Making
Things Talk

Практические методы организации связей между физическими объектами

Arduino, датчики и сети для связи устройств

2-е издание

обучение
на практике

33 ПРОЕКТА СОЕДИНЕНИЯ ГАДЖЕТОВ МЕЖДУ СОБОЙ И С ВНЕШНЕЙ СРЕДОЙ

bhv®

Make:
makezine.com

Making Things Talk

Second Edition

Tom Igoe

MAKER MEDIA
SEBASTOPOL, CA

Том Иго

Arduino, датчики и сети для связи устройств

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2015

УДК 004.4
ББК 32.973.26
И26

Иго Т.

И26 Arduino, датчики и сети для связи устройств: Пер. с англ. — 2-е изд. — СПб.: БХВ-Петербург, 2015. — 544 с.: ил.

ISBN 978-5-9775-3566-3

Рассмотрены 33 проекта на основе микроконтроллерной платы Arduino, в которых показано, как сделать, чтобы электронные устройства могли обмениваться между собой данными и реагировать на команды. Показано, как изменить настройки домашнего кондиционера, «позвонив ему» со своего смартфона; как создавать собственные игровые контроллеры, взаимодействующие по сети; как использовать устройства ZigBee, Bluetooth, инфракрасное излучение и обычное радио для беспроводного получения информации от различных датчиков и др. Рассмотрены языки программирования Arduino, Processing и PHP.

Для широкого круга читателей

УДК 004.4
ББК 32.973.26

Группа подготовки издания:

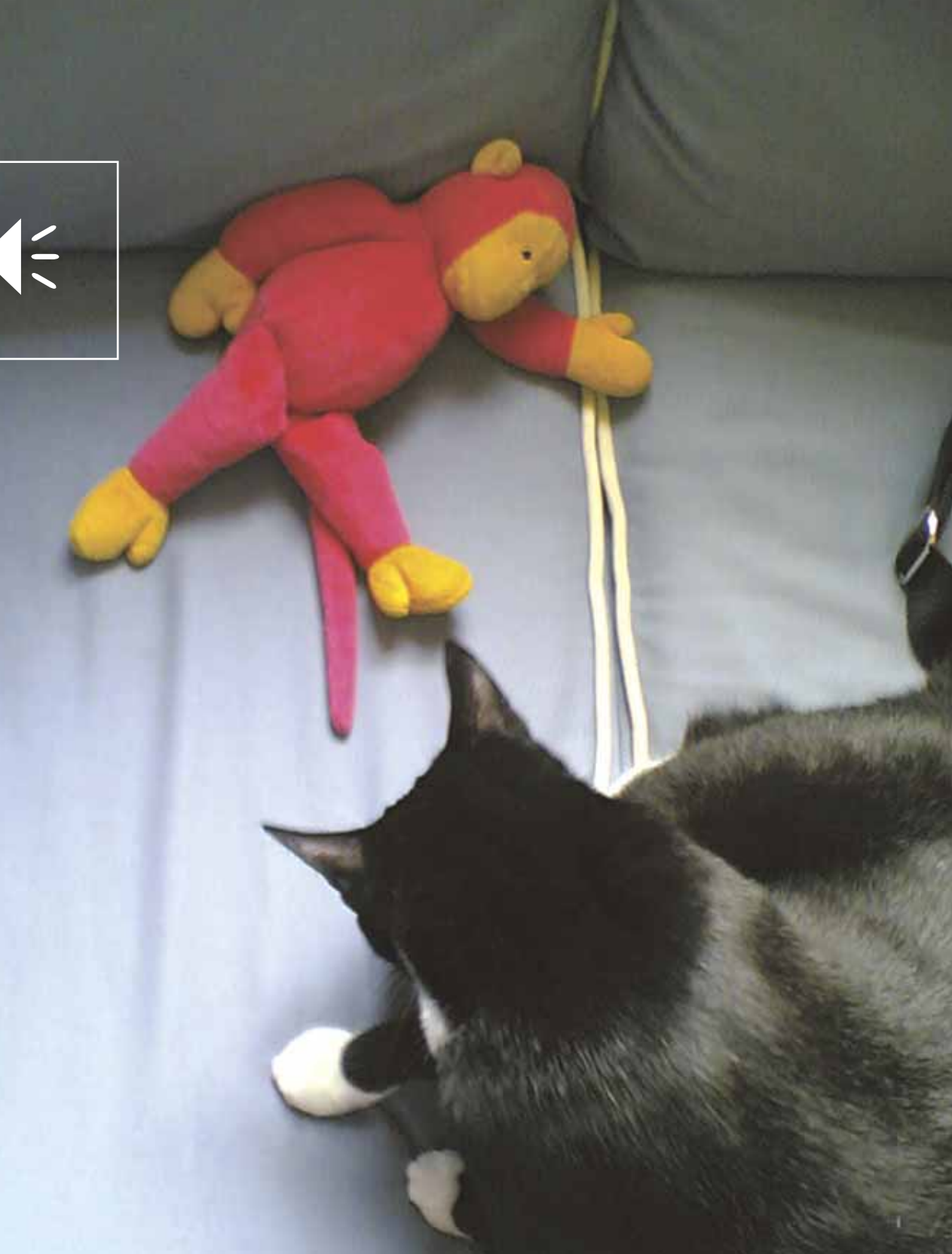
Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Перевод с английского	*Сергея Таранушенко*
Редактор	*Григорий Добин*
Компьютерная верстка	*Людмилы Гауль*
Корректор	*Зинаида Дмитриева*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 30.11.14.
Формат 84×108 1/16. Печать офсетная. Усл. печ. л. 57,12.
Тираж 1500 экз. Заказ №
«БХВ-Петербург», 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография «Наука»
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-1-449-39243-7 (англ.)
ISBN 978-5-9775-3566-3 (рус.)

ОГЛАВЛЕНИЕ



ПРЕДИСЛОВИЕ

Несколько лет тому назад Нил Гершенфельд (Neil Gershenfeld) написал толковую книгу, названную им «Когда вещи начинают думать». В ней он рассматривает мир, в котором обыденные вещи и устройства наделены вычислительными способностями, то есть мир сегодняшний. В частности, там обсуждаются последствия обмена между такими устройствами информацией о наших с вами личностях, возможностях и действиях. Книга хорошая, но ее название, на мой взгляд, — неудачное. Я бы назвал ту книгу «Когда вещи начинают общаться», поскольку, давайте признаем это, даже самые захватывающие идеи чего-либо стоят лишь тогда, когда их обсуждаешь с кем-то другим. И моя книга — «Чтобы вещи общались» — рассказывает, как создавать вещи, способные общаться друг с другом, и как предоставить людям возможность использовать эти вещи для общения между собой.

Компьютерные специалисты вот уже около двух десятков лет используют термин *объектно-ориентированное программирование* для обозначения способа разработки программного обеспечения, в котором программы и подпрограммы рассматриваются как объекты. Подобно физическим объектам, они обладают свойствами и поведением, унаследованными ими от *прототипов* — объектов, от которых они происходят. Каноническая форма любого программного объекта — код, описывающий его тип. Программные объекты позволяют с легкостью сочетать их между собой различными новыми способами. Программный объект можно использовать раз за разом, если известен его *интерфейс* — набор свойств и методов, посредством которых его создатель позволяет осуществлять к нему доступ (а также, если доступна соответствующая справочная информация). Способ, которым объект делает то, что он делает, не играет роли, при условии, что на выходе всегда получается один и тот же результат. Программные объекты наиболее эффективны тогда, когда они понятны применяющему их разработчику и хорошо взаимодействуют с другими программными объектами.

В реальном мире мы окружены разного рода электронными объектами: радиочасами, тостерами, мобильными телефонами, плеерами, детскими (и не только детскими) игрушками и т. п. Чтобы самому создать полезное электронное устройство, требуются серьезные усилия и значительный объем знаний, а чтобы обеспечить подобные устройства возможностью общаться друг с другом, может понадобиться знаний еще вдвое больше. Но так быть не должно. И сейчас появилась возможность собирать электронные устройства из простых модулей. При условии, что вы разбираетесь в интерфейсах таких модулей, можно собрать из них что угодно. Этот подход можно рассматривать как *объектно-ориентированное оборудование*[1]. *Основным условием работоспособности названного подхода является понимание того, как устройства взаимодействуют друг с другом, — независимо от конкретного устройства, будь то тостер, программа электронной почты на вашем ноутбуке или сетевая база данных. Все такие компоненты можно свести воедино, если определить, каким способом они смогут общаться. Эта книга и представляет собой руководство по ряду способов, методов и инструментов, обеспечивающих реализацию такого общения.*

[1] В оригинале: «object-oriented hardware».

Для кого предназначена эта книга?

Эта книга писалась для тех, кто хочет наделить вещи способностью взаимодействовать друг с другом. Например, для преподавателя естественных наук, который намерен показать своим ученикам, как отслеживать погодные условия одновременно в нескольких местах школьного округа. Или для скульптора, желающего заполнить помещение координированно движущимися механическими скульптурами. Или для дизайнера, которому необходимо быстро создавать макеты новых продуктов, моделируя как их форму, так и функции. Или же для обладателя кота, который любит наблюдать за своим питомцем, даже когда находится вне дома. В общем, книга предназначена быть начальным пособием для людей, обладающих ограниченным техническим опытом, но имеющих большой заряд энтузиазма и желающих успешно воплощать в жизнь интересующие их проекты.

Основными инструментами, потребными нам для изучения материала книги, будут персональные компьютеры, веб-серверы и микроконтроллеры — крошечные компьютеры, встроенные во все возрастающее число повседневных устройств. За последнее десятилетие микроконтроллеры и средства для их программирования прошли путь от устройств для посвященных к распространенным, легко используемым инструментам. С устройствами, которые десять лет тому назад ставили в тупик аспирантов,

сегодня легко обращаются учащиеся начальных школ. В течение этого времени мы с моими коллегами обучали людей с самым разнообразным образованием и родом занятий (очень немногие из которых были программистами) применению этих устройств для расширения диапазона физических действий, которые компьютеры могут воспринимать и интерпретировать и на которые способны реагировать.

В последнее время люди, использующие микроконтроллеры в своих устройствах, проявляют все большее желание наделить эти устройства возможностями не только воспринимать окружающий мир и управлять им, но также сообщать другим подобным устройствам о том, что они воспринимают и чем управляют. Если вы скомпоновали что-либо на базе BASIC Stamp[2] или Lego Mindstorms[3] и хотите, чтобы ваше устройство могло обмениваться информацией с другими подобными устройствами, созданными вами или другими самоделкиными, данная книга вам в этом поможет. Она также будет полезной и для программистов со знанием сетей и веб-служб, которые хотят приобщиться к программированию встроенных сетей.

Но если вы из тех, кому нравится разбираться во всех тонкостях технологий, в этой книге вы, возможно, не найдете того, чего ищете. В ней нет подробных примеров кода для Bluetooth или стеков TCP/IP, а также диаграмм цепей для микросхем контроллеров Ethernet. Рассматриваемые в книге компоненты устройств являют собой компромисс между простотой, гибкостью и стоимостью. Они как раз и воплощают ранее обозначенное объектно-ориентированное оборудование, требуя сравнительно небольшого объема проводных соединений и кода. То есть предназначены для того, чтобы позволить вам достичь конечной цели — наделить вещи возможностью общаться между собой — наиболее быстрым и легким способом.

[2] Микроконтроллер с небольшим специализированным интерпретатором BASIC (PBAIC), встроенным в ROM.

[3] LEGO Mindstorms — конструктор (набор сопрягаемых деталей и электронных блоков) для создания программируемого робота. (**http://ru.wikipedia.org/wiki/LEGO_Mindstorms**).

Что вам нужно знать?

Чтобы извлечь как можно больше пользы из этой книги, вам необходимы базовые знания аппаратной и программной части микроконтроллеров, определенное представление об Интернете и доступ к тому и другому.

Многие специалисты, начав программировать микроконтроллеры, творят чудеса с несколькими датчиками и парой сервоприводов, но при этом могут иметь весьма ограниченный опыт в области взаимодействия между микроконтроллером и другими программами на персональном компьютере. Точно так же, многие опытные сетевые и мультимедиа программисты никогда не экспериментировали ни с каким оборудованием, включая микроконтроллеры. Если вы принадлежите к какой-либо из упомянутых категорий, эта книга для вас. Поскольку материал книги предназначен для читателей с самым разнообразным опытом, некоторые темы, в зависимости от вашего личного опыта, могут показаться вам чересчур простыми. В таком случае пропускайте материал, с которым вы уже знакомы, чтобы побыстрее добраться до более интересного для вас материала.

Если вы никогда ранее не имели дело с микроконтроллерами, то прежде, чем приступать к работе с этой книгой, вам следовало бы получить определенные начальные знания в означенной области. Для этого рекомендуется прочитать мою предыдущую книгу «Physical Computing: Sensing and Controlling the Physical World»[4] (издательство Thomson), которую я написал совместно с Даном О'Саливаном (Dan O'Sullivan). В этой книге дается введение в основы электроники, микроконтроллеров и дизайна физического взаимодействия.

[4] «Физические вычисления. Восприятие и управление материальным миром».

Кроме того, вам понадобятся базовые знания программирования. Если у вас нет никакого опыта в этой области, то я рекомендую ознакомиться со средой программирования Processing, информацию о которой можно найти на веб-сайте **www.processing.org.** Там же можно загрузить и саму среду и сопутствующие ей инструменты и материалы. Среда программирования Processing достаточно простая, чтобы в ней обучались программированию люди, которые никогда раньше этим не занимались, но в то же самое время достаточно мощная, чтобы реализовывать с ее помощью весьма сложные проекты. Среда Processing используется в этой книге в тех случаях, когда требуется выполнять программирование в графическом интерфейсе.

Книга содержит примеры кода на нескольких языках программирования. Эти примеры достаточно простые, так что, если вам не подходит язык, на котором они написаны, с помощью комментариев в коде вы можете переписать их на язык, с которым вам более удобно работать.

Содержание книги

В книге представлены основные принципы создания объектов сетевой структуры, а также приводятся примеры, их иллюстрирующие. Каждая глава содержит инструкции по созданию работающих проектов, основанных на изложенных в ней концепциях.

- **Глава 1.** Здесь рассматриваются основные инструменты программирования, используемые в книге, и приводится пример программы «Здравствуй, мир!» для каждого из них.
- **Глава 2.** Введение в основные базовые понятия, необходимые для придания вещам способности общения друг с другом. Рассматриваются характеристики, которые должны быть оговорены прежде всего, а также показано, как разделение этих понятий в уме помогает в поиске и устранении проблем. В качестве примера взаимодействия, обеспечиваемого посредством модема, приведены инструкции по созданию простого проекта одноранговой последовательной связи между микроконтроллером и персональным компьютером на основе радиоканала Bluetooth. Предоставляется информация о протоколах данных, модемных устройствах и схемах адресации.
- **Глава 3.** Введение в более сложную сеть — Интернет. Обсуждаются устройства, необходимые для ее построения, а также основные взаимодействия между этими устройствами. Рассматриваются сообщения, лежащие в основе некоторых из наиболее распространенных процедур, выполняемых ежедневно в Интернете, и приводятся инструкции по отправке таких сообщений. Описано создание первого комплекта программ для отправки через Интернет данных о физических действиях у вас дома.
- **Глава 4.** Создаем первое встроенное устройство. Получаем дополнительный опыт по подключению к Сети с помощью средств консоли командной строки, а также подключаем микроконтроллер к веб-серверу, используя настольный компьютер или ноутбук в качестве посредника.
- **Глава 5.** Следующий шаг в изучении подключений к Сети — рассмотрение сокетных соединений, которые позволяют более длительное взаимодействие. Здесь мы напишем серверную программу, к которой можно подключиться с любого устройства, имеющего доступ в Сеть. Подключение к этой программе осуществляется как из командной строки, так и с микроконтроллера, — чтобы научиться понимать, как разные типы устройств могут подключаться друг к другу через один и тот же сервер.
- **Глава 6.** Введение в беспроводную связь. Предоставляются некоторые характеристики беспроводной связи, включая ее возможности и ограничения. Создаем несколько небольших проектов, позволяющих сказать «Здравствуй, мир!» через эфир несколькими разными способами.

- **Глава 7.** Рассматривается подход к реализации связи, существенно отличающийся от использования сокетов, рассмотренных в *главе 5*. Этот подход использует управляемые сообщениями протоколы — такие как протокол UDP для Интернета или ZigBee и 802.15.4 для беспроводных сетей. В отличие от модели клиент-сервер, которой посвящены предыдущие главы, здесь мы учимся обмениваться информацией между одноранговыми сетевыми объектами по одному сообщению за раз.
- **Глава 8.** Рассмотрены несколько инструментов для определения местонахождения предметов в физическом пространстве (геолокации), приведены некоторые размышления о взаимосвязи между физическим расположением объекта и сетевыми взаимодействиями.
- **Глава 9.** Рассматривается вопрос идентификации в физическом и сетевом пространстве. Излагаются несколько методов для генерирования однозначных сетевых идентификаторов на основе физических характеристик. Некоторое внимание уделено тому, как определять характеристики сетевых устройств.
- **Глава 10.** Рассмотрены сети мобильной телефонной связи, поясняется, что можно делать сейчас с мобильными телефонами и телефонными сетями.
- **Глава 11.** Представлен более подробный обзор разных типов протоколов, рассмотренных в этой книге, с описанием инфраструктуры, потребной для их дальнейшего использования.

