay(100); // mc
}

void initializelsm() {
    write_i2c(accelerometer_address, 0x20, 0x27); // 17
    write_i2c(magnetometer_address, 0x02, 0x00); // 18
}

void updateHeading() {
    updateAccelerometer();
    updateMagnetometer();
}

void updateAccelerometer() { // 19
    Wire.beginTransmission(accelerometer_address);
    Wire.write(0x28 | 0x80); // 20
    Wire.endTransmission(false);
    Wire.requestFrom(accelerometer_address, 6, true); // 21
    int i = 0;
    while(Wire.available() < 6) {
        i++;
        if(i > 1000) {
            Serial.println("Ошибка получения данных акселерометра
❖ через I2C");
            return;
        }
    }

    uint8_t axel_x_l = Wire.read(); // 22
    uint8_t axel_x_h = Wire.read();
    uint8_t axel_y_l = Wire.read();
    uint8_t axel_y_h = Wire.read();
    uint8_t axel_z_l = Wire.read();
    uint8_t axel_z_h = Wire.read();

    acc_x = (axel_x_l | axel_x_h << 8) >> 4; // 23
    acc_y = (axel_y_l | axel_y_h << 8) >> 4;
    acc_z = (axel_z_l | axel_z_h << 8) >> 4;
}

void updateMagnetometer() { // 24
    Wire.beginTransmission(magnetometer_address);
    Wire.write(0x03);
    Wire.endTransmission(false);
    Wire.requestFrom(magnetometer_address, 6, true);
    int i = 0;
    while(Wire.available() < 6) {
        i++;
        if(i > 1000) {
            Serial.println("Ошибка получения данных магнетометра
❖ через I2C ");
            return;
        }
    }

    uint8_t axel_x_h = Wire.read();
    uint8_t axel_x_l = Wire.read();
    uint8_t axel_y_h = Wire.read();
    uint8_t axel_y_l = Wire.read();

```

```

uint8_t axel_z_h = Wire.read();
uint8_t axel_z_l = Wire.read();

mag_x = (int16_t)(axel_x_l | axel_x_h << 8);
mag_y = (int16_t)(axel_y_l | axel_y_h << 8);
mag_z = (int16_t)(axel_z_l | axel_z_h << 8);
}

// Направление на север
void calculateHeading() {
    // Определение единичного вектора
    float dot = acc_x*acc_x + acc_y*acc_y + acc_z*acc_z; // 18
    float magnitude = sqrt(dot); // 19
    float nacc_x = acc_x / magnitude; // 20
    float nacc_y = acc_y / magnitude;
    float nacc_z = acc_z / magnitude;

    // Применение калибровочных данных
    mag_x = (mag_x - magMin_x) / (magMax_x - magMin_x) * 2 - 1.0; // 21
    mag_y = (mag_y - magMin_y) / (magMax_y - magMin_y) * 2 - 1.0;
    mag_z = (mag_z - magMin_z) / (magMax_z - magMin_z) * 2 - 1.0;

    // Направление на восток
    float ex = mag_y*nacc_z - mag_z*nacc_y; // 22
    float ey = mag_z*nacc_x - mag_x*nacc_z;
    float ez = mag_x*nacc_y - mag_y*nacc_x;
    dot = ex*ex + ey*ey + ez*ez; // 23
    magnitude = sqrt(dot);
    ex /= magnitude;
    ey /= magnitude;
    ez /= magnitude; // 24

    // Проекция
    float ny = nacc_z*ex - nacc_x*ez; // 25

    float dotE = -1 * ey; // 26
    float dotN = -1 * ny;

    // Вычисление угла
    heading = atan2(dotE, dotN) * 180 / M_PI; // 27

    heading = round(heading);
    if (heading < 0) heading += 360; // 28
}

void write_i2c(char address, unsigned char reg, const uint8_t data)
{
    Wire.beginTransaction(address);
    Wire.write(reg);
    Wire.write(data);
    Wire.endTransmission();
}

```

- ❶ Адреса акселерометра и магнетометра, принимающие участие в операциях по-
разрядного сдвига (см. главу 8).

- 7 Нормальный режим функционирования данной программы (вывод показания компаса) задается константой `runningMode=1`. Чтобы откалибровать компас, обнулите значение константы (см. раздел “Калибровка компаса”).
- 8 Исходные калибровочные значения помогут вам получить показания компаса перед проведением точной калибровки (в режиме `runningMode=0`).
- 9 Переменные для хранения текущих показаний установлены в нуль.
- 10 Библиотека `Wire.h` обеспечивает поддержку интерфейса I²C в Arduino.
- 11 Вызов функции инициализации.
- 12 Эта функция обновляет глобальные переменные, не возвращая никакого значения.
- 13 В режиме калибровки выводятся только минимальные и максимальные показания компаса.
- 14 В обычном рабочем режиме выводятся показания, откорректированные с учетом наклона (ускорения).
- 15 Включение акселерометра и установка его параметров в значения по умолчанию. Параметры по умолчанию указываются в технической документации к компасу-акселерометру.
- 16 Включение магнетометра и перевод его в режим непрерывной работы.
- 17 Получение данных с компаса согласно настройкам протокола, взятым из технической документации, и последующее обновление глобальных переменных.
- 18 Значение `0x80` в двоичном виде представляется как `0b10000000`: старший разряд (MSB) равен единице, а остальные семь битов представлены нулями. Операция исключающего ИЛИ для числа `0x80` и любого другого числа приводит к результату, в котором старший разряд (MSB) всегда равен 1. Число `0x28`, согласно технической документации, соответствует регистру `OUT_X_L_A`.
- 19 Считывание шести байтов.
- 20 Каждое необработанное значение разделяется на два байта: младший, менее значимый (`axel_x_l`), и старший, более значимый (`axel_x_h`).
- 21 Каждое необработанное значение имеет длину $8+4=12$ бит. Старший байт содержит более значимые 8 бит. Младший байт состоит из оставшихся 4 бит. Остальные 4 бита младшего байта (`axel_x_l`) не учитываются (детально о битовых операциях см. в главе 8).
- 22 Необработанные значения магнетометра (компаса) считываются так же, как данные акселерометра (функцией `updateAccelerometer()`).
- 23 Вычисление скалярного произведения векторов...
- 24 ...для получения модуля.
- 25 Деление каждого измерения на модуль для получения вектора с длиной, равной единице. Таким образом, мы получим единичный вектор, направленный вверх (`nacc_x, nacc_y, nacc_z`).
- 26 Применение данных калибровки.

- 22 Восток располагается в направлении, повернутом на 90 градусов относительно севера. Он также располагается в 90 градусах от направления вверх. Векторное произведение позволяет вычислить вектор, расположенный перпендикулярно указанным двум направлениям.
- 23 Вектор на восток, как и север ранее, нормализован.
- 24 После нормализации вектор (e_x , e_y , e_z) имеет модуль, равный единице и указывающий на восток. Операция $/=$ выполняется подобно операции $+=$, но только для деления. Например, $f_{00} /= 2$ вычисляется так же, как и $f_{00} = f_{00}/2$.
- 25 Выполнение векторного произведения для нахождения вектора N в горизонтальной плоскости.
- 26 Проецирование вектора N из плоскости NE в плоскость XY.
- 27 Определение угла между осью Y и проецированным вектором N. Преобразование радиан в градусы ($\text{deg} = \text{rad} / (2 * \pi) * 360$).
- 28 Перебор значений, обеспечивающих показания компаса в диапазоне от 0 до 360 градусов.

Подключение к Raspberry Pi и программа управления компасом LSM303

На рис. 10.8 показана схема подключения электронного компаса к Raspberry Pi. Подключив все показанные на ней перемычки, выполните программный код из листинга 10.6.

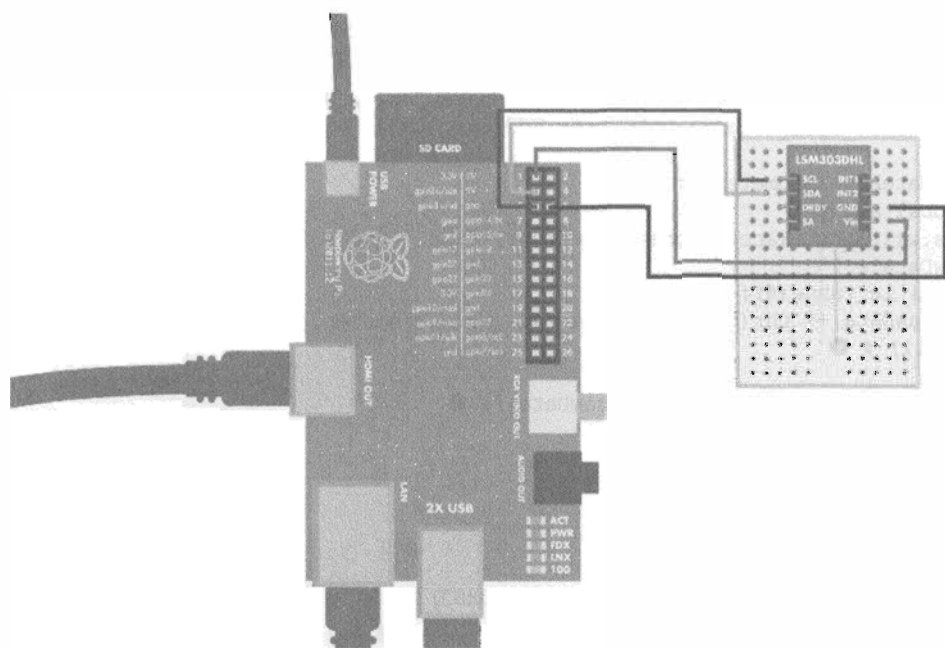


Рис. 10.8. Схема подключения датчика LSM303 к Raspberry Pi

Не забывайте откалибровать прибор, как было описано в разделе “Калибровка компаса”.

Листинг 10.6. lsm303.py

```
# lsm303.py - калибровка и использование компаса-акселерометра
# LSM303DLH
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import smbus # sudo apt-get -y install python-smbus
import struct
import math

accelerometer_address = 0x30 >> 1
magnetometer_address = 0x3C >> 1

calibrationMode = True # ❶

magMax = [ 0.1, 0.1, 0.1 ] # ❷
magMin = [ 0.1, 0.1, 0.1 ]

acc = [ 0.0, 0.0, 0.0 ] # ❸
mag = [ 0.0, 0.0, 0.0 ]

heading = 0

def initlsm():
    global bus
    bus = smbus.SMBus(1)
    bus.write_byte_data(accelerometer_address, 0x20, 0x27) # ❹
    bus.write_byte_data(magnetometer_address, 0x02, 0x00) # ❺

def updateAccelerometer():
    global acc # ❻
    bus.write_byte(accelerometer_address, 0x28 | 0x80) # ❼
    rawData = ""
    for i in range(6):
        rawData += chr(bus.read_byte_data(accelerometer_address,
        0x28+i)) # ❽

    data = struct.unpack('<hhh',rawData) # ❾
    acc[0] = data[0] >> 4 # ❿
    acc[1] = data[1] >> 4
    acc[2] = data[2] >> 4

def updateMagnetometer(): # ⓫
    global mag
    bus.write_byte(magnetometer_address, 0x03)
    rawData = ""
    for i in range(6):
        rawData += chr(bus.read_byte_data(magnetometer_address, 0x03+i))
```



```

data = struct.unpack('>hhh',rawData)
mag[0] = data[0]
mag[1] = data[1]
mag[2] = data[2]

def calculateHeading():
    global heading, acc, mag
    # Нормализация
    normalize(acc) # 11

    # Применение калибровочных данных
    mag[0] = (mag[0] - magMin[0]) / (magMax[0] - magMin[0]) *
2.0 - 1.0 # 12
    mag[1] = (mag[1] - magMin[1]) / (magMax[1] - magMin[1]) *
2.0 - 1.0
    mag[2] = (mag[2] - magMin[2]) / (magMax[2] - magMin[2]) *
2.0 - 1.0

    e = cross(mag, acc) # 13
    normalize(e) # 14

    n = cross(acc,e) # 15

    dotE = dot(e,[0.0, -1.0, 0.0]) # 16
    dotN = dot(n,[0.0, -1.0, 0.0])

    heading = round(math.atan2(dotE, dotN) * 180.0 / math.pi) # 17
    if heading < 0: # 18
        heading += 360 # 20

def normalize(v): # 19
    magnitude = math.sqrt(dot(v,v))
    v[0] /= magnitude
    v[1] /= magnitude
    v[2] /= magnitude

def dot(v1, v2): # 21
    return v1[0]*v2[0] + v1[1]*v2[1] + v1[2]*v2[2]

def cross(v1, v2): # 22
    vr = [0.0, 0.0, 0.0]
    vr[0] = v1[1] * v2[2] - v1[2] * v2[1]
    vr[1] = v1[2] * v2[0] - v1[0] * v2[2]
    vr[2] = v1[0] * v2[1] - v1[1] * v2[0]
    return vr

def main():
    initlsm()
    while True:
        updateAccelerometer() # 23
        updateMagnetometer()

        if calibrationMode: # 24
            magMax[0] = max(magMax[0], mag[0])
            magMax[1] = max(magMax[1], mag[1])

```

```

magMax[2] = max(magMax[2], mag[2])
magMin[0] = min(magMin[0], mag[0])
magMin[1] = min(magMin[1], mag[1])
magMin[2] = min(magMin[2], mag[2])
print("magMax = [ %.1f, %.1f, %.1f ]" % (magMax[0],
↳magMax[1], magMax[2]))
print("magMin = [ %.1f, %.1f, %.1f ]" % (magMin[0],
↳magMin[1], magMin[2]))
else:
    calculateHeading() # ❷
    print(heading)
time.sleep(0.5)

if __name__ == "__main__":
    main()

```

- ❶ При установке переменной `calibrationMode` в значение `True` прибор переводится из рабочего режима в режим калибровки (см. раздел “Калибровка компаса”).
- ❷ В коде указываются определенные нами калибровочные значения, которые можно применять без дополнительной проверки (но лучше все же провести калибровку самостоятельно).
- ❸ Последние показания компаса и калибровочные значения сохраняются в глобальных переменных `acc`, `mag` и `heading`.
- ❹ Управление акселерометром с помощью встроенных команд, взятых из технической документации.
- ❺ Включение магнетометра.
- ❻ Для обновления глобальной переменной ее необходимо объявить соответствующим образом в начале функции.
- ❼ Отправка сообщения с первым битом, равным 1 (`0x80 == 0b1000000`), и последним битом, содержащим адрес регистра `OUT_X_L_A` (`0x28 == 0b101000`). В результате выполнения операции исключающего ИЛИ над этими значениями возвращается число `0b10101000` (см. описание побитовых операций в главе 8).
- ❽ Считывание шести байтов, начиная с адреса акселерометра `0x28`.
- ❾ Извлечение целочисленных значений из байтов, переданных датчиком (`rawData`). Формат получаемых значений такой: короткое (двухбайтовое, 16-битовое) целое число (`h`), представленное в прямом порядке следования (`<`). Точное значение извлекаемых целочисленных значений определяется методом `Struct.unpack()`.
- ❿ Значение `data[0]` 16-битовое, в котором значащие только первые 12 бит, а последние 4 бита не учитываются (см. операцию битового сдвига, детально рассмотренную в главе 8).
- ⓫ Функция `updateMagnetometer()` считывает значения подобно функции `updateAccelerometer()`.

- 17 Нормализация вектора ускорения или преобразование его в единичный вектор. Он направлен вверх, но его длина после нормализации равна единице.
- 18 Применение калибровочных данных.
- 14 Восток располагается в 90 градусах от севера (mag) и в 90 градусах от направления вверх (acc).
- 15 После нормализации e представляет единичный вектор (с длиной равной 1), указывающий на восток.
- 16 Новый вектор n перпендикулярен как направлению гравитации, так и направлению на восток. Таким образом, плоскость NE выровнена горизонтально с плоскостью гравитации, но не совпадает с плоскостью XY устройства.
- 17 Скалярное произведение векторов позволяет спроецировать вектор, указывающий на север, на плоскость XY устройства. Вектор, указывающий на восток, учитывается, поскольку в проецировании принимает участие вся плоскость NE.
- 18 Угол между осью Y и направлением на север будет показанием компаса. Это значение преобразуется из радиан (полный круг представляется как 2π) в градусы (полный круг соответствует 360 градусам), для чего используется операция деления.
- 19 Если показание представляется отрицательным значением...
- 20 ...то оно вычитается из полного круга, чтобы всегда попадать в диапазон от 0 до 360 градусов.
- 21 Операция нормализации приводит к образованию единичного вектора. Этот вектор имеет модуль (длину), равный 1, и указывает в исходном направлении.
- 23 Формула скалярного произведения векторов взята из учебника по математике. В данной программе скалярное произведение применяется для вычисления модуля (длины) вектора. Модуль mag вектора v вычисляется по формуле $\text{sqrt}(\text{dot}(v))$.
- 23 Формула векторного произведения также взята из курса математики. В нашей программе она используется для определения вектора, перпендикулярного двум другим векторам. В формуле $c = \text{cross}(a, b)$, c — это вектор, перпендикулярный (расположенный под углом 90 градусов к) векторам a и b .
- 24 Функции `updateAccelerometer()` и `updateMagnetometer()` не возвращают значения, а просто обновляют глобальные переменные.
- 21 Установка константы `calibrationMode` в значение `True` переводит прибор в режим калибровки.
- 26 Нормальный рабочий режим, заключающийся в выводе показаний компаса с учетом наклона устройства.

Рабочий протокол модуля LSM303

Перед тем как приступить к изучению данного раздела, реализуйте предыдущий проект и выполните соответствующий программный код. Кто-то скажет: “Зачем вникать в тонкости его структурной организации?” Но такой подход далеко не самый дальновидный. Если вы планируете заниматься серьезными проектами, то обязательно изучите предыдущий материал.

Передача измеренных модулем LSM303 значений ведется через стандартный интерфейс I²C. Данный интерфейс имеет высокую степень структуризации, а потому его использование не вызывает особых трудностей, как это бывает, например, в случае стандарта SPI.

В шине I²C устройство LSM303 распознается как ведомое (подчиненное), а микроконтроллер (Arduino или Raspberry Pi) выступает в качестве ведущего устройства. Ведущее устройство является инициатором установки соединения с ведомым устройством, запрашивая у него значения. Обмен данными между устройствами через соединение I²C заключается в отправке и получении значений, определенных в технической документации к устройствам. Найти технические данные для используемого оборудования можно на сайте производителя или задать поиск по названию модели (в нашем случае LSM303DLH) в Интернет.

Чтобы лучше понять принципы управления устройствами через соединение I²C, обратитесь к листингу 8.3. Как и в случае подключения любых других устройств через шину I²C, в данном программном коде применяются шестнадцатеричные числа (`0xA == 10`) и операции битового сдвига (`0b01 << 1 == 0b10`). Детальное описание подробностей применения битовых операций в программном коде вынесено в отдельный раздел главы 8.

Вычисление направления по компасу

Электронный компас умеет определять направление расположения северного (магнитного) полюса. Он представляет эту информацию в виде трехмерного вектора $n = [n_x, n_y, n_z]$. Как вы видите, датчик исходно выводит трехмерные данные в правильном виде. Так зачем нужны все проделываемые нами векторные вычисления?

Векторная математика лежит в основе вычислений, проводимых нами для представления показаний в привычном для нас виде. Датчик будет выводить показания независимо от того, насколько глубоко ваши познания в математике. А вот приведенные ниже рассуждения потребуют от вас серьезных знаний в области высшей математики.

Мы привыкли к тому, что обычный компас показывает направление, представленное в виде числа, выраженного в градусах (в диапазоне от 0 до 360). Наш электронный компас указывает направление на север с помощью двухмерного значения (вектора).

Выводимые им данные показывают угол отклонения устройства от направления на север. Угол отклонения, как правило, выражается в градусах и увеличивается при повороте по часовой стрелке (вправо).

Для преобразования этого вектора в угол, выражаемый в градусах, используется векторное значение, передаваемое с акселерометра.

Все векторные величины, используемые в расчетах, привязываются к системе координат устройства, поэтому ось Y указывает в направлении действия датчика. Таким образом, вектор [0, 1, 0] указывает “прямо” перед датчиком.

Таблица 10.2. Переменные и определения в LM330

Arduino	Raspberry Pi	Значение	Описание
heading	heading	Целочисленное, 0–360 градусов. Угол между направлением на север и осью Y устройства	Угол отклонения, по часовой стрелке
mag_x, mag_y...	mag[]	Целочисленное со знаком, массив целочисленных со знаком	Трехмерный вектор, указывающий на север
acc_x, acc_y...	acc[]	Целочисленное со знаком, массив целочисленных со знаком	Трехмерный вектор, указывающий вверх (противоположно действию гравитации)
[0, 0, 1]			Вектор, направленный перпендикулярно датчику (ось Z)
[0, 1, 0]			Вектор, указывающий вдоль направления действия датчика (ось Y)
[1, 0, 0]			Вектор, указывающий перпендикулярно (вправо) направлению действия датчика (ось X)

Программы для Arduino и Raspberry Pi имеют одинаковую структуру. Для большей наглядности приведенные ниже рассуждения представлены для языка Python. Программный код для Arduino имеет такую же структуру, хотя названия переменных в нем могут несколько отличаться.

Сначала нужно определиться с направлениями: вверх и на север. Они представлены такими векторами:

- acc (противоположно гравитации, трехмерный);
- mag (направление на север, трехмерный).

Угол между этими векторами произвольный.

Калибровка проводится только при определении вектора, указывающего на север mag.

Теперь нужно вычислить направление на восток и представить его отдельным вектором. Этот вектор располагается в пространстве, перпендикулярном векторам

mag и acc. Это означает, что направление, противоположное действию гравитации, и направление на восток располагаются под углом 90 градусов. Подобным образом, направление на север и направление на восток отличаются углом в 90 градусов. Направление на восток проще всего вычисляется как векторное произведение.

```
e = cross(mag, acc)
e = normalize(e)
```

После нормализации переменная e представляет собой единичный вектор (с длиной, равной 1), указывающий на восток. Он, в свою очередь, перпендикулярен направлению, противоположному действию гравитации.

Векторы на север (n) и на восток (e) располагаются в плоскости NE. Эта плоскость располагается под определенным углом к плоскости поверхности устройства XY.

Векторы n и e необходимо выровнять по плоскости, перпендикулярной действию гравитации:

```
n = cross(acc, e)
```

На последнем этапе высчитывается показатель, выводимый пользователям на монитор. Это значение, выраженное в градусах, указывает угол, на который устройство отклонилось от направления на север. С точки зрения устройства нужно спроецировать вектор на плоскость NE, а затем рассчитать угол между проекцией и направлением на север.

```
dotE = dot(e, [0.0, -1.0, 0.0])
dotN = dot(n, [0.0, -1.0, 0.0])
headingRad = math.atan2(dotE, dotN)
headingDeg = headingRad / (2*math.pi) * 360
```

Последняя строка возвращает искомый угол отклонения, выраженный в градусах.

Эксперимент: переключатель на эффекте Холла

Переключатель на эффекте Холла (датчик Холла), показанный на рис. 10.9, регистрирует близкое присутствие магнитного поля.

Датчики Холла обычно применяются для подсчета количества оборотов или скорости вращения объектов. До изобретения технологии GPS многие спидометры работали на датчиках Холла.

Если в текущем проекте к датчику Холла поднести магнит, то программа выведет на монитор сообщение о срабатывании переключателя. Переключатели на эффекте Холла выпускаются в огромных количествах многими производителями. Мы в своих проектах часто используем устройство с красноречивым названием Hall Magnetic Sensor (номер в каталоге 141363), приобретенное в интернет-магазине <http://dx.com>. Этот датчик оснащен встроенным светодиодом, который загорается, если к нему поднести магнит. Его выходы обозначены несколько странно окрашенными проводами: черный — сигнальный (да, он подключается к выводу D2 микроконтроллерной платы, а не к GND); синий — земля; коричневый — питание (+5 В).

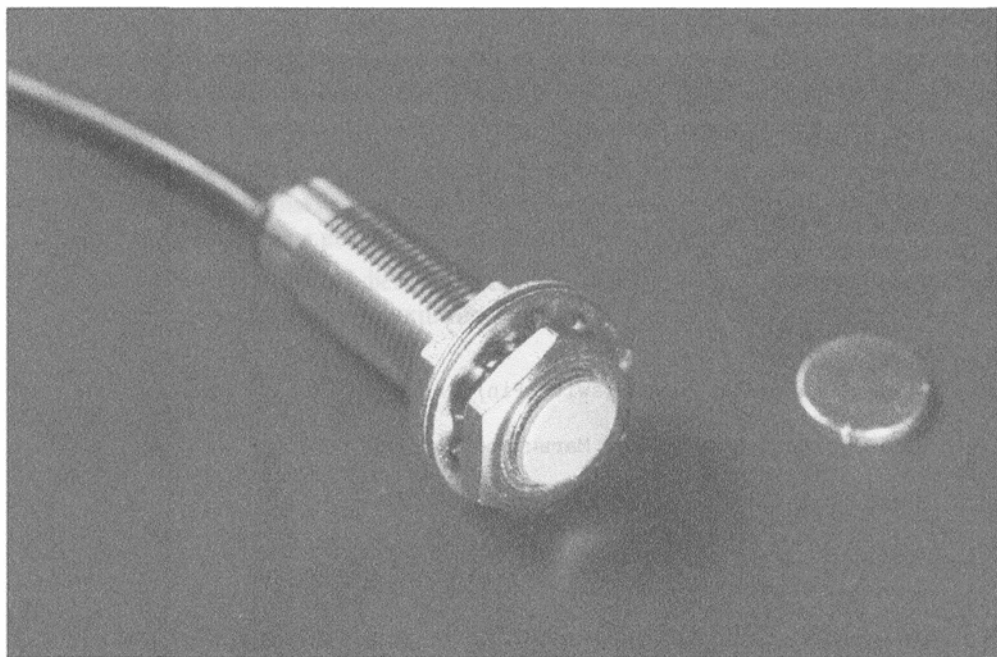


Рис. 10.9. Переключатель на датчике Холла

Подключение к Arduino и программа управления переключателем на эффекте Холла

На рис. 10.10 показана схема подключения датчика Холла к Arduino. Подключите правильно все провода и выполните программный код, представленный в листинге 10.7.

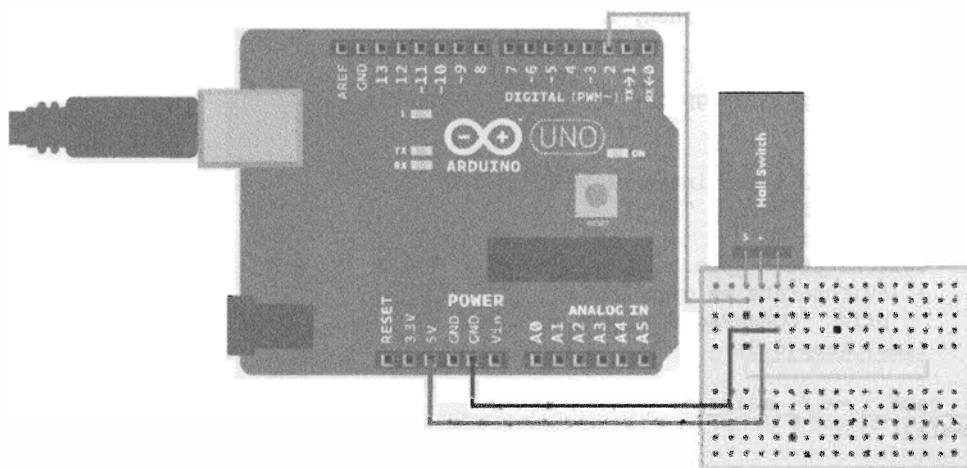


Рис. 10.10. Схема подключения переключателя на эффекте Холла к Arduino

Листинг 10.7. hall_switch.ino

```
// hall_switch.ino - вывод сообщения о регистрации магнитного поля
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int switchPin=2;

void setup() {
  Serial.begin(115200);
  pinMode(switchPin, INPUT);
  digitalWrite(switchPin, HIGH);
}

void loop() {
  int switchState=digitalRead(switchPin); // ❶
  if (switchState == LOW) {
    Serial.println("Внимание! Магнитное поле!");
  } else {
    Serial.println("Нет магнитного поля");
  }
  delay(50);
}
```

❶ Это простой цифровой переключатель, работающий подобно кнопке.

Подключение к Raspberry Pi и программа управления переключателем на эффекте Холла

На рис. 10.11 показана схема подключения переключателя на эффекте Холла к Raspberry Pi. Программный код, выполняемый после реализации этой простой электрической цепи, приведен в листинге 10.8.

Листинг 10.8. hall_switch.py

