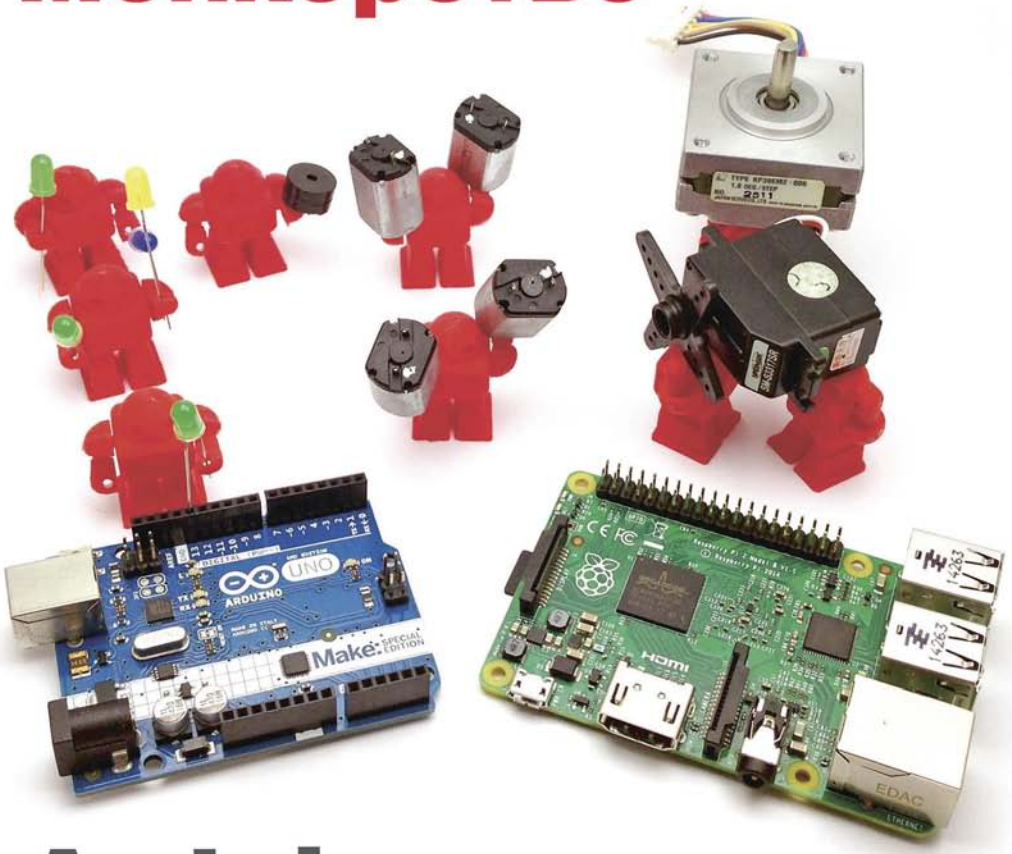
Мейкерство



Arduino и Raspberry Pi

Управление движением, светом и звуком

Саймон Монк

Make:
makezine.com

bhv®

Make: Action

Movement, Light, and Sound with Arduino and Raspberry Pi

Simon Monk

MAKER MEDIA™
SAN FRANCISCO, CA

Саймон Монк

Мейкерство

Arduino и Raspberry Pi

Управление движением, светом и звуком

Санкт-Петербург
«БХВ-Петербург»
2017

УДК 004
ББК 32.973.26
М77

Монк Саймон

М77 Мейкерство. Arduino и Raspberry Pi. Управление движением, светом и звуком: Пер. с англ. — СПб.: БХВ-Петербург, 2017. — 336 с.: ил.

ISBN 978-5-9775-3754-4

Рассказано, как самостоятельно создавать устройства на основе популярных платформ Arduino и Raspberry Pi. Излагаются принципы работы описываемых устройств. Сложные задачи решаются последовательно, через выполнение экспериментов и реализацию увлекательных проектов. Рассказано, как управлять светодиодными индикаторами, электродвигателями различных типов, соленоидами, агрегатами переменного тока, нагревателями, охладителями, дисплеями и звуковыми устройствами. Показано, как наблюдать за этими устройствами через Интернет и дистанционно управлять ими. Описаны проекты по созданию робота для расплющивания алюминиевых банок, сборке поливальной установки для комнатных растений, управляемого микроконтроллером светодиодного светофора, самодельного термостата, куклы, которая танцует и разговаривает, получив сообщение из твиттера, и многие другие.

Для читателей, интересующихся электроникой и робототехникой

УДК 004
ББК 32.973.26

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Перевод с английского	*Михаила Райтмана*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Оформление обложки	*Марины Дамбиевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-1-457-18779-7 (англ.)
ISBN 978-5-9775-3754-4 (рус.)