“ Покупка деталей

Для реализации всех проектов, описанных в этой книге, потребуется много деталей, вследствие чего вам придется познакомиться с большим числом поставщиков. Поскольку в моем городе нет больших розничных продавцов деталей электроники, я покупаю все нужные детали через Интернет. Если вам повезло, и в вашем городе или селении есть возможность покупать необходимые детали в физическом магазине, тем лучше для вас. В противном случае познакомьтесь со следующими интернет-поставщиками:

Jameco (**http://jameco.com**), Digi-Key (**www.digikey.com**) и Farnell (**www.farnell.com**) — розничные поставщики деталей электроники, предлагающие во многом одинаковый ассортимент деталей. А такие поставщики, как Maker Shed (**www.makershed.com**), SparkFun (**www.sparkfun.com**) и Adafruit (**http://adafruit.com**), предлагают специализированные компоненты, наборы и пакеты для быстрой и легкой реализации популярных проектов. Полный список поставщиков дается в *приложении*. Если необходимо заменить какую-либо деталь, которая вам хорошо известна, делайте это без колебаний.

Учитывая, что покупка деталей через Интернет не представляет никаких трудностей, вы можете посчитать достаточным общаться с продавцами исключительно через средства их веб-сайтов. Тем не менее, не стоит пренебрегать возможностью делать это по телефону. Особенно в случаях, когда у вас отсутствуют какие-либо знания в области определенного проекта, будет полезным поговорить с кем-либо о заказываемых деталях и получить о них как можно больше информации. Скорей всего, на другом конце телефонной линии у большинства розничных поставщиков деталей найдутся люди, которые будут рады вам помочь. Номера телефонов в *приложении* представлены для тех поставщиков, для которых они имелись.

От редакции русского издания

В России достаточно обширный ассортимент электронных деталей, компонентов, оборудования и инструментов предлагается фирмами «Амперка» (**www.amperka.ru**), «Линуксцентр» (**www.linuxcenter.ru**) и «Чип и Дип» (**http://chipdip.ru**).

Бонус от фирмы «Амперка»

Специально для читателей этой книги «Амперка» предлагает особое кодовое слово **MTT14**, которое даст вам скидку при оформлении заказа онлайн.

Использование примеров кода

Эта книга предназначена для того, чтобы помочь вам реализовать свои проекты. Соответственно, код из нее можно использовать в своих программах и документации без необходимости испрашивать какое-либо разрешение, если только речь не идет о коммерческом воспроизведении значительного объема кода.

Например, использование нескольких фрагментов кода из этой книги в своей программе разрешения не требует, но для продажи или распространения диска CD-ROM с примерами из книг издательства O'Reilly разрешение требуется. Цитирование теста из этой книги, включая код, разрешения не требует, но включение значительного объема кода из книги в документацию своего продукта требует разрешения.

Мы будем благодарны за предоставление ссылки на источник при цитировании материалов из этой книги. Формат такой ссылки может включать название, имя автора, издателя и ISBN книги. Например: «Making Things Talk: Practical Methods for Connecting Physical Objects, by Tom Igoe. Copyright 2011 O'Reilly Media, 978-1-4493-9243-7»[5]. Если вы подозреваете, что использование вами примеров кода может выходить за рамки принципа добросовестного использования или данных ранее разрешений, то можете уточнить этот вопрос электронной почтой по адресу **permissions@oreilly.com**.

[5] Выходные данные настоящего перевода этой книги на русский язык приведены на обороте титульного листа издания. — *Ред.*

Использование примеров схем

Для сборки проектов из этой книги вам придется разбирать или даже ломать те или иные устройства и нарушать условия их гарантии. Если вам это не по душе, закройте книгу и займитесь чем-либо другим. Эта книга не для тех, кто дрожит от мысли, что если они разберут вещь, то не смогут собрать ее обратно.

Впрочем, хотя определенный дух исследования приветствуется, все же необходимо соблюдать должную предосторожность и не предпринимать никаких рискованных действий при реализации проектов из книги. Каждый набор инструкций создавался с учетом требований безопасности — игнорируйте инструкции по безопасности на свой собственный страх и риск. Обязательно убедитесь в том, что у вас имеется достаточный уровень знаний и опыта для работы над проектом безопасным образом.

Необходимо также иметь в виду, что проекты и схемы в этой книге предназначены только для познавательных целей. Такие частности, как регулирование мощности, автоматический сброс, экранирование от радиопомех и подобные моменты, которые необходимо учитывать, чтобы электронное устройство можно было сертифицировать для рынка, в этой книге не рассматриваются. Если вы хотите разработать продукт для использования кем-то еще, кроме вас самих, полагаться только на информацию из этой книги будет недостаточным.

Благодарности за первое издание

Эта книга является продуктом обсуждения и сотрудничества со многими людьми. Ее создание было бы невозможным без поддержки и ободрения со стороны круга моих соратников.

Я свыше десяти последних лет работал в Программе интерактивных телекоммуникаций[6] на факультете искусств Tisch[7] Нью-Йоркского университета — месте, где мне посчастливилось участвовать в оживленном и задушевном общении со многими талантливыми людьми. И книга моя основана на наработках курса «Сетевые объекты», который я преподавал там в течение нескольких лет. Я надеюсь, что изложенные в ней идеи представляют дух этого заведения и смогут передать вам мое чувство удовлетворенности от работы в нем.

Основатель факультета Ред Бёрнз (Red Burns) оказывает мне поддержку с самого начала моей работы в этой области. Она потакает заоблачным полетам моей фантазии, но, когда требуется, решительно возвращает меня на землю. Она проверяет каждый мой проект, чтобы удостовериться, что я использую технологию не ради самой технологии, но для того, чтобы с ее помощью предоставлять людям новые возможности.

Мой коллега Дан О'Саливан, который сейчас возглавляет программу, познакомил меня с физическими вычислениями, а затем великодушно разрешил мне совместно с ним преподавать эту дисциплину и формировать ее роль в Программе интерактивных телекоммуникаций. Он замечательный советник и сотрудник и постоянно давал мне советы и оценивлл мою работу. Большинство глав этой книги берут начало в длинных беседах с Даном. Их отражение прослеживается по всей книге, что только сделало ее лучше.

Клэй Ширки (Clay Shirky), Дэниель Розин (Daniel Rozin) и Дан Шифман (Dan Shiffman) также тесно сотрудничали со мной в этом проекте. Клэй благосклонно наблюдал, как в моем офисе вырастали кучи деталей, и любезно отрывался от своей собственной работы, чтобы выразить мнение о моих идеях. Дэниель Розин также делился ценными важными знаниями, и его идеи весьма значительно повлияли на книгу. Дан Шифман перечитал много черновиков книги и дал по ним полезные отзывы. Он также подсказал мне много отличных примеров кода.

Мои коллеги Марианна Петит (Marianne Petit), Нэнси Хэкинджер (Nancy Heckinger) и Жан-Марк Гутье (Jean-Mark Gauthier) поддерживали меня на протяжении всей работы над книгой, ободряя и воодушевляя, подменяя меня на работе при необходимости и предоставляя вдохновляющий пример своей собственной работой.

Сделать возможной эту книгу помогли также и все остальные члены преподавательского состава Программы интерактивных телекоммуникаций. В частности, Джордж Агудоу (George Agudow), Эдуард Гордон (Edward Gordon), Мидори Ясуда (Midori Yasuda), Меган Демарест (Megan Demarest), Нэнси Льюис (Nancy Lewis), Роберт Райан (Robert Ryan), Джон Дуэйн (John Duane), Марлон Иванс (Marlon Evans), Тони Ценг (Tony Tseng) и Глория Сед (Gloria Sed) стоически выносили мои безумства по части физических вычислений и сетевых объектов и делали возможной мою работу, работу других членов преподавательского состава, а также и студентов. Исследователи Карлин Мо (Carlyn Maw), Тод Голубек (Todd Holoubek), Джон Шиммель (John Schimmel), Дория Фэн (Doria Fan), Давид Нолен (David Nolen), Питер Керлин (Peter Kerlin) и Майкл Ольсон (Michael Olson) в течение нескольких последних лет помогали преподавательскому составу и студентам реализовать идеи, под влиянием которых были созданы проекты, вошедшие в эту книгу. Преподаватели Патрик Дуайер (Patrick Dwyer), Майкл Шнайдер (Michael Schneider), Грег Шакар (Greg Shakar), Скотт Фицджеральд (Scott Fitzgerald), Джэми Аллен (Jamie Allen), Шан Вэн Эвери (Shawn Van Every), Джеймс Ту (James Tu) и Раффи Крикорян (Raffi Krikorian) использовали на своих уроках инструменты из этой книги или предоставили свои методики для проектов, изложенных в ней.

[6] В оригинале: «Interactive Telecommunications Program».

[7] В оригинале: «Tisch School of the Arts».

Студенты Программы интерактивных телекоммуникаций расширили границы возможного в этой области, и их работа отражена во многих проектах. В конкретных случаях я называю имена, но, в общем, хочу поблагодарить всех студентов, которые прослушали мой курс «Сетевые объекты», за то, что они помогли мне самому подробно разобраться в этом предмете. Особенно велико участие студентов 2006 и 2007 годов, так как им пришлось заниматься по первым черновым версиям этой книги, и они обнаружили в них несколько важных ошибок.

Несколько человек внесли значительный вклад в эту книгу в форме кода, идей или работы. Джефф Смит (Geoff Smith) дал курсу его название: «Сетевые объекты» и познакомил меня с понятием объектно-ориентированного оборудования. Джон Шиммель показал мне, как выполнять вызовы HTTP с помощью микроконтроллера. Дан О'Саливан предоставил серверный код, лежащий в основе всего моего серверного кода. Благодаря совету Дана Шифмана по стилю кодирования, улучшилась читаемость моего кода Processing. Роберт Фалуди (Robert Faludi) предоставил большое количество фрагментов кода, улучшил читаемость примеров кода XBee, а также исправил много ошибок в нем. Макс Уитни (Max Whitney) помогла мне наладить работу связи Bluetooth и сделать кровать для кошки (несмотря на ее аллергию на представителей семейства кошачьих). Денис Кроули (Dennis Crowley) разъяснил мне возможности и ограничения двумерных штрихкодов. Крис Хиткоут (Chris Heathcote) оказал значительное влияние на мое понимание геолокации. Дарел Бишоп (Durrell Bishop) помог мне разобраться с идентификацией объектов. Майк Кунявски (Mike Kuniavsky) и слушатели семинара «Аппаратные наброски»[8], проводившегося в 2006 и 2007 годах, помогли мне увидеть эту работу, как часть большего сообщества, а также познакомили меня со многими новыми инструментами. Коту по имени Лапша[9] пришлось смириться со множеством разных глупостей, чтобы мы могли сделать для него кровать и снять фотографии. Сразу же надо оговориться, что при подготовке этой книги не пострадало ни одно животное, хотя одно из них пришлось задабривать с помощью кошачьей мяты.

Кэйси Рис (Casey Reas) и Бен Фрай (Ben Fry) сделали возможной программную часть этой книги, создав платформу Processing. Без Processing реализация программной части сетевых объектов была бы намного труднее. Также было бы невозможно создание простого, элегантного программного интерфейса для Arduino и Wiring[10]. Аппаратную часть этой книги сделали возможной создатели Arduino и Wiring: Массимо Банци (Massimo Banzi), Джанлука Мартино (Gianluca Martino), Давид Квартиелес (David Cuartielles) и Давид Меллис (David Mellis) — по части Arduino, а также Эрнандо Барраган (Hernando Barragan) — по Wiring и Николас Замбетти (Nicholas Zambetti), соединивший обе эти платформы. Мне повезло работать с ними.

Хотя я старался упоминать в книге многих поставщиков деталей, Натан Сидл (Nathan Seidle) из «SparkFun» заслуживает особого упоминания. Без него эта книга не была бы такой, какой она есть. В то время, как я годами говорил об объектно-ориентированном оборудовании, Натан и ребята из «SparkFun» потихоньку воплощали его в жизнь.

Я также хочу сказать спасибо команде поддержки фирмы «Lantronix» — они всегда предоставляли качественные продукты и превосходную поддержку. Гарри Моррис (Garry Morris), Гэри Маррс (Gary Marrs) и Дженни Эйзенхауэр (Jenny Eisenhauer) всегда бодро и любезно отвечали на мои бесконечные запросы по электронной почте и телефонные звонки.

Идеи для проектов этой книги я заимствовал у своих коллег со всего мира из обсуждений на семинарах и простых встречах. Мои благодарности членам преподавательского состава и студентам, с которыми я работал в программе «Интерактивный дизайн» в Королевском художественном колледже[11], в программе «Цифровые СМИ и искусство» в Калифорнийском университете в Лос-Анджелесе[12], в программе «Интерактивный дизайн» в Колледже архитектуры и дизайна Осло, в Институте интер-

[8] В оригинале: «Sketching in Hardware».

[9] В оригинале: «Noodles». По всей видимости, в кличке кота нашло свое отражение увлечение автора фильмом «Однажды в Америке», где герой Роберта де Ниро носит эту же кличку. — *Ред.*

[10] Еще одно микроконтроллерное устройство наподобие Arduino.

[11] В оригинале: «Royal College of Art».

[12] В оригинале: «University of California at Los Angeles».

активного дизайна Ивреи[13] и в Институте интерактивного дизайна Копенгагена[14].

Источником вдохновения для этой книги послужили многие проекты сетевых объектов. Я выражаю свою благодарность авторам этих проектов, чьи работы используются в качестве примеров: Туан Анх Т. Нгуен (Tuan Anh T. Nguyen), Джоо Йоун Паек (Joo Youn Paek), Дория Фэн (Doria Fan), Маурисио Мело (Mauricio Melo) и Джейсон Кауфман (Jason Kaufman). Тарик Корула (Tarikh Korula) и Джош Руки-Ли (Josh Rooke-Ley) — авторы «Необычайных проектов». Джин-Йо Мок (Jin-Yo Mok), Алекс Байм (Alex Beim), Эндрю Шнайдер (Andrew Schneider), Гилад Лотан (Gilad Lotan), Анжела Пабло (Angela Pablo), Моуна Андраос (Mouna Andraos), Сонали Сридхар (Sonali Sridhar) и Сара Йохансон (Sarah Johansson), а также Франк Ланц (Frank Lantz) и Кевин Славин (Kevin Slavin) — авторы проекта «Территория/Код».

Богатый опыт позволил мне получить сотрудничество с журналом «MAKE». Основатель журнала, Дэйл Докерти (Dale Dougherty), поддерживал все мои идеи, с пониманием относился к возникавшим задержкам и проявлял благосклонность, когда я хотел попробовать что-то новое. Он никогда не говорил «нет», не предложив при этом приемлемой альтернативной версии проекта (которая часто оказывалась лучшей, чем забракованная). Брайан Джепсон (Brian Jepson) сделал все и больше своих редакторских обязанностей в реализации всех проектов, предлагая модификации, выполняя отладку кода, помогая с фотографиями и иллюстрациями и предоставляя всеобщую поддержку. Сказать, что эта книга была бы невозможной без его помощи, было бы большим преуменьшением — лучшего редактора нельзя было бы и пожелать. Спасибо Нэнси Котари (Nancy Kotary) за тщательную вычитку рукописи книги. Кейти Уилсон (Katie Wilson) сделала все, чтобы эта книга выглядела и читалась намного лучше, чем я когда-либо мог надеяться. Также спасибо Тиму Лиллису (Tim Lillis) за иллюстрации. В общем, спасибо всей команде журнала «MAKE».

Спасибо моим агентам: Лауре Левин (Laura Lewin), которая привела процесс в действие, и Нилу Салкинду (Neil Salkind), который его продолжил, а также всей команде поддержки «Студии Б». Наконец, спасибо моей семье и друзьям, которым приходилось выслушивать мои восторженные разглагольствования и жалобы отчаяния в ходе работы над книгой. Я очень люблю всех вас.

[13] В оригинале: «Interaction Design Institute Ivrea».

[14] В оригинале: «Copenhagen Institute of Interaction Design».

Мы хотим знать ваше мнение

Если у вас есть какие-либо замечания или вопросы по этой книге, вы можете задать их издателю по следующему адресу:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (тел. в США или Канаде)
(707) 829-0515 (тел. международный или местный)
(707) 829-0104 (факс)

Список опечаток, примеры и дополнительная информация по книге содержится на ее веб-сайте по адресу: **www.makezine.com/go/MakingThingsTalk**.

Оставить комментарий или задать технические вопросы по книге можно электронной почтой по адресу: **bookquestions@oreilly.com**.

Maker Media — подразделение издательства «O'Reilly Media», полностью посвященное расширяющемуся обществу изобретательных людей, которые верят, что если что-то можно вообразить, его можно сделать. Подразделение состоит из журналов «MAKE», «CRAFT» и «Maker Faire», а также серии книг «Hacks», и поощряет дух «сделай сам» посредством предоставления инструкций и созидательной поддержки.