```
# hall_switch.py - вывод сообщения о регистрации магнитного поля
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio # ❶

def main():
    switchPin = 3 # has internal pull-up # ❷
    gpio.mode(switchPin, "in")
    while (True):
        switchState = gpio.read(switchPin) # ❸
        if(switchState == gpio.LOW):
            print "Обнаружено магнитное поле!"
            time.sleep(0.3)

if __name__ == "__main__":
    main()
```


- ❶ Убедитесь, что копия файла `botbook_gpio.py` располагается в том же каталоге, что и файл текущей программы. Эта библиотека и все файлы примеров книги доступны для загрузки по адресу, указанному во введении. Настройка портов GPIO в Raspberry Pi детально рассмотрена в главе 1.
- ❷ Чтобы избежать образования плавающего напряжения на выводе, воспользуйтесь подтягивающим резистором. К счастью, Raspberry Pi снабжается встроенным подтягивающим резистором для GPIO-выводов 2 и 3.
- ❸ Данное устройство на эффекте Холла представляет собой обычный цифровой переключатель, поэтому функционирует как кнопка.

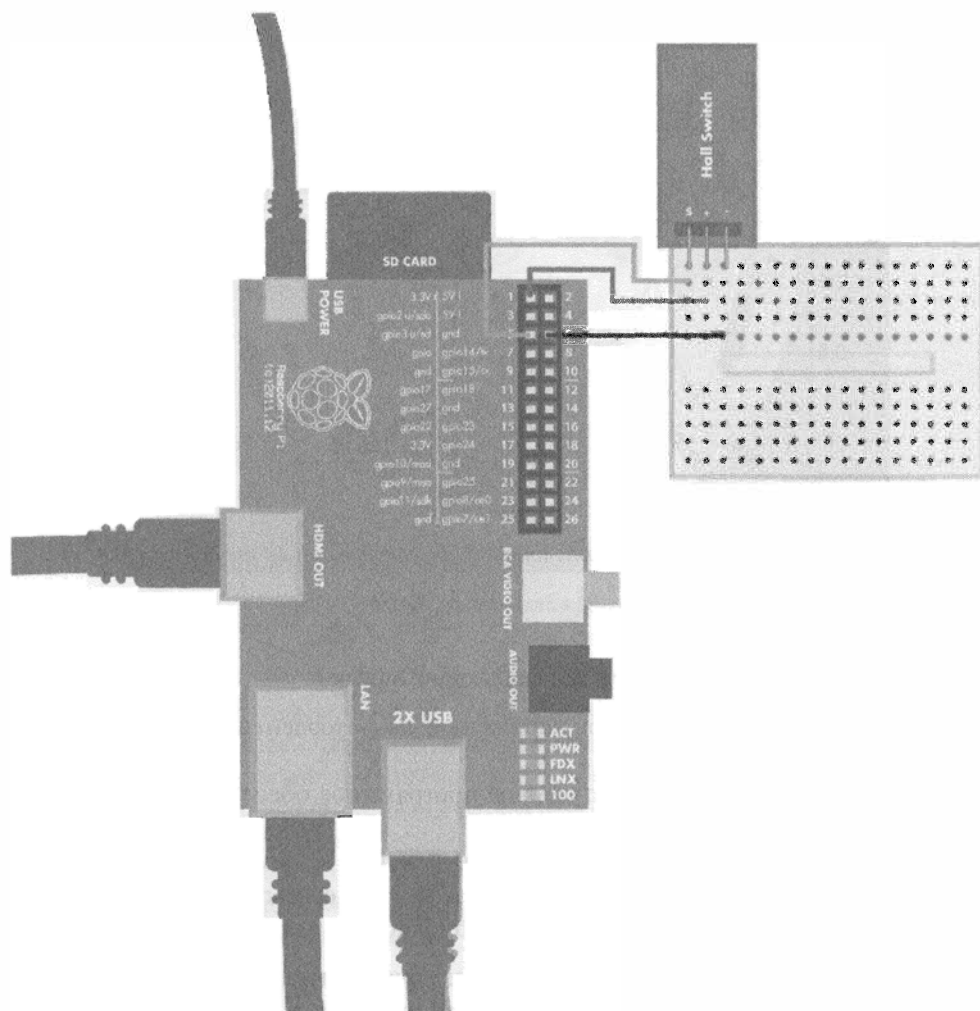


Рис. 10.11. Схема подключения переключателя на эффекте Холла к Raspberry Pi

Пилотный проект: интернет-мониторинг рабочего напряжения фотоэлемента

Давайте создадим на базе Raspberry Pi веб-сервер, отвечающий за удаленное отслеживание напряжения на фотоэлементе (солнечной панели) (рис. 10.12).

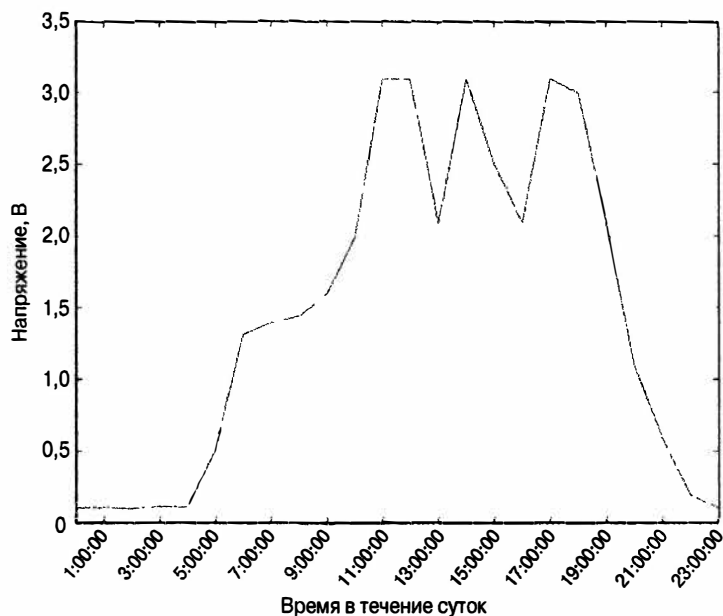


Рис. 10.12. График производительности фотоэлемента в течение суток

Получаемые навыки

В текущем проекте вы приобретете следующие навыки:

- научиться измерять выходное напряжение на фотоэлементе и регистрировать его на веб-сервере;
- запускать в Raspberry Pi популярное программное обеспечение (приложение веб-сервера);
- создавать отсроченные (таймированные) задачи с помощью планировщика cron, учитывающего время и дату даже в случае перезагрузки Raspberry Pi;
- рисовать графики в Python с помощью библиотеки matplotlib.

В больших, часто посещаемых веб-серверах применяются точно такие же принципы структурной организации, как те, что описаны далее. Мы преподаем курс по Apache и cron в Linux-системах для студентов, занимающихся разработкой и поддержкой веб-серверов, поэтому смело утверждаем, что приведенные ниже методики общеприменимы и ни в коей мере не являются особенными для микроконтроллерных и роботизированных систем.

Подключение фотоэлемента

Вы помните светильник IKEA, который мы использовали в качестве корпуса в проекте цветного купола (см. главу 7)? В данном проекте мы будем использовать фотоэлемент, являющийся частью его электронного оборудования. Если в финальном проекте главы 7 вы не использовали корпус от светильника из IKEA, то купите фотоэлемент в ближайшем магазине или выпаяйте его из старого, давно не используемого прибора. После снятия плафона светильника мы первым делом отпаяли от фотоэлемента красный провод, обведенный кружком на рис. 10.13.

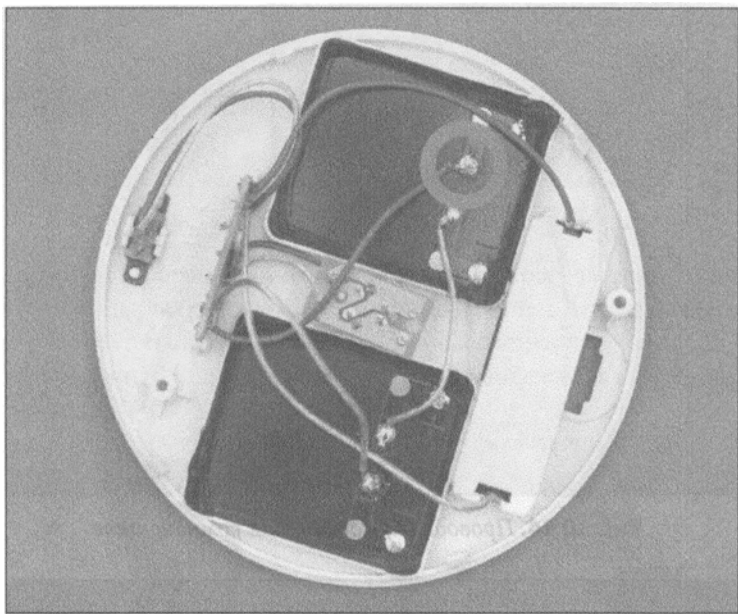


Рис. 10.13. Выпаяйте красный провод

На самом деле вам не нужно идти в IKEA, чтобы добыть детали для своих проектов. Имея в своей коллекции соответствующий фотоэлемент, просто добавьте его в электрическую цепь согласно схеме, показанной на рис. 10.17. Поскольку в текущем проекте применяется датчик тока с максимально поддерживаемым напряжением 13,6 В, убедитесь, что ваш фотоэлемент не подает на выход сигнал большего напряжения, чем то, на которое рассчитан датчик. Кроме того, приобретая новый фотоэлемент, вы избавите себя от утомительного паяния контактов. (У вас ведь еще нет паяльника?)

Разрежьте проволоочную перемычку, как показано на рис. 10.14, и впаяйте в полученный разрыв датчик тока и фотоэлемент (рис. 10.15). Конечный результат показан на рис. 10.16.

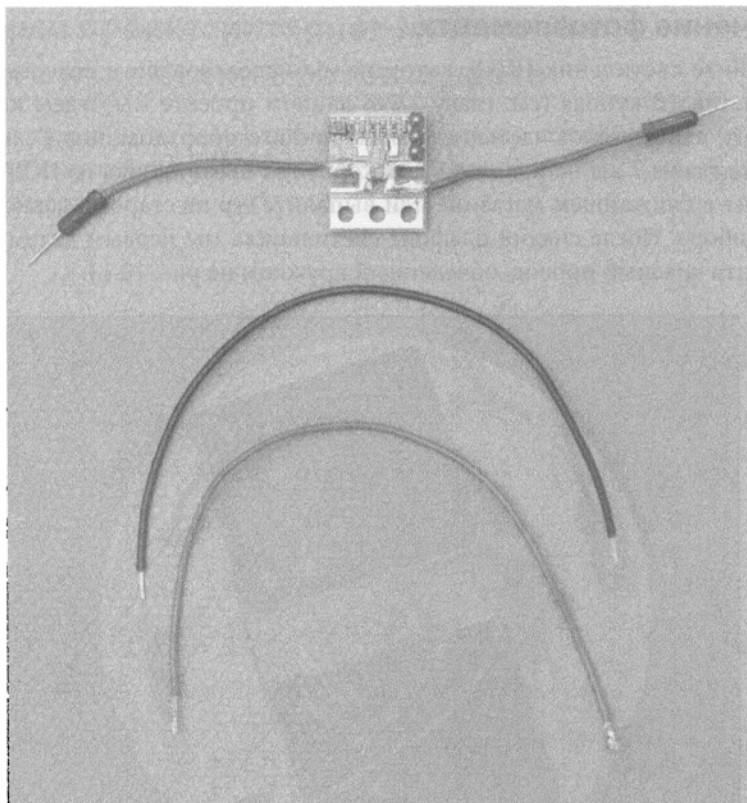


Рис. 10.14. Провода для соединения компонентов

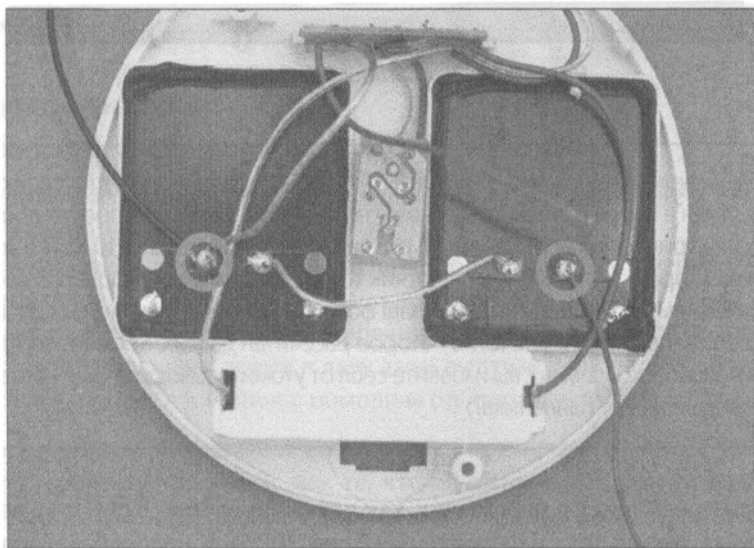


Рис. 10.15. Провода припаяны к модулю фотоэлемента

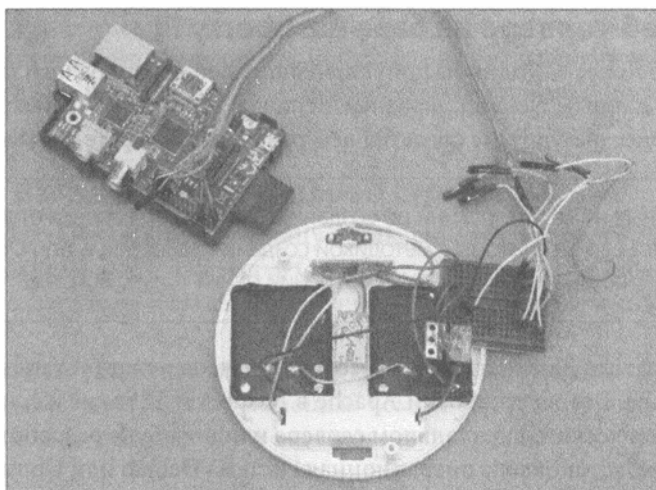


Рис. 10.16. Полностью скомпонованное устройство

Для тестирования оборудования воспользуйтесь программным кодом, рассмотренным в проекте подключения к микроконтроллерной плате датчика тока и напряжения AttoPilot. Проверить работоспособность фотоэлемента проще всего с помощью обычной фотовспышки. В нашем случае на выходе панели наблюдались значения напряжения от 1 до 3 В.

Подключите датчик тока к Raspberry Pi согласно схеме, показанной на рис. 10.17.

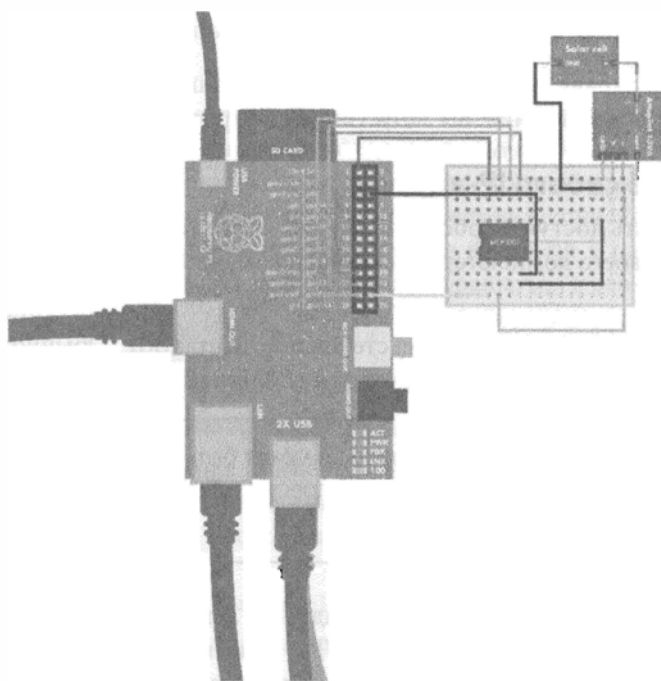


Рис. 10.17. Схема подключения фотоэлемента к Raspberry Pi (см. цветную вклейку)

Создание веб-сервера на базе Raspberry Pi

Приложение Apache — один из популярнейших веб-серверов на планете. Согласно аналитическим данным Netcraft, в разные периоды это программное решение становилось популярнее, чем другие средства веб-разработки вместе взятые.

Приведенные ниже инструкции по созданию веб-сервера содержат детальные пояснения. Если вы знакомы с методикой создания серверов в Linux, то можете упустить их, остановившись на рассмотрении одних только команд.

Чтобы превратить Raspberry Pi в веб-сервер, сначала нужно установить программный пакет Apache. Этапы установки Apache в Raspberry Pi такие же, как при инсталляции его на физическом, виртуальном сервере или ноутбуке разработчика ПО. Если в вашем ноутбуке установлена операционная система Debian или Ubuntu, то вы можете запросто протестировать на нем приведенные ниже команды.

На рабочем столе Raspbian запустите командную оболочку (значок приложения LXTerminal находится слева на рабочем столе).

Обновите список доступных программных пакетов и установите Apache.

```
$ sudo apt-get update
$ sudo apt-get -y install apache2
```

Веб-сервер установлен и запущен. Проверим его работоспособность с помощью браузера, например Midori. Значок для него также находится слева на рабочем столе Raspbian. Введите такой адрес:

```
http://localhost/
```

Вы видите на экране содержимое указанной страницы? Неужели работает? Примите искренние поздравления: вы запустили веб-сервер!

Определение своего IP-адреса

Чтобы получить доступ к собственному веб-серверу с другого компьютера, определите IP-адрес своего компьютера следующей командой:

```
$ ifconfig
```

Ищите в выведенном на экран тексте строку `inet addr`. Обычно она находится под названием установленного соединения `eth0` или `wlan0`.

Внешний IP-адрес вашего компьютера не может быть 127.0.0.1. Это адрес локального сетевого ресурса; по такому адресу компьютер может обращаться к себе только в локальной сети.

Попробуйте перейти по найденному адресу. Запустите Midori и введите сначала `http://`, а затем добавьте в конце URL найденный ранее IP-адрес, например `http://10.0.0.1`. В вашем случае будет другой текст, но принцип адресации сохраняется.

Такой адрес прекрасно работает в локальной сети. Таким же образом вы легко найдете IP-адреса других компьютеров или ноутбуков вашей локальной сети. Вы смогли получить доступ к собственному веб-серверу с другого локального компьютера или ноутбука?

Создание в Raspberry Pi домашней страницы

Создать и поддерживать веб-страницу очень просто, особенно если это начальная (домашняя) страница вашего сервера. Существуют два подхода к решению этой задачи: позволить создавать начальные страницы пользователям и создавать их самостоятельно.

Чтобы разрешить создавать начальные страницы пользователям, вам потребуется изменить системные настройки с помощью команды `sudo`. Приведенные ниже команды загружают модуль `userdir` и перезапускают `Apache`.

```
$ sudo a2enmod userdir
$ sudo service apache2 restart
```

Теперь нам нужно создать в Raspberry Pi каталог для начальной страницы. Для этого вам не нужна команда `sudo`. Будьте внимательны при вводе названия каталога для начальной страницы, `public_html`.

```
$ cd /home/pi/
$ mkdir public_html
```

Добавьте в этот каталог текстовый вариант страницы:

```
$ echo "botbook">/home/pi/public_html/hello.txt
```

Посетите страницу с помощью браузера Midori:

```
http://localhost/~pi/hello.txt
```

Чтобы удостовериться в получении доступа к собственной веб-странице, введите вместо `localhost` IP-адрес своего компьютера (ноутбука). Вы также можете попробовать получить доступ к странице из любого другого компьютера локальной сети.

На этом наше превращение Raspberry Pi в веб-сервер закончено! У вас даже есть начальная страница, размещенная на нем. Давайте вернемся к рассмотрению программного кода, из которого будем периодически обращаться к веб-странице и выводить на нее важные данные.

Подключение к Raspberry Pi и программа мониторинга рабочих параметров фотоэлемента

В текущем программном коде применяется `matplotlib`, бесплатно распространяемая библиотека Python, предназначенная для визуализации математических данных. Установите ее с помощью таких команд:

```
$ sudo apt-get update
$ sudo apt-get -y install python-matplotlib
```

Еще одно эффективное средство построения графиков — это Plotly.

Однократное выполнение программы `voltage_record.py` приводит к созданию новой точки данных и построению графика (с последующим завершением работы). Цикличность выполнения операций в программе не обеспечивается, поскольку сама программа будет перезапускаться каждые пять минут с помощью планировщика `cron`.

История измерений хранится в файле `/home/pi/record.csv`.

Сгенерированный программой график сохраняется в виде файла `/home/pi/public_html/history.png`. Поскольку вы уже установили сервер `Apache` в своем компьютере, можете публиковать данные по адресу `http://localhost/~pi/history.png`. Чтобы сделать доступными ваши данные для других компьютеров локальной сети, замените `localhost` на IP-адрес платы `Raspberry Pi`.

Листинг 10.9. `voltage_record.py`

```
# voltage_record.py - регистрация напряжения на выходе
# фотоэлемента и представление его в виде временного графика
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import os
import matplotlib # ❶
matplotlib.use("AGG") # ❷
import matplotlib.pyplot as plt
import numpy
from datetime import datetime
from datetime import date
from datetime import timedelta
import attopilot_voltage # ❸
import shelve

historyFile = "/home/pi/record" # ❹
plotFile = "/home/pi/public_html/history.png" # ❺

def measureNewDatapoint():
    return attopilot_voltage.readVoltage() # ❻

def main():
    history = shelve.open(historyFile) # ❼

    if not history.has_key("x"): # ❽
        history["x"] = [] # ❶
        history["y"] = []

    history["x"] += [datetime.now()] # ❶
    history["y"] += [measureNewDatapoint()] # ❶

    now = datetime.now() # ❶
    sampleCount = 24 * 60 / 5
    history['x'] = history["x"][-sampleCount:]
    history['y'] = history["y"][-sampleCount:]

    plot = plt.figure() # ❶
    plt.title('Solar cell voltage over last 24h')
```



```
plt.ylabel('Voltage V')
plt.xlabel('Time')
plt.setp(plt.gca().get_xticklabels(), rotation=35)
plt.plot(history["x"], history["y"], color='#4884ff') # 10
plt.savefig(plotFile) # 11
history.close() # 12
```

```
if __name__ == '__main__':
    main()
```

- 1 Библиотеку нужно установить в системе так, как описывалось выше.
- 2 Инструмент AGG (Anti-Grain Geometry) — это прекрасное решение в matplotlib для представления графиков растровыми изображениями (формата PNG).
- 3 Файл программы, создаваемый в предыдущем проекте, необходимо сохранить в подкаталоге attopilot_voltage/ каталога текущей программы (файл voltage_record.py).
- 4 Это вспомогательный файл, в котором сохраняются полученные числовые значения. Функция shelve, применяемая далее, заполняет этот файл данными.
- 5 Создание графика (растрового PNG-изображения). Для публикации его на веб-сервере поместите его в каталог, используемый Apache для хранения данных.
- 6 Команда проведения измерений. Чтобы измерить другие электрические характеристики, измените команду соответствующим образом.
- 7 Открытие файла истории измерений. При необходимости он создается автоматически.
- 8 Если значения переменных не представлены в файле истории...
- 9 ...то они в него добавляются. Файл истории представляет собой каталог данных — значения x и y для точек графика представляются отдельными списками.
- 10 Назначение переменной x временной метки. Обратите внимание на то, что вам не придется заботиться о форматировании даты и синтаксическом анализе выражения, представляющего время на графике.
- 11 Сохранение измеренного значения в переменной y. Каталоги для x и y сильно отличаются, но значения связываются между собой общим индексом. Например, данные в точке y[12] связаны со временем, представленным в точке x[12].
- 12 В текущем временном отрезке отсутствуют точки, полученные больше 24 часов назад. Самый простой способ заключается в сохранении только определенного количества точек данных, например 100.
- 13 Создание новой зависимости в matplotlib.
- 14 Построение графика.
- 15 Сохранение графика в виде PNG-изображения.
- 16 Закрытие файла истории делает невозможным сохранение новых точек данных на диске.

Отсроченные задания и планировщик cron

Планировщик cron идеально подходит для выполнения заданий по расписанию. Даже если вы завершили работу Raspberry Pi, планировщик cron продолжит отслеживать назначенные ранее задания после следующей загрузки.

Сначала удостоверьтесь, что можете выполнить программу Python из командной оболочки, указывая полный путь ее расположения на диске. Укажите точное расположение файла `voltage_record.py` (листинг 10.9).

```
$ python /home/pi/voltage_record/voltage_record.py
```

Вы забыли, в каком каталоге хранится файл `voltage_record.py`? Выполните простую команду: `cd /home/pi; find -iname voltage_record.py`.

Если программа выполняется правильно, то назначьте ее в качестве задания в планировщике cron. Измените настройки файла cron для вашего пользователя с помощью такой команды:

```
$ crontab -e
```

Добавьте в конец файла следующие строки.

```
*/5 * * * * /home/pi/voltage_record/voltage_record.py
*/1 * * * * touch /tmp/botbook-cron
```

Сохраните файл. Если вы редактировали файл командой nano, то нажмите комбинацию клавиш `<Ctrl+X>`, затем клавишу `<Y>`, а потом `<Enter>` или `<Return>`.

Первая из добавляемых строк указывает cron запускать программу каждые следующие 5 минут, не учитывая смену часов, дней недели, месяцев и времен года.

Следующей (второй) строкой дается указание каждую минуту создавать пустой файл `/tmp/botbook-cron`. Она служит исключительно для тестирования рабочего состояния cron, и вы можете удалить ее. Подождите минуту и проверьте, создан ли файл, как предполагалось выше.

```
$ ls /tmp/botbook*
/tmp/botbook-cron
```

Вам ведь знакома команда `ls`? Замечательно! Вы успешно справились с назначением задания в cron.

Если выполнение задания cron приводит к сильному загромождению журнала, то добавьте к команде в файле планировщика cron суффикс `>/dev/null`. Тем самым вы скроете в журнале все, что команда выводит в результате своего выполнения.

Когда бы вам ни понадобилось выполнять в Raspberry Pi повторяющиеся действия, обратитесь за помощью к cron. Даже в случае отключения питания сведения о назначенном задании будут восстановлены после последующей загрузки Raspberry Pi.

Что дальше?

Вот вы и научились “распознавать” электричество. Ваши устройства теперь в состоянии измерить электрический ток и напряжение таких величин, что их непосредственная подача на микроконтроллерную плату вызовет гарантированное повреждение оборудования. Магнитные поля легко регистрируются датчиком Холла, а с помощью электронного компаса можно измерить направление линий магнитного поля в трехмерном пространстве даже с учетом наклона прибора.

Проект отслеживания напряжения на фотоэлементе и публикации полученных данных на веб-сервере можно адаптировать под любые цели. Изменяя способ и расчетные функции, вы сможете регистрировать данные с любого датчика. В нашем проекте данные публикуются для общего обозрения в локальной сети. Попробуйте защитить данные паролем или используйте технологию SSH для создания защищенного соединения при отправке данных на сервер.

В следующей главе мы плавно перейдем от изучения электрических и магнитных полей к управлению звуковыми волнами и снабдим наши устройства средствами распознавания и воспроизведения звука.

Звуковые волны, распространяемые в воздухе, характеризуются областями сжатия и разрежения молекул, входящих в его состав. Звуковые колебания можно записывать и воспроизводить самыми разными способами.

Как любой другой компьютер, Raspberry Pi поддерживает возможность записи и обработки звуковых данных. В следующем проекте мы воспользуемся быстрым преобразованием Фурье (Fast Fourier Transform — FFT) для разделения звукового потока на несколько составляющих. Проведенные вычисления позволят определить частотные характеристики звуковых волн. В Arduino мы воспользуемся микрофоном для измерения уровня громкости звука.

Эксперимент: запись звука и настройка уровня громкости

Звук распознается и записывается микрофоном. В нашем первом проекте мы научимся получать данные об уровне звука, регистрируемого микрофоном, и выводить его значение на монитор.

Уровень громкости — это важнейшая характеристика звуковых колебаний, которая используется во многих проектах. Вам может понадобиться ограничить громкость звука, усреднить ее во времени или же измерить минимальное и максимальное значения. В последнем проекте главы вы научитесь ограничивать уровень громкости звука. В своих экспериментах мы использовали микрофон на коммутационной плате (BOB-09964) компании SparkFun (рис. 11.1).

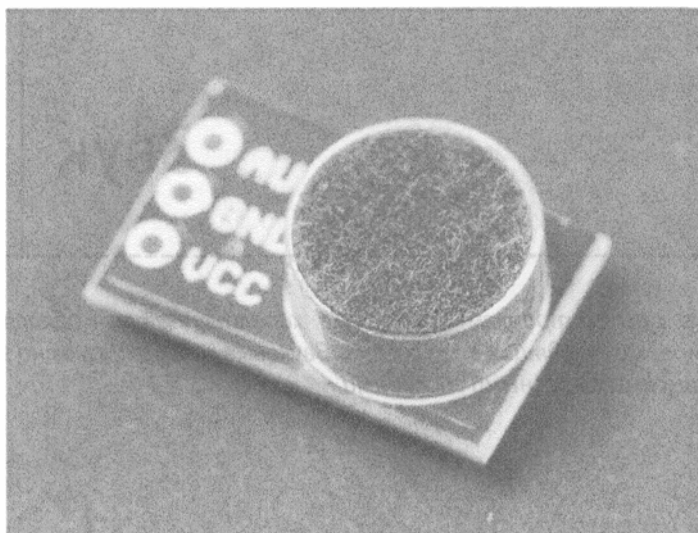


Рис. 11.1. Микрофон на коммутационной плате

Подключение к Arduino и программа управления микрофоном

На рис. 11.2 приведена схема подключения платы микрофона к Arduino. После соединения всех показанных на ней перемычек выполните программный код из листинга 11.1.

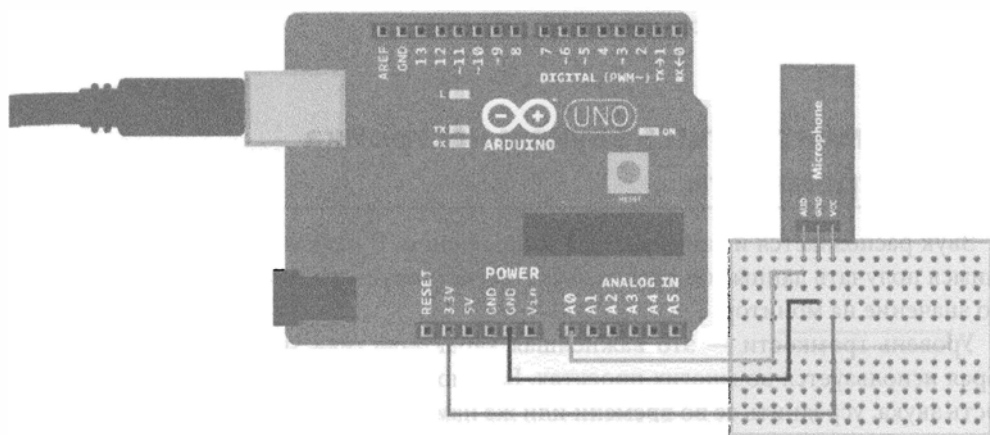


Рис. 11.2. Схема подключения микрофона к Arduino

Листинг 11.1. `microphone.ino`

```
// microphone.ino - вывод извещения и значения уровня громкости
// на монитор последовательного порта
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```

const int audioPin = A0;
const int sensitivity = 850;

void setup() {
  Serial.begin(115200);
}

void loop() {
  int soundWave = analogRead(audioPin); // ❶
  Serial.println(soundWave);
  if (soundWave > sensitivity) { // ❷
    Serial.println("Громко!");
    delay(500);
  }
  delay(10);
}

```

- ❶ Сигнал с микрофона считывается так же, как и с любого другого аналогового резистивного датчика. Функция `analogRead()` возвращает необработанные значения в диапазоне от 0 до 1023 — чем больше значение, тем выше уровень громкости. Способ подключения микрофона и программный код управления заимствован из проекта подключения потенциометра.
- ❷ Вывод сообщения при регистрации звука, уровень громкости которого выше заданной чувствительности.

Подключение к Raspberry Pi и программа управления микрофоном

Схема подключения микрофона к Raspberry Pi приведена на рис. 11.3. Программный код, выполняемый после правильного подключения всех перемычек, приведен в листинге 11.2.

Листинг 11.2. `microphone.py`

```

# microphone.py - считывание звука с аналогового входа
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

```

```

import time
import botbook_mcp3002 as mcp # ❶

def readSound(samples):
    buff = [] # ❷
    for i in range(samples): # ❸
        buff.append(mcp.readAnalog()) # ❹
    return buff

def main():
    while True:
        sound = readSound(1024) # ❺
        print(sound)
        time.sleep(1) # c

```

```
if __name__ == "__main__":
    main()
```

- ❶ Файл библиотеки `botbook_mcp3002.py`, подключаемой в нашем проекте, должен располагаться в том же каталоге, что и файл текущей программы. Кроме того, вам потребуется установить библиотеку `spidev`, импортируемую библиотекой `botbook_mcp3002`. Подробно об этом см. в главе 3 и в комментариях к программному коду файла `botbook_mcp3002.py`.
- ❷ Объявление нового пустого списка.
- ❸ Повторение команд, приведенных ниже, так, что `i` присваивается каждое из значений. В нашей программе это выглядит так: `for i in [0, 1, 2 ... 1023]:.`
- ❹ Считывание значения с микрофона. Добавление нового значения в список (`buff`).
- ❺ Считывание 1024 образцов и получение списка возвращенных значений.

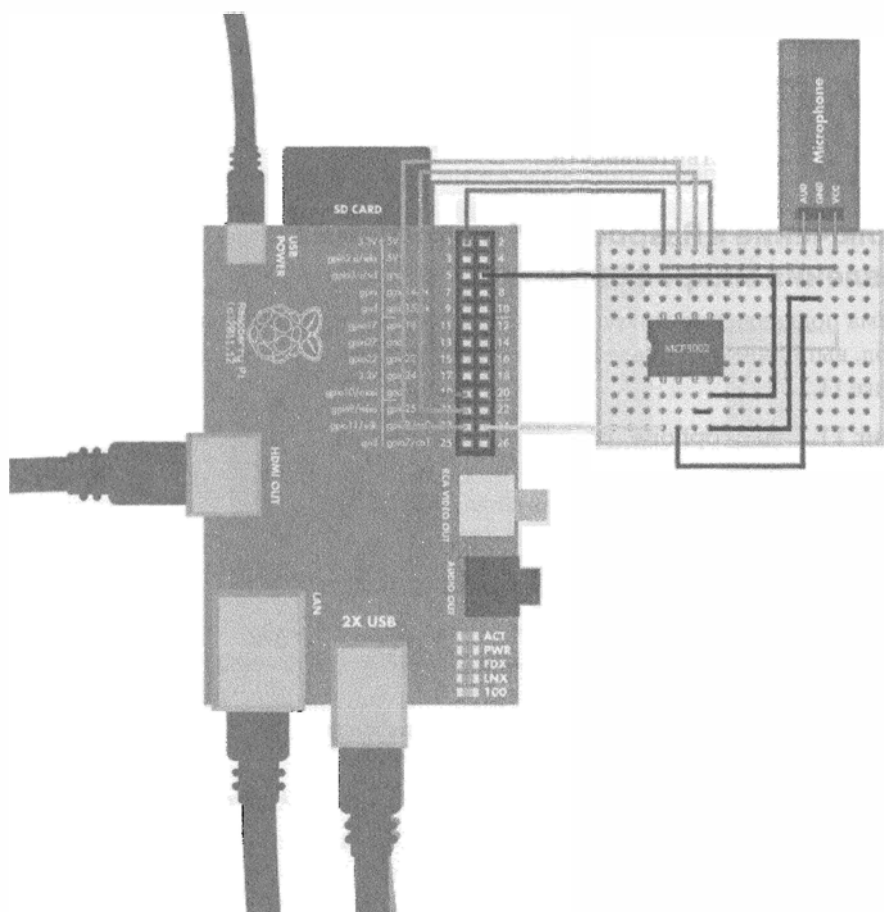


Рис. 11.3. Схема подключения микрофона к Raspberry Pi (см. цветную вклейку)

Эксперимент в окружающей среде: услышать падение булавки

Насколько хорошо вы слышите? Сможете услышать звук упавшей булавки? Попробуем ответить на этот вопрос, заодно протестировав чувствительность микрофона. Подключите микрофон к Arduino в соответствии с инструкциями, приведенными в предыдущем разделе, и выполните соответствующий программный код. Расположите микрофон (датчик звука) на твердой поверхности, например, на деревянном столе или паркетном полу, так, чтобы перед ним было достаточно пространства для падения булавки, бросаемой с небольшой высоты.

Сделайте все возможное, чтобы минимизировать фоновый шум. Уменьшите чувствительность микрофона в программном коде, но так, чтобы сообщение о высоком уровне громкости не появлялось в полной тишине. Тщательно подберите граничное значение, при котором регистрируется малейший звук, но извещение не появляется при наличии только фонового шума.

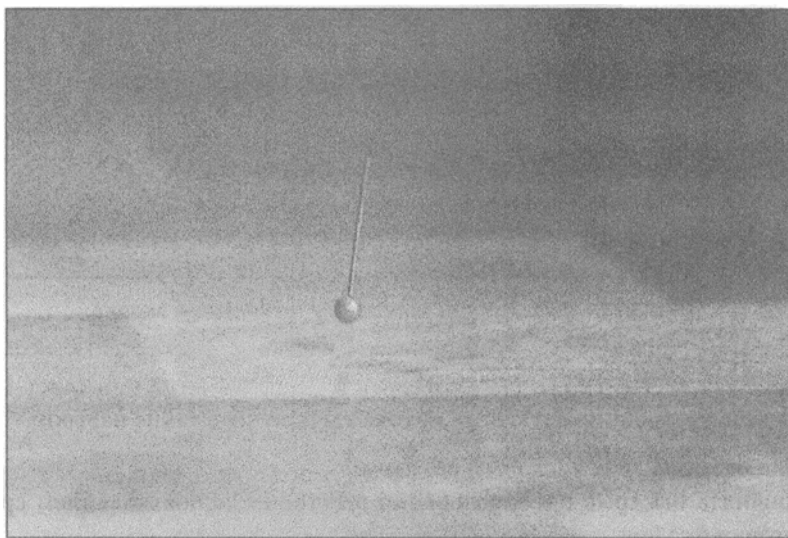


Рис. 11.4. Булавка, падающая на деревянный пол

Откройте окно монитора последовательного порта в среде разработки Arduino. Вы видите на экране сообщение о регистрации звука? Если да, то это означает, что микрофон распознает звук падения булавки. Если извещение не выводится, то еще раз убедитесь в отсутствии в помещении сильного фонового шума и бросьте булавку несколько ближе к микрофону; при необходимости попробуйте повысить чувствительность микрофона.

Чувствительность к слабым звукам находится в прямой зависимости от уровня громкости посторонних (фоновых) шумов. Выключите в комнате все воспроизводящие звук и шум устройства и только после этого поэкспериментируйте с настройкой чувствительности микрофона в программном коде. Удалось зарегистрировать падение булавки по издаваемому ею звуку?

Пилотный проект: визуализация звука через HDMI-порт

У вас уже есть графический эквалайзер с диагональю 50 дюймов? Еще нет, но есть несбыточная мечта о его приобретении? Будьте уверены, мы поможем вам ее реализовать! В данном проекте мы будем анализировать звуковые данные, считываемые микроконтроллерной платой (в нашем случае Raspberry Pi), и выводить их характеристики на 50-дюймовый телевизор. Частоты, представленные в регистрируемых звуковых данных, будем выводить в виде красочных анимированных столбцов (рис. 11.5).

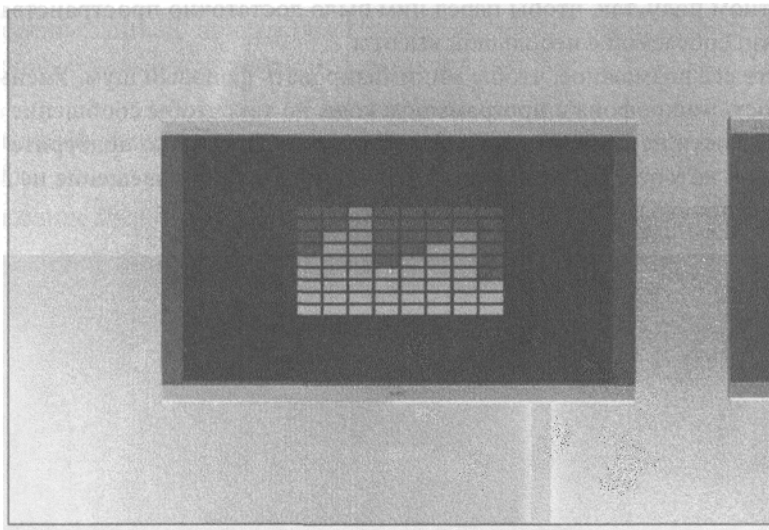


Рис. 11.5. Графический эквалайзер на большом экране

Получаемые навыки

В проекте визуализации звукового потока вы получите такие навыки:

- анализировать звук числовым образом;
- выполнять быстрые преобразования в Python с использованием средств библиотеки SciPy;
- определять частоты, присутствующие в звуковом потоке, с помощью быстрого преобразования Фурье (FFT).

Кроме того, мы вспомним используемые ранее методы программирования, требующие применения инструментов библиотеки pyGame, и способы вывода видеозображения формата FullHD через HDMI-выход микроконтроллерной платы (например, на телевизор или проектор; см. главу 6)

Включение последовательного порта в Raspberry Pi

Чтобы воспользоваться последовательным портом в Raspberry Pi, вам сначала нужно его освободить. По умолчанию он используется интерпретатором входа в систему. Физическое подключение к порту реализуется с помощью кабеля последовательной передачи данных:

```
$ sudoedit /etc/inittab
```

Сделайте комментарием последнюю строку файла, которая указывает захватить последовательный порт. Чтобы закомментировать строку, введите в ее начале символ решетки, указывающий игнорировать ее при выполнении файла:

```
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Перезагрузите Raspberry Pi.