Оглавление

Об авторе

Саймон Монк — профессиональный писатель, его книги в основном посвящены электронике для любителей. Среди них наиболее известны Programming Arduino: Getting Started with Sketche (в русском переводе «Программируем Arduino. Основы работы со скетчами»), The Raspberry Pi Cookbook (в русском переводе «Raspberry Pi. Сборник рецептов») и Hacking Electronics (в русском переводе «Практическая электроника»). Он также помогает своей жене Линде (администратору сайта **Monk-makes.com**) собирать и продавать наборы и другую сопутствующую продукцию к этим книгам. Вы можете отслеживать Саймона в Твиттере, а также найти подробное описание его книг на сайте **simonmonk.org**.

О техническом редакторе

Дункану Амосу уже за 50, и большую часть своего трудового стажа он провел, занимаясь инженерным обеспечением телетрансляций. Кроме этого, Амос проектировал и собирал спутниковые подсистемы, писал книги технической тематики и пользовательские руководства, занимался искусственным осеменением скота, чинил садовую технику, моделировал и собирал мебель. Он занялся микроконтроллерами уже в возрасте, будучи опытным мастером свого дела. Так что он знает, чего стоит умение простыми словами объяснять сложную технику.

Введение 1

Микропроцессорные платы Arduino и Raspberry Pi значительно облегчили любителям самоделок путь в мир электроники. На их основе вы, например, сможете создать для своего дома автоматизированную систему, способную по сети Wi-Fi регулировать домашнее освещение и отопление либо просто управлять какими-либо моторами, приводящими в движение те или иные устройства вашего дома.

Из этой книги вы узнаете, как использовать популярные платформы Raspberry Pi и Arduino с тем, чтобы устройства на их основе могли управлять движением, освещением и звуком.

Arduino и Raspberry Pi

Хотя и Arduino, и Raspberry Pi — это маленькие платы, сравнимые по размеру с кредитной картой, между собой они существенно различаются. Arduino — очень простая микропроцессорная плата, не требующая для своей работы какой-либо операционной системы, тогда как Raspberry Pi — полноценный миниатюрный компьютер, работающий под управлением ОС Linux и оснащенный интерфейсами для подключения внешних электронных устройств.

Raspberry Pi

Если вы новичок в области электроники, но свободно обращаетесь с компьютером, Raspberry Pi покажется вам более привычным устройством. И в самом деле — Raspberry Pi (рис. 1.1) представляет собой уменьшенную версию материнской платы обычного компьютера с операционной системой Linux. На этой плате имеются USB-порты для подключения клавиатуры и мыши, аудиовыход и видеовыход HDMI для подключения к монитору или телевизору. Кроме того, плата Raspberry Pi оборудована Ethernet-портом для подключения к сети, она также совместима с USB-адаптерами Wi-Fi. Энергия на плату подается через разъем Micro-USB.

В качестве хранилища информации на Raspberry Pi используется карта памяти формата MicroSD, а не обычный для компьютеров дисковый накопитель. На этой

карте содержатся как операционная система, так и все пользовательские документы и программы.

> ***Это любопытно...***
>
> Плата Raspberry Pi была сконструирована в Великобритании, задумывалась, прежде всего, как дешевый компьютер для изучения основ информатики и, в частности, программирования на языке Python, ее целевая аудитория — школьники. Поэтому многие считают, что название «Pi» происходит от слога *Py* в слове *Python*.

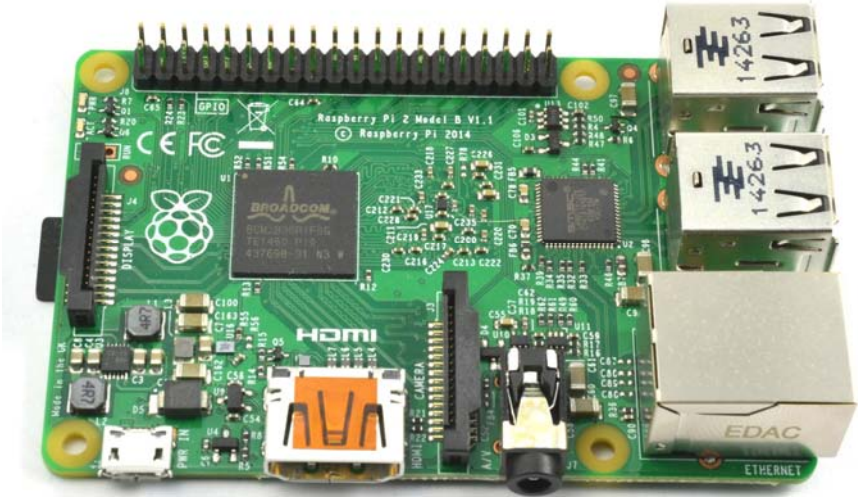


Рис. 1.1. Raspberry Pi 2

Тем не менее, плата Raspberry Pi кое-чем все же отличается от платы обычного ПК или ноутбука, работающего под Linux:

- стоит около 3700 рублей (упрощенный Raspberry Pi — так называемая «модель A+» — стоит дешевле, а модель zero — и еще дешевле);
- использует мощность всего 5 Вт;
- несет два ряда универсальных контактов для ввода/вывода (GPIO[1]), позволяющих подключать электронные устройства непосредственно к плате (эти контакты хорошо заметны на рис. 1.1, слева вверху). Через выводы GPIO можно управлять светодиодами, дисплеями, двигателями — в общем, самыми разными устройствами вывода, о которых пойдет речь в этой книге далее.

Кроме того, с Raspberry Pi можно выходить в Интернет по сети Wi-Fi или кабелю локальной сети, поэтому устройство подходит и для работы над проектами из области Интернета вещей (см. *главу 16*).

Спецификация для Raspberry Pi 2 (это новейшая и наилучшая версия на момент подготовки книги) такова:

[1] GPIO — интерфейс ввода/вывода общего назначения (от англ. General-Purpose Input/Output). — *Ред.*

- четырехъядерный процессор ARM v8 1,2 ГГц;
- модуль Wi-Fi 802.11n;
- модуль Bluetooth 4.1;
- контроллер Ethernet 10/100 BaseT;
- 4 порта USB 2.0;
- видеовыход HDMI;
- разъем для подключения камеры;
- колодка GPIO на 40 контактов (все контакты работают под напряжением 3,3 В).

Для тех, кто ранее не встречался с Raspberry Pi, в *главе 3* приведен «курс молодого бойца» по его настройке и запуску, а также экспресс-курс по языку Python.

Arduino

На рынке представлен весьма широкий спектр различных моделей Arduino. Но в этой книге мы будем иметь дело с наиболее распространенной и популярной моделью Arduino, которая называется Arduino Uno (рис. 1.2). Arduino несколько дешевле Raspberry Pi — плату Arduino Uno вы можете приобрести примерно за 2300 рублей.

Если вы привыкли работать с обычным компьютером, то параметры Arduino могут показаться вам мало на что пригодными. Так, Arduino оснащена всего лишь 34 Кбайт памяти различных типов. То есть, в Raspberry Pi примерно в 30 тыс. раз больше памяти, чем в Arduino, и это даже без учета той энергонезависимой памяти, что имеется на вставляемой в Pi карте MicroSD! Более того, частота процессора Arduino Uno составляет всего 16 МГц. К Arduino нельзя подключить клавиатуру, мышь или монитор, на Arduino нет и операционной системы.

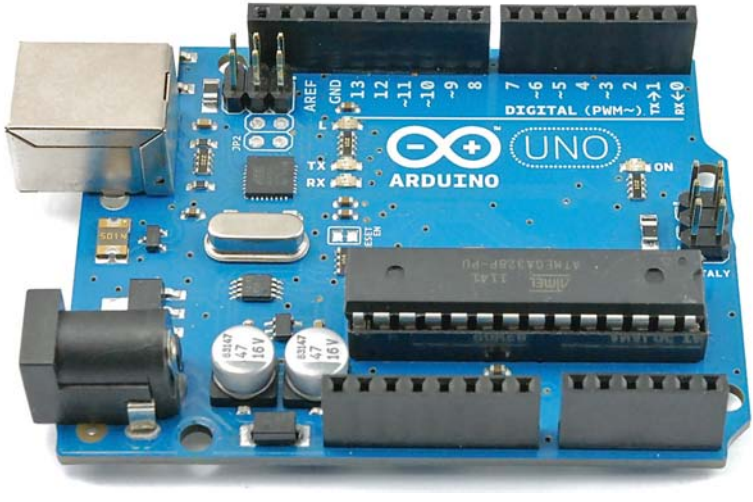


Рис. 1.2. Arduino Uno, версия 3

Может возникнуть вопрос, какой вообще прок в таком слабосильном устройстве? Секрет полезности Arduino — в ее крайней простоте. В нее не надо загружать операционную систему, и она не несет интерфейсов, которые могли бы оказаться ненужными в вашем проекте, — они просто потребляли бы энергию, а само устройство из-за них стоило бы дороже.

В то время как Raspberry Pi — полноценный компьютер, сила Arduino заключается в том, что она хорошо делает то, для чего предназначена, — для подключения электроники и управления ею.