Дополнительную информацию о «Maker Media» можно получить здесь:

- «MAKE»: **www.makezine.com**
- «CRAFT»: **www.craftzine.com**
- «Maker Faire»: **www.makerfaire.com**
- «Maker SHED»: **www.makershed.com**

Примечания ко второму изданию

Две новые тенденции побудили меня переписать эту книгу: развитие движения за открытое оборудование и рост культуры прямого участия, в том числе и в деле создания интерактивных устройств. Это хорошо видно на примере быстрого увеличения круга пользователей Arduino, в частности, и числа участников движения за открытое оборудование, в целом. Последствия этих тенденций все еще в процессе оценки, но одно ясно — объектно-ориентированное оборудование и физические вычисления становятся повседневной реальностью. В настоящее время электроникой занимается намного больше людей, чем я мог бы вообразить в 2005 году.

Прежде чем любая технология может быть принята в общее использование, она должна занять место во всеобщем воображении. Люди без каких бы то ни было знаний технологии должны иметь хоть какое-нибудь представление о том, что это за технология и для чего ее можно использовать. До 2005 года я тратил много времени, объясняя людям, что такое физические вычисления, и что я имел в виду под «сетевыми объектами». В настоящее время все знают, например, контроллеры Wii или Kinect как устройства, которые расширяют диапазон физических воздействий человека, доступный для восприятия компьютерами. Сегодня также трудно найти несетевое электронное устройство.

Понимание широкими массами этих понятий вызывает большое удовлетворение, но что еще более захватывающе, так это видеть все большее распространение использования устройств на их основе. Люди не просто пользуются своими устройствами Kinect для игр, а встраивают их в интерфейсы, помогающие лицам с ограниченными физическими возможностями. Они не просто играются с контроллером Wii, а применяют его для управления музыкальными инструментами. Люди привыкли к тому, что они могут менять целевое использование своих электронных устройств, и делают это.

Когда я стал членом проекта Arduino, то надеялся, что Arduino может стать чем-то более легким для самостоятельного модифицирования, чем бытовые электронные устройства того времени, но в то же самое время будет не таким трудным в освоении, чем микроконтроллерные системы. Я полагал, что хорошим подходом к достижению этой цели станет метод открытых источников, так как он позволил бы принципам платформы Arduino распространиться за пределы созданных нами моделей. Эта надежда была воплощена в жизнь посредством множества производных плат, шилдов, побочных продуктов и вспомогательных устройств, которые стали доступными в течение нескольких последних лет. Чудесно видеть, сколько народу не просто создает электронные устройства, которые могут модифицировать другие люди, но делает это таким образом, что для этого не требуется профессиональная подготовка.

Развитие шилдов и библиотек для Arduino оказалось столь значительным, что второе издание книги можно было бы написать почти без необходимости выполнять какое бы то ни было программирование или создавать схемы заново. Почти для каждого проекта в книге существует уже готовый шилд или библиотека. Однако сборка устройств из готовых деталей позволяет получить только определенный уровень знаний, поэтому я попытался все же показать здесь некоторые из принципов, лежащих в основе электронных коммуникаций и физических интерфейсов. В тех случаях, когда существует простое аппаратное решение, я его приводил, но также показывал содержащуюся в нем схему. Самые лучшие библиотеки кода и схемные решения основаны на подходе, который я называю «стеклянным ящиком». Под этим я имею в виду, что для работы они скрывают все непривлекательные подробности и предоставляют удобный интерфейс, но при этом позволяют, кому хочется, заглянуть вовнутрь и увидеть, что в них происходит. Более того, они хорошо сконструированы, и жуткие подробности не выглядят такими уж и жуткими при их близком рассмотрении. Я надеюсь, что это издание книги выполнит поставленную перед ним цель таким же самым образом.

Справка по программному обеспечению

Со времени, когда я начал работу над этим изданием, в платформу Arduino было внесено несколько существенных изменений. В частности, Arduino IDE[15] тогда находилась в стадии бета-разработки, но к моменту выхода книги будет доступна версия 1.0. Если вы уже знакомы с Arduino, убедитесь, что вы загрузили версию 1.0 бета 1 или более позднюю версию IDE. Для этой книги использовалась версия Arduino 1.0 бета 1, которая доступна по адресу **http://code.google.com/p/arduino/wiki/Arduino1**. Конечная версия 1.0 будет доступна на странице загрузок веб-сайта **www.arduino.cc**. Периодически проверяйте этот сайт на наличие последних обновлений. Исходный код для книги можно загрузить из хранилища gitHub по адресу **https://github.com/tigoe/MakingThingsTalk2**, а извещения об изменениях в коде публикуются в блоге **www.makingthingstalk.com**.

Справка по оборудованию

Чтобы сконцентрироваться на коммуникации между физическими устройствами, я выбрал в качестве базовой аппаратной схемы для этого издания плату Arduino Uno. Все проекты из этой книги будут работать с Arduino Uno (при условии использования соответствующих вспомогательных устройств и шилдов). Несколько проектов были разработаны на специальных моделях Arduino — таких как Arduino Ethernet или Arduino LilyPad, поскольку их форм-фактор оказался наиболее подходящим. Но даже эти проекты были проверены на Arduino Uno. Любой микроконтроллер, совместимый с Arduino Uno, должен исполнять приведенный здесь код и сопрягаться с этими схемами.

Благодарности за второе издание

Круг людей, которые делают эту книгу возможной, продолжает увеличиваться.

Изменения во втором издании были в немалой мере вызваны работой моих партнеров по команде Arduino. Робота с Массимо Банци, Давидом Квартиелесом, Джанлукой Мартино и Давидом Меллисом продолжает быть приятной, стимулирующей и наполненной сюрпризами. Мне повезло сотрудничать с ними.

Руководство Программы интерактивных коммуникаций в университете Нью-Йорка продолжает оказывать мне поддержку во всей моей профессиональной работе. Весь этот проект с книгой был бы невозможным без участия в нем моих коллег по этой программе. Как всегда, Дан О'Саливан продолжал давать мне ценные советы по многим проектам. Даниель Шифман и Шан Вэн Эвери предоставили помощь с версиями Processing для персонального компьютера и для Android. Марианна Петит, Нэнси Хэкинджер, Клэй Ширки и Марина Зурков (Marina Zurkow) оказали важную моральную поддержку. Ред Бёрнз, как всегда, продолжает подавать мне пример того, как наделять людей новыми возможностями, обучая их понимать технологии, которые управляют их жизнями.

Команда постоянных исследователей и адъюнкт-профессоров Программы интерактивных коммуникаций продолжает, как всегда, меняться и, тоже как всегда, оказывать мне неоценимую помощь. Для этого издания мне помогали с примерами, проверяли проекты и заменяли меня в Программе интерактивных коммуникаций такие славные ребята, как Мустафа Багдатли (Mustafa Bagdatli), Каролина Браун (Caroline Brown), Джеремайя Джонсон (Jeremiah Johnson), Мередит Хэссон (Meredith Hasson), Леша Ходгсон (Liesje Hodgson), Крейг Капп (Craig Kapp), Ади Маром (Adi Marom), Ариэль Неварес (Ariel Nevarez), Поль Ротман (Paul Rothman), Итай Бенджамин (Ithai Benjamin), Кристиан Серрито (Christian Cerrito), Джон Диматос (John Dimatos), Шиовай Фенг (Xiaoyang Feng), Кейси Кинцер (Kacie Kinzer), Занна Марш (Zannah Marsh), Кори Меншер (Corey Menscher), Мэт Паркер (Matt Parker) и Тым Твилман (Tymm Twillman).

Члены преподавательского состава Томас Герхардт (Thomas Gerhardt), Скотт Фицджеральд (Scott Fitzgerald), Рори Наджинт (Rory Nugent) и Дастин Робертс (Dustyn Roberts) привнесли мой материал в курс «Введение в физические вычисления».

[15] Integrated Development Environment — интегрированная среда разработки.

Роберт Фалуди остается моим источником для всего, связанного с компанией «Digi International» и их семейством радиоплат XBee.

Мои благодарности Антуанетте ЛаСорса (Antoinette LaSorsa) и Лил Тролструп (Lille Troelstrup) из организации «Adaptive Design Association» за разрешение использовать дизайн их качающейся платформы, проект которой рассматривается в *главе 5*.

Многие люди из списка рассылки наших разработчиков и преподавателей внесли свой вклад в развитие платформы Arduino. В частности, Микал Харт (Mikal Hart), Майкл Марголис (Michael Margolis), Адриан МакЮэн (Adrian McEwen) и Лимор Фрид (Limor Fried) оказали большое влияние на эту книгу, работая над ключевыми библиотеками связи — такими как SoftwareSerial, Ethernet и TextFinder, а также присылая личные советы и дружелюбные ответы на многие из моих вопросов. Книга Майкла Марголиса «Рецептурный справочник по Arduino»[16] (издательство O'Reilly) также послужила в качестве справочного материала для некоторых фрагментов кода, опубликованных мной здесь. Спасибо Райану Маллигану (Ryan Mulligan) и Александру Бревигу (Alexander Brevig) за их библиотеки, адаптированные версии которых использованы в этой книге.

Владельцы компании «Adafruit» Лимор Фрид и Филипп Торроне (Phillip Torrone) в течение всей моей работы над книгой снабжали меня своими советами и критикой и оказывали поддержку. Вообще, компании «Adafruit» и «SparkFun» — это для меня основные источники деталей, поскольку их продукты отличаются отличной и надежной работой. Кроме того, Натан Сидл из «SparkFun» продолжает быть одним из моих основных критиков и советчиков.

Второе издание книги имеет лучшую графику благодаря системе Fritzing. Это средство открытого доступа для создания электронных схем можно загрузить на веб-сайте **http://fritzing.org** — Рето Веттач (Reto Wettach), Андре Кнориг (Andre Knorig) и Джонатан Коэн (Jonathan Cohen) создали замечательный инструмент для разработки схем и плат. Также спасибо Райану Оуэнсу (Ryan Owens) из «SparkFun» за предоставление мне первого доступа ко многим разработанным им чертежам деталей. Спасибо Джорджио Оливеро (Giorgio Olivero) и Джоди Калкин (Jody Culkin) за рисунки, добавленные в это издание.

Мои благодарности Давиду Бойхану (David Boyhan), Джоди Калкин, Заку Ивланду (Zach Eveland) и Габриэле Гутиерес (Gabriela Gutierrez) за чтение частей рукописи и озвучивание своих мнений и рекомендаций.

Также спасибо Киту Кейси (Keith Casey) из «Twilio», создателю Amarino Бонифазу Кауфману (Bonifaz Kaufmann), Андреасу Горансону (Andreas Goransson) за его помощь по Android, а также Кэйси Рис и Бену Фрай за создание версии Processing для Android и за их отзывы и советы по разделу на Android.

Новый материал в этом издании дополнен и новыми проектами. За это спасибо Бенедетте Пиантанелле (Benedetta Piantella), Джастину Даунзу (Justin Downs) из «Groundlab», а также создателям «SIMbalink» Мередит Хэссон, Ариэлю Неваресу и Нахане Шеллинг (Nahana Schelling). Спасибо Тимо Арналлу (Timo Arnall), Элнару Сниву Мартинуссену (Elnar Sneve Martinussen) и Йорну Кнутсену (Jorn Knutsen) (веб-сайт: **www.nearfield.org**) за их помощь и сотрудничество по RFID. За напоминание, каким быстрым может быть протокол DMX-512 и как легко им пользоваться, спасибо Даниэлю Хиршману (Daniel Hirschmann). Спасибо Мустафе Багдатли за его совет по проекту «Каменное лицо», а Франциске Gilbert (Frances Gilbert) и Джейку за их роль в проекте «КотоКам 2». Антон Чехов, примите мои извинения, и спасибо Тали Падану за комическую идею.

Спасибо Джиане Гонсалес (Giana Gonzalez), Юнгхуи Киму (Younghui Kim), Дженифер Магнолфи (Jennifer Magnolfi), Джин-Йо Мок (Jin-Yo Mok), Мэту Паркеру (Matt Parker), Эндрю Шнайдеру (Andrew Schneider), Гиладу Лотану (Gilad Lotan), Анжеле Пабло (Angela Pablo), Джеймсу Барнетту (James Barnett), Моргану Ноэлю (Morgan Noel), кошке Лапше и кукле Мартышкину за проекты моделирования.

[16] «Arduino Cookbook».

Как всегда, спасибо команде журнала «MAKE», особенно моему редактору и сотруднику Брайану Джепсону. Благодаря его терпению и настойчивости второе издание книги увидело свет. Благодарю технического редактора Скотта Фицджеральда, который также помог собрать воедино все части книги. То, что в этой книге есть часть про Интернет, это заслуга Скотта. Спасибо моему агенту Нилу Салкинду и всем из «Студии Б».

В последние недели работы над этим изданием книги несколько близких друзей пришли мне на помощь и сделали возможным то, с чем я бы не смог справиться своими собственными силами. Зак Ивланд (Zach Eveland), Денис Хэнд (Denise Hand), Джэнифер Магнолфи, Клайв Томпсон (Clive Thompson) и Макс Уитни (Max Whitney) днем и вечером помогали мне нарезать, паять, монтировать и собирать многие из последних проектов, а также составляли мне компанию, покуда я работал над материалом книги. Джо Хобайка (Joe Hobaica) пожертвовал несколько дней на управление производством для завершения книги. Он организовал фотодокументирование большинства новых проектов, спланировал мой поток работ, вел список задач, покупал разные детали, проверял связность, а также напоминал мне, когда спать и есть. Все эти люди напомнили мне, что работу над наделением вещей способностью общаться друг с другом лучше всего выполнять, общаясь с друзьями.


FOCUS
GAIN

Глава 1

СРЕДСТВА

Эта книга представляет собой что-то вроде сборника кулинарны рецептов, и в ее первой главе я познакомлю вас с основными используемыми при готовке ингредиентами. Описанные здесь понятия и инструменты найдут применение во всех остальных главах книги. Для каждого инструмента предоставляется достаточно информации, чтобы вы могли сказать «Здравствуй, мир!» с его помощью. Скорее всего, некоторые из рассматриваемых в этой главе инструментов или подобные им вам уже приходилось держать в руках. В таком случае вы можете спокойно пропускать знакомый вам материал и переходить к инструментам, с которыми вам еще не приходилось сталкиваться. Использование этих инструментов в проектах, представленных в следующих главах, демонстрирует для большинства из них лишь малую толику того, на что они способны, поэтому рекомендуется самостоятельно разобраться с менее знакомыми вам инструментами, чтобы получить более полное представление о том, что они могут делать. Для этого в книге приводятся ссылки на материалы для дальнейшего самостоятельного изучения малознакомых вам инструментов.

Happy Feedback Machine[1], созданная Туаном Анхом Т. Нгуеном (Tuan Anh T. Nguyen)

Основное удовольствие, получаемое от взаимодействия с этим устройством, дае щелкание тумблерами и вращение ручек. Огоньки и звуки, которые оно при это производит, — вторичны, и большинство людей, в него играющих, запоминаю свои ощущения, а не его поведение.

[1] Это название можно приблизительно перевести так: «Машина удовольствия на основе обратно

Все начинается с прикосновения

Все рассматриваемые в этой книге объекты — как осязаемые, так и неосязаемые — демонстрируют определенное поведение. Программные объекты отправляют и получают сообщения, хранят данные или делают и то, и другое. Физические объекты двигаются, светятся или издают звуки. Первый вопрос, который нужно задать об объекте, — что он делает? А следующий вопрос — каким образом принудить объект делать то, что он должен делать? Или, иными словами, какой у этого объекта интерфейс?

Интерфейс объекта состоит из трех компонентов. Первый компонент — *физический интерфейс*. Это то, что мы трогаем: ручки, тумблеры, ключи, а также и датчики, которые реагируют на наши действия. Элементы, которые соединяют объекты, также являются частью физического интерфейса. Любая сеть объектов начинается и заканчивается физическим интерфейсом. И хотя некоторые объекты в сети (например, программные объекты) не обладают физическим интерфейсом, люди создают умозрительные модели работы системы на основе того или иного физического интерфейса. Компьютер есть нечто намного большее, чем просто комплект из клавиатуры, мыши и экрана, но он представляется нам состоящим именно из этого, потому что как раз эти устройства мы видим и трогаем. В систему можно заложить всякого рода замечательные функции, но если эти функции не очевидны в проявлениях, которые люди могут видеть, слышать или трогать, они никогда не будут к ним обращаться. Возможно, вы вспомните часы в видеомагнитофонах, постоянно мигавшие на отметке 12:00, потому что никто не мог разобраться, как их установить. Следовательно, из-за некачественного физического интерфейса страдает вся система.

Следующий компонент интерфейса — *программный интерфейс*: команды, посылаемые объектам, чтобы вызвать их реакцию. В ряде проектов мы будем разрабатывать свой программный интерфейс, в некоторых — использовать уже существующий. Наилучшие программные интерфейсы это те, которые имеют простые, единообразные функции, выдающие предсказуемый вывод. К сожалению, не все программные интерфейсы столь просты, как нам бы хотелось, поэтому будьте готовы к определенному экспериментированию, чтобы программные объекты делали то, что они должны делать. При изучении нового программного интерфейса будет полезно применять к нему такой же мысленный подход, как и к физическому интерфейсу. Не пытайтесь сразу же использовать все имеющиеся функции, а сначала изучите действие каждой из них по отдельности. Мы же не начинаем учиться играть на пианино сразу с разучивания фуги Баха... Так и здесь — изучение программного интерфейса не начинается с разработки на нем полноценного приложения, а продвигается постепенно, с изучения одной функции за другой. Эта книга содержит большое число проектов. И если какая-либо из функций проекта окажется вам непонятной, напишите простую программу с использованием только этой функции, чтобы разобраться с ней, после чего возвращайтесь к работе над всем проектом.