Подключение к Raspberry Pi и программа визуализации звука

Давайте проведем начальную настройку системы. В Raspberry Pi запустите терминал и выполните в нем такие команды:

```
$ sudo apt-get update
```

```
$ sudo apt-get -y install python-pygame python-numpy
```

На рис. 11.6 показана схема подключения оборудования, используемого в текущем проекте Raspberry Pi. После подключения всех показанных на ней устройств выполните программный код, приведенный в листинге 11.3.

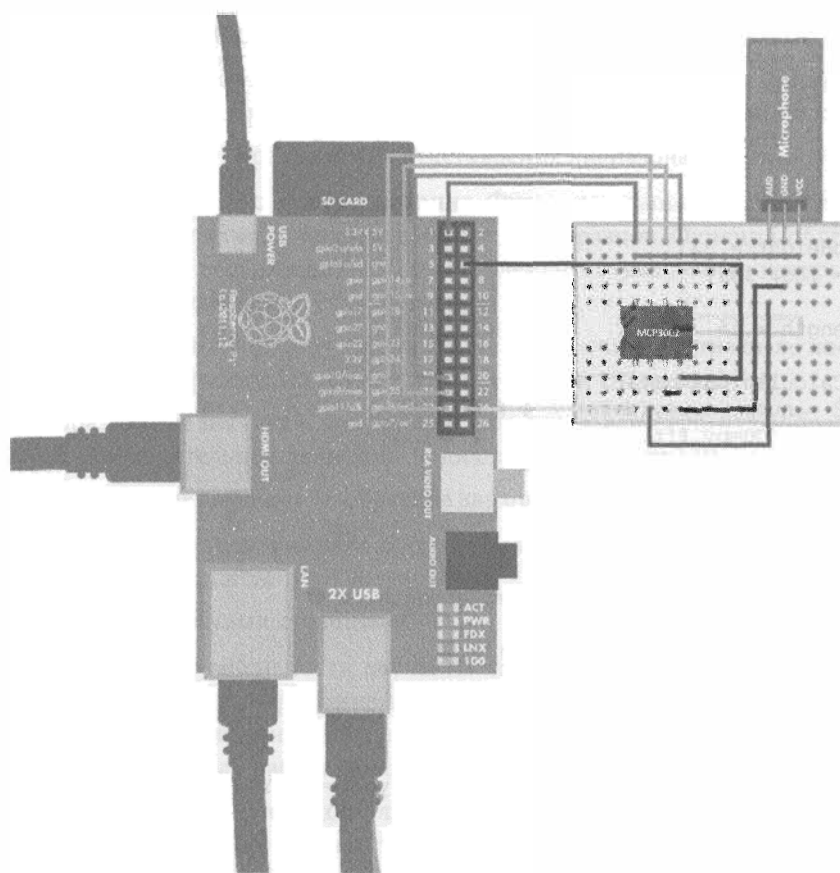


Рис. 11.6. Схема подключения микрофона к Raspberry Pi для визуализации звука (см. цветную вклейку)

Листинг 11.3. equalizer.py

```
# equalizer.py - графический эквалайзер, предназначенный для
# визуализации звука, регистрируемого микрофоном
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import pygame # sudo apt-get -y install python-pygame
import math
import numpy # sudo apt-get -y install python-numpy

import microphone # ❶
from pygame.locals import *

pygame.init()

width = 800
height = 640

size = width, height
background = 0, 0, 0

screen = pygame.display.set_mode(size, pygame.FULLSCREEN)
fullBar = pygame.image.load("equalizer-full-small14.jpg") # ❷
emptyBar = pygame.image.load("equalizer-empty-small14.jpg")
clock = pygame.time.Clock()
pygame.mouse.set_visible(False)
mainloop = True

barHeight = 36
barWidth = 80
barGraphHeight = 327
barPos = [55, 130]

sampleLength = 16

def fftCalculations(data): # ❸
    data2 = numpy.array(data) / 4 # ❹
    fourier = numpy.fft.rfft(data2) # ❺
    ffty = numpy.abs(fourier) # ❻
    ffty = ffty / 256.0 # ❼
    return ffty

while mainloop:
    buff = microphone.readSound(sampleLength) # ❶

    barData = fftCalculations(buff) # ❷

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            mainloop = False
        if (event.type == KEYUP) or (event.type == KEYDOWN):
            if event.key == K_ESCAPE:
                mainloop = False
    screen.fill(background)
    # Представление столбцов
    for i in range(8): # ❸
```

```

bars = barData[i+1] # ⑩
bars = int(math.floor(bars*10)) # ⑪
if bars > 10:
    bars = 10
bars -= 1
screen.blit(emptyBar, (barPos[0]+i*(barWidth+10),
↳barPos[1]), (0, 0, barWidth, barHeight*(10-bars)))
if bars >= 0:
    barStartPos = (barPos[0] + i * (barWidth + 10),
    barPos[1] + barGraphHeight - barHeight * bars + 6)
    barSourceBlit = (0, barGraphHeight - barHeight *
↳bars+6,
    barWidth, barHeight*bars)
    screen.blit(fullBar, barStartPos, barSourceBlit) # ⑫
pygame.display.update()

```

- ① Программа `microphone.py`, рассмотренная в начале главы, должна располагаться в том же каталоге, что и текущая программа (`equalizer.py`).
- ② Библиотека `pyGame` позволяет загружать JPG-изображения. Создаются такие изображения во многих графических редакторах, например `Inkscape`, `Gimp`, `Photoshop` и даже `Illustrator`.
- ③ При вызове этой функции в основной программе считанные данные будут представляться 16 образцами, представляющими диапазон 0–1024.
- ④ Масштабирование образцов для представления диапазона значений 0–256.
- ⑤ Выполнение быстрого преобразования Фурье над записанными звуковыми образцами. Получение диапазонов частот для образцов. Функция `rfft()` возвращает массив, содержащий $16/2+1$ (9) элементов, каждый из которых соответствует определенной звуковой частоте.
- ⑥ Отбрасывание мнимой части чисел, чтобы на их основе можно было построить график или диаграмму. Например, `abs(1+1j)` округляется до 1,4.
- ⑦ Преобразование значений, полученных в результате быстрого преобразования Фурье, в относительные величины (в диапазоне от 0,0 до 1,0).
- ⑧ Считывание микрофоном 16 образцов звуковых данных.
- ⑨ Сохранение значений звуковых данных после быстрого преобразования Фурье в переменной `barData`.
- ⑩ Для каждого из восьми столбцов (0–7)...
- ⑪ ...определение интенсивности звучания соответствующих частот. Поскольку нумерация задается выражением `i+1`, первый элемент (0) не рассматривается. Этот не рассмотренный элемент представляет постоянную составляющую, среднее значение для набора положительных и отрицательных пиков.
- ⑫ Вычисление высоты столбцов или количества элементов, представляющих их (значение 1,0 или 100% соответствует максимально высокому столбцу, состоящему из девяти элементов).
- ⑬ Отображение столбцов необходимой высоты на экране.

Быстрое преобразование Фурье

Быстрое преобразование Фурье применяется в настоящем проекте для определения звуковых частот, присутствующих в записываемом сигнале. Впоследствии на основе полученных данных строится диаграмма графического эквалайзера. Они также могут использоваться в анализаторах спектра звукового сигнала и при выводе данных на экране осциллографа.

Для простоты рассмотрим всего два тона с частотами 5 и 40 Гц. Мы заведомо выбрали недостоверные частоты, ведь слышимые нами звуки представляются колебаниями с частотами от 20 Гц до 20 000 КГц.

На рис. 11.7 показана временная характеристика звуковых колебаний на частоте 5 Гц. Вдоль вертикальной оси указывается амплитуда колебаний или степень сжатия и разрежения вещества в воздухе. На рис. 11.8 вы можете видеть такую же характеристику, но для колебаний с частотой 40 Гц.

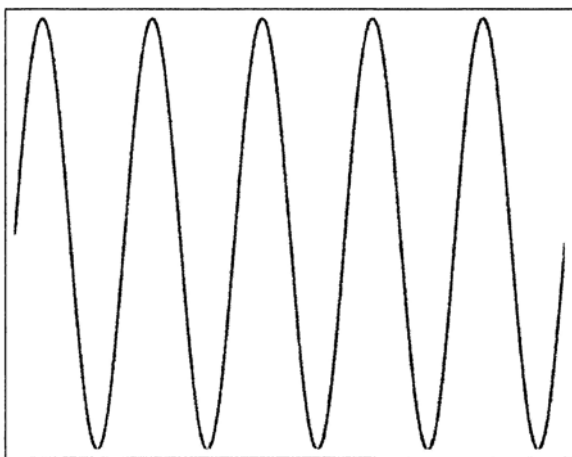


Рис. 11.7. Низкая звуковая частота. Синусоида с частотой 5 Гц

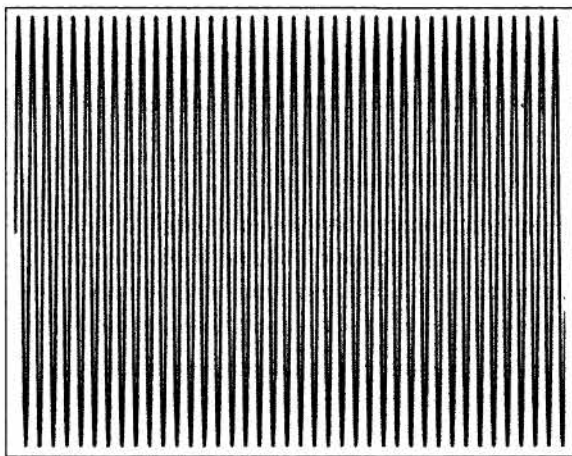


Рис. 11.8. Высокая звуковая частота. Синусоида с частотой 40 Гц

При одновременном воспроизведении звук будет представляться одной звуковой волной сложной формы, подобной показанной на рис. 11.9. Как видите, колебания низкой частоты задают общую форму волны, а более высокочастотные колебания отвечают за периодические пульсации низкочастотного сигнала. Наложение двух волн разной частоты всегда приводит к подобному результату, особенно если их частоты сильно отличаются.

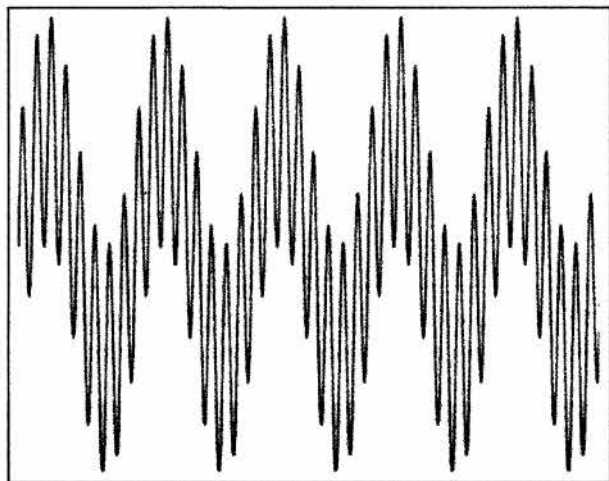


Рис. 11.9. Наложение двух волн

При записи музыки и звука они всегда представляются подобной комбинированной волной, но гораздо более сложной формы. Именно такое исходное представление звуковых данных впоследствии кодируется в такие известные звуковые форматы, как mp3, wav и ogg. Возникает справедливый вопрос: а можно ли разложить такую комбинированную звуковую волну на составляющие ее компоненты? Несомненно! И поможет нам в этом преобразование Фурье!

В нашем проекте мы как раз используем этот способ преобразования для разложения звука на частотные составляющие. Как вы могли заметить при анализе программного кода текущего проекта, все необходимые вычисления выполняются специальной функцией, в качестве аргументов которой подставляются данные, считываемые микрофоном (рис. 11.10).

В результате выполнения быстрого преобразования Фурье вы получите частоты, которыми представлен исходный звуковой сигнал, 5 и 40 Гц. Таким образом, можно разложить на отдельные гармоники любые звуковые данные. На рис. 11.11 показан графический эквалайзер в процессе работы.

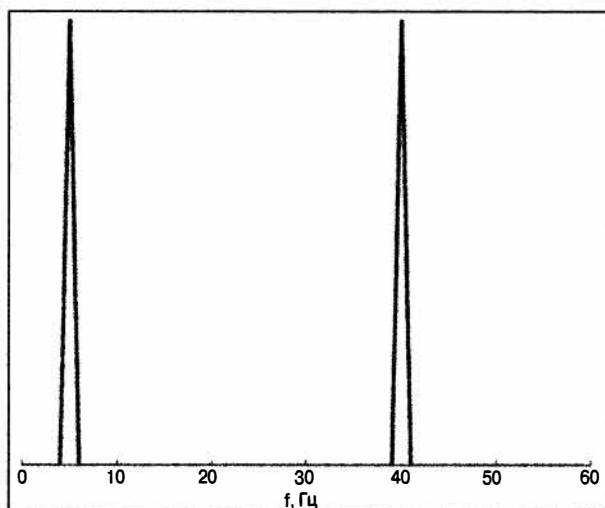


Рис. 11.10. Быстрое преобразование Фурье позволяет определить частоты составляющих сигнал звуковых волн

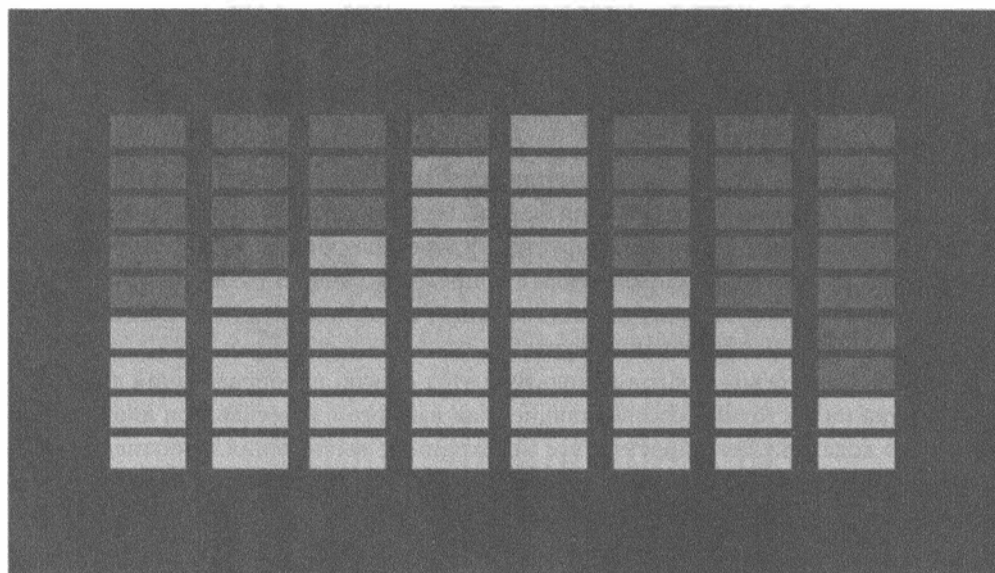


Рис. 11.11. Эквалайзер

Что дальше?

На этом мы закончим с обработкой звуковых данных. В этой главе вы научились распознавать в Arduino различные уровни громкости звучания, а также задавать порог чувствительности, с которого микрофон начинает регистрировать звук. На базе Raspberry Pi мы создали устройство анализа частотных характеристик звукового

потока. В процессе реализации проекта вы закрепили свои знания средств библиотеки `pygame` и применили быстрое преобразование Фурье для разложения звуковой волны на отдельные частотные компоненты.

Нам осталось только осмотреться вокруг и попробовать оценить то, что происходит каждый день за окном. Погода и климатические изменения, происходящие каждый день, станут темой следующей главы. Мы попробуем детально подойти к изучению характеристик среды обитания, начиная с определения температуры и влажности помещения и кончая определением погодных условий на улице в любое время суток.

Погода все еще неподвластна человеку. Пока неподвластна.

Самые важные характеристики погодных условий, измеряемые не только метеорологами, но и обычными людьми, — температура и влажность. Долгосрочные прогнозы строятся на основании наблюдений за изменением атмосферного давления.

Эксперимент: насколько жарко в помещении?

Датчик LM35 измеряет температуру окружающей среды, представляя ее выходным напряжением. Поскольку этот температурный датчик относится к семейству аналоговых резистивных устройств, то его проще подключать к Arduino, чем к Raspberry Pi. Для подключения температурного датчика к Raspberry Pi вам понадобится внешний аналого-цифровой преобразователь.

Датчик LM35, как и любой другой потенциометр, снабжен тремя выводами (рис. 12.1). Выходное напряжение U преобразовывается в температуру T согласно простой формуле:

$$T = 100 \times U$$

Некоторые примеры преобразований приведены в табл. 12.1. Как видите, рабочий диапазон температурного датчика LM35 составляет от 2 до 150 °C. Выходное напряжение представляется диапазоном 0–1,5 В.

Вам требуется измерить температуры ниже нуля по шкале Цельсия? Воспользуйтесь температурным датчиком LM36 или выберите любой другой датчик, поддерживающий соответствующий диапазон температур (он даже может быть указан в технической документации к датчику LM35).

Таблица 12.1. Зависимость температуры окружающей среды от напряжения на выходе температурного датчика LM35

Температура, °C	Напряжение, В	Описание
2	0,02	Самая низкая измеряемая температура
20	0,2	Комнатная температура
100	1	Температура кипения воды
150	1,5	Самая высокая измеряемая температура

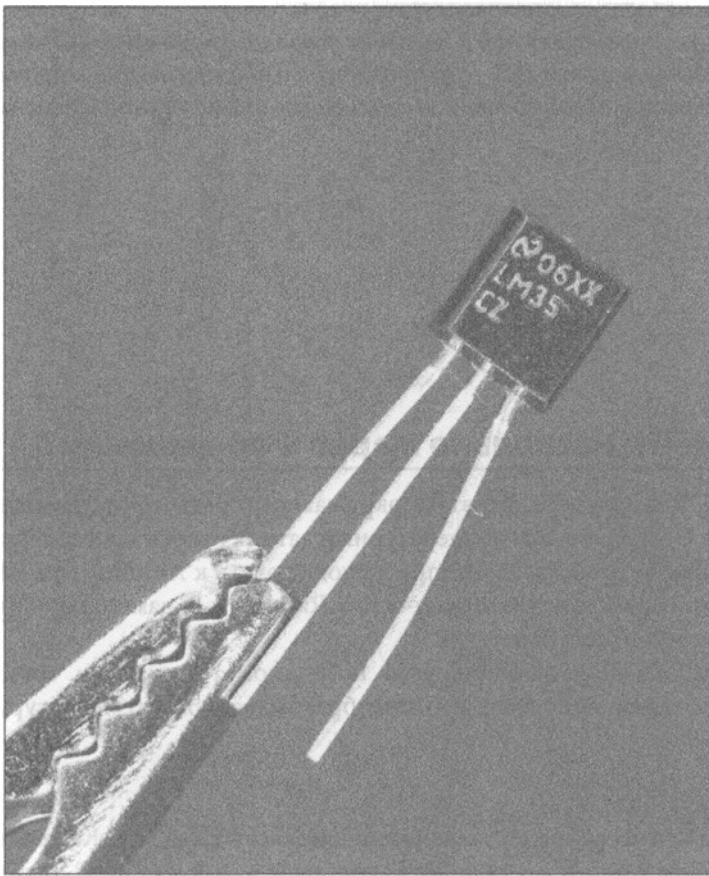


Рис. 12.1. Датчик LM35

Подключение к Arduino и программа управления температурным датчиком LM35

На рис. 12.2 показана схема подключения температурного датчика к Arduino. Собрав показанную электрическую цепь, выполните программный код из листинга 12.1.

Листинг 12.1. temperature_lm35.ino

```
// temperature_lm35.ino - измерение температуры датчиком LM35
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int lmPin = A0;

void setup()
{
    Serial.begin(115200);
    pinMode(lmPin, INPUT);
}

float tempC()
{
    float raw = analogRead(lmPin); // ❶
    float percent = raw/1023.0; // ❷
    float volts = percent*5.0; // ❸
    return 100.0*volts; // ❹
}

void loop()
{
    Serial.println(tempC());
    delay(200); // мс
}
```

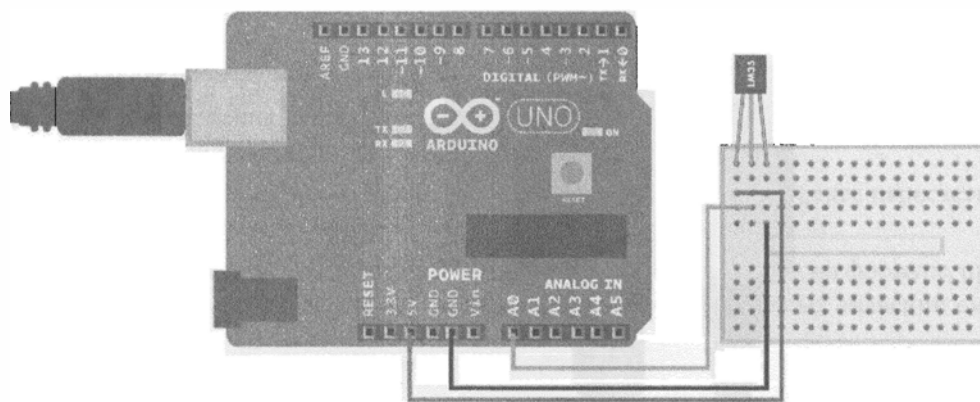


Рис. 12.2. Схема подключения температурного датчика к Arduino

- ❶ Температурный датчик LM35 — это обычное аналоговое резистивное устройство, поэтому напряжение на нем считывается стандартной функцией `analogRead()`. Эта функция возвращает значение в диапазоне от 0 до 1023.
- ❷ Преобразование необработанного значения (0–1023) в пересчетный коэффициент для сигнала высокого уровня (HIGH, 5 В). Пересчетный коэффициент принимает значение из диапазона от 0,0 до 1,0.

- 3 Вычисление напряжения на выходе с учетом полученного ранее пересчетного коэффициента (в диапазоне от 0 до 5 В).
- 4 Возвращение температуры, выраженной в градусах Цельсия. Расчетная формула взята из технической документации к температурному датчику LM35.

Подключение к Raspberry Pi и программа управления температурным датчиком LM35

На рис. 12.3 показана схема подключения температурного датчика к Raspberry Pi. После ее реализации выполните программный код, представленный в листинге 12.2.

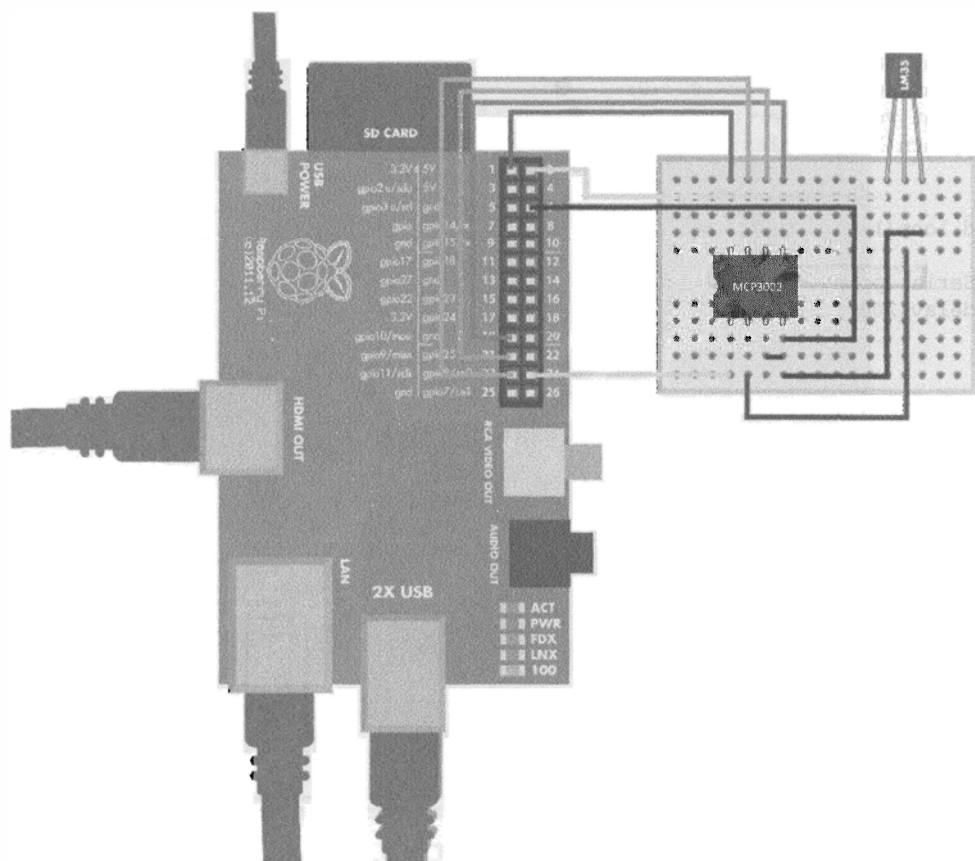


Рис. 12.3. Схема подключения температурного датчика к Raspberry Pi (см. цветную вклейку)

Поскольку максимальное выходное напряжение на датчике LM35 составляет 1,5 В, что соответствует 150 °С, то риска повреждения платы Raspberry Pi нет, так как на ее вход можно подавать сигналы с максимальным напряжением 3,3 В.