Наконец, третий компонент — *электрический интерфейс*, состоящий из импульсов электрической энергии, пересылаемых от одного устройства к другому и несущих закодированную в них информацию. Если только вы не разрабатываете новые объекты или соединения между ними, вам никогда не придется иметь дело с этим интерфейсом. Но если вы такую разработку осуществляете, то чтобы знать, как согласовывать объекты с небольшими различиями в их электрических интерфейсах, вам нужно разбираться в них достаточно серьезно.

Все дело в импульсах

Для взаимодействия друг с другом объекты используют *протоколы связи*. Протокол — это набор взаимно согласованных правил для взаимодействия между двумя или несколькими объектами.

Последовательные протоколы — такие как RS-232, USB и IEEE 1394 (который также называется FireWire и i.Link) — используются для подключения к компьютерам принтеров, приводов жестких и оптических дисков, клавиатур, мышей и прочих периферийных устройств. Сетевые протоколы — такие как Ethernet и стек протоколов TCP/IP — служат для соединения многих компьютеров посредством сетевых хабов (концентраторов), роутеров (маршрутизаторов) и коммутаторов. Протокол связи обычно определяет скорость обмена сообщениями, компоновку данных в сообщении и синтаксис обмена данными. Протокол для физических объектов также определяет электрические характеристики и иногда даже физическую форму соединителей. Однако протоколы не уточняют, что передается от объекта к объекту, — команды для выполнения объектом каких-либо действий полагаются на протоколы так же, как построение грамотных фраз в тексте книги опирается на качественную грамматику языка.

Тем не менее у всех протоколов, от простейшего для обмена сообщениями между микросхемами до наиболее сложного сетевого, есть один общий аспект — все они имеют дело с импульсами электричества. Цифровые устройства обмениваются информацией, передавая электрические импульсы (сигналы) по связывающему их соединению. Обмен информацией по проводному USB-соединению между мышью и компьютером тоже осуществляется с помощью электроимпульсов. Да и сетевые соединения также реализуются посредством передачи по проводам электрических импульсов. Для более протяженных расстояний и в целях увеличения пропускной способности линий вместо металлических проводов могут использоваться волоконно-оптические кабели, информация по которым передается в виде световых импульсов. А когда реализовать физическое соединение неудобно или невозможно, обмен информацией осуществляется посредством радиосигналов с использованием соответствующих *приемопередатчиков*[2]. *При этом значение передаваемых сигналов данных не зависит от носителя, по которому они передаются. Одну и ту же последовательность импульсов можно передать как по металлическим проводам или волоконно-оптическим кабелям, так и по радиоканалам. Учитывая, что все обмены сообщениями начинаются с последовательности импульсов (и что где-то есть руководство, описывающее последовательность этих импульсов), мы можем работать с любой встретившейся на нашем пути системой связи.*

[2] Как следует из названия, приемопередатчик представляет собой радиоустройство, позволяющее как передавать, так и принимать радиосигналы.

Компьютеры всех видов и размеров

В этой книге мы встретимся, по крайней мере, с четырьмя разными типами компьютеров, сгруппированными по их физическим интерфейсам. Самый знакомый из них — это персональный компьютер. Будь это настольный компьютер или ноутбук, персональный компьютер имеет клавиатуру, экран и мышь, и вы, скорее всего, пользуетесь ими каждый рабочий день. Эти три элемента — клавиатура, экран и мышь — составляют физический интерфейс персонального компьютера.

Второй тип компьютера, рассматриваемый в этой книге, — *микроконтроллер*, не обладающий физическим интерфейсом, посредством которого люди могут непосредственно с ним взаимодействовать. Это всего лишь электронная микросхема с контактами ввода и вывода, которые могут издавать и принимать электрические импульсы. Взаимодействие с микроконтроллером осуществляется с помощью трехшаговой процедуры:

1. Ко входам микроконтроллера подключаются датчики для преобразования разных типов энергии: движения, тепловой, звуковой — в электрическую энергию.

2. К выходам микроконтроллера подключаются двигатели, динамики и другие устройства для преобразования электрической энергии в физическое действие.

3. Создается и исполняется программа, определяющая, как изменения на входах воздействуют на выходы.

Иными словами, физический интерфейс микроконтроллера определяется самими пользователями.

Третий тип компьютера, с которым мы будем иметь дело в этой книге, — *сетевой сервер*. Это, в принципе, тот же самый настольный компьютер, и он даже может иметь клавиатуру, монитор и мышь. Различие заключается в том, что хотя сетевой сервер может делать все, что и обычный персональный компьютер, его основной задачей является осуществление обмена данными по сети. Большинство пользователей не думают о серверах как о физических устройствах, поскольку взаимодействуют с ними только по сети, используя свои локальные компьютеры в качестве интерфейса. При этом следует иметь в виду, что наиболее важным интерфейсом сервера для пользователей является его программный интерфейс.

Четвертая группа компьютеров это такой себе винегрет: мобильные телефоны, музыкальные синтезаторы, контроллеры двигателей и т. п. Некоторые из них обладают полноценным физическим интерфейсом, другие оснащены минимальным физическим, но развитым программным, у большинства же имеется урезанный интерфейс обоих типов. Хотя обычно мы не думаем о таких устройствах, как о компьютерах, в действительности они являются таковыми. И если рассматривать такие устройства, как объекты с интерфейсами, которыми можно манипулировать, будет легче понять, как все они обмениваются информацией, независимо от их конечного назначения.

“ Хорошие привычки

Взаимодействие сетевых объектов в чем-то схоже с личными отношениями. Основная проблема и тех, и других заключается в том, что посылая сигнал, мы никогда полностью не можем быть уверены в том, понимает ли получатель, что мы хотим этим сигналом сказать. Кроме того, существует бесчисленное множество вариантов потери сигнала или искажения его значения при передаче.

Вы-то, возможно, знаете, что чувствуете, но ваш партнер — нет. Все, что у нее (или у него) имеется, чтобы судить о ваших намерениях, — это слова, которые вы произносите, и действия, которые вы осуществляете. Точно так же, вы сами можете знать, какое сообщение отправляет ваш компьютер, как он его отправляет, и что означают все его биты, но у принимающего это сообщение компьютера может не быть ни малейшего понятия, что они означают, если только он не обладает программой для интерпретации полученного сообщения. Все что у него есть — это лишь импульсы (биты) электрической энергии. И чтобы обеспечить надежный, однозначный обмен информацией (будь то в личных

отношениях или в сетевых), нужно следовать нескольким очень простым правилам:

- слушайте больше, чем говорите;
- никогда не полагайте, что на другом конце услышали именно то, что вы сказали;
- предварительно договоритесь о правилах обмена информацией;
- просите уточнить неясные сообщения.

Слушайте больше, чем говорите

Самый лучший способ произвести хорошее первое впечатление и поддерживать добрые отношения — это быть хорошим слушателем. Слушать — задача более трудная, чем говорить. Говорить можно в любое время, когда нам угодно, но так как мы не знаем, когда другой человек может сказать что-либо, нам нужно слушать все время. Применительно к сетевым отношениям это означает, что создавать нужно такие программы, которые большую часть времени ожидают сообщения, а отправляют сообщения только тогда, когда это необходимо. Отправлять сообщения на постоянной основе часто легче, чем только тогда, когда нужно, но это может вызывать всякого рода проблемы. Ограничить отправку сообщений обычно не сложно, и полученные выгоды намного превосходят затраты.

Никогда не предполагайте

Слушатель не всегда слышит то, что, как вы думаете, вы ему говорите. Иногда причиной этого является неверное толкование услышанного, в других же случаях это следствие проблем с приемом. Если просто полагать, что наше сообщение было получено без каких бы то ни было сложностей, и продолжать действовать по этому предположению, мы получим кучу проблем. Не следует также сначала и до конца разрабатывать всю логику системы и все шаги, через которые должны проходить сообщения, а затем собрать и тестировать всю систему сразу.

Конечно, продумать всю систему заранее было бы неплохо, но создавать и тестировать ее следует небольшими шагами. Большинство ошибок при разработке таких проектов происходит при обмене информацией между объектами. Всегда сначала проверьте связь между объектами, отправив простое сообщение типа «Здравствуй, мир!», прежде чем продолжать реализовывать более сложные аспекты проекта. Держите процедуру для отправки этого сообщения под рукой на случай, если снова потребуется проверить связь между объектами.

Неправильный подход к обмену сообщениями — только одна оплошность, которую можно совершить. Для большинства проектов в этой книге потребуется создавать физический, программный и электрический компоненты интерфейса. Одной из самых распространенных ошибок, совершаемых при разработке подобных гибридных проектов, является предположение, что все проблемы находятся в одном месте. Довольно часто я ломал голову над программной частью, пытаясь решить проблему с передачей сообщения, чтобы только потом обнаружить, что получающее устройство даже не было подключено или было не готово получать сообщения. Не полагайте также, что проблемы со связью находятся в том элементе системы, с которым вы лучше всего знакомы. Как раз наоборот, очень часто они находятся в элементе, с которым вы знакомы менее всего и, соответственно, подсознательно избегаете искать их в нем. В случае проблем с прохождением сообщения, вспомните о каждом звене в цепочке от отправителя к получателю и проверьте каждое из них. А затем проверьте те звенья, которые вы пропустили при первой проверке.

Установите правила обмена информацией

В хороших личных отношениях вырабатывается общий язык, основанный на общем жизненном опыте. Мы стараемся наилучшим образом выразить что-либо, чтобы наш спутник по жизни был наиболее восприимчив к этому, а также вырабатываем быстрый способ для выражения понятий, которые приходится повторять регулярно. Качественный обмен данными также полагается на общие способы изложения ситуации, которые называются *протоколами*. Иногда может создаваться специальный протокол для всех объектов системы, в других же случаях могут использоваться уже существующие протоколы. При работе с уже существующим

протоколом убедитесь, что вы понимаете все его особенности, прежде чем использовать его для обмена сообщениями. Если у вас есть желание и возможность создать свой собственный протокол, при его определении в обязательном порядке примите во внимание требования как отправителя, так и получателя. Например, вы решили использовать протокол, который легко программируется для вебсервера, но может оказаться не под силу микроконтроллеру. Немного времени, потраченного на обдумывание сильных и слабых сторон обоих участников обмена данными, и немного компромисса касательно требований каждого из них позволят реализовать канал связи намного легче.

Попросите разъяснения

Принятые сообщения могут быть искажены до неузнаваемости. Например, вы слышите что-то не вполне логичное, но, тем не менее, реагируете на услышанное так, как вы это поняли, и только потом узнаете, что ваш собеседник имел в виду нечто совсем иное, а не то, о чем вы подумали. Поэтому в таких случаях, чтобы не сделать глупой ошибки, лучше сначала попросить уточнить сказанное. Точно так же в сетевых коммуникациях разумно прежде всего проверять, имеют ли смысл получаемые сообщения, и в противном случае запрашивать повтор передачи. Отправив сообщение, следует реально в этом удостовериться, а не только лишь полагать, что оно было действительно отправлено. Дело в том, что иногда не сказать ничего может быть хуже, чем сказать что-то не то. Мелкие проблемы в отношениях между людьми могут превратиться в крупные, если никто не говорит о наличии этих проблем. То же самое относится и к сетевым взаимодействиям. Например, одно устройство может ожидать до бесконечности сообщения от другого, не зная, что это устройство, например, вовсе не включено. Поэтому, когда нет подтверждения получения сообщения, отправьте сообщение повторно. Но не делайте это слишком быстро и дайте принимающему устройству достаточное время на ответ. Подтверждение сообщений может показаться лишней тратой времени, но это сэкономит много времени и усилий при разработке сложной системы.

Инструментарий

Учитывая, что в дальнейшем вы будете иметь дело с физическими, программными и электрическими интерфейсами объектов, вам потребуются «физические» инструменты, программное обеспечение и компьютерное оборудование.

«Физические» инструменты

Если вам ранее приходилось работать с электроникой или микроконтроллерами, у вас, скорее всего, уже имеются некоторые инструменты. На рис. 1.1 показаны инструменты и приспособления, наиболее часто применяемые для реализации проектов из этой книги. Такие, вполне обычные, инструменты можно приобрести у многих поставщиков (кое-кто из них упомянут в табл. 1.1). В пояснениях, приведенных под рис. 1.1, также упомянуты некоторые поставщики этих инструментов с указанием номера позиции инструмента по каталогу поставщика.

О поставщиках деталей

В этой книге упоминаются несколько поставщиков деталей. Я покупаю свои детали у разных поставщиков — в зависимости от того, кто из них может предложить самую качественную версию детали по самой низкой цене. Но иногда легче просто сразу покупать весь нужный комплект деталей у одного поставщика, который имеет в наличии их все, вместо того, чтобы искать в обширном справочнике того или иного поставщика, предлагающего какую-то одну деталь по более низкой цене. Упомянутые в табл. 1.1 поставщики ни в коем случае не являются обязательными, и вы можете свободно покупать детали у любого другого поставщика. Более полный список поставщиков приводится в *приложении*.

Кроме инструментов вам потребуются и некоторые часто используемые электронные детали и компоненты. Они также приведены в табл. 1.1 с указанием номера детали по каталогам соответствующих поставщиков. Не все поставщики имеют в наличии все потребные детали, поэтому в табл. 1.1 есть много пропусков.

О российских поставщиках деталей

В России достаточно обширный ассортимент электронных деталей, компонентов, оборудования и инструментов предлагается фирмами «Амперка» (**www.amperka.ru**), «Линуксцентр» (**www.linuxcenter.ru**) и «Чип и Дип» (**http://chipdip.ru**).

Бонус от фирмы «Амперка»

Специально для читателей этой книги «Амперка» предлагает особое кодовое слово **MTT14**, которое даст вам скидку при оформлении заказа онлайн.

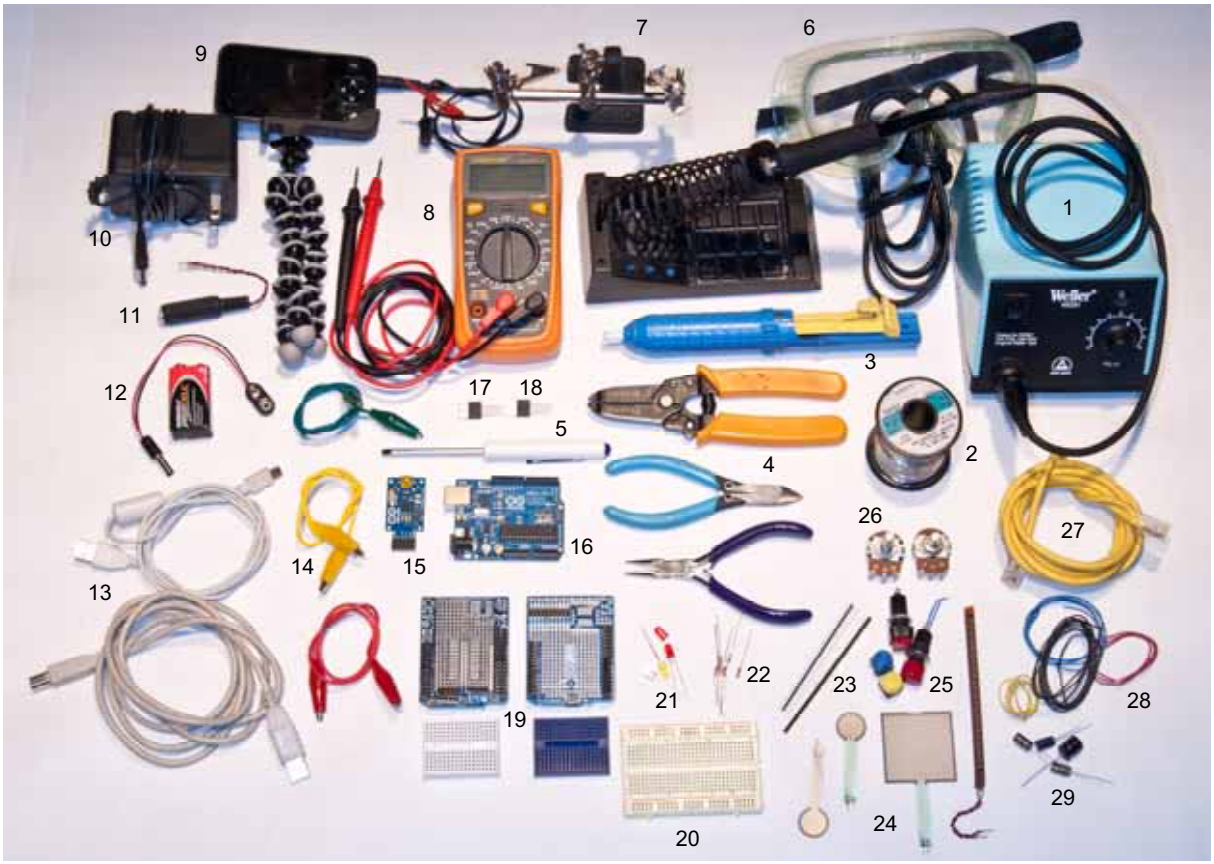


Рис. 1.1. Инструменты и детали, используемые в проектах этой книги. Их краткое описание дается далее

1. **Паяльник.** Лучше всего использовать паяльник средней ценовой категории. Дешевые паяльники быстро выходят из строя, но качественный паяльник наподобие Weller WLC-100 замечательно подходит для работы с электронными деталями небольшого размера. Не пользуйтесь так называемыми паяльниками холодной пайки. При их работе иногда возникает искра, которая может повредить чувствительные к статическому электричеству детали типа микроконтроллеров. Поставщики: «Амперка» (**http://amperka.ru**): AMP-X149; Jameco (**http://jameco.com**): 146595; Farnell (**www.farnell.com**): 1568159; RadioShack (**http://radioshack.com**): 640-2801 и 640-2078.

2. **Припой.** Лучше всего использовать припой 21-23 AWG[3]. Если есть возможность, пользуйтесь бессвинцовым припоем, который менее вреден для здоровья. Поставщики: «Амперка»: AMP-X150; Jameco: 668271; Farnell: 419266; RadioShack: 640-0013.

3. **Отсос для удаления припоя.** Инструмент, полезный для демонтажа схем. «Амперка»: AMP-X154; Jameco: 305226; SparkFun (**www.SparkFun.com**): TOL-00082; Farnell: 3125646.

4. **Инструмент для снятия изоляции, кусачки, длинногубцы.** Все эти инструменты необходимы для работы с проволокой. Не пользуйтесь их версией «три в одном», так как она создаст вам много проблем. Хорошие инструменты этого типа не столь дорогие, чтобы на них экономить.

   - **Инструмент для снятия изоляции:** Jameco: 159291; Farnell: 609195; SparkFun: TOL-00089; RadioShack: 640-2129A.
   - **Кусачки:** «Амперка»: 1PK-501A; Jameco: 161411; Farnell: 3125397; SparkFun: TOL-00070; RadioShack: 640-2043.
   - **Длинногубцы:** Jameco: 35473; Farnell: 3127199; SparkFun: TOL-00079; RadioShack: 640-2033.

5. **Мини-отвертка.** Приобретите комбинированную отвертку: с крестообразным и плоским жалами. Она будет вам постоянно нужна. «Амперка»: 8PK-2061; Jameco: 127271; Farnell: 4431212; RadioShack: 640-1963.

6. **Защитные очки.** Рекомендуется пользоваться ими при пайке, сверлении и выполнении прочих подобных операций. SparkFun: SWG-09791; Farnell: 1696193.

7. **«Третья рука».** Это приспособление намного облегчает удержание деталей при пайке и прочих операциях. «Амперка»: AMP-X156; Jameco: 681002; Farnell: 1367049.

8. **Мультиметр.** Не обязательно покупать дорогой инструмент. Достаточно будет такого, который позволяет измерять напряжение, ток, сопротивление и электропроводность. «Амперка»: AMP-X032; Jameco: 220812; Farnell: 7430566; SparkFun: TOL-00078; RadioShack: 22-182.

9. **Осциллограф.** Профессиональные осциллографы достаточно дорогие, но DSO Nano стоит всего лишь около $100, его будет вполне достаточно для работы. «Амперка»: TOL01241P; SparkFun: TOL-10244 (v2); Seeed Studio (**www.seeed-studio.com**): TOL114C3M; Maker SHED (**www.makershed.com**): MKSEEED11.

10. **Источник постоянного тока 9–12 В.** Он будет нужен вам постоянно. Не обязательно покупать его в магазине — можно использовать блок питания от какого-либо электронного устройства. В обязательном порядке определите полярность разъема, чтобы не подать питание обратной полярности на какой-нибудь компонент и не сжечь его. Большинство устройств в проектах используют разъем питания постоянного тока, в который вставляется штекер с внутренним диаметром 2,1 мм и внешним — 5,5 мм (см. позицию 11), поэтому ваш источник питания должен быть оснащен разъемом таких же размеров. «Амперка»: EN1000S (3–12 В, 1000 мА); Jameco: 170245 (12 В, 1000 мА); Farnell: 1176248 (12 В, 1000 мА); SparkFun: TOL-00298; RadioShack: 273-355 (9 В, 800 мА).

11. **Разъем (штекер) источника питания** с внутренним диаметром 2,1 мм и внешним — 5,5 мм. Эта деталь требуется для подключения микроконтроллера или макетной платы к источнику постоянного тока. Размер разъема наиболее употребим для источников питания, которые подойдут к схемам проектов, рассматриваемых в этой книге. «Амперка»: AMP-X042; Jameco: 159610; Digi-Key (**www.digikey.com**): CP-024A-ND; Farnell: 3648102.

12. **Разъем для батарейки типа «Крона» и сама батарейка.** Эти детали пригодятся, когда нужно запитать проект от батарейки. «Амперка»: AMP-W003; SparkFun: PRT-09518; Adafruit (**http://adafruit.com**): 80; Digi-Key: CP3-1000-ND и 84-4K-ND; Jameco: 28760 и 216452; Farnell: 1650675 и 1737256; RadioShack: 270-324 и 274-1569.

13. **USB-кабели.** Для проектов из этой книги нужны будут USB-кабели как типа A-на-B (стандартный USB-кабель), так и типа A-на-мини-B (USB-кабель, используемый для цифровых камер). «Амперка»: AMP-W004; SparkFun: CAB-00512, CAB-00598; Farnell: 1838798, 1308878.

14. **Тестовые проводники с разъемами типа «крокодил».** Удерживать несколько деталей при измерении параметров схемы задача не из легких — эти проводники намного облегчают ее. Jameco: 10444; RS (**www.rs-online.com**): 483-859; SparkFun: CAB-00501; RadioShack: 278-016.

15. **Переходник USB/TTL-Serial.** Преобразовывает USB-сигнал в сигнал TTL. Переходники для использования на макетных платах, наподобие показанного здесь модуля FT232, дешевле, чем бытовые устройства, и более удобны для использования в рас-

[3] AWG, American Wire Gauge System — американская система оценки проводов (стандарты на их диаметр). Подробности см. здесь: **http://ru.wikipedia.org/wiki/Американский_калибр_проводов**.

сматриваемых в этой книге проектах. «Амперка»: A000059; SparkFun: BOB-00718; Arduino Store (**store.arduino.cc**): A000014.

16. **Модуль микроконтроллера.** На рисунке показан микроконтроллер Arduino Uno. В США его можно приобрести в магазинах SparkFun и Maker SHED (**http://store.arduino.cc/ww/**), а также у многих международных дистрибьюторов. Информацию для своего региона см. на веб-сайте **http://arduino.cc/en/Main/Buy**.

17. **Регулятор напряжения.** Регуляторы напряжения сглаживают меняющееся входное напряжение в ровное (более низкое) выходное напряжение. Наиболее применяемое напряжение питания для проектов в этой книге: 5 В и 3,3 В. Будьте осторожны при использовании незнакомых регуляторов. Ознакомьтесь с их техническими характеристиками, чтобы знать назначение их выводов.

    - **3,3 В. «Чип и Дип»:** 9020002518; Digi-Key: 576-1134-ND; Jameco: 242115; Farnell: 1703357; RS: 534-3021.
    - **5 В. «Амперка»:** AMP-X065; Digi-Key: LM7805CT-ND; Jameco: 51262; Farnell: 1703357; RS: 298-8514.

18. **Транзистор TIP120.** Транзисторы функционируют как цифровые переключатели, позволяя управлять высокими токами или напряжениями с помощью намного меньших токов или напряжений. Существует много разных типов транзисторов, но для проектов в этой книге используются транзисторы TIP120. Обратите внимание, что внешне этот транзистор выглядит точно так же, как и регулятор напряжения рядом с ним (поз. 17). Иногда электронные компоненты с разным функционалом упаковываются в одинаковые физические корпуса, поэтому, чтобы определить назначение компонента, нужно проверить маркировку на его корпусе. «Чип и Дип»: 47667; Digi-Key: TIP120-ND; Jameco: 32993; Farnell: 9804005.

19. **Шилды для прототипов.** Это платы расширения для модуля микроконтроллера Arduino, которые представляют собой пластины с решеткой монтажных отверстий, в которые можно впаивать компоненты. Такие платы используются для создания прототипов своих схем. Можно, однако, использовать и макетные платы (также здесь показаны) — они удобны для быстрой реализации прототипов проектов. «Амперка»: A000082; Adafruit: 51; Arduino Store: A000024; SparkFun: DEV-07914; Maker SHED: MSMS01.

    - **Макетные платы для прототипных шилдов.** «Амперка»: AMP-X008; SparkFun: PRT-08802; Adafruit: входит в комплект платы; Digi-Key: 923273-ND

20. **Беспаечная макетная плата.** Иметь несколько таких плат под рукой не помешает. Лично мне нравятся платы с двумя длинными шинами с каждой стороны, что позволяет подключить питание и землю на обеих сторонах. «Амперка»: AMP-X003; Jameco: 20723 (2 шины с каждой стороны); Farnell: 4692810; Digi-Key: 438-1045-ND; SparkFun: PRT-00137; RadioShack: 276-002.

21. **Светодиоды для трассировки сигналов.** Для разработчика аппаратных устройств светодиоды то же самое, что и распечатки для разработчика программного обеспечения. Они позволяют быстро определить наличие напряжения между двумя точками и путь прохождения сигнала. Поэтому всегда следует иметь под рукой несколько таких компонентов. «Амперка»: AMP-X009-R4; Jameco: 3476; Farnell: 1057119; Digi-Key: 160-1144- ND; RadioShack: 278-016.

22. **Сопротивления (резисторы).** Для реализации проектов из этой книги вам понадобятся резисторы различных номиналов. Наиболее применяемые номиналы перечислены в табл. 1.1.

23. **Штыревые разъемы.** Эти разъемы будут вам нужны все время. Не помешает иметь под рукой также несколько разъемов гнездового типа. «Амперка»: AMP-X013; Jameco: 103377; Digi-Key: A26509-20-ND; Farnell: 1593411.

24. **Аналоговые датчики** (переменные сопротивления). Существует большое множество аналоговых датчиков для измерения разных физических свойств. Упомянутые здесь являются самыми простыми из аналоговых датчиков, и их легко встраивать в пробные схемы. Датчики изгиба и силы давления удобны для проверки схем и/или программ.

    - **Датчики изгиба.** «Амперка»: AMP-X127; Jameco: 150551; Images SI (**www.imagesco.com**): FLX-01.
    - **Датчики силы давления.** «Амперка»: AMP-X126; Parallax (**www.parallax.com**): 30056; Images SI: FSR-400, 402, 406, 408.

25. **Кнопки.** Для проектов в этой книге хорошо подойдут два типа кнопок. Первый — предназначенные для монтирования на печатные платы (подобные применяемым на платах Arduino и Wiring), которые в наших проектах на макетных платах используются в основном как кнопки сброса. Второй — предназначенные для монтирования в панелях, которые служат в качестве элементов интерфейса для конечных пользователей. Но, в принципе, можно использовать практически любой вид кнопок.

    - **Для монтирования на печатных платах.** «Амперка»: AMP-X026-Y; Digi-Key: SW400-ND; Jameco: 119011; SparkFun: COM-00097.
    - **Для монтирования на панелях.** «Чип и Дип»: 48828; Digi-Key: GH1344-ND; Jameco: 164559PS.

26. **Потенциометры.** Потенциометры необходимы для настройки параметров проектов. «Амперка»: AMP-X021; Jameco: 29081; SparkFun: COM-09939; RS: 91A1A-B28-B15L; RadioShack: 271-1715; Farnell: 1760793.

27. **Кабели Ethernet.** Для проектов потребуется пара сетевых кабелей. «Амперка»: AMP-X050; Jameco: 522781; RadioShack: 55010852.

28. **Провода черного, красного, синего и желтого цвета.** Для монтажа на беспаечной макетной плате лучше всего подойдет провод стандарта 22 AWG. Запаситесь, по крайней мере, тремя цветами, и всегда используйте красный для плюса питания, а черный — для общего. Немного организованности в использовании проводов может быть очень полезно.

    - **Черный** — Jameco: 36792.
    - **Синий** — Jameco: 36767.
    - **Зеленый** — Jameco: 36821.
    - **Красный** — Jameco: 36856; RadioShack: 278-1215.
    - **Желтый** — Jameco: 36919.
    - **Разные цвета** — «Чип и Дип»: 40119; RadioShack: 276-173.

29. **Конденсаторы.** Для реализации проектов из этой книги вам понадобятся конденсаторы различных номиналов. Наиболее применяемые номиналы перечислены в табл. 1.1.

Новые аппаратные компоненты

В следующих главах рассматриваются некоторые аппаратные компоненты, которые появились, когда работа над этим изданием книги уже велась, — это плата Arduino Ethernet, шилд Arduino Wi-Fi, беспроводной шилд, RFID-шилд, адаптер RS232/USB и ряд других. К этому моменту упомянутые в табл. 1.1 поставщики радиодеталей еще не назначили им номер, поэтому такие компоненты нужно искать по их названию. К тому времени, когда эта книга попадет вам в руки, поставщики должны уже иметь их в наличии.

Таблица 1.1. *Распространенные компоненты для проектов по электронике*

Код, полное имя и веб-сайт поставщика:

AMP — «Амперка» (**http://amperka.ru**)
CHD — «Чип и Дип» (**http://chipdip.ru**)
D — Digi-Key (**http://digikey.com**)
R — RS (**www.rs-online.com**)
J — Jameco (**http://jameco.com**)
F — Farnell (**www.farnell.com**)

Компонент, номинал	*Код поставщика и номер детали*
Сопротивления	
100 Ω	**AMP** — AMP-R100R-10, **D** — 100QBK-ND, **J** — 690620, **F** — 9337660, **R** — 707-8625
220 Ω	**AMP** — AMP-R220R-10, **D** — 220QBK-ND, **J** — 690700, **F** — 9337792, **R** — 707-8842
470 Ω	**AMP** — AMP-R470R-10, **D** — 470QBK-ND, **J** — 690785, **F** — 9337911, **R** — 707-8659
1K	**AMP** — AMP-R1K-10, **D** — 1.0KQBK, **J** — 29663, **F** — 1735061, **R** — 707-8669
10 K	**AMP** — AMP-R10K-10, **D** — 10KQBK-ND, **J** — 29911, **F** — 9337687, **R** — 707-8906
22 K	**CHD** — 40844, **D** — 22KQBK-ND, **J** — 30453, **F** — 9337814, **R** — 707-8729

Таблица 1.1 *(продолжение)*

Компонент, номинал	*Код поставщика и номер детали*
100 K	**AMP** — AMP-R100K-10, **D** — 100KQBK-ND, **J** — 29997, **F** — 9337695, **R** — 707-8940
1 M	**CHD** — 51741, **D** — 1.0MQBK-ND, **J** — 29698, **F** — 9337709, **R** — 131-700
Конденсаторы	
0,1 μF керамический	**AMP** — AMP-CC104-10, **D** — 399-4151-ND, **J** — 15270, **F** — 3322166, **R** — 716-7135
1 μF электролитический	**CHD —**31890, **D** — P10312-ND, **J** — 94161, **F** — 8126933, **R** — 475-9009
10 μF электролитический	**AMP** — AMP-CE10U-10, **D** — P11212-ND, **J** — 29891, **F** — 1144605, **R** — 715-1638
100 μF электролитический	**CHD —** 16360, **D** — P10269-ND, **J** — 158394, **F** — 1144642, **R** — 715-1657
Регуляторы напряжения	
3,3 В	**CHD** — 9020002518, **D** — 576-1134-ND, **J** — 242115, **F** — 1703357, **R** — 534-3021
5 В	**AMP** — AMP-X065, **D** — LM7805CT-ND, **J** — 51262, **F** — 1860277, **R —** 298-8514
Аналоговые датчики	
Изгиба	**AMP** — AMP-X127, **D** — 905-1000-ND, **J** — 150551, **R** — 708-1277
Давления	**AMP** — AMP-X126, **D** — 1027-1000-ND, **J —** 2128260
Светодиоды	
T1, зеленый, прозрачный	**AMP** — AMP-X009-G, **D** — 160-1144-ND, **J** — 34761, **F** — 1057119, **R** — 247-1662
T1, красный, прозрачный	**AMP** — AMP-X009-R, **D** — 160-1665-ND, **J** — 94511, **F** — 1057129, **R** — 826-830
Транзисторы	
2N2222A	**AMP** — AMP-X035-5, **D** — P2N2222AGOS-ND, **J** — 38236, **F** — 1611371, **R** — 295-028
TIP120	**CHD** — 47667, **D** — TIP120-ND, **J** — 32993, **F** — 9804005
Диоды	
1N4004-R	**AMP** — AMP-X045-5, **D** — 1N4004-E3, **J** — 35992, **F** — 9556109, **R** — 628-9029
3,3 В стабилитрон (1N5226)	**AMP** — AMP-X051, **D** — 1N5226B-TPCT-ND, **J** — 743488, **F** — 1700785
Кнопки	
Для печатных плат	**AMP** — AMP-X026-Y, **D** — SW400-ND, **J** — 119011, **F** — 1555981
Для панелей	**CHD** — 48828, **D** — GH1344-ND, **J** — 164559PS, **F** — 1634684, **R** — 718-2213

Таблица 1.1 *(окончание)*

Компонент, номинал	*Код поставщика и номер детали*
Беспаечные макетные платы	
разные	**AMP** — AMP-X003, **D** — 438-1045-ND, **J** — 20723, 20600, **F** — 4692810
Монтажный провод	
красный	**D** — C2117R-100-ND, **J** — 36856, **F** — 1662031
черный	**D** — C2117B-100-ND, **J** — 36792, **F** 1662027
синий	**J** — 36767, **F** — 1662034
желтый	**J** — 36920, **F** — 1662032
Потенциометр	
10 K	**AMP** — AMP-X021, **D** — 29081
Штыревые разъемы	
Прямые штырьки	**AMP** — AMP-X013, **D** — A26509-20-ND, **J** — 103377, **S** — PRT-00116
Штырьки под прямым углом	**CHD** — 929136020, **D** — S1121E-36-ND, **S** — PRT-00553
Гнездовые разъемы	**CHD** — 362520493, **S** — PRT-00115
Разъем для батареи «Крона»	
9 В	**AMP** — AMP-W003, **D** — 2238K-ND, **J** — 101470PS, **S** — PRT-00091

Программные продукты

Среда Processing

В этой книге используется многоцелевая программная среда, носящая название *Processing*. Она основана на платформе Java и предназначена для тех пользователей, которые хотят реализовывать свои проекты без необходимости подробно изучать программирование. Среда Processing — весьма полезный инструмент для объяснения понятий программирования, поскольку для осуществления значительных действий: создания сетевого соединения, подключения к внешнему устройству через последовательный порт, управление камерой — требуется сравнительно небольшой объем кода Processing. Этот бесплатный инструмент с открытым исходным кодом можно загрузить с веб-сайта **www.processing.org**.