Листинг 12.2. temperature_lm35.py

```
# temperature_lm35.py - вывод температуры в помещении
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp # ❶

# На вход датчика LM35 подается напряжение меньше 2 В, поэтому
# дополнительный нагрузочный резистор не используется
def main():
    while True:
        rawData = mcp.readAnalog() # ❷
        percent = rawData / 1023.0 # ❸
        volts = percent * 3.3 # ❹
        temperature = 100.0 * volts # ❺
        print("Текущая температура: %.1f C " % temperature)
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

- ❶ Файл библиотеки botbook_mcp3002.py необходимо поместить в такой же каталог, что и файл текущей программы (temperature_lm35.py). Вам также понадобится библиотека spidev, востребованная в библиотеке botbook_mcp3002. См. комментарии в начале программного кода botbook_mcp3002.py или соответствующий раздел в главе 3.
- ❷ Считывание выходного напряжения на датчике с использованием модуля MCP3002 и библиотеки botbook_mcp3002. Необработанное значение представляется целым числом из диапазона от 0 до 1023.
- ❸ Преобразование необработанного значения в пересчетный коэффициент для напряжения высокого уровня, 3,3 В.
- ❹ Вычисление выходного напряжения согласно полученному ранее пересчетному коэффициенту (в диапазоне от 0 до 3,3 В).
- ❺ Преобразование выходного сигнала в градусы Цельсия согласно формуле, приведенной в технической документации. Примеры получаемых значений см. в табл. 12.1.

Эксперимент в окружающей среде: изменение температуры

Воздух — очень хороший тепловой изолятор. Вы легко заметите, что показания датчика LM35 при помещении в различные температурные среды — сауну, холодильник или балкон — будут изменяться очень медленно (рис. 12.4).

Чтобы добиться быстрого изменения показаний, поднесите кусочек льда вплотную к датчику LM35, не касаясь его. Будьте предельно внимательны и избегайте намокания проводов, микроконтроллерной или макетной платы. Теперь датчик “остыл”, и изменение температуры регистрируется быстрее.

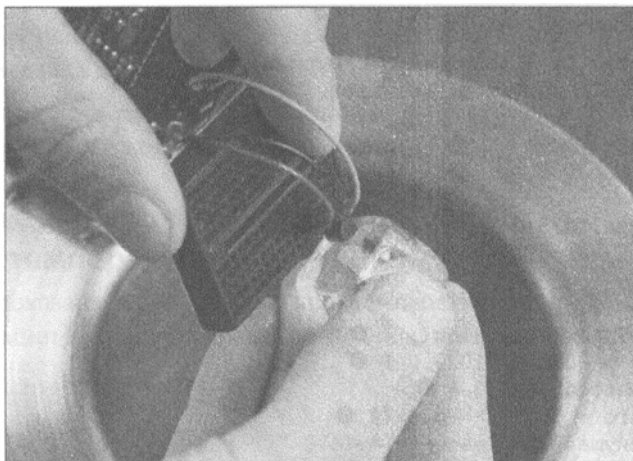


Рис. 12.4. Измерение отрицательных температур

Эксперимент: определение влажности

Датчик DHT11 измеряет вблизи себя и влажность, и температуру (рис. 12.5).

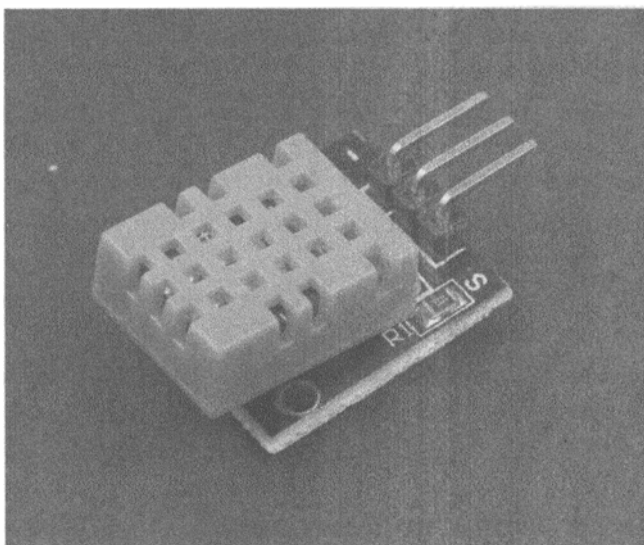


Рис. 12.5. Датчик влажности DHT11

Протокол, согласно которому датчик DHT11 передает данные, несколько странный. Датчик передает сигнал с очень короткими импульсами высокой частоты. В платформе Arduino нет операционной системы, поэтому временные интервалы в ней обрабатываются точнее, чем в Raspberry Pi, и, как следствие, длительность импульсов определяется проще. Но даже несмотря на это, код управления датчиком DHT11 в Arduino требует специальных приемов программирования, выходящих за рамки вызова одной только функции `pulseIn()`.

В платформе Raspberry Pi временные интервалы измеряются достаточно неточно, поэтому говорить о надежности измерения короткопериодических импульсов не приходится. Именно поэтому рассматриваемый нами датчик влажности подключается к Raspberry Pi через Arduino, которая фактически и считывает показания с DHT11. Подключение к Arduino реализуется через последовательное соединение USB. В случае необходимости подобный подход реализуется при подключении к микроконтроллерной платформе любого другого датчика.

На сигнальный выход в дежурном режиме обычно подается сигнал высокого уровня (HIGH). Начало считывания определяется отправкой микроконтроллером импульса низкого уровня (LOW) длительностью 18 мс.

Датчик DHT11 в качестве выходных данных отправляет 5-байтовый пакет. Поскольку один байт состоит из 8 битов, размер пакета данных составляет $5 \text{ байт} \times 8 \text{ битов} = 40 \text{ битов}$.

Влажность выдыхаемого воздуха

Для тестирования датчика нам необходимо добиться значительного изменения влажности возле него. Как это сделать? Легче всего купить бытовой ультразвуковой увлажнитель воздуха, который с помощью пьезоэлектрического генератора колебаний создает облако микрочастиц воды. Для любителей тропического отдыха всегда доступен вариант путешествия в Таиланд. Это простые, но очень затратные способы для наших скромных проектов. Удивительно, но все мы являемся отличными генераторами влажного воздуха. Поднесите датчик как можно ближе к губам и интенсивно подышите на него (рис. 12.6). Изменяя интенсивность дыхания, следите за значениями, выводимыми на монитор последовательного порта Arduino.

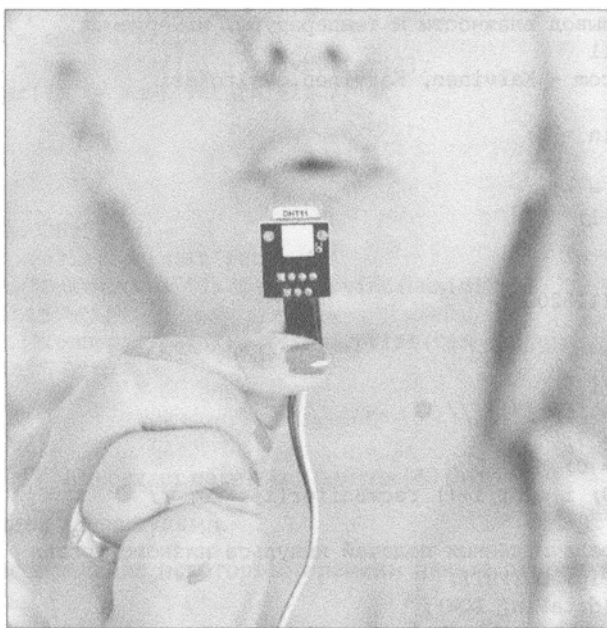


Рис. 12.6. Влажность выдыхаемого воздуха также определяется датчиком DHT11

Подключение к Arduino и программа управления датчиком DHT11

На рис. 12.7 показано, как датчик влажности подключается к Arduino. Соедините согласно представленной схеме все компоненты устройства и выполните программный код из листинга 12.3.

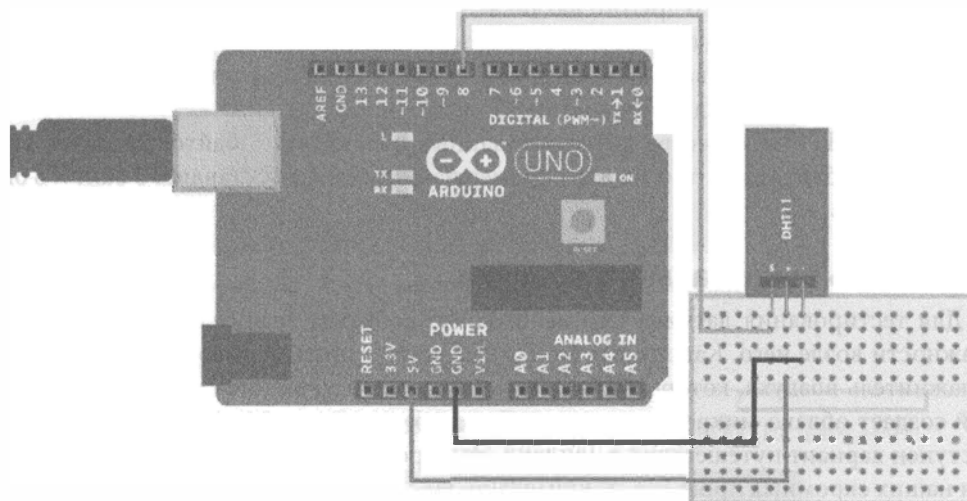


Рис. 12.7. Схема подключения датчика влажности к Arduino

Листинг 12.3. dht11.ino

```
// dht11.ino - вывод влажности и температуры, измеряемых
// датчиком DHT11
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int dataPin = 8;

int temperature = -1;
int humidity = -1;

void setup() {
    Serial.begin(115200);
}

int readDHT11() {
    uint8_t recvBuffer[5]; // ❶
    uint8_t cnt = 7;
    uint8_t idx = 0;
    for(int i = 0; i < 5; i++) recvBuffer[i] = 0; // ❷

    // Начало передачи данных подачей импульса низкого уровня
    pinMode(dataPin, OUTPUT);
    digitalWrite(dataPin, LOW);
    delay(18);
    digitalWrite(dataPin, HIGH);
```

```

delayMicroseconds(40); // ❶
pinMode(dataPin, INPUT);
pulseIn(dataPin, HIGH); // ❷
// Считывание передаваемых данных
unsigned int timeout = 10000; // итерации
for (int i=0; i<40; i++) // ❸
{
    timeout = 10000;
    while(digitalRead(dataPin) == LOW) {
        if (timeout == 0) return -1;
        timeout--;
    }

    unsigned long t = micros(); // ❹

    timeout = 10000;
    while(digitalRead(dataPin) == HIGH) { // ❺
        if (timeout == 0) return -1;
        timeout--;
    }

    if ((micros() - t) > 40) recvBuffer[idx] |= (1 << cnt); // ❻
    if (cnt == 0) // еще байт?
    {
        cnt = 7; // начало со старшего разряда
        idx++; // следующий байт!
    }
    else cnt--;
}

humidity = recvBuffer[0]; // % // ❼
temperature = recvBuffer[2]; // C
uint8_t sum = recvBuffer[0] + recvBuffer[2];
if(recvBuffer[4] != sum) return -2; // ❽
return 0;
}

void loop() {
    int ret = readDHT11();
    if(ret != 0) Serial.println(ret);
    Serial.print("Влажность: "); Serial.print(humidity);
    Serial.println(" %");
    Serial.print("Температура: "); Serial.print(temperature);
    Serial.println(" C");
    delay(2000); // мс
}

```

- ❶ Датчик DHT11 передает пакет из 5 байтов (40 битов).
- ❷ Заполнение буфера нулями.
- ❸ Ожидание в течение некоторого времени начала отправки датчиком DHT11 данных.
- ❹ Получение сигнала ответа с датчика DHT11.

- 9 Для каждого из 40 битов выполняются операции, представленные блоком кода, который заключен в фигурные скобки.
- 6 Время работы, выраженное в микросекундах (мкс); применяется при определении длительности импульса.
- 7 Измерение длительности импульса.
- 8 Если длительность импульса больше 40 мкс, то бит приравнивается к 1. В противном случае текущий бит равен предварительно установленному значению, 0. Для переключения соответствующего бита в рассматриваемом байте применяется операция битового сдвига (например, $1 \ll 2 == 0b100$). Над указанным битом и значением из предыдущего байта выполняется операция побитового исключающего ИЛИ. Поэтому, если значение уже равно 1, то оно не изменяется. Если же значение равно 0, а битовый сдвиг ($0b100$) приводит к результату 1, то значение приравнивается к 1. См. описание битовых операций в главе 8.
- 9 Влажность указывается в первом байте полученного пакета данных.
- 10 Контрольная сумма (4-й байт) --- это сумма байтов, представляющих влажность и температуру.

Подключение к Raspberry Pi и программа управления датчиком DHT11

Настало время скомбинировать Arduino и Raspberry Pi в одном устройстве, чтобы воспользоваться преимуществами сразу обеих платформ. Как уже упоминалось, управлять датчиком DHT11 в Raspberry Pi весьма проблематично. Поэтому подключать его мы будем непосредственно к Arduino. В приведенном далее программном коде данные считываются платой Arduino, но впоследствии передаются и обрабатываются в Raspberry Pi, что расширяет возможности применения датчика влажности в различных проектах.

Конечно, управление датчиком DHT11 в Raspberry Pi возможно, хотя и требует применения сложного программного кода. Как бы там ни было, но получаемый результат все равно нельзя назвать удовлетворительным. В комбинированной среде за низкоуровневые задачи отвечает Arduino, а Raspberry Pi занимается только управлением выводимыми данными. Для лучшего понимания комбинированной системы после выполнения программного кода внимательно изучите раздел “Доступ к Arduino из Raspberry Pi”.

Чтобы получить доступ к последовательному порту в Raspberry Pi, вам сначала нужно освободить его, поскольку он занят интерпретатором входа в систему. Детально решение этой задачи рассматривалось в главе 11. Создайте устройство, схематически показанное на рис. 12.8, и выполните программный код из листинга 12.4.

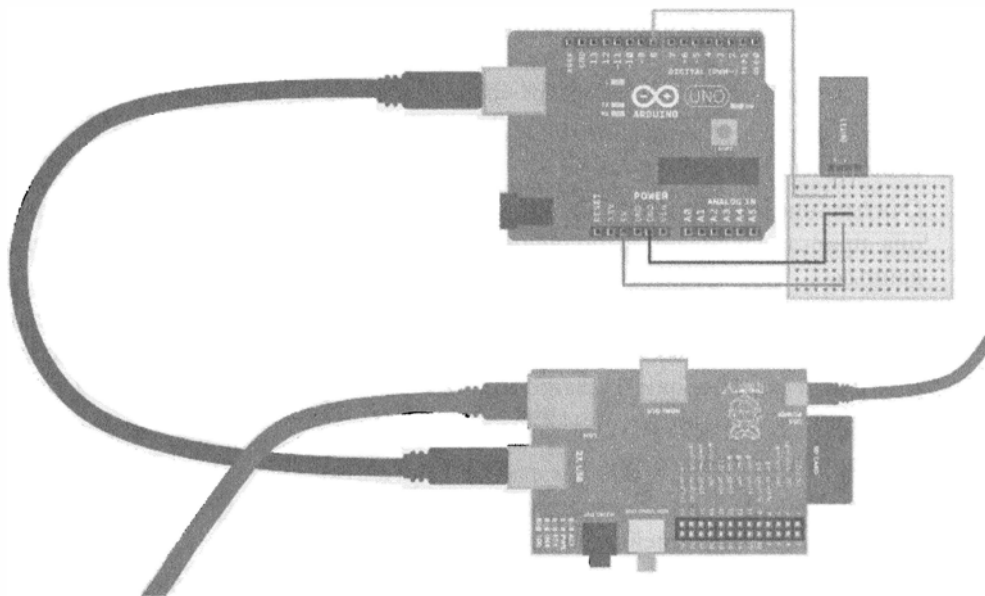


Рис. 12.8. Схема подключения Arduino с датчиком влажности к Raspberry Pi

Листинг 12.4. dht11_serial.py

```
# dht11_serial.py - вывод влажности и температуры, измеряемых
# датчиком DHT11
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import serial # ❶

def main():
    port = serial.Serial("/dev/ttyACM0", baudrate=115200,
    timeout=None) # ❷
    while True:
        line = port.readline() # ❸
        arr = line.split() # ❹
        if len(arr) < 3: # ❺
            continue # ❻
        dataType = arr[2]
        data = float(arr[1]) # ❼
        if dataType == '%':
            print("Влажность: %.1f %% " % data)
        else:
            print("Температура: %.1f C " % data)

        time.sleep(0.01)

if __name__ == "__main__":
    main()
```

- ❶ Для установки указанной библиотеки вам может понадобиться выполнить такую команду:
`sudo apt-get update && sudo apt-get -y install python-serial`
- ❷ Открытие последовательного USB-соединения для передачи данных. В Arduino необходимо установить такую же скорость передачи данных, 115 200 бит/с.
- ❸ Считывание полной текстовой строки. Функция `readline()` — блокирующая, она приостанавливает выполнение программы до получения передаваемых данных.
- ❹ Разделение текстовой строки на список значений. Это позволяет упростить обработку выводимых Arduino данных. Например, при выводе через последовательный порт строки Влажность: 25 % она разделяется на такой список значений: `['Влажность:', '25', '%']`.
- ❺ Если длина строки неверна...
- ❻ ...то текущая строка кода игнорируется, а выполнение программы продолжается следующей итерацией цикла `while`.
- ❼ В результате преобразования типов данных строковые значения "25" становятся числовыми `25.0`.

Доступ к Arduino из Raspberry Pi

Arduino — это простая и надежная, работающая в реальном времени микроконтроллерная система. Кроме того, в Arduino встроен аналого-цифровой преобразователь (АЦП). В противовес ей платформа Raspberry Pi на порядок производительней и работает под управлением отдельной операционной системы Linux. Почему бы не совместить эти платформы?

При такой схеме Arduino отвечает за управление датчиками и вывод считанных данных, а Raspberry Pi управляет Arduino.

Давайте сначала подключим датчик к Arduino и добьемся вывода данных с него через последовательный порт. Для этого сначала подключите плату Arduino к настольному компьютеру или ноутбуку, в котором установлена соответствующая среда разработки. Вывод данных на монитор последовательного порта выполняется функцией `Serial.println()`.

Для упрощения двухплатформенной системы рассмотрите возможность приобретения модуля AlaMode для Raspberry Pi, позволяющего монтировать плату Arduino непосредственно поверх Raspberry Pi. Эта плата расширения устраняет необходимость применения USB-кабеля, поскольку монтируется прямо на штекерную колодку Raspberry Pi.

Чтобы наблюдать за выводимыми данными, выполните команду `Tools⇒Serial Monitor` (Сервис⇒Монитор порта). Как только получите на экране значения, которые вас удовлетворяют, переходите к добавлению в систему платы Raspberry Pi.

Подключить Arduino к Raspberry Pi очень просто: воспользуйтесь стандартным USB-кабелем.

Передача данных с Arduino по-прежнему осуществляется через последовательное USB-соединение. Для получения данных с последовательного порта в Raspberry Pi применяются программные средства Python и библиотеки pySerial. Пример системы, требующей использования средств библиотеки pySerial, приведен на рис. 12.8.

Платформа Arduino выводит сообщения на монитор последовательного порта в виде текста, после чего в Python из нее извлекаются числовые значения. В большинстве прототипов устройств, в которых используются датчики, вам не нужно изобретать другой метод обмена данными. Рассмотрим, например, как обрабатывается следующая текстовая строка, передаваемая Arduino через последовательный порт:

```
Влажность: 83 %
```

В Python проще всего получить ее в виде строки, а затем воспользоваться функцией `split()`, чтобы разделить и представить как список значений.

```
>>> s="Влажность: 83 %"
>>> s.split()
[Влажность:', '83', '%']
```

Ссылаться на отдельные элементы списка можно, заключив их в квадратные скобки, `[]`. Первый элемент списка представлен числом нуль. Следующий код применяется для вывода второго элемента списка:

```
>>> x = s.split()[1]
>>> x
'83'
```

Теперь вам нужно преобразовать строковое значение `'83'` в целочисленное `83`, чтобы в дальнейшем оперировать им в программе как числом. Это необходимо, чтобы применить полученное значение в математических вычислениях, что невозможно в случае использования данных строкового типа.

```
>>> int(x)
83
```

Описанным образом можно передавать данные в Raspberry Pi из любого Arduino-совместимого датчика.

Датчик атмосферного давления GY65

Атмосферное давление, как никакая другая характеристика, применяется синоптиками для составления прогнозов погоды. Высокое давление всегда свидетельствует о ясной солнечной погоде. Низкое давление предполагает дождь, снег и высокую облачность.

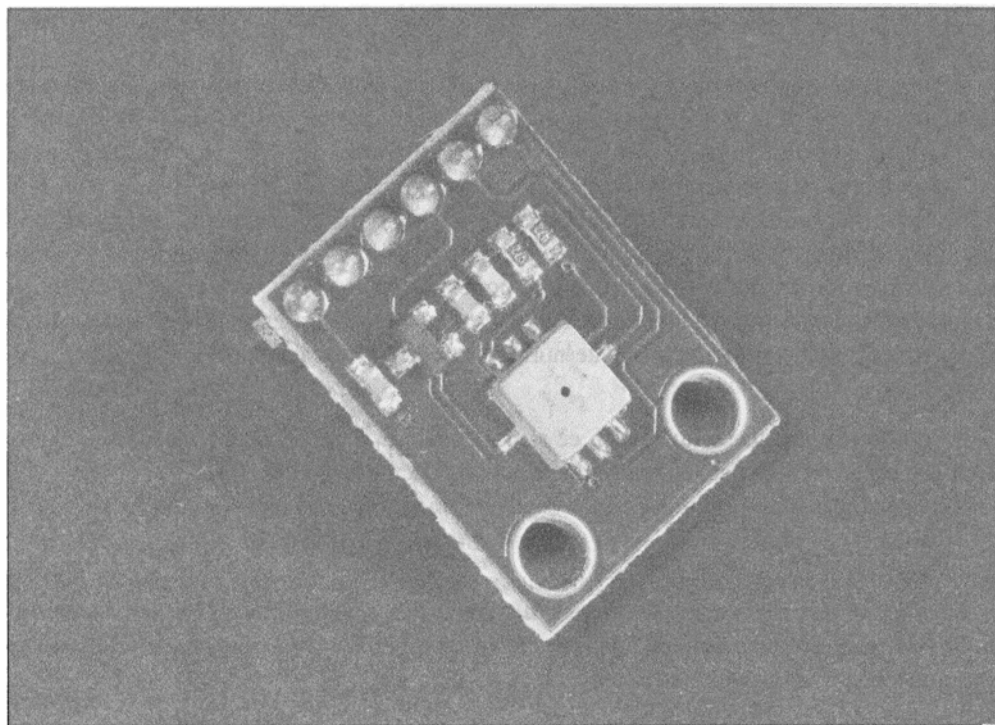


Рис. 12.9. Датчик атмосферного давления GY65

Нормальное атмосферное давление составляет около 100 кПа. Точное значение зависит от высоты над уровнем моря:

$$100 \text{ кПа} = 100\,000 \text{ Па} = 1000 \text{ мбар} = 1 \text{ бар}$$

Степень изменения атмосферного давления в зависимости от погодных условий не очень велика. Максимальное зарегистрированное атмосферное давление составляет 108,4 кПа, а минимальное — 87 кПа (внутри воронки урагана). При нормальных атмосферных условиях колебание давления изменяется в пределах диапазона, ширина которого не превышает 1 кПа.

Изменение давления в нижних слоях атмосферы выглядит несущественно на фоне разницы давления в других самых обыденных ситуациях. Так, например, давление в велосипедной шине составляет около 400 кПа, а давление за бортом самолета, летящего на высоте 10 000 метров над землей, — меньше 30 кПа.

Какой же уровень атмосферного давления считать высоким? Метеорологи оперируют понятием относительно высокого атмосферного давления, сравнивая его с атмосферным давлением окружающих регионов.

Предсказывать погоду по изменению атмосферного давления не очень сложно. Если атмосферное давление растет, то погода будет улучшаться. Чем сильнее изменение давления, тем солнечнее будет на улице вскоре.

В самом простом прогнозе погоды сравнивается текущее атмосферное давление со значением нормального атмосферного давления, характерного для высоты над

уровнем моря вашего региона. Высота над уровнем моря легко определяется с помощью GPS и Google Maps. Согласно информации SparkFun, основного дистрибьютора модуля GY65, увеличение атмосферного давления на 0,25 кПа говорит об наступлении хорошей, солнечной погоды. Таким же образом уменьшение атмосферного давления на 0,25 кПа вызывает ухудшение погодных условий с обильными осадками и понижением температуры.

На самом деле GY65 — это название не датчика, а коммутационной платы для датчика BMP085. Поэтому технические характеристики в Интернете лучше искать для датчика атмосферного давления BMP085, а не модуля GY65.

Подключение к Arduino и программа управления датчиком GY65

На рис. 12.10 показана схема подключения датчика атмосферного давления к Arduino. После построения соответствующей электрической цепи выполните программный код из листинга 12.5.

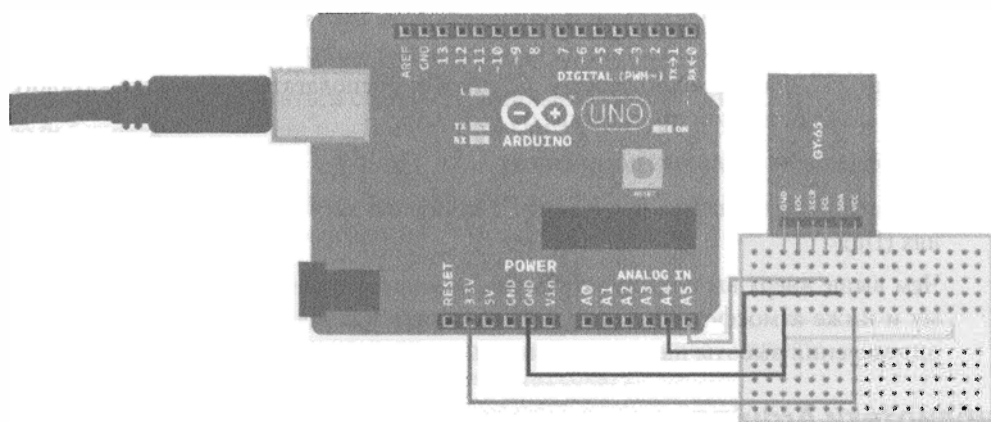


Рис. 12.10. Схема подключения датчика атмосферного давления к Arduino

В программном коде управления датчиком атмосферного давления в Arduino используется библиотека `gy_65`, доступная для загрузки с сайта, указанного во введении. Если вам интересно узнать в деталях, как правильно управлять протоколами I²C/SMBus, то обратитесь к разделу “Подключение к Raspberry Pi и программа управления датчиком GY65”

Листинг 12.5. `gy_65.ino`

```
// gy_65.ino - вывод высоты над уровнем моря, давления и
// температуры, измеряемых модулем GY65 (BMP085)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>
#include <gy_65.h> // ❶

void setup() {
```

```

Serial.begin(115200);
readCalibrationData(); // ❷
}

void loop() {
  float temp = readTemperature();
  float pressure = readPressure(); // ❸
  float altitude = calculateAltitude(pressure); // ❹
  Serial.print("Высота: ");
  Serial.print(altitude,2);
  Serial.println(" м");
  Serial.print("Давление: ");
  Serial.print(pressure,2);
  Serial.println(" Па");
  Serial.print("Температура: ");
  Serial.print(temp,2);
  Serial.println("C");
  delay(1000);
}

```

- ❶ Библиотека `gy_65` делает управление датчиком пустяковым занятием. Она берет на себя всю ответственность за управление модулем при подключении его к шине I²C. Файл библиотеки `gy_65` должен располагаться в том же каталоге, что и файл основной программы, `gy_65.ino`.
- ❷ Обновление глобальных переменных.
- ❸ Атмосферное давление, получаемое с помощью датчика, выражается в паскалях (Па).
- ❹ Чем выше вверх вы поднимаетесь, тем меньше концентрация воздуха, а потому и ниже атмосферное давление, что делает модуль GY65 применимым для определения высоты над уровнем моря.

Библиотеки Arduino

Если программа для проекта содержит объемный код, то лучше разделить его на несколько файлов. В случае проекта Arduino по управлению датчиком GY65 складывается именно такая ситуация. Подобная программа ожидается при реализации последнего проекта текущей главы — «Пилотный проект: прогноз погоды с выводом на электронную бумагу».

Файл библиотеки должен располагаться в том же каталоге, что и файл основной программы (всегда важно создать в каталоге основной программы проекта специальный подкаталог, предназначенный для хранения только библиотек Arduino). Ниже показана структура такого каталога.

```

gy_65.ino # основная программа
gy_65/ # подкаталог для хранения библиотеки
gy_65/gy_65.cpp # файл с программным кодом библиотеки
gy_65/gy_65.h # файл прототипов всех функций библиотеки

```

Правильность размещения основной программы `gy_65.ino` не вызывает сомнений. В каждом проекте Arduino всегда есть основная программа.

Библиотека располагается в отдельном подкаталоге. Программный код библиотеки представлен файлом `gy_65.cpp`. Этот программный код мало чем отличается от программного кода проектов Arduino, рассматриваемых ранее.

Заголовочный файл, `gy_65.h`, содержит прототипы функций, используемых в файле библиотеки (`*.cpp`). Прототипы — это копии первых строк каждой функции, встречающейся в файле `gy_65.cpp`. Например, в заголовочном файле может быть всего одна строка:

```
float readTemperature();
```

Описание библиотеки Arduino gy_65

Управление модулем GY65 реализуется и без применения библиотек. Это требует глубоких знаний протоколов обмена данными с GY65. Если вы готовы к рассмотрению серьезных программных структур, то приготовьтесь к изложению трудного для понимания материала.

Протокол обмена данными проще всего рассмотреть на реальном примере: изучая программный код библиотеки и программы управления датчиком в Raspberry Pi (рис. 12.11).

В этом разделе мы поговорим о создании собственных библиотек C++ для Arduino. Всегда нелишне ознакомиться с рекомендациями разработчиков Arduino по написанию библиотек.

Заголовочный файл `gy_65.h` (листинг 12.6) содержит прототипы всех функций. Он включает список функций, применяемых в файле `*.cpp`, в котором непосредственно и выполняются эти функции.

Листинг 12.6. `gy_65.h`

```
// gy_65.h - заголовочный файл библиотеки, подключаемой при
// измерении высоты, давления и температуры модулем GY65 (BMP085)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
void readCalibrationData();
float readTemperature();
float readPressure();
float calculateAltitude(float pressure);
```

Полный программный код функций, объявленных в заголовочном файле, реализован в библиотеке `gy_65.cpp` (листинг 12.7). Если какая-то часть программного кода библиотеки трудна для понимания, то внимательно изучите главу 8, в которой описаны системы счисления, побитовые операции и принципы обмена данными через шину I²C.

Листинг 12.7. `gy_65.py`

```
// gy_65.cpp - библиотека, подключаемая при измерении высоты,
// давления и температуры модулем GY-65 (BMP085)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
#include <Arduino.h>
#include <Wire.h>
#include "gy_65.h"
```

```

const char i2c_address = 0x77;
int OSS = 0;
const long atmosphereSeaLevel = 101325; // Па

struct calibration_data // ❶
{
    int16_t ac1;
    int16_t ac2;
    int16_t ac3;
    int16_t ac4;
    int16_t ac5;
    int16_t ac6;
    int16_t b1;
    int16_t b2;
    int16_t mb;
    int16_t mc;
    int16_t md;
};

calibration_data caldata;

long b5;

int16_t swap_int16_t(int16_t value) // ❷
{
    int16_t left = value << 8;
    int16_t right = value >> 8;
    right = right & 0xFF;
    return left | right ;
}

unsigned char read_i2c_unsigned_char(unsigned char address) // ❸
{
    unsigned char data;
    Wire.beginTransmission(i2c_address);
    Wire.write(address);
    Wire.endTransmission();
    Wire.requestFrom(i2c_address,1);
    while(!Wire.available());
    return Wire.read();
}

void read_i2c(unsigned char point, uint8_t *buffer, int size)
{
    Wire.beginTransmission(i2c_address);
    Wire.write(point);
    Wire.endTransmission();

    Wire.requestFrom(i2c_address,size);

    int i = 0;

    while(Wire.available() && i < size) {
        buffer[i] = Wire.read();
        i++;
    }
}

```

```

    }

    if(i != size) {
        Serial.println("Ошибка получения данных через I2C");
    }
}

int read_i2c_int(unsigned char address) {
    int16_t data;
    read_i2c(address, (uint8_t *)&data, sizeof(int16_t));
    data = swap_int16_t(data);
    return data;
}

void readCalibrationData() // 4
{
    Wire.begin();
    read_i2c(0xAA, (uint8_t *)&caldata, sizeof(calibration_data)); // 5

    uint16_t *p = (uint16_t *)&caldata; // 6
    for(int i = 0; i < 11; i++) { // 7
        p[i] = swap_int16_t(p[i]);
    }
}

float readTemperature() { // 8
    // Получение необработанного значения температуры
    Wire.beginTransaction(i2c_address);
    Wire.write(0xF4); // Register
    Wire.write(0x2E); // Value
    Wire.endTransmission();
    delay(5); // 9
    unsigned int rawTemp = read_i2c_int(0xF6);

    // Вычисление температуры
    long x1, x2;
    float t;
    x1 = (((long)rawTemp - (long)caldata.ac6) *
    ↵ (long)caldata.ac5) / pow(2,15);
    long mc = caldata.mc;
    int md = caldata.md;
    x2 = (mc * pow(2,11)) / (x1 + md);
    b5 = x1 + x2;
    t = (b5 + 8) / pow(2,4);
    t = t / 10;
    return t; // Градусы Цельсия
}

long getRealPressure(unsigned long up){ // 10
    long x1, x2, x3, b3, b6, p;
    unsigned long b4, b7;
    int b1 = caldata.b1;
    int b2 = caldata.b2;
    long ac1 = caldata.ac1;
    int ac2 = caldata.ac2;

```

```

int ac3 = caldata.ac3;
unsigned int ac4 = caldata.ac4;

b6 = b5 - 4000;
x1 = (b2 * (b6 * b6) / pow(2,12)) / pow(2,11);
x2 = (ac2 * b6) / pow(2,11);
x3 = x1 + x2;

b3 = (((ac1*4 + x3) << OSS) + 2) / 4;
x1 = (ac3 * b6) / pow(2,13);
x2 = (b1 * ((b6 * b6) / pow(2,12)) ) / pow(2,16);
x3 = ((x1 + x2) + 2) / 4;
b4 = (ac4 * (unsigned long)(x3 + 32768) ) / pow(2,15);

b7 = ((unsigned long)up - b3) * (50000 >> OSS);
if (b7 < 0x80000000) p = ( b7 * 2 ) / b4;
else p = (b7 / b4) * 2;

x1 = (p / pow(2,8)) * (p / pow(2,8));
x1 = (x1 * 3038) / pow(2,16);
x2 = (-7357 * p) / pow(2,16);
p += (x1 + x2 + 3791) / pow(2,4);

long temp = p;
return temp;
}

float readPressure() { // ⑩
    // Считывание ненормализованного давления
    Wire.beginTransmission(i2c_address);
    Wire.write(0xF4); // Регистр
    Wire.write(0x34 + (OSS << 6));
    Wire.endTransmission();

    delay(2 + (3 << OSS));

    unsigned char msb,lsb,xlsb;
    unsigned long rawPressure = 0;
    msb = read_i2c_unsigned_char(0xF6);
    lsb = read_i2c_unsigned_char(0xF7);
    xlsb = read_i2c_unsigned_char(0xF8);

    rawPressure = (((unsigned long) msb << 16) | ((unsigned long)
    ↵ lsb << 8) | (unsigned long) xlsb) >> (8-OSS);
    return getRealPressure(rawPressure);
}

float calculateAltitude(float pressure) { // ⑪
    float pressurePart = pressure / atmosphereSeaLevel;
    float power = 1 / 5.255;
    float result = 1 - pow(pressurePart, power);
    float altitude = 44330*result;
    return altitude; // м
}

```

- ❶ Объявление глобальных переменных и структур. Калибровочные данные запрашиваются из энергонезависимой памяти (EEPROM).
- ❷ Получение двухбайтового целочисленного значения и изменение порядка следования байтов на противоположный, поскольку данные с датчика передаются в несколько ином формате, чем принимает Arduino.
- ❸ Следующие функции применяются исключительно ради повышения уровня структуризации программы: `read_i2c_unsigned_char()`, `read_i2c()` и `read_i2c_int()`. Они являются оболочками для средств библиотеки `Wire.h`, упрощая использование последних в текущей программе.
- ❹ Считывание калибровочных данных с энергонезависимой памяти датчика. Их формат детально описывается в технической документации к датчику BMP085.
- ❺ Для извлечения калибровочных данных из энергонезависимой памяти применяется отдельная структура `calibration_data`. Байты данных из памяти датчика заносятся в память Arduino в виде структуры, организованной так, что длины переменных соответствуют размеру передаваемых необработанных значений. Переданные с датчика данные заносятся в переменную `caldata`, а доступ к отдельным значениям реализуется с помощью структуры.
- ❻ Поскольку порядок следования байтов в данных, передаваемых с датчика, отличается от поддерживаемого в Arduino, байты в каждом двухбайтовом целочисленном значении необходимо поменять местами. Вначале создается двухбайтовый указатель, исходно ссылающийся на начало структуры `caldata`...
- ❼ ...который затем применяется для просмотра всей структуры `caldata` и замены байтов местами. Обратите внимание, как двухбайтовый (16-битовый) указатель в каждой итерации ссылается на новое двухбайтовое целочисленное значение вместо того, чтобы напрямую ссылаться на отдельные байты.
- ❽ Функция `readTemperature()` считывает необработанное значение температуры, передаваемое через I²C, а затем вычисляет абсолютное значение температуры, применяя формулу, которая указывается в технической документации.
- ❾ В технических характеристиках датчика указывается минимальная задержка, равная 4,5 мс.
- ❿ Функция `getRealPressure()` принимает необработанное значение атмосферного давления, считанное функцией `readPressure()`, и преобразовывает его в значение, выраженное в паскалях (Па). В этой функции применяются калибровочные данные, считываемые функцией `readCalibrationData()`, а вычисления ведутся согласно формуле, указанной на странице 13 технической документации к датчику BMP085.
- ⓫ Запрос атмосферного давления через шину I²C. Получаемое значение и его формат определяются в технической документации к датчику.
- ⓬ Вычисление высоты над уровнем моря согласно показаниям атмосферного давления. Применяется общая барометрическая формула, приведенная в технической документации к датчику BMP085 на странице 14.

Подключение к Raspberry Pi и программа управления датчиком GY65

Программный код управления датчиком атмосферного давления в Raspberry Pi не выглядит очень сложным, поэтому сначала выполните его для тестирования оборудования и только после этого приступайте к детальному изучению. Детальный анализ программы показывает, что код намного сложнее для понимания, чем выглядит при беглом просмотре. В Raspberry Pi передача данных с датчика GY65 через I²C выполняется без использования специальной библиотеки, поэтому программный код управления датчиком значительно длиннее, чем в случае с Arduino. В данном коде используются методики программирования, известные вам по примерам из предыдущих глав, а потому требующие от вас познаний в области побитовых операций и других систем счисления (см. главу 8).

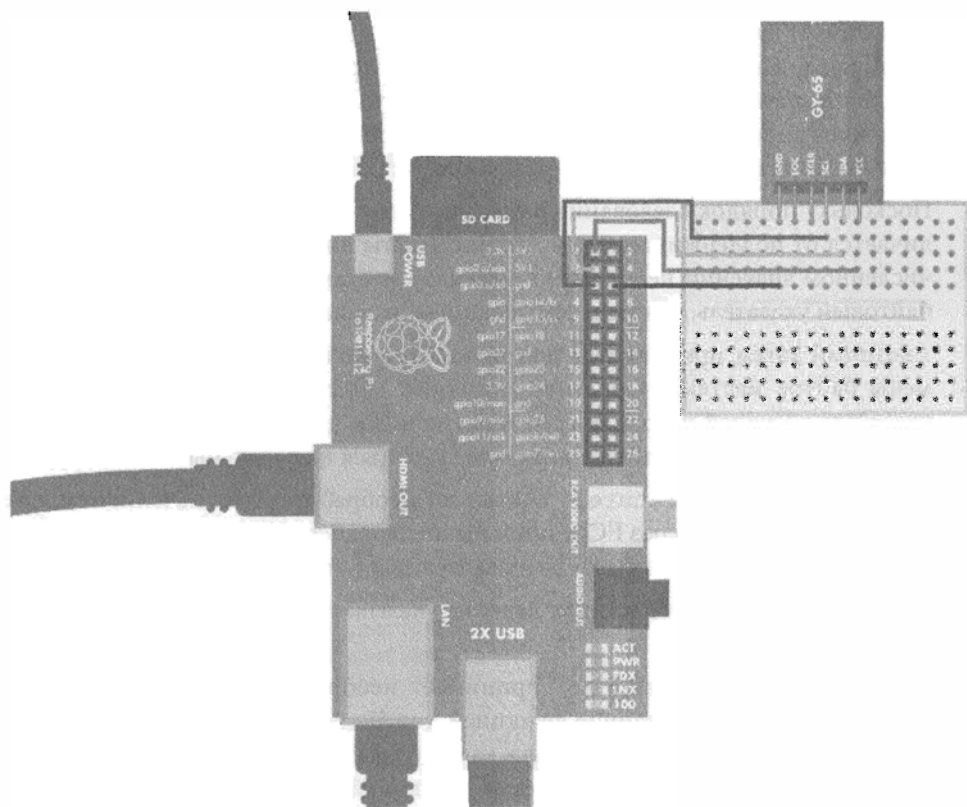


Рис. 12.11. Схема подключения датчика атмосферного давления к Raspberry Pi

Листинг 12.8. gy_65.py

```
# gy_65.py - вывод высоты, атмосферного давления и температуры на
# монитор последовательного порта
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```



```

import smbus # sudo apt-get -y install python-smbus # ❶
import time
import struct

bus = None
address = 0x77
caldata = None

atmosphereSeaLevel = 101325.0
OSS = 0
b5 = 0

def readCalibrationData(): # ❷
    global bus, caldata
    bus = smbus.SMBus(1)
    rawData = ""

    for i in range(22):
        rawData += chr(bus.read_byte_data(address, 0xAA+i)) # ❸
        caldata = struct.unpack('>hhhhhhhhhhh', rawData) # ❹

def readTemperature():
    global b5
    bus.write_byte_data(address, 0xF4, 0x2E) # ❺
    time.sleep(0.005) # ❻
    rawTemp = bus.read_byte_data(address, 0xF6) << 8 # ❼
    rawTemp = rawTemp | bus.read_byte_data(address, 0xF7)
    x1 = ((rawTemp - caldata[5]) * caldata[4]) / 2**15
    x2 = (caldata[9] * 2**11) / (x1 + caldata[10])
    b5 = x1 + x2
    temp = (b5 + 8) / 2**4
    temp = temp / 10.0
    return temp

def readPressure(): # ❽
    bus.write_byte_data(address, 0xF4, 0x34 + (OSS << 6))
    time.sleep(0.005)
    rawPressure = bus.read_byte_data(address, 0xF6) << 16
    rawPressure = rawPressure | bus.read_byte_data(address, 0xF7) << 8
    rawPressure = rawPressure | bus.read_byte_data(address, 0xF8)
    rawPressure = rawPressure >> (8 - OSS)

    # Вычисление давления в паскалях
    b6 = b5 - 4000

    x1 = (caldata[7] * ((b6 * b6) / 2**12)) / 2**11
    x2 = caldata[1] * b6 / 2**11
    x3 = x1 + x2
    b3 = (((caldata[0] * 4 + x3) << OSS) + 2) / 4
    x1 = caldata[2] * b6 / 2**13
    x2 = (caldata[6] * ((b6 * b6) / 2**12)) / 2**16
    x3 = ((x1 + x2) + 2) / 2*2
    b4 = (caldata[3] * (x3 + 32768)) / 2**15
    b4 = b4 + 2**16 # преобразование в значение без знака

```

```

b7 = (rawPressure - b3) * (50000 >> OSS)
if b7 < 0x80000000:
    p = (b7 * 2) / b4
else:
    p = (b7 / b4) * 2
x1 = (p / 2**8) * (p / 2**8)
x1 = (x1 * 3038) / 2**16
x2 = (-7357 * p) / 2**16
p = p + (x1 + x2 + 3791) / 2**4
return p

def calculateAltitude(pressure): # ❶
    pressurePart = pressure / atmosphereSeaLevel;
    power = 1 / 5.255;
    result = 1 - pressurePart**power;
    altitude = 44330*result;
    return altitude

def main():
    readCalibrationData()
    while True:
        temperature = readTemperature()
        pressure = readPressure()
        altitude = calculateAltitude(pressure)
        print("Высота %.2f м" % altitude)
        print("Давление %.2f Па" % pressure) # ❷
        print("Температура %.2f C" % temperature)
        time.sleep(10)

if __name__ == "__main__":
    main()

```

- ❶ Стандарт SMBus является частью технологии I²C. Библиотека `smbus` значительно упрощает его поддержку в программном коде. Для ее подключения в программе в Raspberry Pi сначала необходимо установить пакет `python-smbus` (детально об этом см. в главе 8).
- ❷ Датчик атмосферного давления хранит 176 бит калибровочных данных, занесенных в энергонезависимую память EEPROM.
- ❸ Получение данных по адресам 0xAA, 0xAB и т.д. до 0xBF. Добавление прочитанных символьных значений к строке. Строка заканчивается после считывания 22 байтов (176 битов). В Arduino вам не приходилось волноваться о длине строки, поскольку длина применяемой структуры определялась длинами переменных, заключенных в ней.
- ❹ Извлечение 11 двухбайтовых коротких целочисленных значений (h), представленных в обратном порядке (>). Таким образом, из строки извлекаются все байты.
- ❺ Построение команды считывания температуры (0x2E), передаваемой в регистр команд датчика (0xF4).
- ❻ Дождемся окончания измерительной операции.

- 7 Получение ответа и преобразование необработанного значения в температуру, выраженную в градусах Цельсия. Расчетная формула взята из технической документации к датчику.
- 8 Функция `readPressure()` работает подобно функции `readTemperature()`, но измеряет атмосферное давление.
- 9 Чем выше вверх вы подниметесь, тем ниже будет атмосферное давление. Исходя из этого принципа общая барометрическая формула позволяет оценить высоту вашего расположения относительно уровня моря.
- 10 $100\,000\text{ Па} = 1\text{ бар}$.

Эксперимент в окружающей среде: автоматический полив (датчик влажности почвы)

Датчик влажности почвы — это простой аналоговый резистивный прибор. Он погружается в почву и показывает, нужен ли вашим растениям дополнительный полив.

В воде, которая течет у нас из крана, а также взятой из артезианских источников, растворено много солей и других химических веществ. Они делают воду проводящей электрической ток средой. Работа датчика влажности почвы (рис. 12.12) основана на измерении проводимости грунта, в которую он помещается.

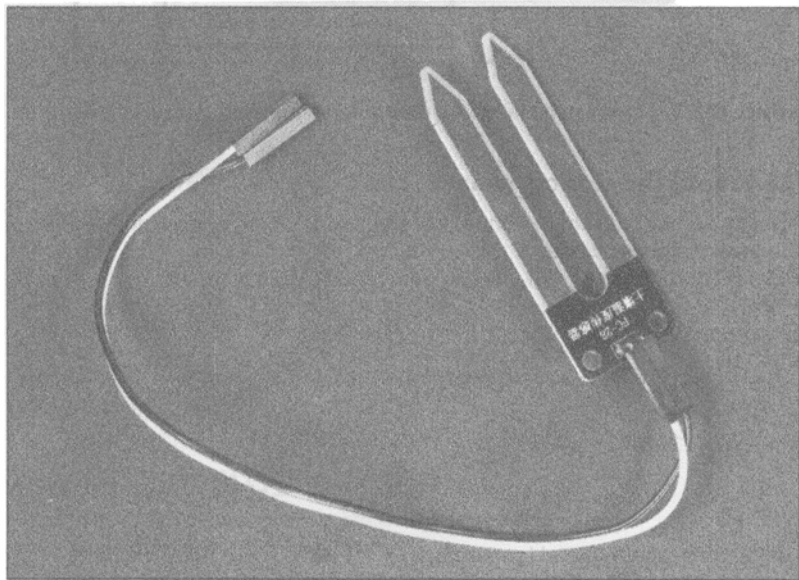


Рис. 12.12. Датчик влажности почвы

Некоторые датчики влажности почвы представляют собой отдельный прибор, не нуждающийся во внешней электронике для проведения измерений. Нам нужен самый простой датчик, который не снабжается автономным электронным управлением.

Исходя из этого создадим самодельное устройство, состоящее из двух металлических щупов (зондов) и платы Arduino или Raspberry Pi, которая будет проводить измерение аналогового сопротивления и выводить данные на экран. Для построения работающей электрической цепи нам понадобится резистор с сопротивлением 1 МОм (маркированный коричневой, черной и зеленой полосками).

Подключение к Arduino и программа управления датчиком влажности почвы

На рис. 12.13 показана схема подключения датчика влажности почвы к Arduino. Программный код управления готовым анализатором приведен в листинге 12.9.

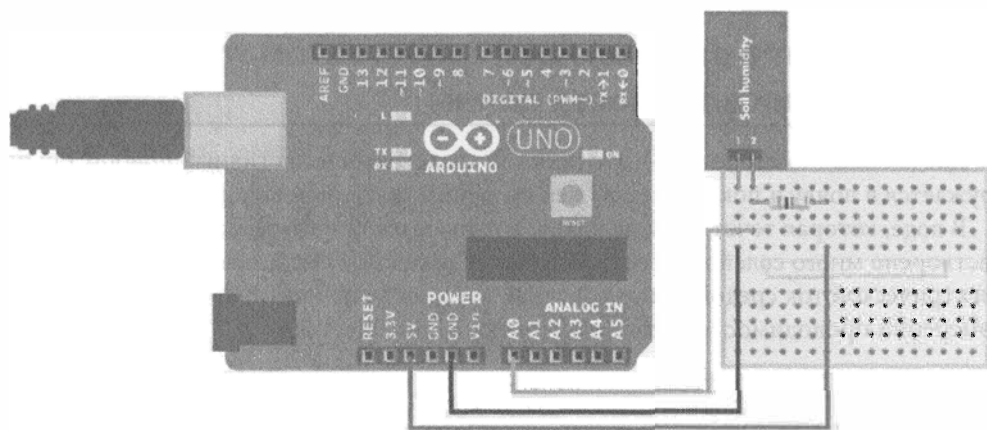


Рис. 12.13. Схема подключения датчика влажности почвы к Arduino

Листинг 12.9. `soil_humidity_sensor.ino`

```
// soil_humidity_sensor.ino - определение влажности почвы по
// измеренному сопротивлению
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = A0;
int soilHumidity = -1;

void setup() {
    Serial.begin(115200);
}

void loop() {
    soilHumidity = analogRead(sensorPin); // ❶
    Serial.println(soilHumidity);
    delay(100);
}
```

❶ Это простой аналоговый резистивный датчик.

Подключение к Raspberry Pi и программа управления датчиком влажности почвы

На рис. 12.14 показана схема подключения датчика влажности почвы к Raspberry Pi. Собрал соответствующую электрическую цепь, выполните программный код, представленный в листинге 12.10.

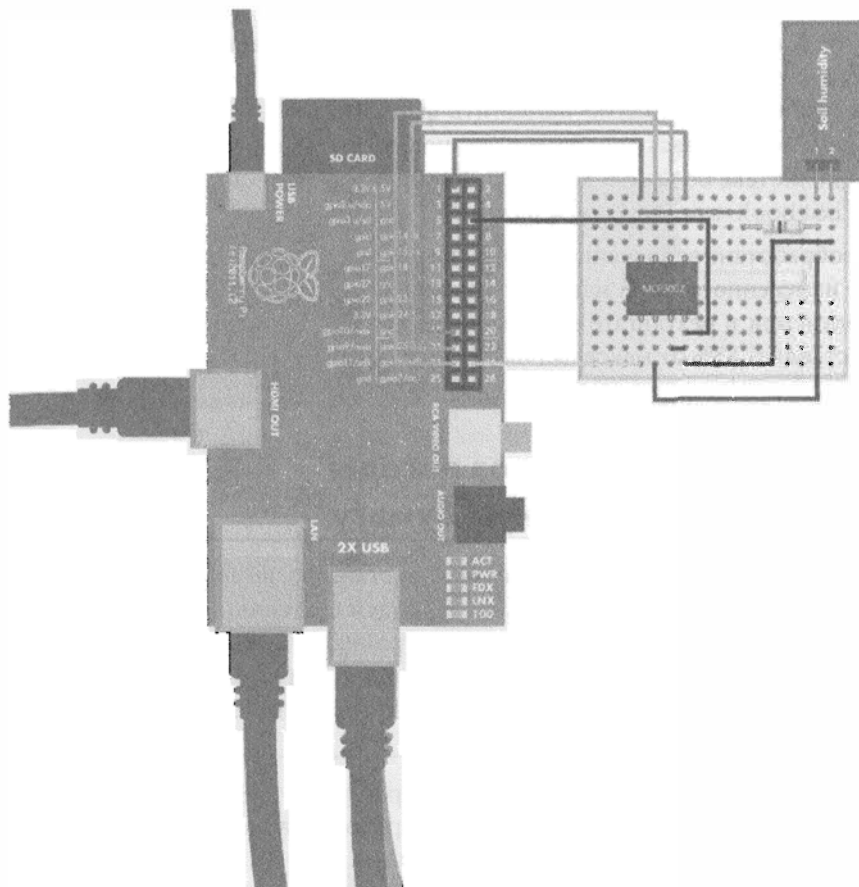


Рис. 12.14. Схема подключения датчика влажности почвы к Raspberry Pi (см. цветную вклейку)

Листинг 12.10. `soil_humidity_sensor.py`

```
# soil_humidity_sensor.py - определение влажности почвы по
# измеренному сопротивлению
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import time
import botbook_mcp3002 as mcp
```

```
def main():
    while True:
```

```
h = mcp.readAnalog() # ❶
h = h / 1024 * 100 # ❷
print("Влажность почвы: %d %% " % h)
time.sleep(5)
```

```
if __name__ == "__main__":
    main()
```

- ❶ Это простой аналоговый резистивный датчик. Как и в случае других датчиков такого же типа, работа которых рассмотрена в книге, в каталоге с файлом текущей программы необходимо поместить файл библиотеки `botbook_mcp3002.py`. Кроме того, от вас потребуется установить библиотеку `spidev`, подключаемую в библиотеке `botbook_mcp3002`. Детально об этом см. в комментариях к файлу `botbook_mcp3002.py` и в главе 3.
- ❷ Необработанное значение преобразуется в процентное значение от максимально измеренного. Если максимально измеренное значение `h` равняется 1, то при получении в результате измерения значения `.53` можно утверждать, что оно составляет 53% от максимального.

Пилотный проект: прогноз погоды с выводом на электронную бумагу

Мы создадим простую метеорологическую станцию, выводящую прогноз погоды на электронную бумагу. Прогноз погоды мы будем составлять на основе наблюдений за изменением атмосферного давления. Электронная бумага в качестве носителя выбрана не случайно: данные на ней прекрасно видны при ярком наружном освещении, а изображение не пропадает даже при отключении питания.

Далее рассматривается самый сложный проект книги. Если вы детально не изучили все ранние проекты, описанные в других главах, то вернитесь к некоторым из них, чтобы восполнить пробелы в знаниях.

Получаемые навыки

Проект метеорологической станции научит вас:

- создавать устройство, выводящее прогноз погоды в графическом виде;
- составлять прогноз погоды на основе данных об изменении атмосферного давления;
- выводить изображения на электронной бумаге, отображающиеся даже при прекращении подачи электропитания;
- переводить Arduino в спящий режим для экономии электроэнергии.



Рис. 12.15. Прогноз погоды, отображенный на электронной бумаге



Рис. 12.16. Монитор на электронной бумаге

Подключение датчиков к Arduino и программа получения прогноза погоды

В следующем коде используется несколько уже изученных ранее методик программирования. Сначала мы загрузим его в микроконтроллерную платформу и выполним, а затем детально разберемся с принципами работы программы.

Чтобы создать собственную версию станции, можно обойтись без понимания всего кода. Добившись нормальной работоспособности станции, внимательно изучите функцию `drawScreen()`, так как именно она отвечает за отображение данных. Вы можете несколько видоизменить ее, например, подкорректировать положение знака “плюс” на экране, задав другое значение переменной `pos`.

```
int pos = 10;
drawCharacter(pos, 70, font, '+');
```

В программном коде рассматриваемого проекта реализуются следующие задачи:

- считывание показаний датчика атмосферного давления GY65 (см. раздел “Датчик атмосферного давления GY65”);
- обработка шестнадцатеричных и двоичных чисел, а также выполнение побитовых операций (см. главу 8);
- перевод Arduino в спящий режим с целью экономии электроэнергии, для чего применяются низкоуровневые команды микроконтроллера ATmega (управляющего работой Arduino);
- отображение результатов на электронной бумаге средствами библиотеки EPD;
- сохранение изображения в виде заголовочного файла (с названием *имя_файла.h*), о чем рассказывается в разделе “Хранение изображений в заголовочных файлах”.

В проекте метеорологической станции применяется плата Arduino Mega. Чтобы выполнить программный код на другой платформе, его нужно частично изменить.

На рис. 12.17 показана схема подключения компонентов проекта метеорологической станции к Arduino Mega. Создав согласно ей электрическую цепь, выполните программный код из листинга 12.11.

Листинг 12.11. `weather_station.ino`

```
// weather_station.ino - вывод прогноза погоды на электронную
// бумагу
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <inttypes.h>
#include <ctype.h>

#include <SPI.h>
#include <Wire.h>
#include <EPD.h> // ❶
```

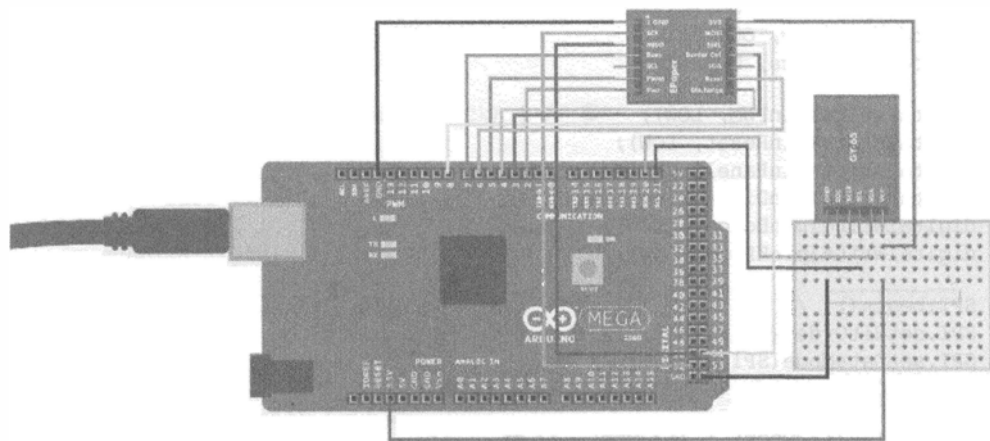



Рис. 12.17. Схема подключения компонентов прибора, выводящего на экран прогноз погоды, на базе Arduino Mega (см. цветную вклейку)

```
#include <gy_65.h> // ❷
#include <avr/sleep.h>
#include <avr/power.h>

#include "rain.h" // ❶
#include "sun.h"
#include "sunccloud.h"
#include "fonts.h"

uint8_t imageBuffer[5808]; // 264*176/8

const int pinPanelOn = 2;
const int pinBorder = 3;
const int pinDischarge = 4;
const int pinPWM = 5;
const int pinReset = 6;
const int pinBusy = 7;
const int pinEPDcs = 8;

EPD_Class EPD(EPD_2_7, pinPanelOn, pinBorder, pinDischarge,
pinPWM, pinReset, pinBusy, pinEPDcs, SPI);

float weatherDiff;
float temperature;

const int sleepMaxCount = 10; // минуты
volatile int arduinoSleepingCount = sleepMaxCount;

void setup() {
  Serial.begin(115200);
  pinMode(pinPanelOn, OUTPUT);
  pinMode(pinBorder, OUTPUT);
  pinMode(pinDischarge, INPUT);
  pinMode(pinPWM, OUTPUT);
```

```

pinMode(pinReset, OUTPUT);
pinMode(pinBusy, OUTPUT);
pinMode(pinEPDcs, OUTPUT);

digitalWrite(pinPWM, LOW);
digitalWrite(pinReset, LOW);
digitalWrite(pinPanelOn, LOW);
digitalWrite(pinDischarge, LOW);
digitalWrite(pinBorder, LOW);
digitalWrite(pinEPDcs, LOW);

SPI.begin(); // ④
SPI.setBitOrder(MSBFIRST); // ⑤
SPI.setDataMode(SPI_MODE0); // ⑤
SPI.setClockDivider(SPI_CLOCK_DIV4); // ⑤

WDTCR |= (1<<WDCE) | (1<<WDE); // ⑤
WDTCR = 1<<WDP0 | 1<<WDP3; // ⑤
WDTCR |= _BV(WDIE); // ⑩
MCUSR &= ~(1 << WDRF); // ⑩

for(int i = 0; i < 5808; i++) // ⑩
    imageBuffer[i] = 0;

readCalibrationData(); // ⑩
}

char characterMap[14] = {'+', '-', 'C', 'd', '0', '1', '2',
    '3', '4', '5', '6', '7', '8', '9'}; // ⑩

void drawCharacter(int16_t x, int16_t y, const uint8_t *bitmap,
    char character) {
    int charIndex = -1;
    for(int i = 0; i < 14; i++) { // ⑩
        if(character == characterMap[i]) {
            charIndex = i;
            break;
        }
    }
    if(charIndex == -1) return;
    drawBitmap(x, y, bitmap, charIndex*25, 0, 25, 27, 350); // ⑩
}

void drawBitmap(int16_t x, int16_t y, const uint8_t *bitmap,
    int16_t x2, int16_t y2, int16_t w, int16_t h, int16_t
    source_width) { // ⑩

    int16_t i, j, byteWidth = source_width / 8;

    for(j=y2; j<y2+h; j++) { // ⑩
        for(i=x2; i<x2+w; i++) {
            byte b= pgm_read_byte(bitmap+j * byteWidth + i / 8);
            if(b & (128 >> (i & 7))) {
                drawPixel(x+i-x2, y+j-y2, true);
            }
        }
    }
}

```

```

    }
}

void drawPixel(int x, int y, bool black) { // ⑩
    int bit = x & 0x07;
    int byte = x / 8 + y * (264 / 8);
    int mask = 0x01 << bit;
    if(black == true) {
        imageBuffer[byte] |= mask;
    } else {
        imageBuffer[byte] &= ~mask;
    }
}

void drawBufferToScreen() { // ⑪
    for (uint8_t line = 0; line < 176 ; ++line) {
        EPD.line(line, &imageBuffer[line * (264 / 8)], 0, false,
        ↵EPD_inverse);
    }
    for (uint8_t line = 0; line < 176 ; ++line) {
        EPD.line(line, &imageBuffer[line * (264 / 8)], 0, false,
        ↵EPD_normal);
    }
}

void drawScreen() { // ⑫
    EPD.begin();
    EPD.setFactor(temperature);
    EPD.clear();
    if(weatherDiff > 250) { // Па
        // Солнечно
        drawBitmap(140,30,sun,0,0,117,106,117); // ⑬
    } else if ((weatherDiff <= 250) || (weatherDiff >= -250)) {
        // Небольшая облачность
        drawBitmap(140,30,suncloud,0,0,117,106,117);
    } else if (weatherDiff < -250) {
        // Дождь
        drawBitmap(140,30,rain,0,0,117,106,117);
    }
    // Вывод температуры
    String temp = String((int)temperature);

    int pos = 10;
    drawCharacter(pos,70,font,'+'); // ⑭
    pos += 25;
    drawCharacter(pos,70,font,temp.charAt(0));
    pos += 25;
    if(abs(temperature) >= 10) {
        drawCharacter(pos,70,font,temp.charAt(1));
        pos += 25;
    }
    drawCharacter(pos,70,font,'d');
    pos += 25;
    drawCharacter(pos,70,font,'C');
}

```

```

drawBufferToScreen(); // ❶

EPD.end();

for(int i = 0; i < 5808; i++) // ❷
    imageBuffer[i] = 0;
}

void loop() {
    Serial.println(temperature); // ❸
    if(arduinoSleepingCount >= sleepMaxCount) { // ❹
        readWeatherData(); // ❺
        drawScreen();
        arduinoSleepingCount = 0; // ❻
        arduinoSleep(); // ❼
    } else {
        arduinoSleep();
    }
}

const float currentAltitude = 40.00; // Установка высоты над
                                   // уровнем моря в метрах
const long atmosphereSeaLevel = 101325; // Па
const float expectedPressure = atmosphereSeaLevel * pow((1-
currentAltitude / 44330), 5.255);

void readWeatherData(){
    temperature = readTemperature();
    float pressure = readPressure();
    weatherDiff = pressure - expectedPressure;
}

ISR(WDT_vect) // ❶
{
    arduinoSleepingCount++;
}

void arduinoSleep() { // ❷
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    sleep_enable();
    sleep_mode(); // ❸
    sleep_disable(); // ❹
    power_all_enable();
} // ❺

```

- ❶ Библиотека, применяемая для вывода данных на электронную бумагу. Можете загрузить ее из Интернета по такому адресу:

<https://github.com/repaper/gratis/tree/master/Sketches/libraries>.

Перед использованием библиотеку необходимо установить в системе.

- ❷ Библиотека управления датчиком GY65, разработанная нами. Она уже использовалась в проекте измерения атмосферного давления (см. раздел “Датчик атмосферного давления GY65”). Разместите копию библиотеки в

каталоге текущего проекта (если вы загрузили файл текущей программы с сайта, указанного во введении, то файл библиотеки у вас уже есть).

- ❸ Изображения сохраняются как заголовочные файлы.
- ❹ Настройка SPI перед передачей через него данных на монитор на электронной бумаге.
- ❺ Настройка SPI так, чтобы первым передавался наиболее значимый бит (о битовых операциях см. в главе 8).
- ❻ Перевод SPI в режим SPI_MODE0: полярность тактового сигнала задается как CPOL 0, а фаза тактового сигнала устанавливается регистром CPHA 0. Буквально это означает, что выборка данных производится по переднему фронту сигнала синхронизации (сигнал изменяется с LOW на HIGH). Передача же данных выполняется по заднему фронту сигнала синхронизации (сигнал изменяется с HIGH на LOW). Режимы (в частности, SPI_MODE0) детально описаны в документации к Arduino (Help⇒Reference⇒Libraries⇒SPI (Справка⇒Содержание⇒Libraries⇒SPI)). Настройки SPI для датчика указываются в технической документации к нему.
- ❼ Команда SPI_CLOCK_DIV4 указывает установить частоту синхросигнала SPI равной 1/4 частоты микропроцессора Arduino. В данном проекте используется плата Arduino Mega, поэтому частота синхросигнала шины SPI будет составлять $16 \text{ МГц} / 4 = 4 \text{ МГц}$.
- ❽ Включение сторожевого таймера. Сего помощью Arduino переводится в спящий режим. Установка флагов WDCE и WDE в регистре WDTCR. Эти команды настолько низкоуровневые, что их назначение описывается в документации к ATMega (а не в базовой документации к Arduino). В Arduino Mega установлен микропроцессор ATMega 1280, поэтому документацию к нему можно найти, задав поиск в Интернете по фразе “ATMega 1280 технические характеристики”. Оператор |= представляет операцию побитового исключающего ИЛИ (операция исключающего ИЛИ выполняется над текущим значением переменной и приводит к замене этого значения полученным результатом). Символ << представляет операцию битового сдвига (см. главу 8).
- ❾ Настройка сторожевого таймера на выход Arduino из спящего режима каждые 8 секунд.
- ❿ Включение поддержки прерываний для сторожевого таймера. Флаг WDIE (Watch Dog Interrupt Enable) извещает о включении режима обработки прерывания для сторожевого таймера.
- ⓫ Сброс флага сторожевого таймера (WDRF) в регистре MCUSR.
- ⓫ Инициализация буфера изображения (в котором хранится изображение до вывода на экран) нулями.
- ⓫ Детально принципы управления датчиком GY65 рассмотрены в разделе “Датчик атмосферного давления GY65”.

- 14 Список из 14 символов в файле `font.h` — большого изображения, представляющего эти символы.
- 15 Поиск индекса символа в файле `font.h`.
- 16 Вывод символа на экран. Обычно шрифт представляется растровым файлом как набор изображений символов, расположенных рядом. Вычисления нужны для того, чтобы выбрать из набора только необходимые символы.
- 17 Функция `drawBitmap()` принимает в качестве аргументов исходное растровое изображение, его положение и размеры. Функция “прорисовывает” каждый пиксель изображения и сохраняет во внутреннем буфере (но не выводит на экран).
- 18 Попиксельная обработка двумерной области рисования, что позволяет в каждой итерации отображать один пиксель изображения.
- 19 Сохранение одного пикселя изображения в буфере (но не вывод на монитор). Каждый пиксель изображения занимает один бит данных. Таким образом, каждый байт (8 бит) содержит данные о 8 пикселях изображения. Ширина экрана составляет 264 пикселя, поэтому каждая строка пикселей занимает в памяти $264/8 = 33$ байта. В процессе попиксельной обработки каждый бит данных изображения хранит значение 1 или 0.
- 20 Передача блока битов “прорисованного” растрового изображения из буфера (`imageBuffer`) на экран. В этой операции принимают участие программные средства библиотеки EPD. Изображение на экране монитора состоит из 176 строчек, каждая из которых содержит 264 пикселя. Поскольку каждый пиксель представляется одним битом данных, то каждая строка изображения занимает $264/8 = 33$ байта. Передача растрового изображения заключается в построчном выводе каждого байта данных из буфера с помощью функции `EPD.line()`.
- 21 В функции `drawScreen()` задействуются как средства библиотеки `gy_65.h` (для получения результатов измерений), так и библиотеки EPD (для вывода результатов на электронную бумагу). Чем большая степень изменения атмосферного давления регистрируется датчиком, тем хуже намечается погода. Если вы хотите создать свою версию программы, то именно функция `drawScreen()` должна стать вашей отправной точкой в изменении настроек метеорологической станции.
- 22 Занесение изображения из файла `sun.h` в буфер для последующего вывода на экран. С помощью этой функции можно вывести на экран любой другой растровый файл. Функция `drawBitmap` имеет такой вид:

```
drawBitmap(imageBufferX, imageBufferY, source Image,
    sourceX, sourceY, sourceWidth, sourceHeight,
    totalSourceWidth)
```

Первые два параметра (`imageBufferX`, `imageBufferY`) указывают положение вставляемого изображения. Остальные параметры определяются характеристиками самого изображения: имя файла растрового изображения

(sourceImage) и его геометрические размеры (sourceX, sourceY, sourceWidth, sourceHeight). Последний параметр — это общая ширина растрового изображения (totalSourceWidth).