Рис. 1.2. Окно редактора кода среды Processing

Поскольку среда Processing основана на Java, в программы Processing можно включать классы и

методы Java. Среда может исполняться под Mac OS X, Windows и Linux. Существуют также версии Processing для Android и JavaScript, так что это довольно-таки разносторонняя среда. Если по какой-либо причине вам не нравится работать в среде Processing, можно использовать примеры кода и комментарии из этой книги в качестве исходного кода для создания таких примеров в любой среде, которой вы отдаете предпочтение.

Загрузив и установив среду Processing на своем компьютере, запустите ее на выполнение. Откроется окно, наподобие показанного на рис. 1.2.

Теперь давайте создадим нашу первую программу в Processing. Введите следующий текст в окно редактора, а затем нажмите кнопку **Run** (Исполнить) — самую левую кнопку на панели инструментов Processing.

```
println(«Здравствуй, мир!»);
```

Не ахти какая программа, но это классическая первая программа в любом языке. Исполнение этой программы выводит текст **Здравствуй, мир!** в текстовом поле внизу окна редактора. Как видим, ничего сложного.

Программы Processing называются *скетчами* (sketch), и все данные скетча сохраняются в файле с именем скетча. Редактор Processing весьма прост и не содержит каких-либо примочек, отвлекающих внимание. Панель инструментов редактора состоит из кнопок для запуска и остановки исполнения скетчей, открытия существующих скетчей, сохранения скетчей и экспортирования скетчей в апплеты Java. Кроме того, скетчи можно экспортировать в виде автономных приложений. Файлы скетчей по умолчанию сохраняются в подпапке Processing папки Документы, но их можно сохранять в любой папке.

Далее мы рассмотрим более сложную программу, которая демонстрирует некоторые основные структуры программирования Processing.

Контекст использования кода

Все примеры кода в этой книге сопровождаются комментариями, указывающими контекст, в котором они должны использоваться: Processing, режим Android Processing, Arduino, PHP и т. п.

```
/*
Программа рисования треугольников
Контекст: Processing

Рисует треугольник при отпущенной левой кнопке мыши.
Очищает окно рисования при нажатии левой кнопки мыши.
*/

// объявляем переменные
float redValue = 0;    // переменная для красного цвета
float greenValue = 0;  // переменная для зеленого цвета
float blueValue = 0;   // переменная для синего цвета

// метод setup() исполняется один раз в начале программы:

void setup() {
size(320, 240); // устанавливаем размер окна рисования
background(0);  // устанавливаем черный фон окна рисования
fill(0);// задаем цвет для заполнения фигур (0 = черный)
smooth(); // рисуем фигуры со сглаженными кромками
}
```

```
// Метод draw() исполняется непрерывно, пока открыто окно
// рисования.
// Он обновляет окно, а также выполняет любые
// запрограммированные в нем действия:

void draw() {

// Выбираем произвольные значения красной, зеленой и синей
// составляющих цвета:
redValue = random(255);
greenValue = random(255);
blueValue = random(255);

// задаем цвет линии:
stroke(redValue, greenValue, blueValue);

// рисуем при отпущенной левой кнопке мыши (игнорируя все
// обычные правила):
if (mousePressed == false) {
// рисуем треугольник
triangle(mouseX, mouseY, width/2, height/2,pmouseX,
  pmouseY);
}
// очищаем окно рисования при нажатии левой кнопки мыши:
else {
background(0);
fill(0);
}
}
```

Любая программа Processing должна содержать два метода (процедуры): `setup()` и `draw()`. Метод `setup()` исполняется один раз в начале программы, устанавливая все начальные условия, — такие как размер окна апплета, начальные значения переменных и т. п. А метод `draw()` — основной цикл программы, исполняющийся непрерывно, пока открыто окно апплета.

Для переменных Processing нужно задавать тип их данных. В приведенной здесь программе переменные `redValue`, `greenValue` и `blueValue` — плавающего типа, то есть они могут содержать числа с плавающей запятой. Другие распространенные типы данных переменных, с которыми нам придется работать, это:

- `int` — целочисленные значения;
- `boolean` — значения `true` и `false`;
- `string` — текст;
- `bytes`.

Синтаксис программ в среде Processing близок по стилю синтаксису языка C. Так же, как и переменные, все функции в нем тоже имеют тип данных. Многие функции имеют тип данных `void`, то есть они не возвращают никаких значений в результате исполнения. Все строки кода должны заканчиваться точкой с запятой, а блоки кода заключаются в фигурные скобки. Условные операторы (`if-then`), операторы цикла (`for-next`) и комментарии также используют синтаксис языка C. За исключением цикла `for-next`, все эти операторы продемонстрированы в приведенной ранее программе.

Пример цикла `for-next` демонстрируется в следующем фрагменте кода. Попробуйте создать свой скетч с использованием этого цикла. Для создания нового скетча выполните команду **New** в меню **File** среды Processing.

```
for (int myCounter = 0; myCounter <=10; myCounter++) {
  println(myCounter);
}
```

Пользователям BASIC

Если вы никогда не использовали цикл `for-next` в стиле C, этот пример может показаться вам чем-то страшным. В действительности, ничего сложного в нем нет. Все, что приведенный здесь код делает, это сначала устанавливает значение переменной `myCounter`, а затем исполняет заключенные в фигурные скобки инструкции до тех пор, пока это значение остается меньшим или равным 10. Инструкция `myCounter++` указывает программе добавить единицу к значению `myCounter` при каждом прохождении цикла. Эквивалентный код в BASIC выглядит следующим образом:

```
for myCounter = 0 to 10
   Print myCounter
next
```

Processing — удобный язык для экспериментирования, так как позволяет очень быстро создавать интерактивную графику. Он также может служить в качестве простого введения в программирование на языке Java. Если вы уже знаете этот язык, то можете вставлять код Java непосредственно в программы Processing. Возможности Processing можно расширять с помощью библиотек кода. В этой книге мы будем часто пользоваться двумя библиотеками кода для Processing: библиотекой последовательных ресурсов и библиотекой сетевых ресурсов.

Дополнительную информацию по синтаксису Processing легко найти в руководстве по этому языку, которое можно загрузить с веб-сайта **www.processing.org**. Узнать больше о программировании в Processing можно из книги «Processing: A Programming Handbook for Visual Designers and Artists»[4] (издательство MIT Press), написанной создателями этого языка Кэйси Рисом и Беном Фрайем, или из их более короткой книги «Getting Started with Processing»[5] (издательство O'Reilly). Также можно воспользоваться отличной книгой для начинающих «Learning Processing»[6] (издательство Morgan Kaufmann), написанной Даниэлем Шифманом. Кроме указанных, существует и множество других книг по Processing, так что вы сможете выбрать ту, которая вам больше подходит.

Приложения удаленного доступа

Одним из наиболее эффективных инструментов, которым мы будет пользоваться для отладки проектов из этой книги, является программа удаленного доступа, позволяющая получить доступ к *консоли командной строки* удаленного компьютера. Если вам никогда раньше не приходилось работать с интерфейсом командной строки, это средство может поначалу оказаться неудобным, но привыкнуть к нему несложно. Умение работать с интерфейсом командной строки особенно важно для захода на веб-серверы, так как работа со сценариями PHP, используемыми для этого, как раз осуществляется через консоль командной строки.

Дело в том, что большинство поставщиков веб-хостинга используют Linux, BSD, Solaris или другую UNIX-подобную операционную систему. Поэтому, для работы с веб-сервером на такой системе приходится подключаться к нему с помощью консоли командной строки.

Примечание

Если вы уже знаете, как создавать документы PHP и HTML и загружать их на веб-сервер, раздел «PHP» (см. далее) можно пропустить.

[4] «Processing: Справочник по программированию для визуальных конструкторов и творческих личностей».

[5] «Основы работы с Processing».

[6] «Изучаем Processing»

Хотя командная строка предоставляет наиболее прямой способ работы с PHP, некоторые предпочитают косвенный способ, создавая текстовые файлы на своем компьютере, а затем загружая их на удаленный компьютер. В зависимости от ограничений вашего веб-хостинга, это может оказаться единственной доступной вам опцией, — впрочем, на рынке есть много недорогих хостинговых компаний, предлагающих полный доступ к веб-серверу из командной строки. Но даже если вы предпочитаете работать с веб-сервером таким образом, для некоторых проектов в этой книге командная строка является единственным вариантом, поэтому весьма полезно получить некоторое представление о ней уже сейчас.

Для компьютеров Windows существует несколько программ для удаленного доступа, но мы будем пользоваться программой PuTTY (**www.puttyssh.org**). Процедура установки несложная — загрузите с веб-сайта программы установщик Windows для нее и запустите его на исполнение.

На компьютерах под Mac OS X и Linux для удаленного доступа можно использовать программу OpenSSH, которая входит в состав обеих этих операционных систем. Для работы с программой OpenSSH под Linux или Mac OS X нужно сначала запустить программу эмуляции командной строки этих ОС и выполнить в ней команду `ssh`, которая программу OpenSSH и запустит. В Mac OS X программа эмуляции командной строки называется Terminal и находится в подкаталоге Utilities каталога Applications. В Linux эмулятор командной строки может называться xterm, rxvt, Terminal или Konsole.

Программы ssh и telnet

Программа ssh — это более современная версия старой программы удаленного доступа для UNIX, называющейся telnet. Программа ssh обеспечивает более высокий уровень безопасности, зашифровывая отправляемые данные, таким образом защищая их на случай перехвата. В программе telnet шифрование отправляемых данных не применяется. Для связи между компьютерам следует использовать программу ssh всегда, когда это возможно. Иногда в этой книге используется программа telnet, но это лишь по той причине, что только эта программа может обеспечить требуемую для данного примера функциональность. Программу telnet можно рассматривать как старого приятеля — возможно он не самый популярный во дворе, возможно иногда он говорит слишком много и невпопад, но он всегда был на твоей стороне и ты знаешь, что ему можно доверять, когда все другие подвели тебя.

Установка соединения SSH

Mac OS X и Linux

Откройте программу терминала. Обычно приложение терминала предоставляет простое текстовое окно с приветствием наподобие следующего:

```
Last login: Wed Feb 22 07:20:34 on ttyp1
ComputerName:~ username$
```

Чтобы подключиться к требуемому веб-серверу, выполните в командной строке команду:

```
ssh имя_пользователя@myhost.com
```

Разумеется, вместо параметров `имя_пользователя` и `myhost.com` введите свое имя пользователя и адрес веб-сервера соответственно.

Windows

Запустите программу PuTTY (рис. 1.3).

Введите имя своего веб-сервера в поле **Host Name**, установите переключатель **SSH** в разделе **Connection type** и нажмите кнопку **Open**.

Когда программа выполнит подключение к удаленному компьютеру, будет выведен запрос ввести пароль. Введите пароль (вводимый текст не будет отображаться) и нажмите клавишу <Enter>.

Рис. 1.3. Главное окно программы PuTTY

Работа с командной строкой

Установив соединение с удаленным веб-сервером, вы увидите примерно такое сообщение:

```
Last login: Wed Feb 22 08:50:04 2006 from
216.157.45.215
[userid@myhost ~]$
```

Это значит, что вам доступна командная строка удаленного компьютера, и любая введенная вами команда на этом компьютере будет выполнена. Для начала узнаем, в какой папке мы находимся. Для этого выполним следующую команду:

```
pwd
```

Эта команда состоит из первых букв фразы `print working directory`, которая означает «напечатать рабочую папку». Она указывает компьютеру отобразить имя и путь текущей папки. (Многие команды UNIX весьма кратки — чтобы их легче было вводить. С другой стороны, аббревиатурные команды труднее запомнить.) В ответ сервер выведет путь и имя текущей папки:

```
/home/igoe
```

Это рабочая папка вашей учетной записи на сервере. На многих веб-серверах такая папка содержит вложенную папку, которая называется public_html или www и содержит файлы веб-сайта. Файлы, находящиеся в домашней папке (то есть файлы вне папок public_html или www), недоступны для просмотра посетителями веб-сайта.

Организация папок и файлов на веб-сервере

О точной организации папок и файлов в вашей домашней папке нужно проконсультироваться с администрацией хостинга.

Для вывода списка файлов папки служит команда `ls` (`list`):

```
ls -l .
```

Точки в конце команды

Точка в конце команды означает «текущая папка», а две точки — «родительская папка текущей папки».

Параметр `-l` означает `list long`, то есть требование выводить подробный список. В результате исполнения этой команды выводится примерно такой ответ:

```
total 44
drwxr-xr-x 13 igoe users 4096 Apr 14 11:42
public_html
drwxr-xr-x 3 igoe users 4096 Nov 25 2005 share
```

Это список всех файлов и вложенных папок, содержащихся в текущей папке, с указанием их *атрибутов*.

- В первом элементе строки указываются разрешения для действий с данным объектом: чтение, изменение или исполнение.
- Во втором элементе указывается количество ссылок на данный файл из других областей системы — эта информация в большинстве случаев не особо требуется.
- В третьем элементе указывается владелец объекта, а в четвертом — группа владельцев.
- В пятом элементе указывается размер объекта, а в шестом — дата и время последнего изменения.
- Наконец, в последнем элементе указывается имя объекта.

В среде UNIX все файлы, имена которых начинаются с точки, в общем случае не отображаются. Это требуется для некоторых особых файлов — таких, например, как файлы управления доступом, которые рассматриваются далее. Чтобы просмотреть содержимое папки, включая ее скрытые файлы, с командой `ls` нужно использовать параметр `-la`:

```
ls -la
```

Для перехода в другую папку служит команда `cd` (`change directory` — изменить каталог). Например, перейти в папку public_html можно, выполнив следующую команду:

```
cd public_html
```

Чтобы переместиться на один уровень выше в иерархии папок, выполняется следующая команда:

```
cd ..
```

Для возврата в домашнюю папку к команде `cd` добавляется параметр ~ (тильда):

```
cd ~
```

Возвратиться в домашнюю папку можно, просто выполнив команду `cd` без параметров.

Чтобы перейти во вложенную папку, эта папка указывается после родительской папки через косую черту — / (слеш):

```
cd public_html/cgi-bin
```

Чтобы указать *абсолютный* путь из главной папки сервера (которая называется root — корневая), в начале пути файла ставится такая же косая черта — /. Пути без косой черты в их начале называются *относительными*.

Для создания новой папки служит команда `mkdir` (`make directory`):

```
mkdir имя_папки
```

Эта команда создает новую папку в текущей папке. При просмотре содержимого пустой папки (каковой будет вновь созданная папка) с помощью команды `ls -la` выводятся только две строчки:

```
drwxr-xr-x 2 tqi6023 users 4096 Feb 17 10:19 .
drwxr-xr-x 4 tqi6023 users 4096 Feb 17 10:19
..
```

Первая строчка, с одной точкой в конце, — это ссылка на данную папку, а вторая, с двумя точками в конце, — ссылка на родительский каталог. Такие две ссылки будут находиться в любой папке, и удалить их нельзя.

Чтобы удалить папку, служит команда `rmdir` (`remove directory`) с указанием папки, подлежащей удалению:

```
rmdir имя_папки
```

Удалить можно только пустую папку, поэтому перед удалением папки нужно удалить все ее содержимое — как файлы, так и вложенные папки. Команда `rmdir` выполняется немедленно, без запроса на подтверждение удаления, поэтому нужно быть осторожным в ее использовании. Не следует также удалять какие бы то ни было папки или файлы, которые вы не создавали.

Управление доступом к файлам

Выполните команду `ls -l`, чтобы вывести список файлов текущей папки, и рассмотрите поближе *разрешения* для файлов. Например, обозначение:

```
drwx------
```

означает, что данный объект является папкой (`d` — directory), и создавший ее пользователь (который также называется *владельцем*) может просматривать или читать ее (`r` — read), записывать в нее (`w` — write), а также исполнять ее (`x` — execute). Рассмотрим другой пример, разрешения:

```
-rw-rw-rw
```

Тире вначале означает, что данный объект является файлом (а не папкой), и что владелец файла, группа владельцев (обычно владелец является членом этой группы), а также любой другой, кто имеет доступ к файлу, может просматривать его и выполнять запись в него. Первая группа `rw-` задает разрешения владельца, вторая — группы, а третья — всех остальных. Владелец объекта может изменить его разрешения с помощью команды `chmod`:

```
chmod go-w имя_файла
```

Параметры команды указывают пользователей, которых затрагивает данная команда, и изменяемые разрешения. В приведенном примере мы удаляем разрешения на запись (`-w`) для группы (`g` — group) владельцев файла и всех других (`o` — others), кроме владельца файла. А в следующем примере мы присваиваем права записи и исполнения для группы и других пользователей:

```
chmod go +wx имя_файла
```

Как мы уже поняли, в параметрах команды `chmod` буква `u` означает пользователя (user), `g` — группу (group), а `o` — прочие (others). Мы также знаем, что буква `r` означает разрешения чтения (read), `w` — записи (write), а `x` — исполнения (execute). Знак + (плюс) означает присвоение разрешений, а знак – (минус) — лишение их. Будьте осторожны, чтобы случайно не лишить разрешений самого себя (пользователя). Кроме того, выработайте привычку не предоставлять доступа к файлам для групп и прочих, если только в этом нет необходимости, — на крупных поставщиках услуг веб-хостинга иметь соседями по серверу сотни других пользователей — обычное дело.

Создание, просмотр и удаление файлов

Для работы с файлами полезными будут еще две программы командной строки: nano и less. Программа nano — это текстовый редактор. Это очень простой редактор, поэтому для объемных работ будет лучше редактировать текст на своем компьютере с помощью другого, более удобного редактора, а затем загружать готовый текст на сервер. Но для небольших правок прямо на сервере nano незаменим. Чтобы создать с помощью nano новый файл, выполните следующую команду:

```
nano имя_файла.txt
```

Откроется окно редактора (рис. 1.4).

Все команды для работы в nano вводятся с клавиатуры совместно с клавишей <Ctrl>. Например, для выхода из программы нужно нажать комбинацию клавиш <Ctrl>+<X>. Наиболее употребляемые команды перечислены в нижней части окна редактора.

Насколько редактор nano удобен для быстрого создания и редактирования небольших текстовых файлов, настолько для чтения таких файлов хорошо подходит редактор less. Этот редактор отображает текстовый файл по одному экрану за раз. Для просмотра текстового файла в less нужно просто ввести имя этого редактора, а за ним имя требуемого файла:

```
less имя_файла.txt
```

В результате содержимое файла будет выводиться поэкранно, с приглашением в виде двоеточия (:) в конце экрана. Нажатие клавиши пробела выводит следующий экран текста. Чтобы завершить работу программы, нажмите клавишу <q>. Определенно, похвастаться никакими крутыми наворотами less не может, но со своей основной задачей — просмотром длинных файлов — он справляется «на отлично».

С помощью редактора less можно также просматривать вывод других команд и почти любой программы командной строки, используя для этого оператор канала `|` (pipe operator). Например, следующая команда передает вывод команды `ls` для просмотра в редакторе `less`:

```
ls -la . | less
```

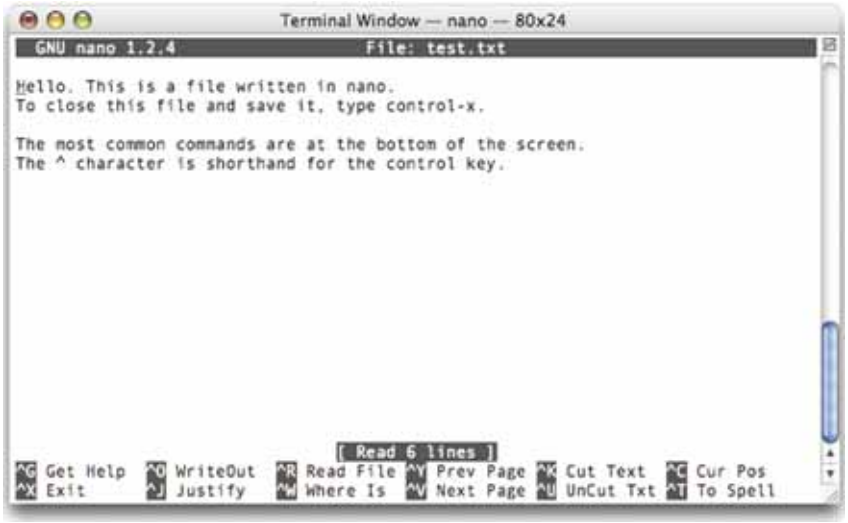


Рис. 1.4. Окно текстового редактора nano

Удалить любой файл можно с помощью команды `rm` (`remove`):

```
rm имя_файла
```

Подобно команде `rmdir`, команда `rm` не выводит запроса на подтверждение удаления файла, поэтому нужно быть осторожным в ее использовании.

Оболочка UNIX содержит много других команд, но рассмотренных здесь для начала будет вполне достаточно. Список наиболее употребляемых команд оболочки можно получить, выполнив в терминале команду `help`, а информацию об определенной команде — выполнив команду `man имя_команды`.

Чтобы завершить сеанс связи с сервером и закрыть подключение, выполните команду `logout`.

Дополнительную информацию по работе с командной строкой в UNIX- и Linux-системах можно почерпнуть в книге «UNIX Operating System» (издательство O'Reilly), авторы Джерри Пик (Jerry Peek), Грейс Тодино-Гонгет (Grace Todino-Gonguet) и Джон Стренг (John Strang).

PHP

Большинство серверных программ в этой книге написаны на языке PHP. Язык PHP — один из самых распространенных языков сценариев, которые исполняются на веб-сервере. Серверные сценарии (server-side scripts) — это программы, расширяющие возможности веб-сервера за пределы простого предоставления статических HTML-страниц. Они позволяют пользователю получить доступ к базам данных посредством браузера, сохранять данные веб-сеанса в текстовый файл, отправлять почтовые сообщения с веб-браузера и многое другое. Для большинства проектов в этой книге вам нужно иметь учетную запись веб-хостинга. Если у вас уже есть такая запись, она, скорее всего, предоставляет возможности PHP. Если же нет, то при выборе поставщика (провайдера) веб-хостинга убедитесь, что ваша учетная запись будет иметь такие возможности.

Чтобы начать работать с PHP, нужно установить соединение с вашей учетной записью на веб-хостинге, используя программу ssh, как это было показано в предыдущем разделе. Некоторые веб-хостинги не предоставляют возможности ssh-подключения, так что проверьте, предоставляет ли ваш провайдер веб-хостинга такую услугу. Если нет, подыщите другого провайдера веб-хостинга, разрешающего ssh-подключение, чтобы обрести гибкость, предоставляемую возможностью работы с веб-сервером с помощью командной строки.

Итак, подключившись к веб-серверу, выполните следующую команду:

```
php -v
```

Сервер должен ответить сообщением, подобным следующему:

```
PHP 5.3.4 (cli) (built: Dec 15 2010 12:15:07)
Copyright (c) 1997-2010 The PHP Group
Zend Engine v2.3.0, Copyright (c) 1998-2010
Zend Technologies
```

Полученное сообщение содержит информацию о версии PHP, установленной на веб-сервере. Код для проектов в этой книге писался под PHP5, поэтому для исполнения наших проектов на веб-сервере должна быть установлена эта или более поздняя версия PHP. Язык PHP, как уже отмечалось, позволяет создавать веб-страницы, которые выводят результаты поиска в базах данных, отправляют сообщения другим серверам, отправляют сообщения электронной почты и многое другое.

Большинство сценариев PHP не исполняются непосредственно из командной строки. Вместо этого веб-серверу (самым распространенным из которых является Apache) отправляется запрос на предоставление файла сценария PHP. Для этого следует ввести адрес требуемого документа в адресную строку веб-браузера и нажать клавишу <Enter>. Веб-сервер исполняет сценарий в запрошенном файле и отправляет запрашивающему браузеру результаты исполнения.

Вопрос исполнения сценариев PHP более подробно рассматривается в *главе 3*. А пока давайте создадим пару простых рабочих PHP-программ. Откройте текстовый редактор, скопируйте в него следующий код и сохраните файл под именем hello.php в папке public_html (или эквивалентной домашней папке вашего веб-сайта — например, www или web/public, в зависимости от веб-сервера вашего хостинга) на своем веб-сайте:

```
<?php
echo «<html><head></head><body>\n»;
echo «hello world!\n»;
echo «</body></html>\n»;
?>
```

Затем в командной строке выполните следующую команду:

```
php hello.php
```

Результат исполнения этой команды должен быть следующим:

```
<html><head></head><body>
hello world!
</body></html>
```

Теперь откройте этот файл в браузере. Для этого в строку адреса браузера введите адрес вашего веб-сайта, косую черту и имя файла: `www.example.com/hello.php`, где `www.example.com` — URL (адрес) вашего веб-сайта. Полученная от веб-сервера страница будет выглядеть примерно так (рис. 1.5).

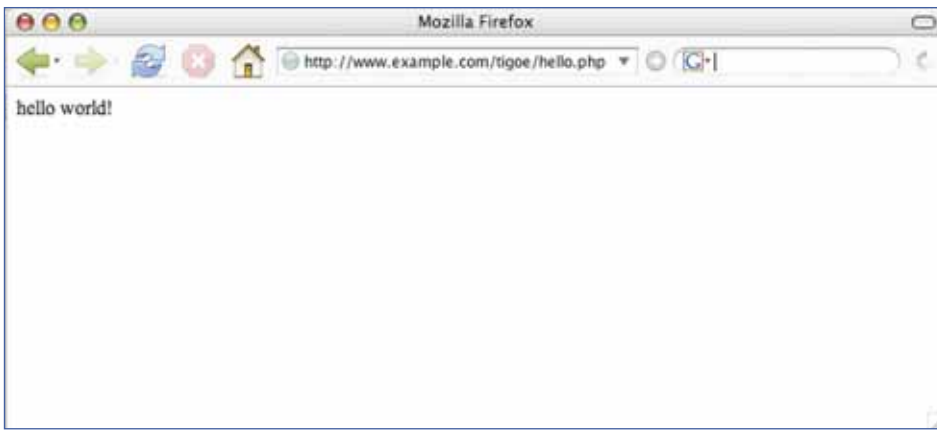


Рис. 1.5. Результат исполнения первого сценария PHP в браузере

Причины неудачи

Если вместо вывода, показанного на рис. 1.5, выводится исходный код PHP-сценария, возможно, что PHP-файл был открыт как локальный файл, а не как файл серверного сценария. В таком случае URL в адресной строке браузера будет начинаться с **file://**, вместо необходимого **http://**.

Другой причиной может стать то, что ваш веб-сервер не настроен для работы с PHP. Еще одной причиной может быть использование веб-сервером другого расширения для файлов сценариев PHP — например, *php4*. Обратитесь к поставщику вашего веб-хостинга за дополнительной информацией.

Как вы могли заметить, при просмотре файла PHP в браузере на экран выводится HTML-текст. Код PHP можно совмещать с кодом HTML в одном файле. Для этого блок кода PHP, вставляемый в код HTML, нужно выделить тегами `<?` и `?>`, которые обозначают начало и окончание блока кода PHP. Если при попытке открыть сценарий PHP в браузере выводится сообщение об ошибке, спросите у системного администратора веб-сервера, нет ли у них каких-либо дополнительных требований, чтобы сценарии PHP, например, находились на сервере в особых папках, или не требуются ли какие-нибудь специальные разрешения для файлов PHP.

Теперь рассмотрим более сложный PHP-сценарий. Сохраните файл с этим кодом в папке public_html вашего веб-сервера под именем time.php:

```
<?php
/*
        Date printer
        Контекст: PHP

   Выводит дату и время в странице HTML.
*/
// Получаем и форматируем дату:
$date = date("Y-m-d h:i:s\t");

// выводим начало страницы HTML:
echo "<html><head></head><body>\n";
echo "Здравствуй, мир!<br>\n";
// Добавляем дату:
echo "Сегодняшнее число:  $date<br>\n";
// завершаем HTML:
echo «</body></html>\n»;
?>
```

Чтобы просмотреть результаты выполнения этого сценария, введите в строку адреса браузера адрес вашего веб-сайта, косую черту и имя файла: `www.example.com/time.php`, где `www.example.com` — URL вашего веб-сайта. В результате исполнения этого сценария в браузере выводится следующий текст:

```
Здравствуй, мир!
Сегодняшнее число: 2014-04-13 11:32:08
```

Как можно видеть, эта программа использует переменную `$date` и вызывает встроенную функцию PHP `date()`, которая присваивает значение этой переменной. В PHP типы данных переменных объявлять не требуется. Имя переменной должно начинаться со знака `$`, а переменная может содержать целое число, число с плавающей точкой или строку. Так же, как и в Processing, в PHP используется C-подобный синтаксис, поэтому операторы условия `if-then`, циклы и комментарии будут выглядеть знакомыми.

Переменные в PHP

Переменные в PHP обрабатываются несколько по-иному, чем в Processing или Arduino. В последних двух средах переменным можно присваивать любые имена — при условии, что не используются слова, которые являются командами. Кроме того, переменная объявляется указанием ее типа перед именем при первом ее использовании. В PHP указывать тип переменной не требуется, но нужно, чтобы ее имя начиналось со знака `$`, как это можно было видеть в предыдущем сценарии PHP. Там переменной `$date` присваивается строковое значение посредством функции `date()`.

В PHP существует несколько функций для проверки состояния переменных. Например, функция `isset()` проверяет переменную на присвоение ей значения, а функции `is_bool()`, `is_int()` и `is_string()` проверяют на наличие в переменной значения определенного типа — булева, целого или строкового, соответственно.

PHP также содержит три важных встроенных переменных, называемых *переменными среды*, — их необходимо знать: `$_REQUEST`, `$_GET` и `$_POST`. Эти переменные предоставляют результаты исполнения запроса HTTP. Сценарии PHP могут вызываться как из форм HTML, так и пользователем, вводом в строку адреса (URL) сценария с последующей строкой переменных. Но в любом случае результаты исполнения сценария возвращаются в переменных среды. Переменная `$_GET` возвращает результат для сценариев, вызванных с помощью запроса HTTP `GET`, переменная `$_POST` возвращает результаты запросов HTTP `POST`, а переменная `$_REQUEST` — результаты запросов любого типа.

Так как запросы HTTP могут содержать несколько разных фрагментов информации (например, поля типичной веб-формы), все переменные среды имеют тип «массив». Чтобы получить определенный элемент такой переменной, его необходимо запросить по его имени. Например, если заполняемая веб-форма содержит поле с названием `Name`, заполняемое в него значение будет сохранено в элементе переменной `$_REQUEST`, называющемся `$_REQUEST['Name']`. Если форма была составлена посредством запроса HTTP `POST`, это значение также можно получить из элемента `$_POST['Name']`. Кроме рассмотренных переменных среды существуют и другие, разговор о которых пойдет позднее, но эти три являются наиболее полезными для получения информации от клиента, будь то веб-браузер или микроконтроллер. Далее в книге мы рассмотрим эти переменные более подробно, а также увидим их применение.

Дополнительную информацию по PHP, включая хорошие учебные пособия, можно получить на веб-сайте этого языка по адресу **www.php.net**. Для более глубокого изучения PHP можно порекомендовать книгу Давида Склара (David Sklar) «Learning PHP 5» (издательство O'Reilly).

Инструменты для работы через последовательный порт

Рассмотренные в предыдущем разделе *программы эмуляции терминала* позволяют получать доступ к удаленным компьютерам через Интернет, но этим их возможности не исчерпываются. Прежде чем получило широкое распространение взаимодействие между компьютерами по сети с помощью стека протоколов TCP/IP, для доступа к удаленным компьютерам использовались модемы, подключенные к последовательному порту компьютера. В те далекие времена на заре развития компьютерных коммуникаций многие пользователи через модемный доступ подключались к доскам объявлений и с помощью системы меню программ эмуляции терминала оставляли сообщения, загружали файлы и обменивались сообщениями с другими пользователями тех же досок объявлений.

В настоящее время последовательный порт компьютера в основном служит для подключения к компьютеру периферийных устройств. Через последовательный порт осуществляется и обмен данными между компьютером и микроконтроллером в области программирования микроконтроллеров. В проектах из этой книги для подключения микроконтроллера к последовательному порту компьютера используются программы терминала.