- 33 Добавление символа в буфер.
- 34 Вывод данных из буфера (imageBuffer) на электронную бумагу. Без этой операции монитор метеорологической станции будет пустовать.
- 35 Очистка буфера изображения.
- 36 Используя малознакомые устройства, такие как дисплей на электронной бумаге, всегда неплохо подстраховаться и продублировать получаемые с датчика данные на мониторе последовательного порта.
- 37 Сторожевой таймер периодически выводит Arduino из режима сна. Если прошло достаточно времени...
- 38 ...то выполняется основная функциональная часть программы.
- 39 Сброс счетчика спящего режима...
- 40 ...и переход в режим сна для экономии электроэнергии.
- 41 Программа обслуживания прерываний, или ISR (Interrupt Service Routine), вызывается автоматически и запускается при каждом выходе Arduino из спящего режима до выполнения функциональной части основной программы. В текущих инструкциях указывается выводить Arduino из спящего режима каждые 8 секунд (ISR () вызывается каждые 8 секунд).
- 42 Переход Arduino в спящий режим для экономии электроэнергии. Как видите, это требует серьезной предварительной подготовки.
- 43 Эта команда указывает Arduino прекратить выполнение программного кода и перейти в спящий режим. Выполнение программы продолжается только после выхода Arduino из спящего режима по команде сторожевого таймера.
- 44 После “пробуждения” Arduino выполняет программный код с текущей строки.
- 45 Вам удалось разобраться в хитросплетении программного кода? Можете искренне гордиться собой, ведь совсем скоро вы станете мастером программирования!

Эксперимент в окружающей среде: без источника питания

В электронной бумаге питание востребовано только при изменении изображения на экране. При выводе статического изображения на электронной бумаге электроэнергия не расходуется.

Вы можете это легко проверить. С помощью Arduino выведите на монитор на электронной бумаге некое изображение. Затем прекратите подачу электропитания на Arduino или же просто отключите монитор от розетки (аккумуляторов). Изображение останется неизменным и никуда не денется. При отображении статической картинки вы не заметите совсем никакой разницы, если отключите монитор от источника питания (рис. 12.18).

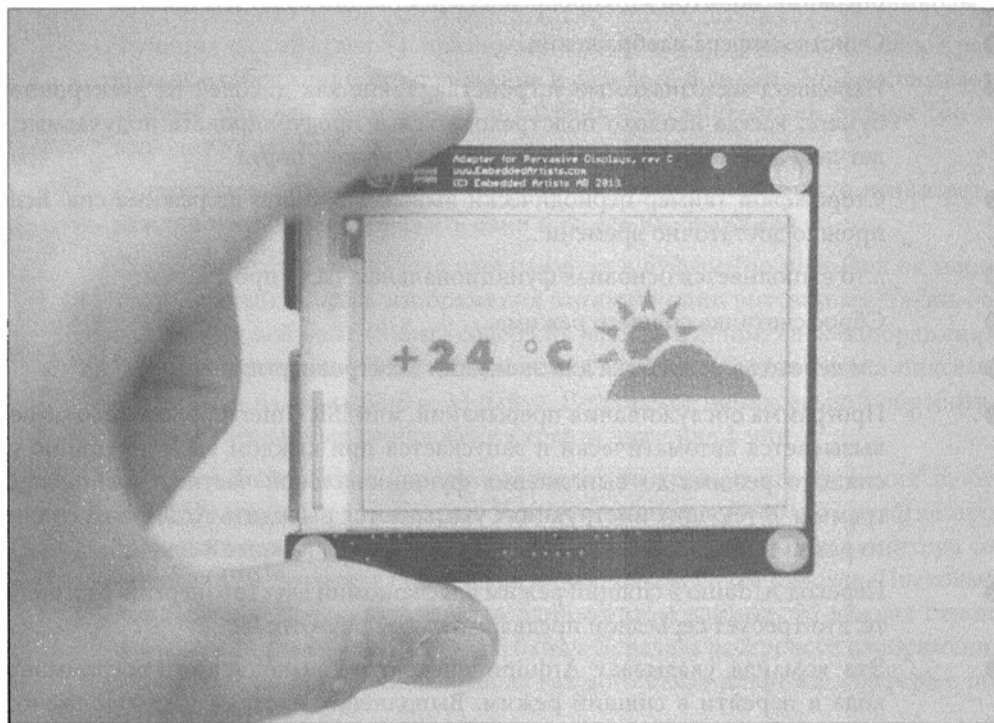


Рис. 12.18. Статическое изображение продолжает демонстрироваться на электронной бумаге даже после прекращения подачи питания

Хранение изображений в заголовочных файлах

Только что созданная вами метеорологическая станция безусловно должна уметь отображать на экране символ хорошей солнечной погоды. Как же быть, если такой символ представляется пользовательским изображением? Вы можете нарисовать такой символ в любом графическом редакторе, например Gimp или Photoshop, а затем сконvertировать сохраненное в растровом формате изображение в заголовочный файл, написанный на языке C, чтобы иметь возможность подключать его к проектам Arduino. Сначала создайте необходимое изображение в любимом графическом приложении. В качестве бесплатного графического редактора используйте программу GIMP. Сохраните растровое изображение в формате BMP. Разместите файл изображения в том же каталоге, что и другие графические файлы проекта (images/).

Теперь вам нужно сконвертировать изображение формата BMP (`sun.bmp`) в заголовочный файл `C` (`sun.h`). Для этого воспользуйтесь прилагаемым к проекту файлом сценария `image2c-botbook.py`.

Вначале настройте среду разработки: установите Python и Python Imaging Library. Пользователи Windows могут загрузить установочный пакет Python с сайта <http://python.org>. Пользователи Mac получают среду разработки Python автоматически при установке операционной системы, хотя и в Mac, и в Windows необходимо дополнительно установить программный пакет Python Imaging Library, загружаемый с такого сайта:

<http://www.pythonware.com/products/pil/>

Пользователям Linux проще всего. Для установки всех необходимых библиотек им нужно выполнить всего две команды:

```
$ sudo apt-get update
$ sudo apt-get -y install python python-imaging
```

Если в вашем случае файл изображения называется не `foo.bmp`, то измените имя файла в сценарии соответствующим образом. Теперь все готово для конвертирования файла. Приведенные ниже команды выполняются в Linux, Windows и Mac. Приглашение, представленное символом доллара (`$`), вам вводить не нужно. Обратите внимание на то, что пользователи Windows могут увидеть в командной оболочке несколько иной символ приглашения. Первая команда выполняется в предположении, что вы находитесь в каталоге с файлом, содержащим программный код рассматриваемого проекта. Этот файл доступен для загрузки на сайте, указанном во введении.

```
$ cd arduino/weather_station/
$ python image2c-botbook.py
BMP (117, 106) 1
```

Файл преобразован в необходимый формат. Можете убедиться в этом, воспользовавшись командой `ls` или `dir` для нахождения файла `foo.h` в рабочем каталоге. Таким образом, из растрового BMP-изображения вы получили заголовочный файл `C`. В файле `foo.h` растровое изображение представляется программным кодом `C`.

```
// файл, сгенерированный сценарием image2c-botbook.py
const unsigned char sun [] PROGMEM= {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ..
0x7F, 0xF0, 0x00, 0xFF, 0xC0, 0x00, 0x1F, 0xF8, 0x00, // ..
```

Такой формат изображения очень удобен для разработки программ. Каждый байт данных содержит сведения о восьми пикселях растрового рисунка. Например, шестнадцатеричное значение `0xFC` соответствует числу 252. В битовом представлении оно выглядит как `0b11111100`. В растровом изображении это двоичное число обозначает восемь рядом расположенных пикселей: черный, черный, черный, черный, черный, черный, белый, белый.

Преобразование растровых файлов в программный код `C`

За преобразование BMP-изображения в заголовочный файл `C` в нашем случае отвечает сценарий `image2c-botbook.py`.

При необходимости можете проделать эту же операцию над PNG-файлами.

В программном коде сценария применяются шестнадцатеричные числа и побитовые операции, с которыми вы познакомились в главе 8.

Листинг 12.12. Преобразование графического файла в заголовочный файл C

```
# image2c-botbook - конвертирование BMP-изображения в файл C для
# вывода на ЖК-мониторы и экраны на электронной бумаге
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
import Image # ❶
import math
imageName = "images/foo.bmp" # ❷
outputName = "foo.h" # ❸

im = Image.open(imageName) # ❹
print im.format, im.size, im.mode

width, height = im.size
pixels = list(im.getdata()) # ❺
length = int(math.floor(width * height / 8.0)) # ❻
carray = length * [0x00] # ❼
    for y in xrange(0, height): # ❶
        for x in xrange(0, width): # ❷
            pixel = pixels[y * width + x] # ❸

            bit = 7 - x & 0x07 # ❹
            byte = x / 8 + y * (width / 8) # ❺

            mask = 0x01 << bit # ❻
            if pixel == 0:
                carray[byte] |= mask # ❼
            else:
                carray[byte] &= ~mask

fileHandle = open(outputName, "w") # ❶
index = 0
fileHandle.write("//Преобразован сценарием image2c-botbook.py\n")
fileHandle.write("const unsigned char sun [] PROGMEM= {\n")
for b in carray:
    fileHandle.write("0x%02X, " % b) # ❸
    index += 1
    if index > 15:
        fileHandle.write("\n")
        index = 0
fileHandle.write("};\n")
```

- ❶ Программный пакет PIL (Python Imaging Library) необходимо предварительно установить так, как описывалось выше.
- ❷ Исходное изображение, сохраненное в формате BMP. Измените имя файла и путь его расположения по своему усмотрению.
- ❸ Имя преобразованного заголовочного файла. По необходимости замените его своим вариантом.

- 4 Библиотека PIL поддерживает преобразование многих графических форматов.
- 5 Разделение изображения на отдельные пиксели. Половина задачи решена.
- 6 Размер файла в байтах (1 байт = 8 бит).
- 7 Создание целевого массива C и заполнение его нулями.
- 8 Для каждой строки...
- 9 ...и каждого пикселя строки выполняются идентичные операции.
- 10 Текущий пиксель.
- 11 Индекс бита в текущем байте.
- 12 Индекс текущего байта.
- 13 Создание битовой маски для текущего бита из текущего байта. Например, маска 0b00000001 создается для изменения последнего бита текущего байта на ноль.
- 14 Смена бита.
- 15 Заголовочные файлы C представляются в формате обычных текстовых файлов.
- 16 Сохранение байтов в шестнадцатеричном коде.

Корпус для метеостанции

В качестве корпуса для метеорологической станции мы использовали многофункциональную пластиковую коробку. Вы можете приспособить под корпус что угодно, главное, чтобы размер был подходящим. Чтобы прорезать окошко под экран на электронной бумаге, сначала разметьте его форму на задней крышке. Просверлите четыре отверстия (диаметр сверла 10–20 мм) в углах будущего окна, а затем лезвием или лобзиком прорежьте боковые стороны. Окончательно отшлифуйте края полученного прямоугольника напильником и наждачной бумагой (рис. 12.19).

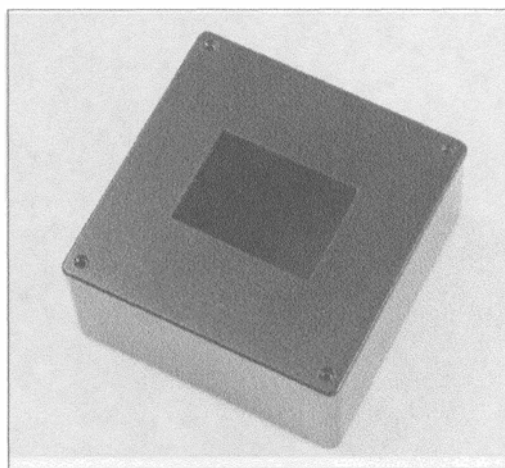


Рис. 12.19. Окно для экрана

Закрепите экран на электронной бумаге с помощью термоклея, аккуратно выровняв его по краям отверстия, как показано на рис. 12.20.

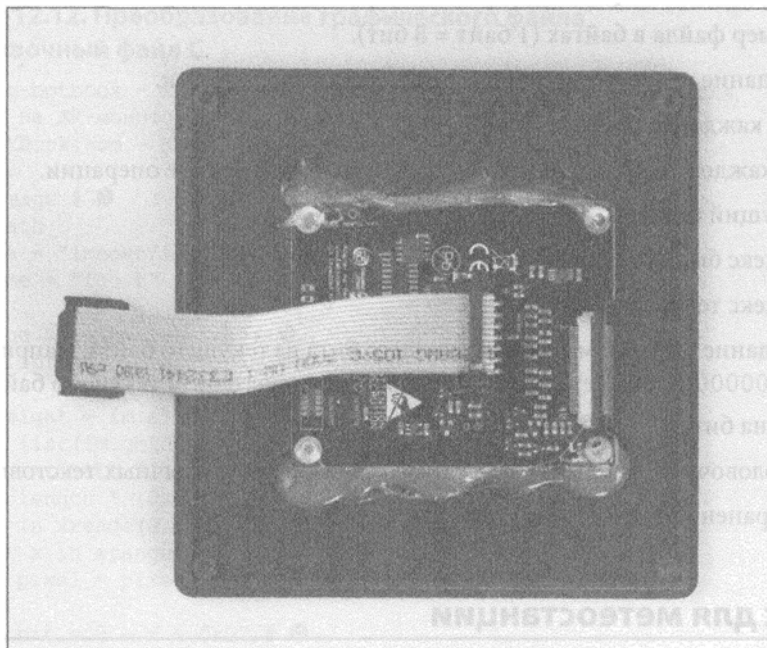


Рис. 12.20. Монитор закрепляется термоклеем

С тыльной стороны корпуса мы просверлили несколько аккуратных вентиляционных отверстий, позволяющих воздуху свободно заходить внутрь прибора и беспрепятственно выходить наружу (рис. 12.21). Если не обеспечить вентиляцию устройства, то существует вероятность его перегрева и нарушения нормальной работоспособности метеостанции. Готовый прибор выглядит так, как показано на рис. 12.22.

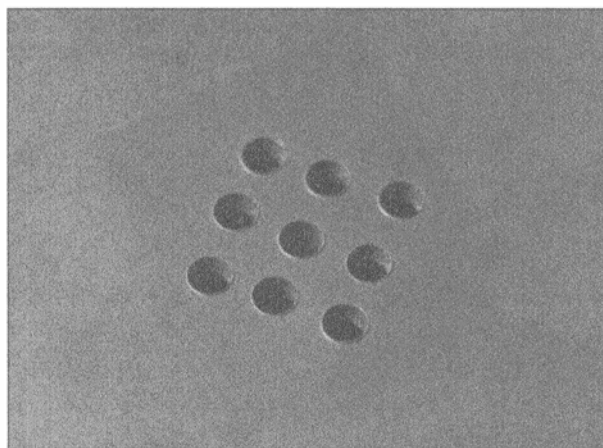


Рис. 12.21. Вентиляционные отверстия

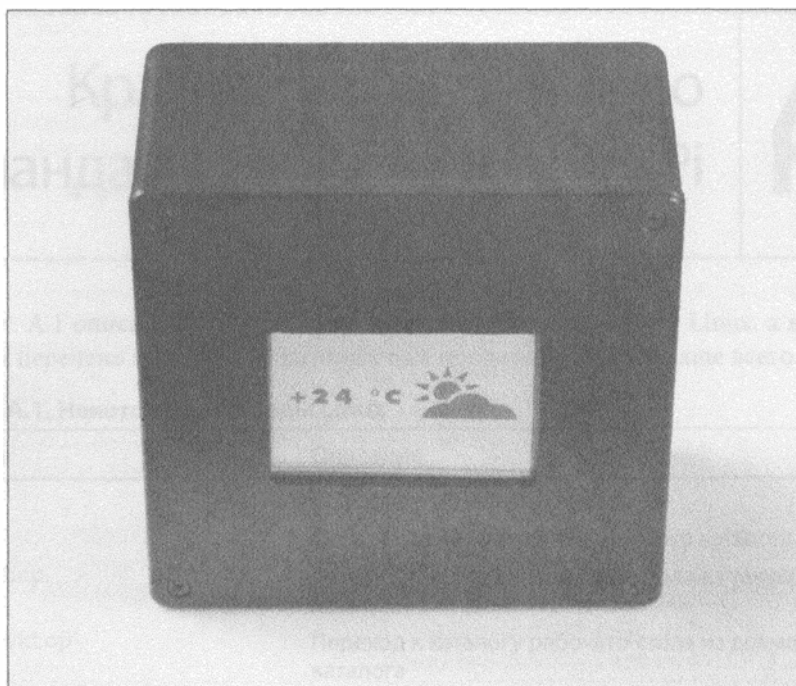


Рис. 12.22. *Готовое устройство*

Вот и все! Мы с вами закончили создание простой метеорологической станции, выводящей прогноз погоды на экран на электронной бумаге. На этом проекте также заканчивается данная книга. Но хочется верить, что ваша карьера разработчика микроконтроллерных устройств только начинается. Теперь, когда вы хорошо знакомы с принципами использования датчиков самых разных типов, разве есть проект, который вам не по силам? Хочется искренне пожелать вам удачи!

Краткий справочник по командам Linux в Raspberry Pi



В табл. А.1 описаны команды, наиболее часто используемые в Linux, а в табл. А.2 приведен перечень каталогов, в которых вам придется работать чаще всего.

Таблица А.1. Некоторые команды Linux

Команда	Описание
<code>pwd</code>	Отображение текущего каталога
<code>ls</code>	Отображает содержимое текущего каталога
<code>cd Desktop</code>	Переход к каталогу рабочего стола из текущего каталога
<code>cd ~/Desktop</code>	Переход к каталогу рабочего стола из домашнего каталога
<code>cd ~</code>	Переход к домашнему каталогу
<code>nano foo.txt</code>	Редактирование файла. Нажмите комбинацию клавиш <Ctrl+C>, далее клавишу <Y>, а затем <Enter> или <Return>, чтобы сохранить файл
<code>passwd</code>	Изменяет пароль (сначала запрашивается старый пароль, но не отображает вводимые вами символы)
<code>startx</code>	Загружает графический интерфейс из сеанса командной оболочки
<code>sudo apt-get update</code>	Обновляет список доступного для загрузки программного обеспечения (требуется сетевое соединение)
<code>sudo apt-get install ipython</code>	Устанавливает программу ipython
<code>sudo shutdown -P now</code>	Подготовка Raspberry Pi к безопасному выключению питания
<code>sudoedit /etc/motd</code>	Редактирует файл от имени суперпользователя, не считаясь с безопасностью системы, в отличие от команды <code>sudo nano /etc/motd</code>
<code>ifconfig</code>	Отображает IP-адрес сетевого адаптера Raspberry Pi. Адрес 127.0.0.1 представляет текущую систему (localhost) и используется для обмена данными между программами, запущенными в Raspberry Pi. Найдите адрес, указанный для адаптера Ethernet или Wi-Fi
<code>sudo raspi-config</code>	Выводит меню конфигурирования большинства плат Raspberry Pi (вам обязательно нужно будет перезагрузить систему)
<code>mkdir bar/</code>	Создает каталог bar/

Команда	Описание
<code>rm -r bar/</code>	Удаляет каталог <code>bar/</code> и все его содержимое без возможности отмены выполненного действия
<code>ssh login@example.com</code>	Подключение к удаленному компьютеру <code>example.com</code> как пользователь <code>login</code>
<code>exit</code>	Выход из оболочки или SSH-сеанса
<code>less /var/log/syslog</code>	Отображает текстовый файл с возможностью страничного просмотра (нажмите клавишу <Пробел> для отображения следующей страницы и клавишу <Q> для выхода)
<code>tail -F /var/log/syslog</code>	Отслеживает содержимое указанного текстового файла по мере добавления в него текста (комбинация клавиш <Ctrl+C> завершает команду и инициирует переход к командной оболочке)
<code>man ssh</code>	Просмотр справочного руководства для команды <code>ssh</code> (нажмите клавишу <Пробел> для отображения следующей страницы и <Q> для выхода)

Таблица А.2. Часто используемые каталоги

Каталог	Назначение
<code>/home/pi/</code>	Ваш (пользователя <code>pi</code>) домашний каталог, содержащий все файлы Raspberry Pi
<code>/var/log/</code>	Содержит все системные файлы журналов, например <code>/var/log/syslog</code> и <code>/var/log/auth.log</code>
<code>/etc</code>	Содержит все файлы системных настроек
<code>/sys</code>	Виртуальная файловая система для считывания и обработки меняющихся данных (непрерывно изменяющихся в процессе функционирования системы, например входных и выходных портов)
<code>/media</code>	Съемные накопители, например <code>/media/cdrom/</code> или <code>/media/usbdisk/</code>
<code>/</code>	Корневой каталог, включающий все остальные каталоги и файлы системы

Предметный указатель

A

Adafruit, 26; 94
Android, 37; 269
Apache, 40; 356
Arduino, 57
 Leonardo, 296
 Mega, 296; 309; 310; 406
 Uno, 58
ASCII, 118; 264; 309

B

Bash, 197

G

GPIO, 42; 45

H

HDMI, 30; 189; 194; 368
 безопасный режим, 32
 монитор, 32

I

I²C, 22; 42; 251; 261; 346
IDE, 58
IMAP, 116
IP-адрес, 356

L

Linux, 31; 34; 37; 46; 261
LNK, 36
LXTerminal, 46; 93; 197; 261; 356

M

Maker Shed, 26
Midori, 38
MIME, 118

P

Parallax, 26; 66
PWR, 35
Python, 53; 115; 347; 415

R

Raspberry Pi, 29
Raspbian, 32; 356
RGB, 192; 218
 светодиод, 224
RX, 61

S

SMTP, 116
SparkFun, 26
SPI, 21; 42; 93; 251; 411
sudo, 39

T

TTL, 309
TX, 61

U

Ubuntu, 58
USB, 290

W

Wi-Fi, 30; 62
Wii Nunchuk, 269
WLAN-адаптер, 30

A

АЦП, 107; 137; 146; 224; 388
Автоматическая регистрация, 198
Автоматический запуск, 197
Адаптер
 Ethernet, 62
 контроллера, 275
Адресное пространство, 255
Акселерометр, 175; 243; 273; 340; 344
 двунаправленный, 244
Активный датчик, 66
Алкотестер, 110
Амплитуда колебаний, 372
Анализатор газа, 104
Аналого-цифровой преобразователь, 89;
 92; 107; 137; 146; 224

Аналоговый вход, 42
Аналоговый датчик, 21; 92
Анод, 44; 225; 228
Антирадар, 79
Атмосферное давление, 389

Б

Байт, 266
Беспачное макетирование, 43
Библиотека, 392
 SpiDev, 92
 импорт, 54; 72
 установка, 93
Биометрическая идентификация, 285
Бит, 266
Битовое маскирование, 267
Битовый сдвиг, 268
Блок инерциальных измерений, 250
Булево значение, 205
Булевый оператор, 266
Буфер, 302; 312; 314; 385
Быстрое преобразование Фурье, 363; 372

В

Веб-сервер, 38; 356
Ведущее устройство, 294; 346
Вектор, 346
Взлом сигнализации, 185
Вибродатчик, 166
Вибродвигатель, 180
Видеоискатель, 66
Визуализация звука, 369
Виртуальный файл, 47
Владелец файла, 52
Влажность, 382; 386
Восьмеричное число, 262
Время, 183
Высота над уровнем моря, 391; 397

Г

Газ, 103
Газовый датчик, 104
Гироскоп, 175; 244
Глобальная переменная, 89; 92; 177; 236
График, 352
Графический эквалайзер, 368
Громкость, 363

Д

Давление, 144
 атмосферное, 389
Дактилоскопический сканер, 294
Дальномер, 66
Датчик, 19
 FlexiForce, 144
 Ping, 66
 Холла, 331; 348
адаптация, 183
активный, 66
аналоговый, 21; 90; 92; 104
атмосферного давления, 389
вибрации, 166
влажности, 382
 почвы, 401
давления, 125
движения, 181
дыма и газа, 103
емкостный, 140
импульсный, 21
инфракрасный, 80; 181
нажима, 144
наклона, 163
напряжения, 327; 355
опрос, 167
пламени, 201
прикосновения, 140; 148
пьезорезистивный, 144
резистивный, 87; 107; 208
составной, 85
срабатывание, 160
температуры, 378
тока, 327; 355
угла поворота, 170
ультразвуковой, 65; 73
уровня алкоголя, 110
ускорения, 244
установка, 120
цвета, 218
цифровой, 21
чувствительность, 149; 220
Двоичное число, 262
Двунаправленный датчик ускорения, 244
Двухкоординатный джойстик, 174
Демон, 40; 52
Детектор линий, 212

- Джойстик, 195
 - под большой палец, 174
- Длина волны, 218
- Длительность
 - импульса, 153; 154; 245; 250
 - сигнала, 72; 244
- Домашний каталог, 198
- Домашняя страница, 357
- Дополнительный модуль, 62
- Дребезг контактов, 129
- Дым, 103
 - концентрация, 105
- Дымовая сигнализация, 114
- Е**
- Единицы измерения, 97; 266
 - СИ, 70
- Емкостный датчик, 140
- Естественное освещение, 89
- Ж**
- Жесткий диск, 243
- З**
- Завершение программы, 148
- Заголовочный файл, 393; 415
- Загрузчик, 58
- Задержка, 55; 72; 119; 148; 156; 260
- Заземление, 140; 149; 273
- Запись звука, 363
- Запрос данных, 260
- Заряд, 140
- Звонок, 151
- Звук, 78; 97; 363
 - скорость, 70
- Земля, 44
- Зуммер, 94; 98; 186
- И**
- ИК-датчик, 80; 181
- ИК-излучатель, 84
- Игровой контроллер, 179
- Идентификатор
 - ошибки, 295
 - подтверждения приема, 295
 - устройства, 302
- Идентификационный номер, 309
- Идентификация, 285
 - отпечатка пальца, 302
- Импульс, 170
 - длительность, 153; 245
- Импульсный датчик, 21
- Индикатор
 - L, 61
 - LNK, 36
 - PWR, 35
 - RX, 61
 - TX, 61
- Инициализация
 - класса, 260
 - контроллера, 273
 - соединения, 256
- Интенсивность свечения, 230
- Интерфейс, 21
 - I²C, 22; 251
 - SPI, 22; 93; 251
 - командный, 37
- Инфракрасное зрение, 83
- Инфракрасное излучение, 65; 80; 181; 201
- К**
- Калибровка, 78; 397; 400
 - датчика, 220
 - компаса, 336; 344
- Карта памяти, 30
 - форматирование, 35
- Касание, 125; 140
- Катод, 44; 225
- Клавиатура, 32
 - цифровая, 286
- Класс, 260
- Кнопка, 125; 316; 324
 - нажатие, 126
 - состояние, 177
- Колебания, 78; 95; 363
- Командная оболочка, 37; 197; 265
- Командная строка, 40
- Командный терминал, 59
- Компас-акселерометр, 334
- Компонент, 26
- Конденсатор, 158
- Консоль Python, 263; 265
- Константа, 89
- Контакт, 45
 - состояние, 92; 127
 - экспорт, 55

- Контроллер
Wii Nunchuk, 269
игровой, 180
- Контрольная сумма, 301; 307; 314; 386
- Контроль осанки, 94
- Координатное пространство, 192
- Корпус, 237; 417
- Кортеж, 224; 315
- Коэффициент заполнения, 230; 246
- Л**
- Линейное преобразование, 231
- Логическое значение, 267
- М**
- Магнетометр, 340; 344
- Магнитное поле, 325; 330
- Магнитный полюс, 325
- Макетная плата, 122
- Манипулятор, 277
- Маршрут, 212
- Массив, 273
- Менеджер пакетов, 39
- Метка радиочастотной идентификации, 308
- Механический манипулятор, 277
- Микроконтроллер, 57
- Микропереключатель, 125; 131
- Микросхема, 108
- Микрофон, 363
чувствительность, 367
- Модуль, 62
радиочастотной идентификации, 308
составного датчика, 86
- Монитор порта, 68; 82; 183
- Монтажный провод, 43
- Мощность тока, 326
- Мультиметр, 326
- Мышь, 32
- Н**
- Нажим, 144
- Накопление заряда, 149
- Наложение звуковых волн, 373
- Направленное освещение, 210
- Напряжение, 326; 352; 377
считывание, 108; 129
- Напряженность магнитного поля, 330
- Нарастающий фронт, 170
- Начальная страница, 357
- Низкоуровневая операция, 266
- Нормализация вектора, 345; 348
- Нормальное атмосферное давление, 390
- О**
- Обработчик событий, 193
- Общий катод, 225
- Объявление переменных, 236; 273
- Операционная система, 31
Android, 37
Linux, 37
Raspbian, 32
настройка, 39
- Опрос датчика, 167
- Освещение
автоматическое, 163
направленное, 210
- Отпечатки пальцев, 292; 316
- Отсроченное задание, 360
- Оттенок, 219
- Охранная сигнализация, 144; 181
- П**
- Парковка головок диска, 243
- Пароль, 118
ввод, 34
- Передача данных, 61; 139
- Переключатель, 164
на эффекте Холла, 348
- Переменная
глобальная, 89; 177
объявление, 91
- Перемигивание, 43
- Период сигнала, 97; 246
- Плавное изменение яркости, 230
- Плавность, 237
- Пламя, 201
- Планировщик, 360
- Плата, 122; 197
Arduino, 60; 127
беспачного макетирования, 43
расширения, 62; 232
- Побитовая операция, 266
- Побитовое И, 139
- Побитовое ИЛИ, 268
- Подтягивающий резистор, 127; 172; 288
- Подчиненное устройство, 255; 294
- Пожарная сигнализация, 106

- Полнооборотный сервопривод, 152
- Порт
 - ввода-вывода, 53
 - общего назначения, 42
 - последовательный, 61; 68; 183; 302
- Порядок следования байтов, 269
- Последовательное соединение, 21; 251
- Последовательный порт, 61; 68; 82; 183; 302; 303
- Потенциометр, 104; 135; 175
 - подстроечный, 155; 205; 216
- Почтовое сообщение, 114
- Почтовый клиент, 116
- Правило третьей полосы, 50
- Прерывание, 72; 167; 171; 411
 - срабатывание, 168
- Прибор ночного видения, 84
- Привилегии, 53; 93; 262
- Привилегированный пользователь, 39
- Приглашение командной строки, 40; 197
- Прием данных, 61
- Прикосновение, 125; 140
- Проволочная перемычка, 43
- Прогноз погоды, 404
- Промышленный интерфейс, 251
- Промышленный протокол, 42
- Пропускная способность, 68
- Пространство
 - адресное, 255
 - имен, 72; 179
- Протокол
 - I²C, 42
 - SPI, 42
 - обмена данными, 393
- Прямоугольный сигнал, 95; 153; 246
- Пьезозуммер, 186
- Пьезорезистивный датчик, 144
- Пьезоэлектрик, 95
- Р**
 - Рабочее поле, 192
 - Рабочий диапазон, 154
 - Рабочий режим, 181; 202
 - Радиочастотная идентификация, 285; 308
 - Размыкание контактов, 127
 - Разъем, 43
 - Arduino, 57
 - Расстояние, 70; 72
 - измерение, 65
 - Растровое изображение, 414
 - Регистр, 255
 - Регистрация пользователя, 198
 - Регулятор, 170; 205
 - громкости, 163
 - Редактирование файлов, 199
 - Режим
 - входа, 83
 - прерывания, 72
 - работы датчика, 181
 - считывания сигнала, 128
 - Резистивный датчик, 87; 208
 - Резистор, 48; 149
 - номинал, 50
 - переменный, 135
 - подтягивающий, 127; 145; 166; 172; 288
 - Роботозахват, 277
- С**
 - СИ, 70
 - Самобалансирующееся устройство, 243
 - Свет, 201
 - Светодиод, 47; 85; 206; 220
 - RGB, 224
 - включение, 54
 - встроенный, 127; 204
 - мигание, 60; 137
 - Сдвиг битов, 139
 - Сенсорная поверхность, 143
 - Сенсорный звонок, 151
 - Сервопривод, 152; 277; 316
 - тестирование, 154
 - Сигнализация
 - взлом, 185
 - дымовая, 114
 - охранная, 144; 181
 - пожарная, 106
 - Сигнал
 - аналоговый, 137
 - вывод, 62
 - высокого уровня, 62
 - длительность, 72; 244
 - звуковой, 70
 - низкого уровня, 140
 - отраженный, 75

период, 97
прямоугольный, 95; 153
с микрофона, 365
считывание, 83; 204; 221
Сила тока, 326
Синхроимпульс, 170
Сирена, 97
Система счисления, 262
Сканер отпечатков пальцев, 294
Скетч, 58
Скользящее среднее, 229; 237; 277; 281
Скорость
 вращения, 348
 звука, 70
 угловая, 244
Событие, 167; 193
Соппротивление, 127
Составной датчик, 85
Спадающий фронт, 170
Спящий режим, 413
Среда разработки, 58
Стандартизированный интерфейс, 251
Статическое изображение, 414
Степенная функция, 232
Сторожевой таймер, 411
Структура, 255
Суперпользователь, 39; 46; 93; 200
Сценарий, 52; 198
Считывающее устройство, 309

Т

ТТЛ, 77
Текстовый файл, 39
Тело почтового сообщения, 119
Температура, 377
 среды, 78; 255
Температурный датчик, 378
Терминал, 58; 265
Термоусаживаемая пленка, 87
Тип данных, 255
Трехмерное пространство, 244
Трехмерный вектор, 346

У

Угарный газ, 105
Углекислый газ, 103
Угловая скорость, 244; 256
Угол поворота, 154; 170

Указатель, 302
Ультразвук, 65
 датчик, 73
Уровень громкости, 363
Ускорение, 243; 244; 273
 измерение, 249
 свободного падения, 244; 256
Устройство
 ввода-вывода, 32
 ведомое, 255
 ведущее, 346
 самобалансирующееся, 243

Ф

Файл, 47
 библиотеки, 392
 виртуальный, 47
 владелец, 52
 редактирование, 199
 сценария, 198
 текстовый, 39
Файловая система, 46
Фасеточный глаз, 85
Форматирующая строка, 108; 119; 148; 224;
 260
Фоторезистор, 85; 89; 205; 206
Фотоэлемент, 352; 353
Фронт сигнала, 170
Функция, 89; 97; 156
 обратного вызова, 168
 плавности, 237
 преобразования чисел, 264
 степенная, 232

Ц

Цвет, 201; 218
Целочисленное значение, 255
Цикл, 156; 193; 257
Цифровая клавиатура, 286
Цифровой вход, 42
Цифровой выход, 42
Цифровой датчик, 21

Ч

Частота, 97; 218
 колебаний, 372
 сигнала, 153; 246
 смены кадров, 193

Чувствительность

датчика, 149

детектора линий, 216

микрофона, 367

Ш

ШИМ, 230

Шестнадцатеричное число, 262

Шилд, 62

Шина, 255; 346

Широтно-импульсная модуляция, 230

Шлейф, 232

Штрих-код, 285

Шум, 229

Щ

Щуп, 133

Э

Эквалайзер, 368

Электрическая цепь, 44

Электричество, 325

Электромагнитная волна, 218

Электронная бумага, 404

Электронная почта, 115

Электронный компас, 325; 335; 346

Электронный компонент, 25

Энергонезависимая память, 397

Эффект

Холла, 331; 348

пьезоэлектрический, 95

Эхо-сигнал, 67; 77

Я

Ядро, 36

Яркость, 228

максимальная, 230

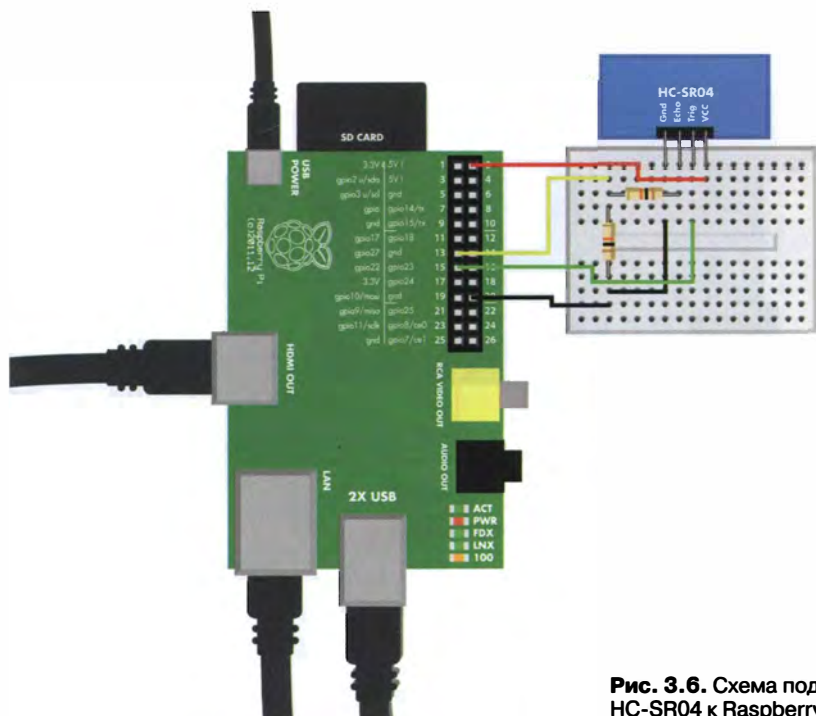


Рис. 3.6. Схема подключения датчика HC-SR04 к Raspberry Pi

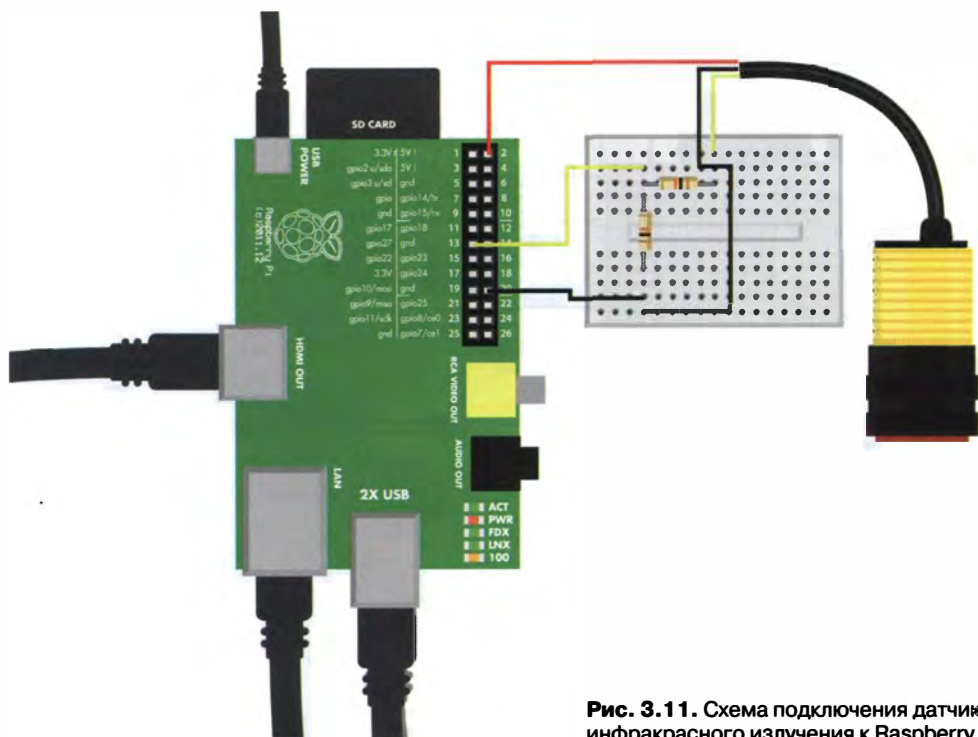


Рис. 3.11. Схема подключения датчика инфракрасного излучения к Raspberry Pi

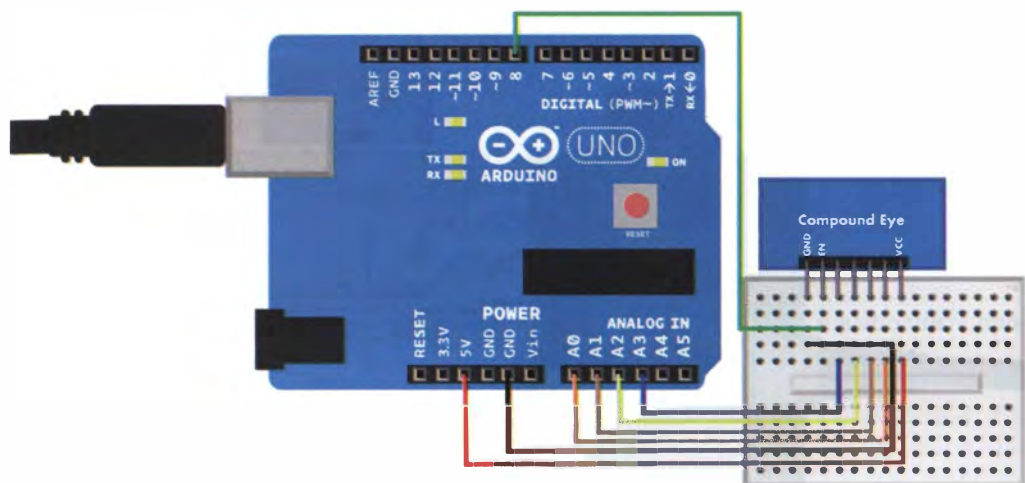


Рис. 3.16. Схема подключения составного ИК-датчика к Arduino

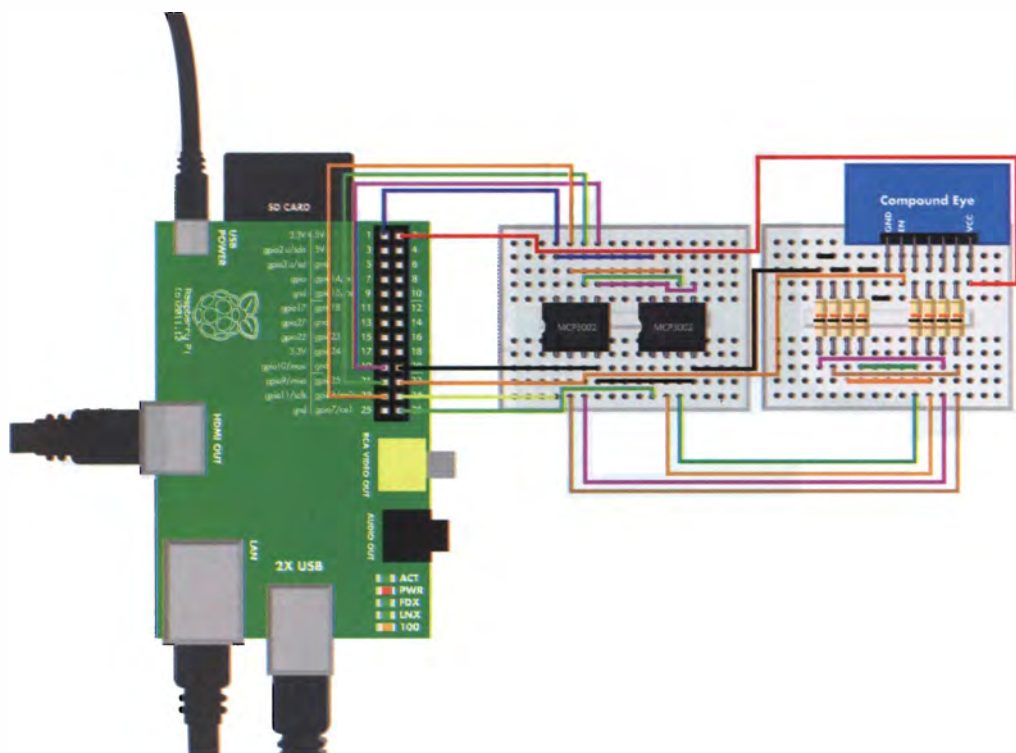


Рис. 3.17. Схема подключения составного ИК-датчика к Raspberry Pi

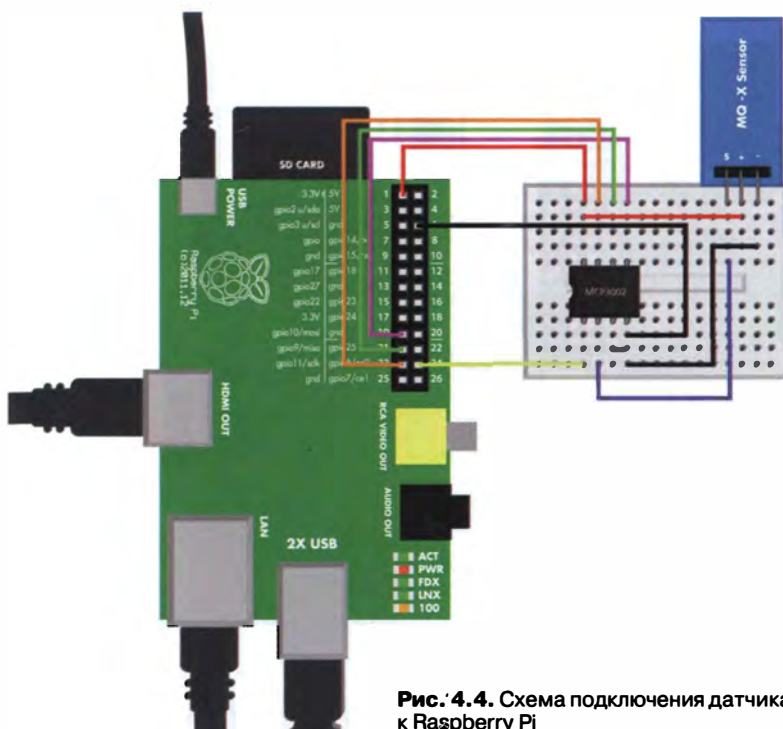


Рис. 4.4. Схема подключения датчика MQ-2 к Raspberry Pi

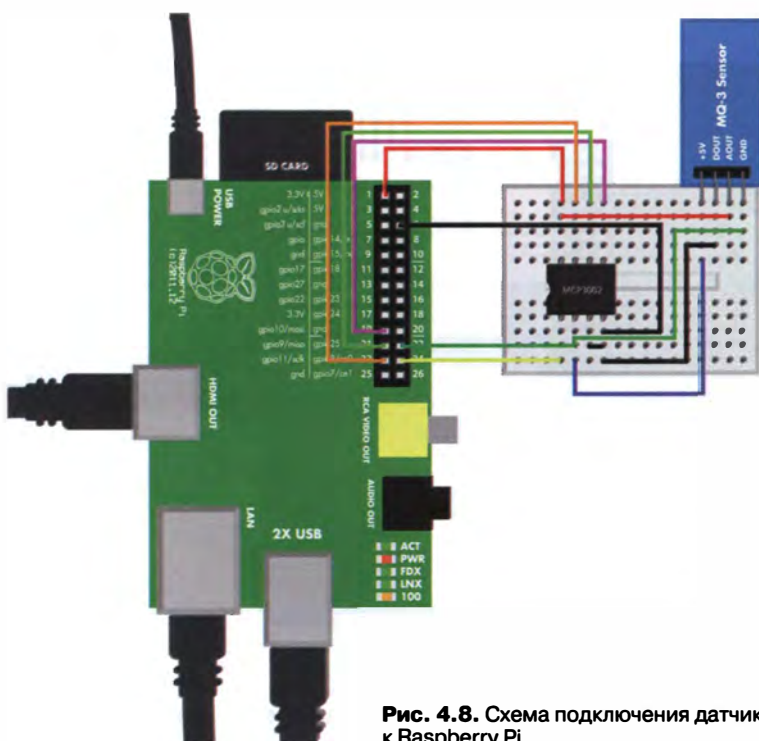


Рис. 4.8. Схема подключения датчика MQ-3 к Raspberry Pi

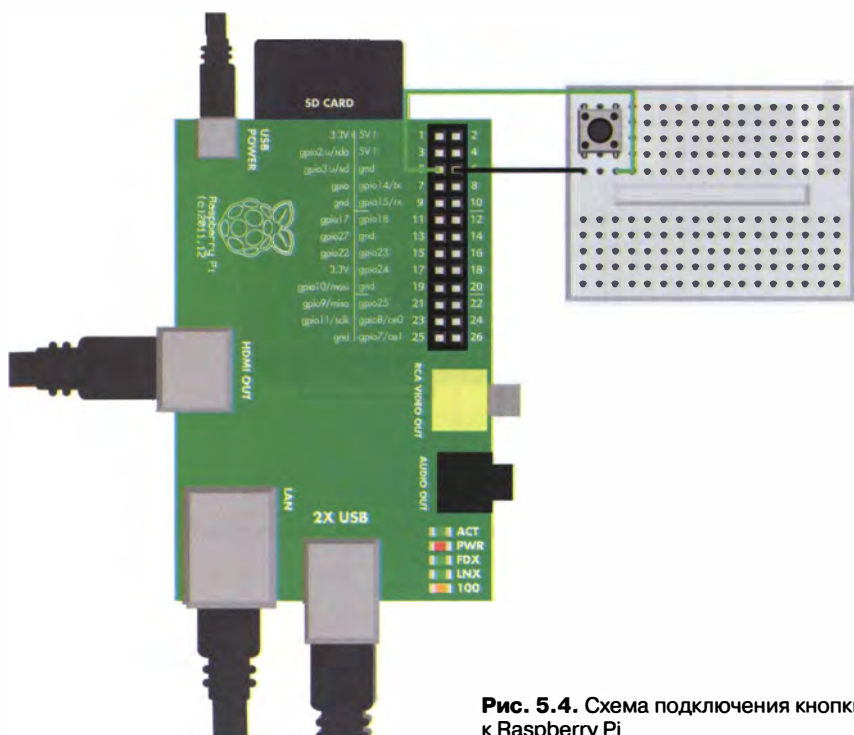


Рис. 5.4. Схема подключения кнопки к Raspberry Pi

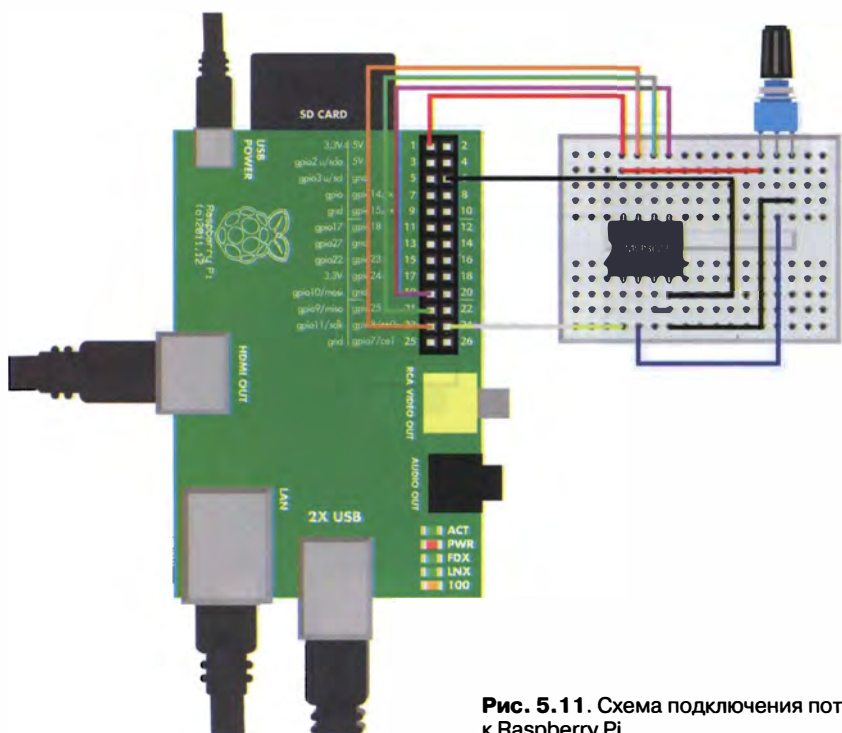


Рис. 5.11. Схема подключения потенциометра к Raspberry Pi

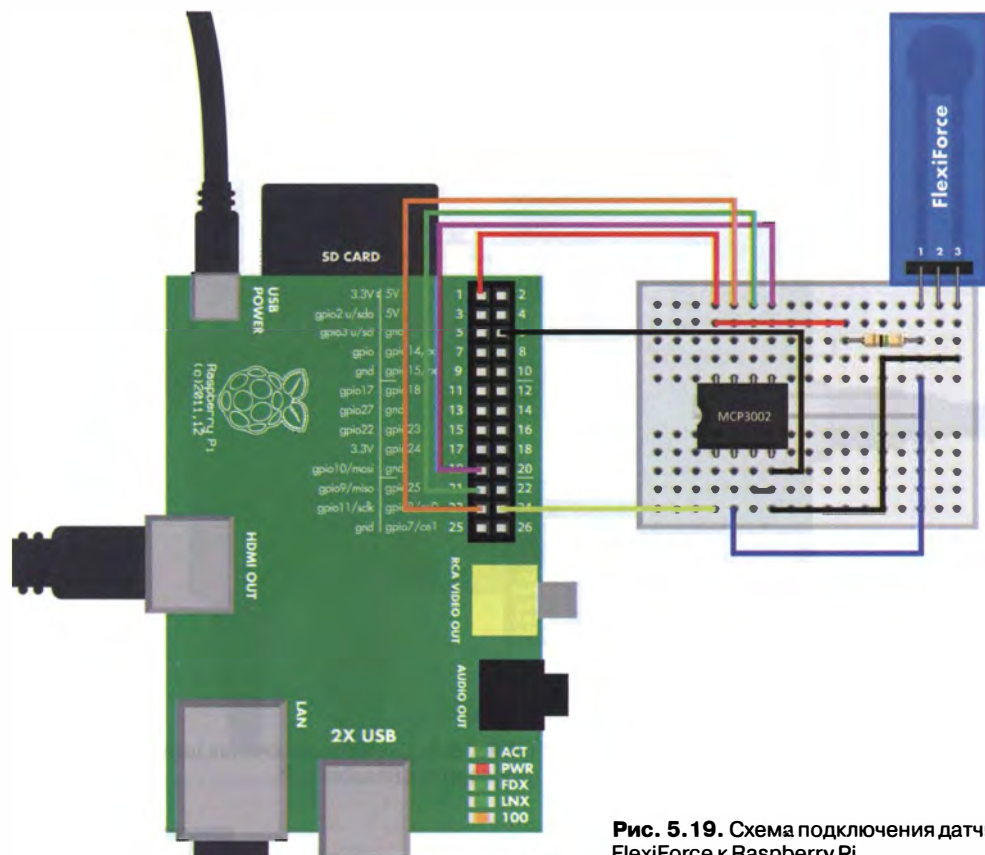


Рис. 5.19. Схема подключения датчика FlexiForce к Raspberry Pi

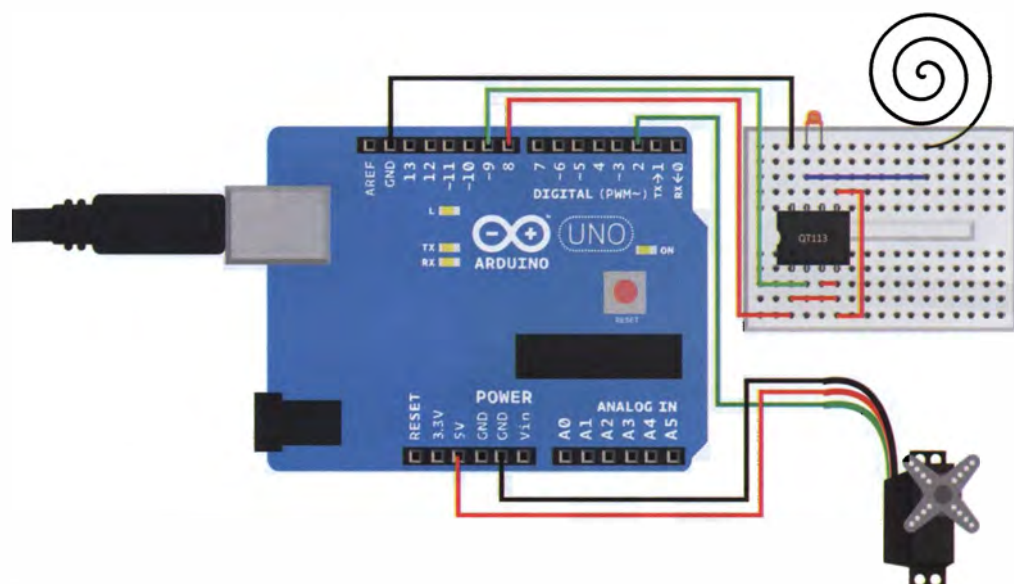


Рис. 5.25. Схема подключения компонентов отдельного звонка к Arduino

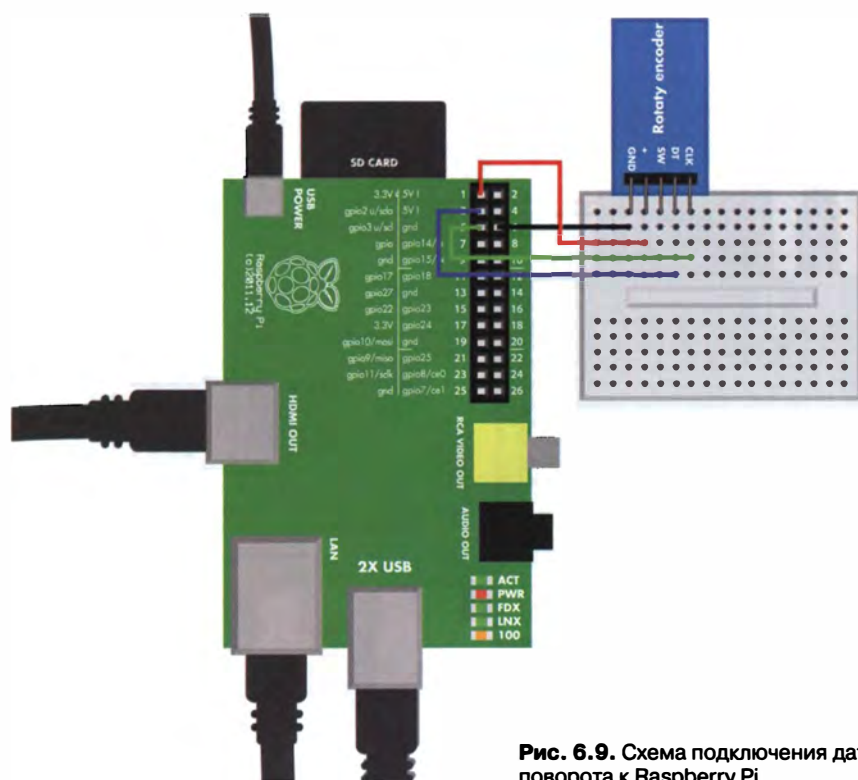


Рис. 6.9. Схема подключения датчика угла поворота к Raspberry Pi

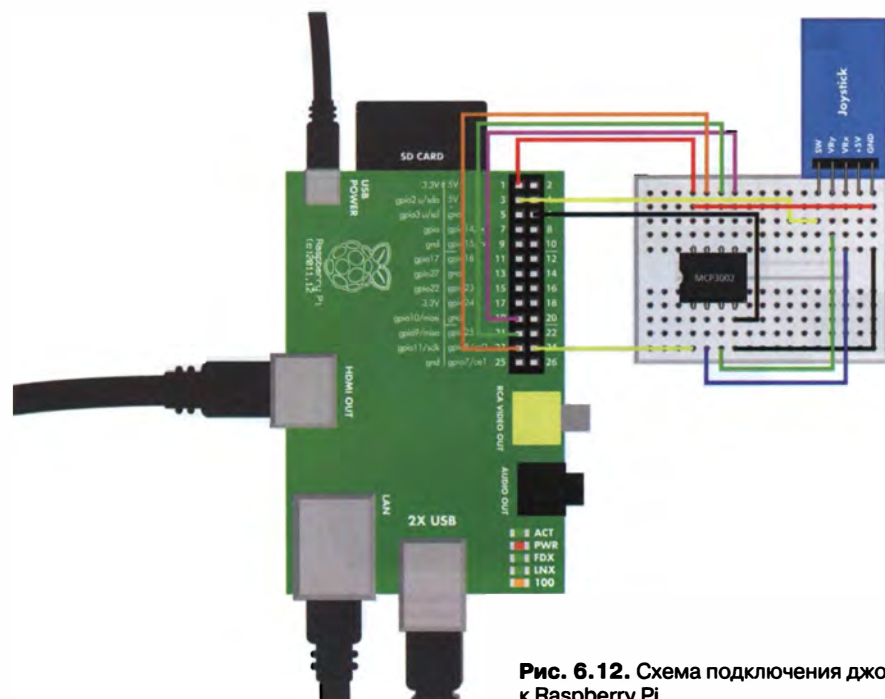


Рис. 6.12. Схема подключения джойстика к Raspberry Pi

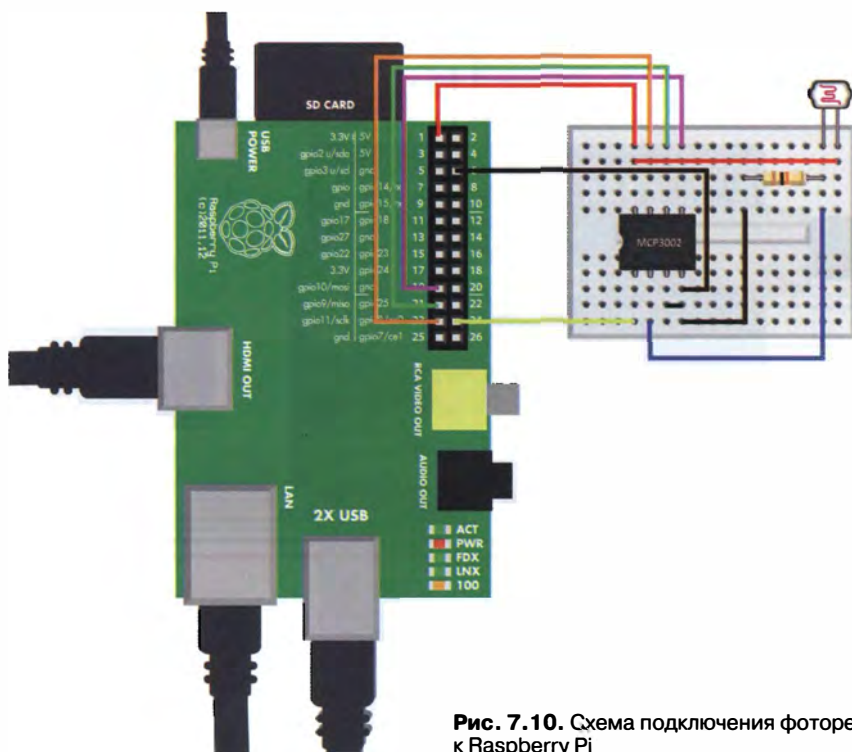


Рис. 7.10. Схема подключения фоторезистора к Raspberry Pi



Рис. 7.17. Хотите — верьте, хотите — нет, но датчик воспримет все эти объекты как белые