Существует большое число терминальных программ, как платных, так и бесплатных. Одной из таких программ является замечательная бесплатная программа CoolTerm, разработанная Роджером Майером (Roger Meier), загрузить ее можно с веб-сайта **http://freeware.the-meiers.org**. Программа работает как на компьютерах под Mac OS X, так и под Windows, в общем, это сейчас моя самая любимая программа. Если вы решите воспользоваться ею, поступите правильно и сделайте пожертвование, так как автор разрабатывал ее в свое свободное время.

Для пользователей Windows хорошей альтернативой может стать программа PuTTY, поскольку она способна открывать как последовательные терминалы, так и терминалы ssh. Программа PuTTY работает и на компьютерах под Linux. На компьютерах под Linux и Mac OS X можно пойти и другим путем — использовать классическую программу GNU screen, исполняющуюся в окне терминала. Но эта программа обладает меньшей функциональностью, чем программа CoolTerm.

Подключение к последовательному порту в Windows

Прежде всего, вам нужно знать имя последовательного порта, к которому подключен микроконтроллер. Для этого откройте диспетчер устройств, выполнив в консоли командной строки команду `devmgmt.msc`. Развернув раздел **Порты (COM и LPT)** диспетчера устройств, вы увидите все устройства, подключенные к компьютеру через последовательные и параллельные порты, названия последовательных портов там имеют вид COM1, COM2 и т. д.

Посмотрите, к какому последовательному порту подключен микроконтроллер Arduino, а затем запустите программу PuTTY. Выберите раздел **Session** в левой панели программы и в области **Connection type** правой панели установите переключатель **Serial**, а затем введите имя порта, к которому подключен микроконтроллер Arduino, в текстовое поле **Serial line**. Пример этих настроек показан на рис. 1.6, *а*.

Затем выберите раздел **Serial** (в самом низу левой панели) и удостоверьтесь, что значение поля **Serial line to connect to** в правой панели совпадает с именем порта подключения Arduino. Далее настройте последовательную линию на скорость 9600 бод (поле **Speed (baud)**), 8 битов данных (поле **Data bits**), 1 остановочный бит (поле **Stop bits**), без проверки четности (значение `None` в поле **Parity**) и без управления потоком данных (значение `None` в поле **Flow control**). Пример этих настроек показан на рис. 1.6, *б*.

Теперь нажмите кнопку **Open**, в результате чего откроется окно терминала последовательного подключения. Все, что вы введете в это окно, будет отсылаться на последовательный порт, а все данные, поступающие на последовательный порт, — выводиться в это окно.

Запустите программу для обмена данными

Если на подключенном к последовательному порту микроконтроллеру Arduino не исполняется программа для обмена данными с последовательным портом (как ее запустить, мы узнаем далее), никакие данные в окно выводиться не будут.

Кто получит порт?

Только одно приложение может одновременно работать с последовательном портом. Иными словами, если последовательный порт открыт одной программой, никакая другая программа не будет иметь к нему доступа, пока первая программа не завершит свою работу с этим портом. Когда программа пытается открыть последовательный порт, она запрашивает исключительный контроль над ним. При этом она или выполняет запись в специальный файл, называющийся *файлом блокировки*, или просит операционную систему выполнить эту запись от ее имени. Завершив работу с последовательным портом, программа снимает блокировку. Иногда при сбое программы, работающей с последовательным портом, порт остается заблокированным и недоступным для других программ. В таких случаях единственным способом разблокировать последовательный порт будет перезагрузка системы, что снимает все блокировки. Впрочем, можно и подождать, пока операционная система сама не осознает, что блокировку порта нужно снять. Во избежание этой проблемы всегда закрывайте последовательный порт при переключении с одного приложения, использующего его, на другое. В этом отношении пользователи Linux и Max OS X должны выработать привычку всегда закрывать окно сеанса работы с последовательным портом, нажав комбинацию клавиш <Ctrl>+<A>, а затем — <Ctrl>+<\>. Пользователям Windows следует прерывать подключение, закрыв окно программы PuTTY. Если не закрывать подключение должным образом, вам придется каждый раз перезагружать компьютер.

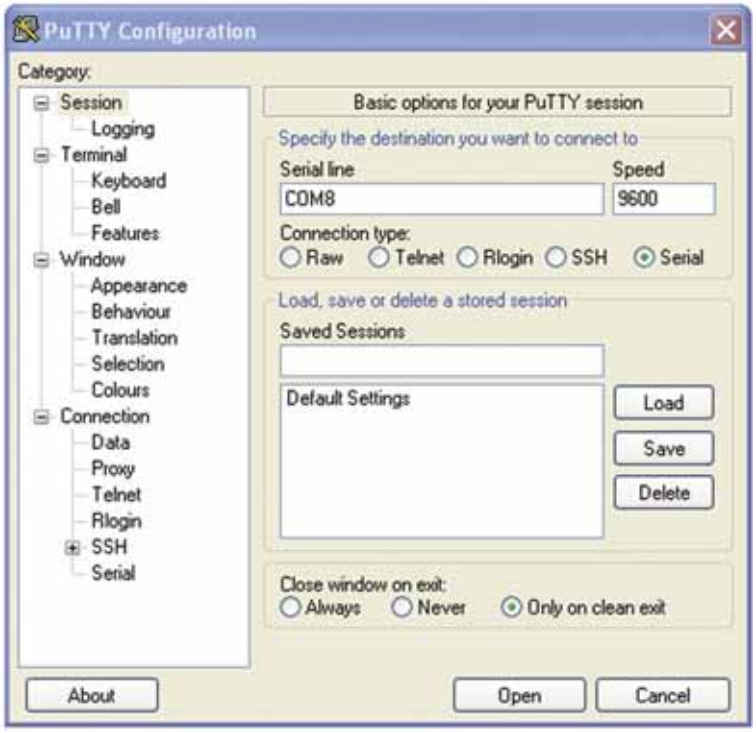


Рис. 1.6, *а*. Установка типа подключения и последовательного порта в PuTTY

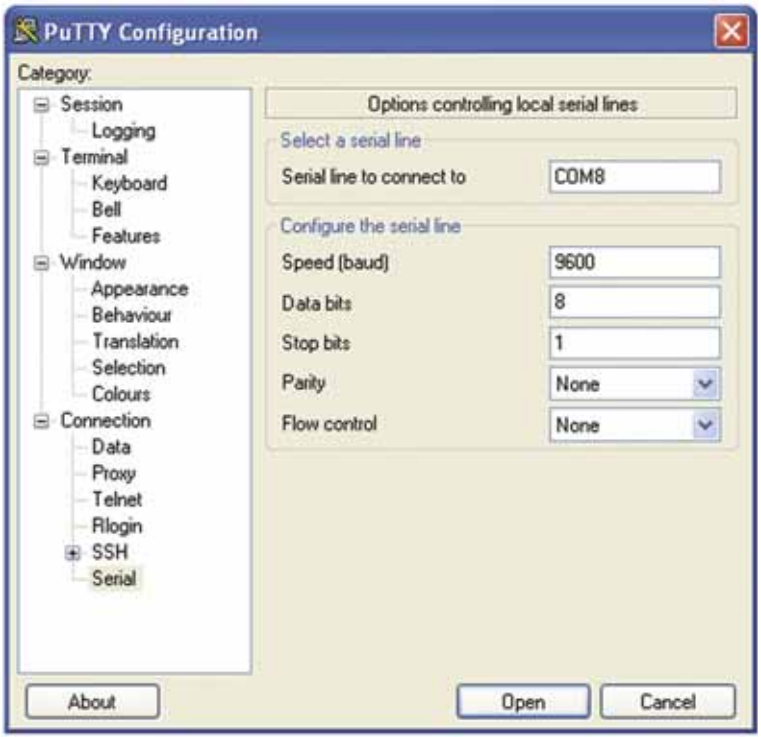


Рис. 1.6, *б*. Настройка параметров последовательного соединения

Подключение к последовательному порту в Mac OS X

Запустите программу CoolTerm (рис. 1.7) и щелкните мышью по значку **Options**.

В открывшемся диалоговом окне **Connection Options** выберите из раскрывающегося списка **Port** последовательный порт, к которому подключен микроконтроллер Arduino. Имена портов в Mac OS X представлены в формате вида:

```
/dev/tty.usbmodem241241
```

Чтобы узнать наверняка, к какому порту подсоединен микроконтроллер Arduino, проверьте список портов при отсоединенном устройстве, а затем — при подсоединенном. Порт, появившийся в списке после подсоединения микроконтроллера, и будет его портом. Чтобы открыть порт и подключить к нему микроконтроллер, нажмите кнопку **Connect** на панели инструментов главного окна программы. Чтобы отключить микроконтроллер и закрыть порт, нажмите кнопку **Disconnect**.

Любители острых ощущений из числа пользователей Mac OS X, с учетом того, что эта система основана на UNIX, могут вместо использования программы CoolTerm подключить микроконтроллер к последовательному порту, следуя приведенным далее инструкциям для Linux.

Подключение к последовательному порту в Linux

Эта процедура подключения может быть использована как в Mac OS X, так и в Linux. Откройте окно терминала и выполните следующую команду:

```
ls /dev/tty.* # — для Mac OS X
ls /dev/tty* # — для Linux
```

В результате исполнения этой команды в терминале будет выведен список последовательных портов, доступных в системе. Имена последовательных портов в Mac OS X и Linux имеют формат вида:

```
/dev/tty/usbmodem241221
```

Выберите необходимый последовательный порт и выполните команду:

```
screen номер_порта скорость_обмена
```

Например, чтобы открыть последовательный порт в Mac OS X для подключения к микроконтроллеру Arduino со скоростью 9600 битов в секунду, выполняется команда:

```
screen /dev/tty.usbmodem241241 9600
```

Соответствующая ей команда в Linux должна иметь вид:

```
screen /dev/ttyUSB0 9600
```

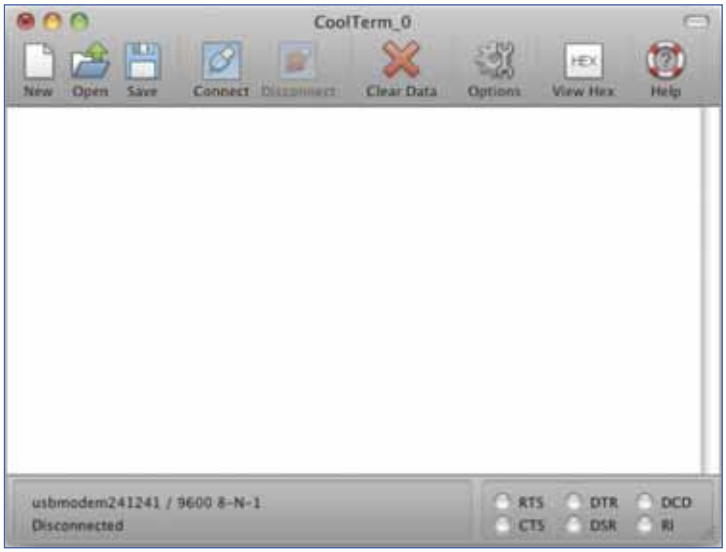


Рис. 1.7. Окно программы CoolTerm

В результате выполнения такой команды окно терминала очищается, устанавливается подключение микроконтроллера к последовательному порту и все, что вводится в терминал, отправляется на открытый последовательный порт. При этом вводимый текст на экране не отображается, но информация, получаемая на последовательный порт от микроконтроллера, выводится на экране в виде символов ASCII. Чтобы закрыть последовательный порт, нажмите комбинацию клавиш <Ctrl>+<A>, а затем — <Ctrl>+<\>.

В следующем разделе мы рассмотрим, как взаимодействовать с микроконтроллером через последовательный порт с помощью соответствующей программы связи.

Аппаратное обеспечение

Arduino, Wiring и подобные микроконтроллеры

Для проектов в этой книге в основном используется микроконтроллерный модуль Arduino. Этот модуль (а также другой микроконтроллерный модуль — Wiring) был разработан в Институте интерактивного дизайна итальянского города Иврея (Ivrea) в 2005 году. Оба модуля основаны на одном и том же семействе микроконтроллеров ATmega от Atmel (**www.atmel.com**), и для программирования обоих применяется особый «диалект» языка C/C++. Этот «диалект» основан на языке Processing, так же, как и используемая обоими модулями среда IDE[7]. В частности, в средах IDE для Arduino и Wiring используются некоторые элементы языка Processing — такие как методы `setup()` и `loop()`[8], функция `map()` и другие.

Когда готовился первый вариант этой книги, существовала одна версия платы Wiring, четыре или пять версий платы Arduino и почти никаких клонов. В настоящее время существует несколько моделей Arduino, две новые модели Wiring и множество клонов Arduino. В целом, клоны Arduino используются для решения обширного диапазона задач. Многие из них вполне совместимы с Arduino, и программировать их можно непосредственно из среды IDE Arduino. Ряд клонов располагают своими собственными средами IDE и будут работать только с некоторыми проектами из этой книги. Есть и такие клоны, которые имеют совместимую физическую конструкцию, но для их программирования применяются другие языки.

Рассматриваемые далее проекты прошли серьезное тестирование на платах Arduino и, при возможности, на классической плате Wiring. И хотя существуют некоторые различия между кодами для плат Arduino и Wiring, код для одной из этих плат, как правило, будет работать и на другой. Что же касается применения кода Arduino на клонах, то об этом следует узнавать у производителя конкретного клона. Многие из них очень активно участвуют в форумах Arduino и будут рады предоставить необходимую помощь.

Среды IDE для плат Arduino и Wiring очень похожи друг на друга. Эти бесплатные среды программирования с открытым исходным кодом можно загрузить с их веб-сайтов — соответственно: **www.arduino.cc** и **www.wiring.org.co**.

Аппаратное обеспечение для обоих модулей также является открытым, и его можно приобрести у различных онлайновых розничных поставщиков, указанных на этих сайтах. А те, кому нравится создавать устройства своими собственными руками, могут загрузить схемы печатных плат модулей и инструкции по их сборке. Однако, все же, лучше приобрести уже готовый модуль, так как это будет намного быстрее и для большинства людей намного надежней. На рис. 1.8 показаны некоторые из доступных плат Arduino (и одна плата Wiring).

Одно из самых лучших качеств плат Arduino и Wiring — это то, что они работают одинаковым образом с платформами Windows, Mac OS X и Linux. Такое в области микроконтроллерных сред разработки случается нечасто.

Другим положительным моментом этих устройств является расширяемость их сред. Точно так же, как

[7] IDE, Integrated development environment — интегрированная среда разработки.

[8] Метод `draw()` языка Processing изначально назывался `loop()`.

в программы на Processing можно включать классы и методы Java, в программы, созданные в средах IDE для Wiring и Arduino, можно вставлять код C/C++, написанный на разновидности языка C — AVR-C. Дополнительную информацию по этому вопросу можно найти на сайтах модулей.

А для знакомства с Arduino можно рекомендовать замечательную книгу «Getting Started with Arduino»[9] (издательство O'Reilly), написанную Массимо Банци (Massimo Banzi).

[9] «Введение в Arduino».

Шилды Arduino

Одной из особенностей плат Arduino, благодаря которой с ними так легко работать, являются модули расширения, называемые шилдами (shield), которые позволяют добавлять к основному модулю Arduino дополнительные функциональности. Шилды, созданные сторонней компанией или частным разработчиком, существуют для большинства задач. Вам нужен синтезатор MIDI? Есть такой шилд. Нужен вывод NTSC-видео? Такой шилд также имеется. Требуется Wi-Fi или Ethernet? И такие шилды есть (и широко используются в проектах этой книги). На рис. 1.9 показаны примеры нескольких готовых шилдов и макетных плат для разработки собственных.

Увеличивающаяся доступность шилдов стала одним из главных факторов в распространении плат Arduino. Шилды качественной разработки и с детальной документацией позволяют создавать различные проекты людям, не имеющим вообще никакого опыта работы с электроникой. Для одних проектов в этой книге мы возьмем уже готовые шилды, а для других — соберем свои собственные.

Наиболее часто в книге используются следующие шилды: шилд Ethernet, позволяющий подключать микроконтроллер к Интернету через сеть Ethernet; шилд для беспроводной связи, позволяющий подключать микроконтроллер к радиомодулям XBee компании Digi и другим радиомодулям с такими же характеристиками; макетные шилды для разработки специальных схем.

Информация о технических характеристиках шилдов доступна на веб-сайте **www.arduino.cc**. Если у вас есть опыт создания печатных плат, попробуйте сделать свой собственный шилд. Вы можете обнаружить, что это довольно занимательно.

До недавнего времени шилды для плат Arduino были физически несовместимы с платами Wiring. Но компания Rogue Robotics (**www.roguerobotics.com**) только что начала предлагать адаптер для плат Wiring, который позволяет подключать к ней шилды для плат Arduino.

Однако будьте осторожны! Не каждый шилд совместим с каждой платой. Рабочее напряжение некоторых плат клонов отличается от напряжения плат Arduino, поэтому они могут быть несовместимыми с шилдами, работающими на напряжении 5 вольт. Если вы используете микроконтроллер иной, нежели Arduino, проверьте, что его технические характеристики позволяют использовать его с требуемыми шилдами. Эту информацию можно получить у производителя платы.