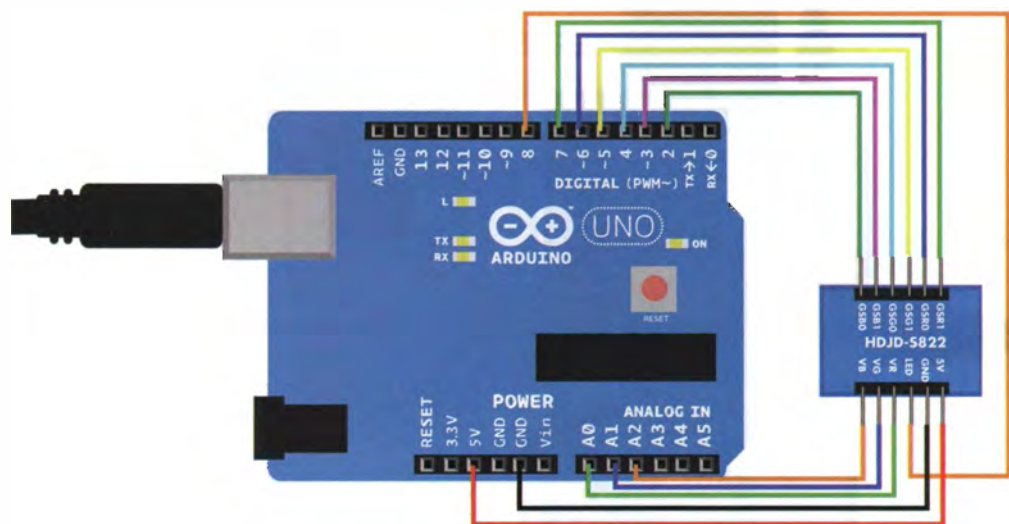


Рис. 7.20. Схема подключения датчика цвета к Arduino

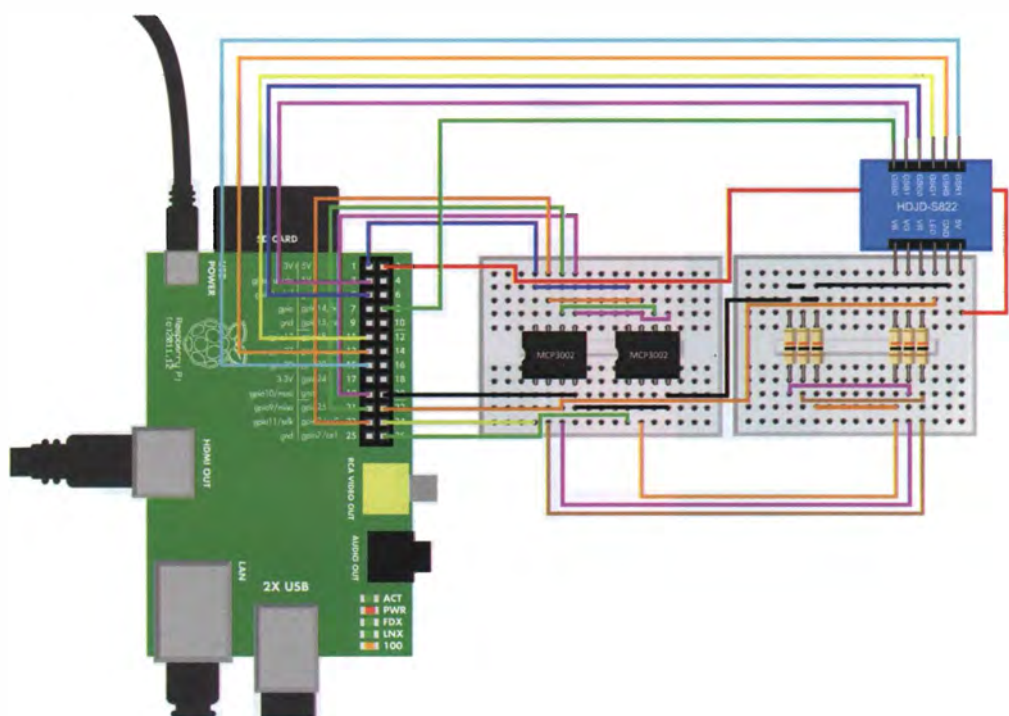


Рис. 7.21. Схема подключения датчика цвета к Raspberry Pi

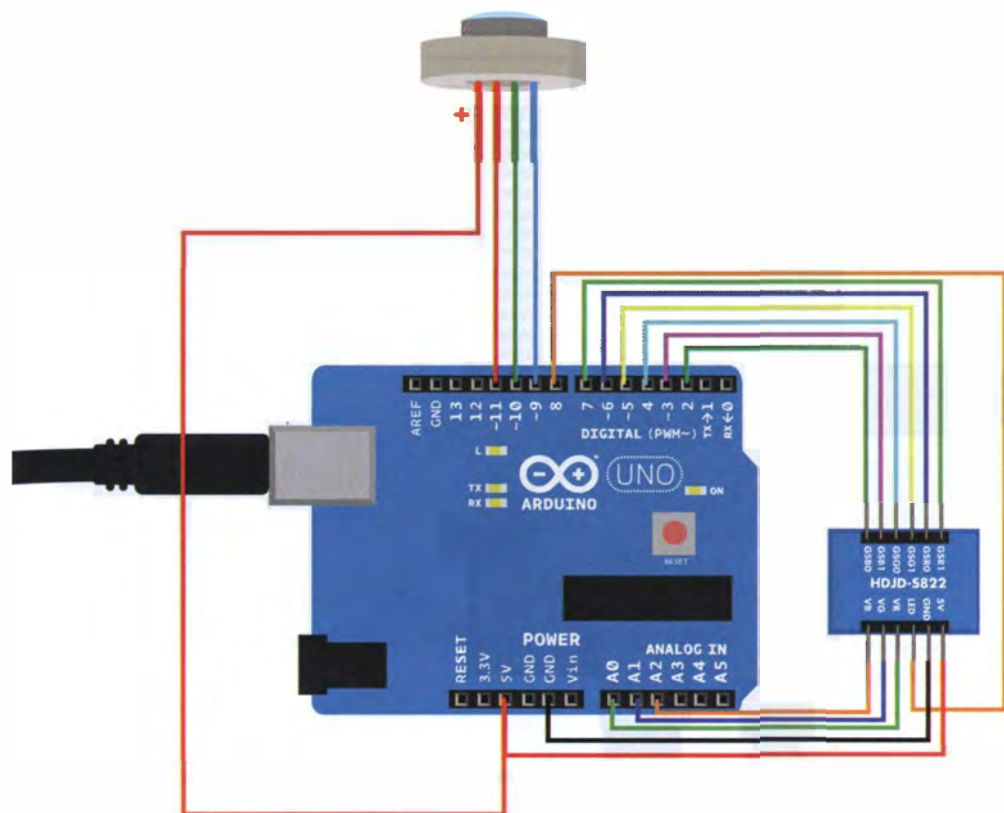


Рис. 7.27. Схема подключения компонентов цветовой сферы

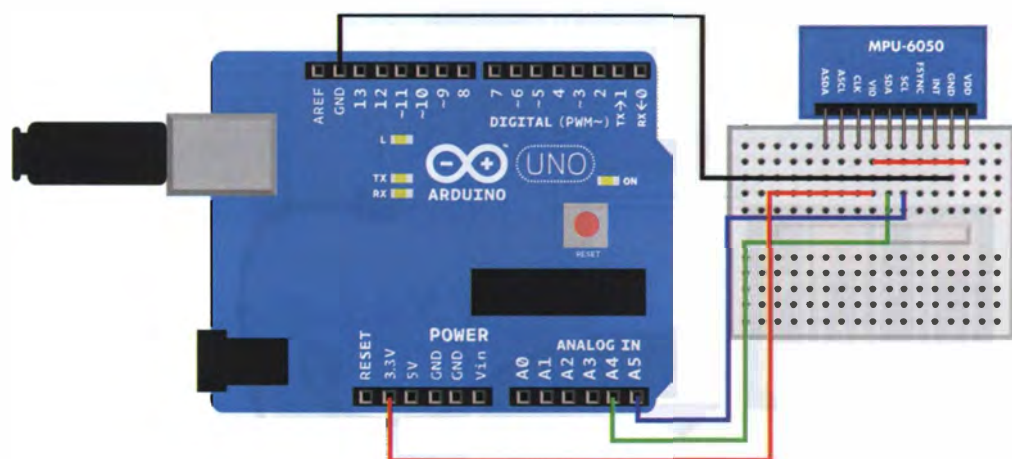


Рис. 8.5. Схема подключения модуля MPU 6050 к Arduino

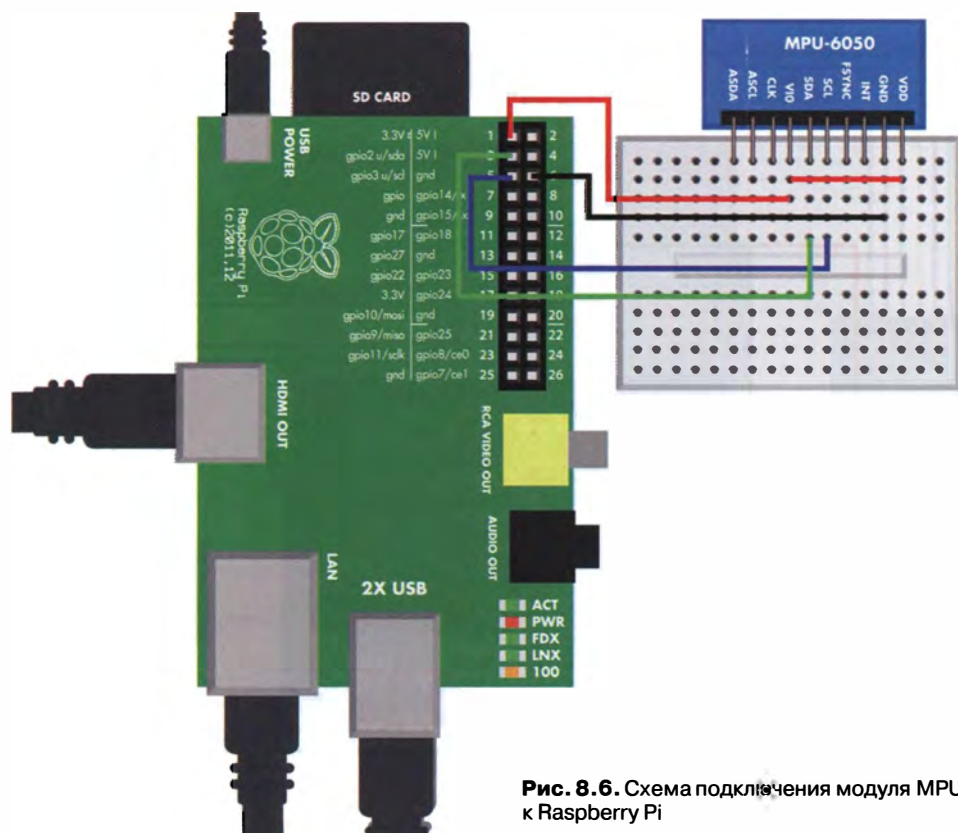


Рис. 8.6. Схема подключения модуля MPU 6050 к Raspberry Pi

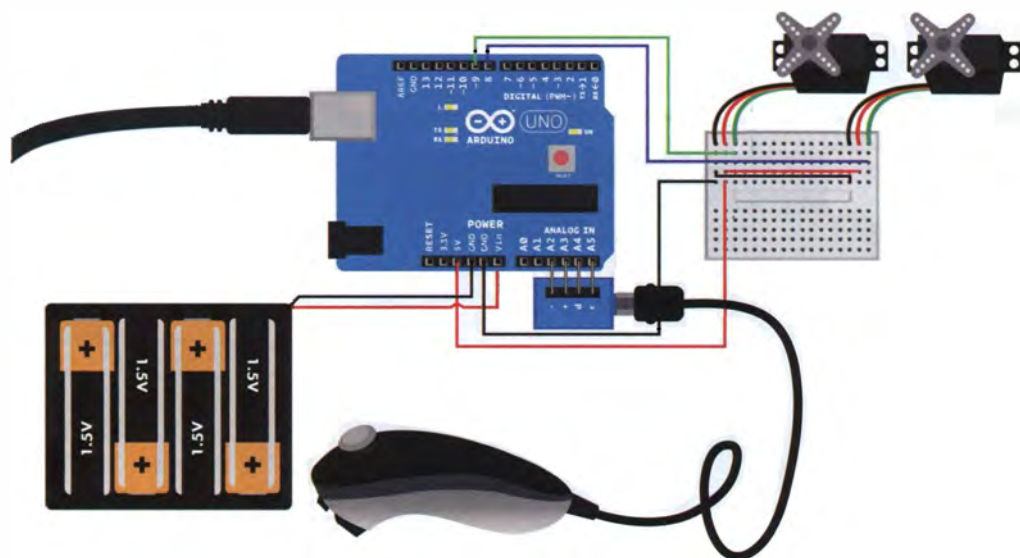


Рис. 8.12. Схема подключения компонентов системы управления манипулятором

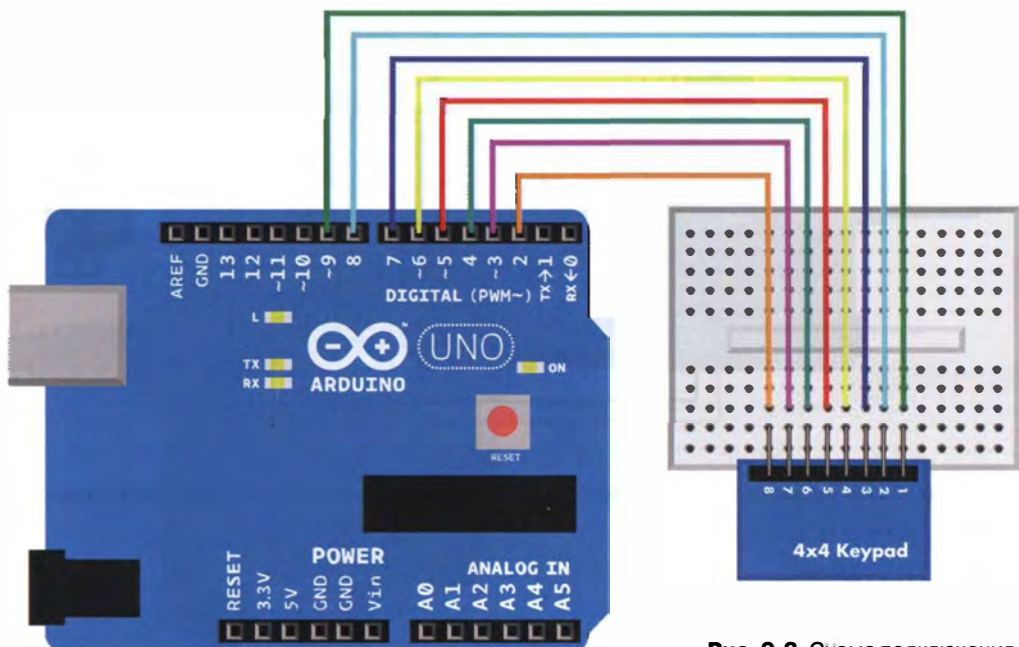


Рис. 9.2. Схема подключения цифровой клавиатуры к Arduino

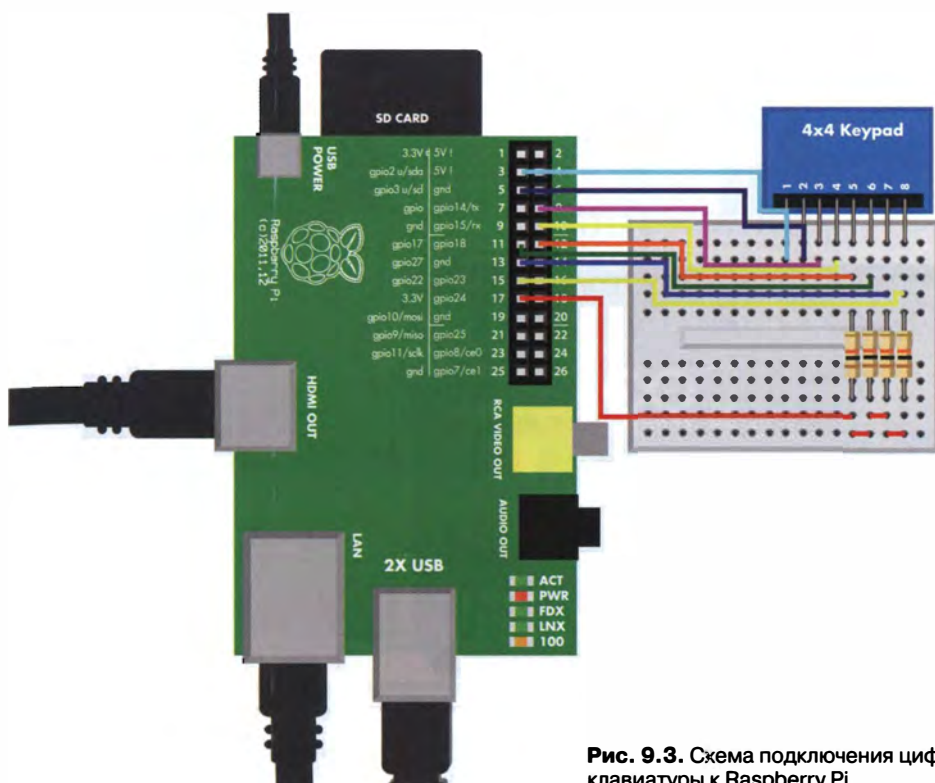


Рис. 9.3. Схема подключения цифровой клавиатуры к Raspberry Pi

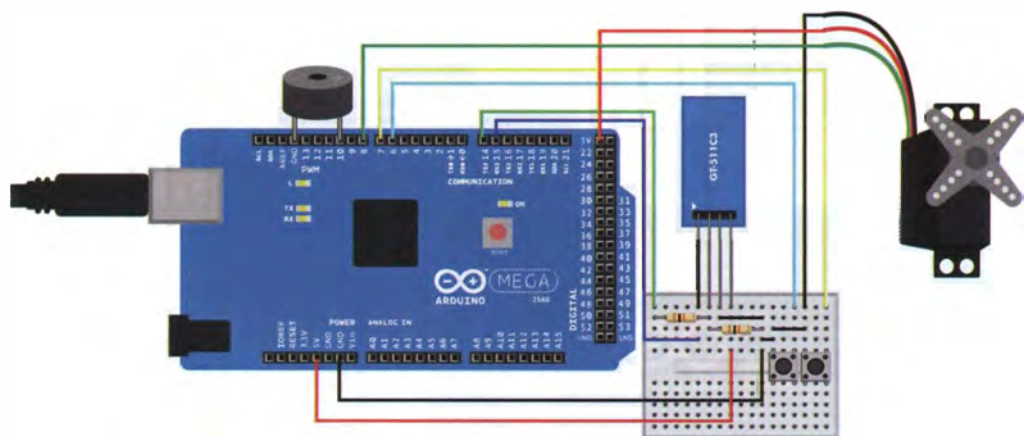


Рис. 9.16. Схема подключения компонентов системы управления “древним” сундуком к Arduino

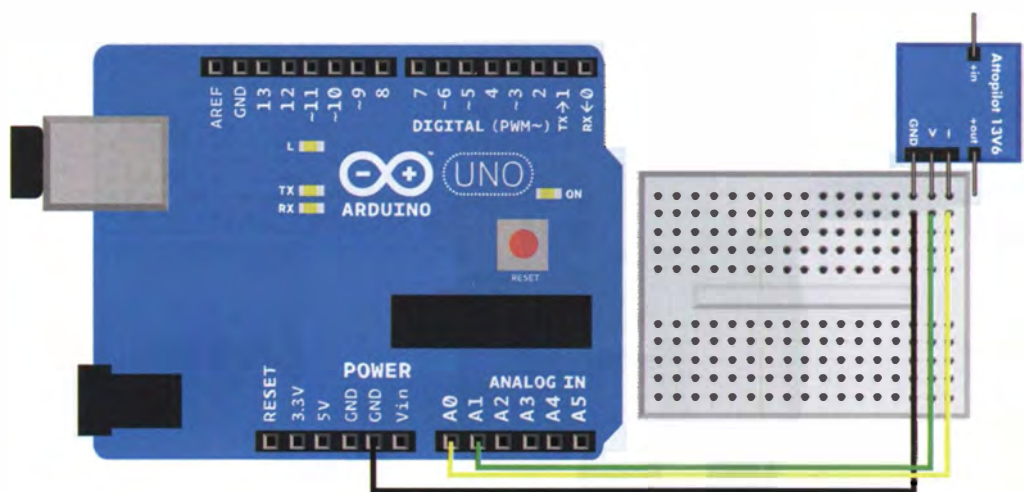


Рис. 10.1. Схема подключения датчика AttoPilot к Arduino

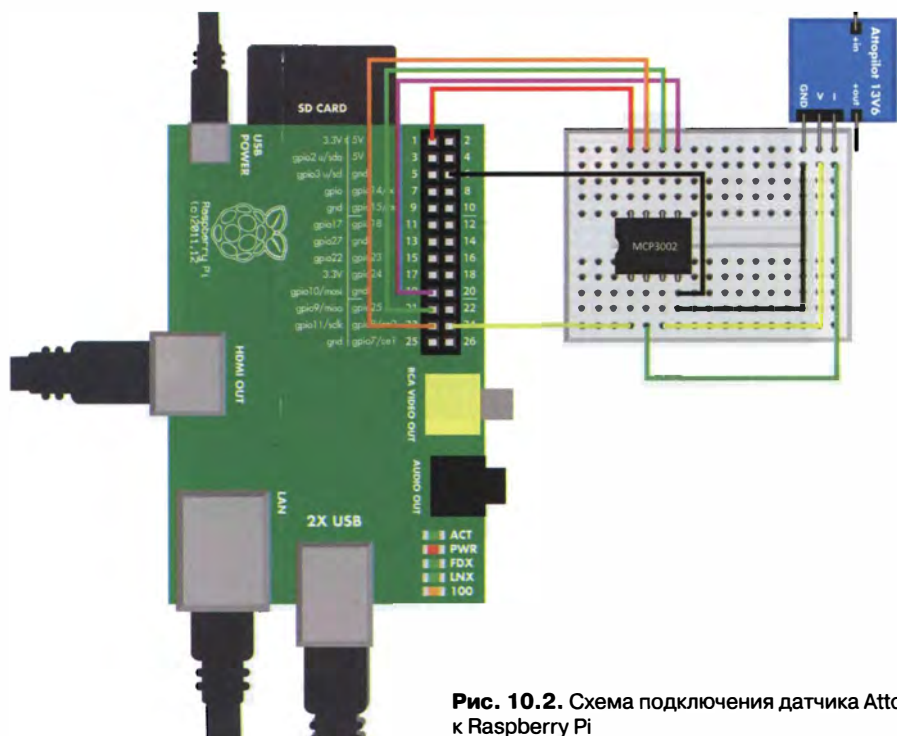


Рис. 10.2. Схема подключения датчика AttoPilot к Raspberry Pi

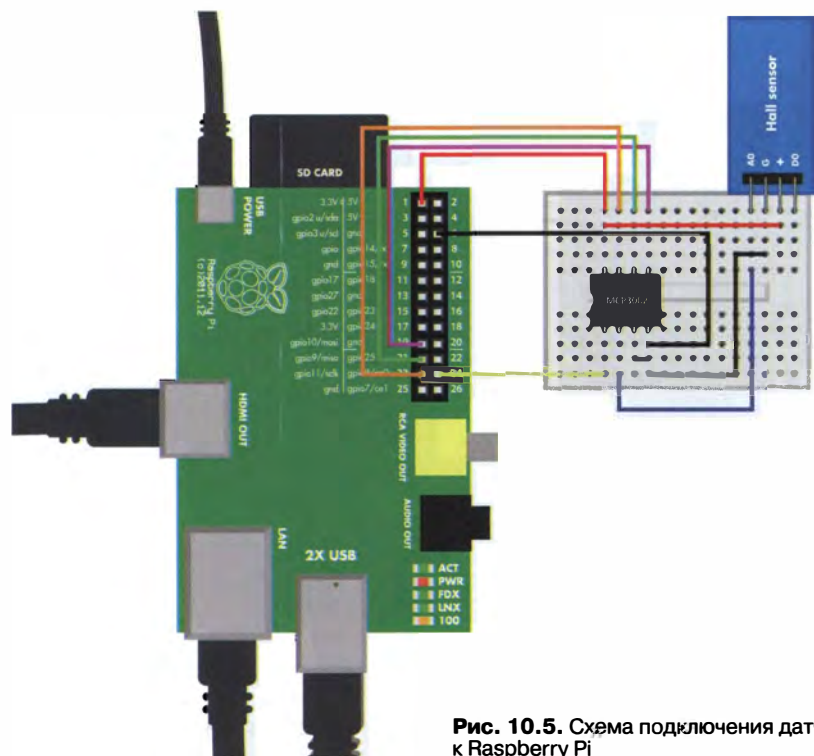


Рис. 10.5. Схема подключения датчика Холла к Raspberry Pi

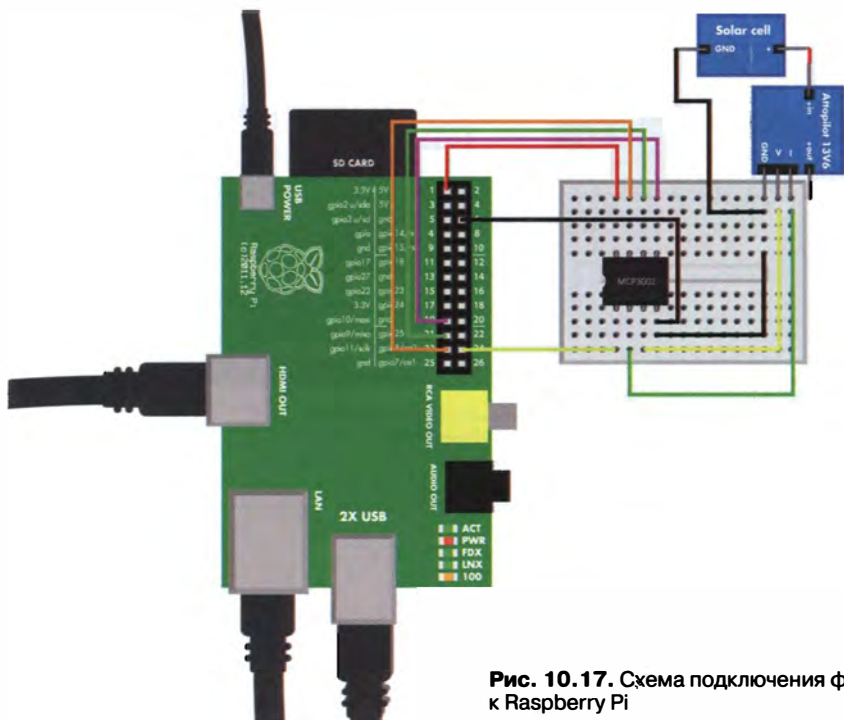


Рис. 10.17. Схема подключения фотоэлемента к Raspberry Pi

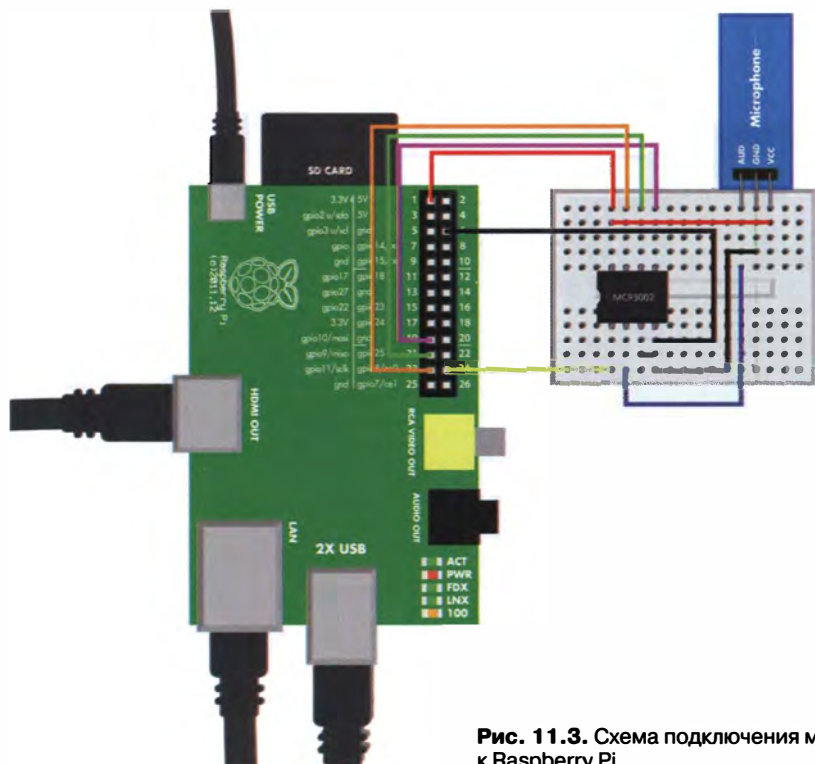


Рис. 11.3. Схема подключения микрофона к Raspberry Pi

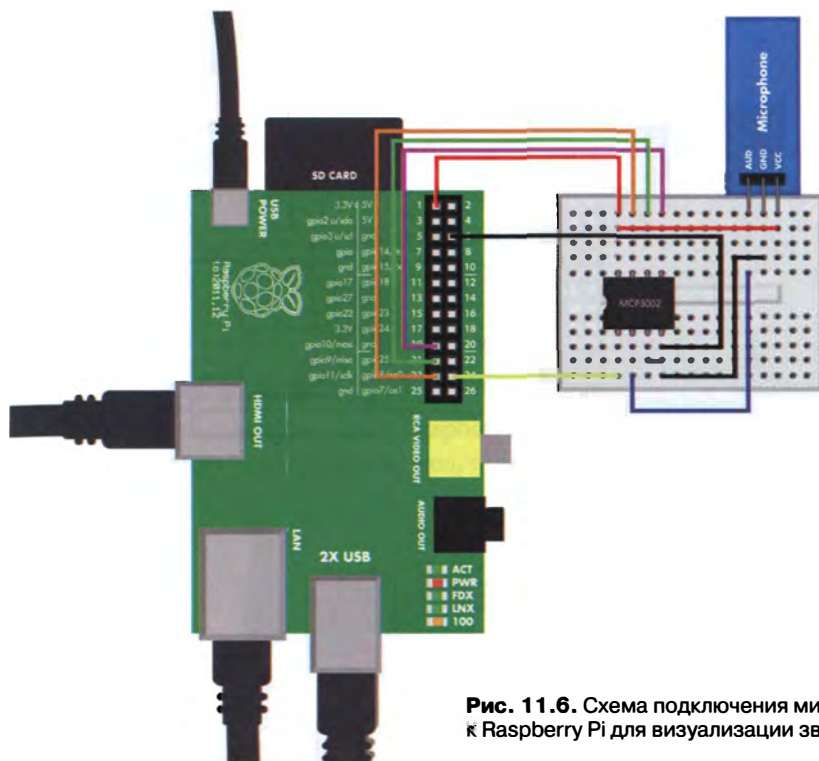


Рис. 11.6. Схема подключения микрофона к Raspberry Pi для визуализации звука

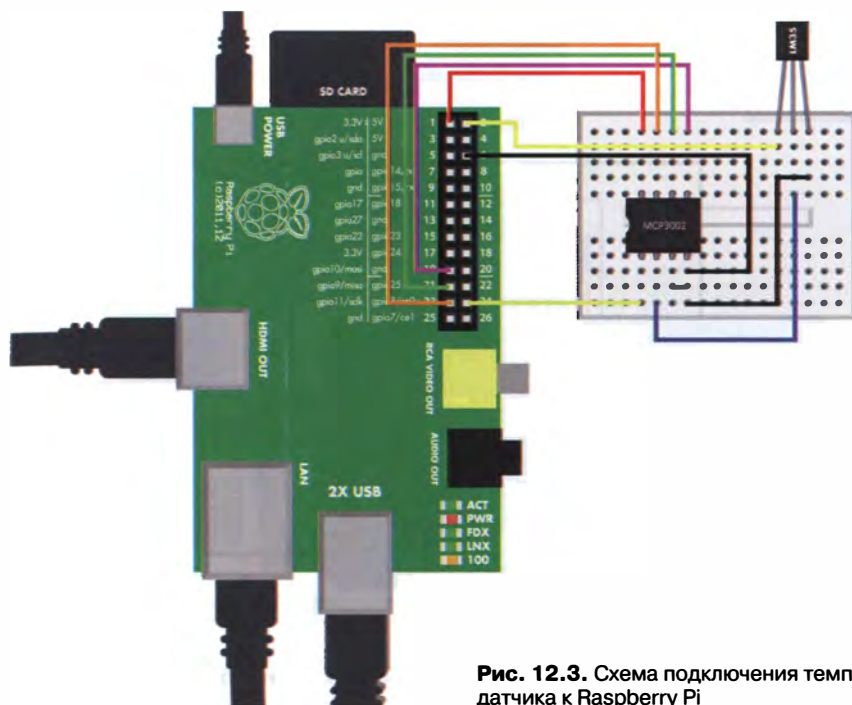


Рис. 12.3. Схема подключения температурного датчика к Raspberry Pi

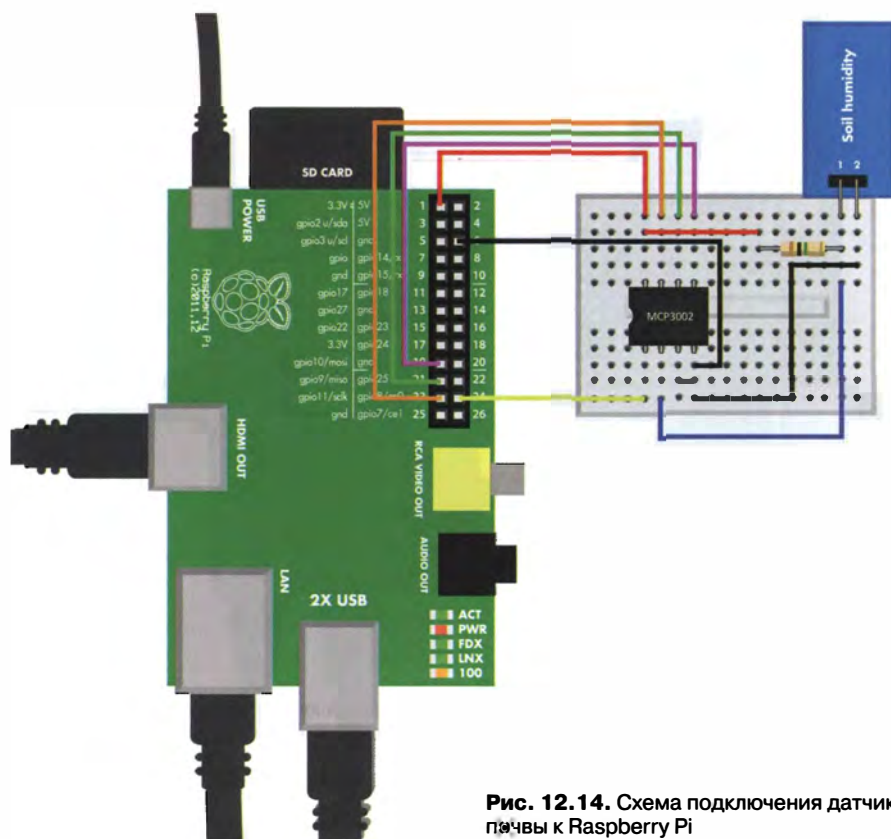


Рис. 12.14. Схема подключения датчика влажности почвы к Raspberry Pi

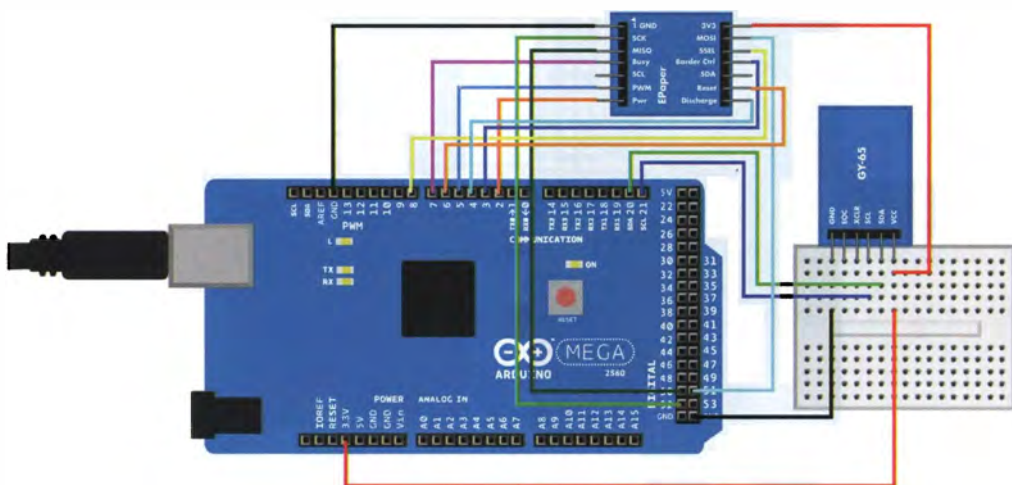


Рис. 12.17. Схема подключения компонентов метеостанции на электронной бумаге, управляемой Arduino Mega