Чтобы запрограммировать Arduino, понадобится обычный компьютер (если хотите, в этом качестве можно использовать даже Raspberry Pi). На таком компьютере понадобится запустить специальную интегрированную среду разработки (IDE), с помощью которой вы сможете написать свою программу и загрузить ее во встроенную флэш-память Arduino.

На Arduino в каждый момент времени можно запустить лишь одну программу. Будучи запрограммирована, Arduino запомнит эту программу и автоматически станет выполнять ее, как только вы подадите на нее питание.

Платы Arduino проектируются с расчетом на подключение *шилдов* — плат, вставляемых в гнезда ввода/вывода Arduino и обеспечивающих ей дополнительные аппаратные возможности. Например, через шилды Arduino можно оснащать различного вида дисплеями, а также адаптерами Ethernet и Wi-Fi.

Программа для Arduino пишется на языке программирования C (подробнее о программировании и использовании платы Arduino рассказано в *главе 2*).

Выбираем устройство: Arduino или Raspberry Pi?

В этой книге рассказывается, как подключать электронные устройства и к Arduino, и к Raspberry Pi, — хотя некоторые проекты лучше реализуются на Arduino, а некоторые — на Raspberry Pi. В то же самое время между этими двумя крайностями существует ряд устройств, способных взаимодействовать как с Arduino, так и с Raspberry Pi, и эта книга поможет вам работать и с ними.

Принимаясь за новый проект, я руководствуюсь железным правилом: по умолчанию использую Arduino. Однако если в проекте присутствует как минимум одно из следующих требований, то, пожалуй, будет лучше остановиться на Raspberry Pi:

- потребуется подключение к Интернету или к локальной сети;
- понадобится большой экран;
- потребуется подключать клавиатуру и мышь;
- понадобятся периферийные устройства, подключаемые через USB, — например, веб-камера.

Конечно, приложив некоторые усилия и сделав определенные затраты, можно оснастить Arduino дополнительными шилдами и удовлетворить большинство из упомянутых требований. Однако, пойдя по такому пути, вам будет сложнее заставить

все это работать, поскольку ни одна из этих возможностей, в отличие от Raspberry Pi, не поддерживается в Arduino «из коробки».

Веские доводы в пользу применения Arduino, а не Raspberry Pi, заключены в следующем:

- *стоимость* — Arduino Uno дешевле Raspberry Pi;
- *скорость запуска* — Arduino не дожидается, пока запустится операционная система. Присутствует лишь небольшая задержка (около секунды), в течение которой система проверяет, не загрузили ли в нее новую программу, а затем она запускается;
- *надежность* — Arduino по сути своей гораздо более простое и отказоустойчивое устройство, чем Raspberry Pi, в нем отсутствуют издержки, связанные с работой операционной системы;
- *энергопотребление* — Arduino потребляет примерно вдесятеро меньше энергии, чем Raspberry Pi. Если вам требуется решение, которое будет работать на батарейках или солнечных панелях, то лучше остановиться на Arduino;
- *ток на выходе GPIO* — GPIO-контакт Raspberry Pi следует использовать для подачи тока силой не более 16 мА. В то же время, контакт Arduino рассчитан на 40 мА. Так что, в некоторых случаях можно подключить какое-либо устройство (скажем, яркий светодиодный индикатор) непосредственно к Arduino, тогда как к Raspberry Pi его подключить было бы нельзя.

И Arduino, и Raspberry Pi — отличные устройства для любительских проектов, и в некоторой степени выбор того или иного устройства является делом вкуса.

Подключая внешние электронные компоненты к плате Raspberry Pi, важно иметь в виду, что Raspberry Pi работает на напряжении 3,3 В, — в отличие от платы Arduino, работающей на напряжении 5 В. Если вы подадите 5 В на GPIO-контакт Raspberry Pi, то, скорее всего, повредите или сожжете либо сам контакт, либо весь Raspberry Pi в целом.

Альтернативы

Arduino Uno и Raspberry Pi — своеобразные крайние позиции в спектре устройств, которые могут использоваться для управления электроникой. Неудивительно, что на рынке появилась уйма других устройств, занимающих то или иное промежуточное положение, а некоторые призваны объединить все достоинства Arduino и Raspberry Pi.

Новые устройства появляются постоянно. Благодаря «свободной» природе Arduino существует множество специфических нишевых вариантов этой платформы — например, для управления беспилотниками или беспроводными датчиками.

На рис. 1.3 показано распределение наиболее популярных устройств в этой области.

Как можно видеть, ниже Arduino Uno располагается плата Adafruit Trinket, уступающая ей как по цене, так и по производительности. На этой интересной плате

имеется несколько GPIO-контактов, в остальном же она вполне совместима с Arduino. Такая плата вполне подойдет для проекта, в котором нужно задействовать всего пару вводов/выводов.

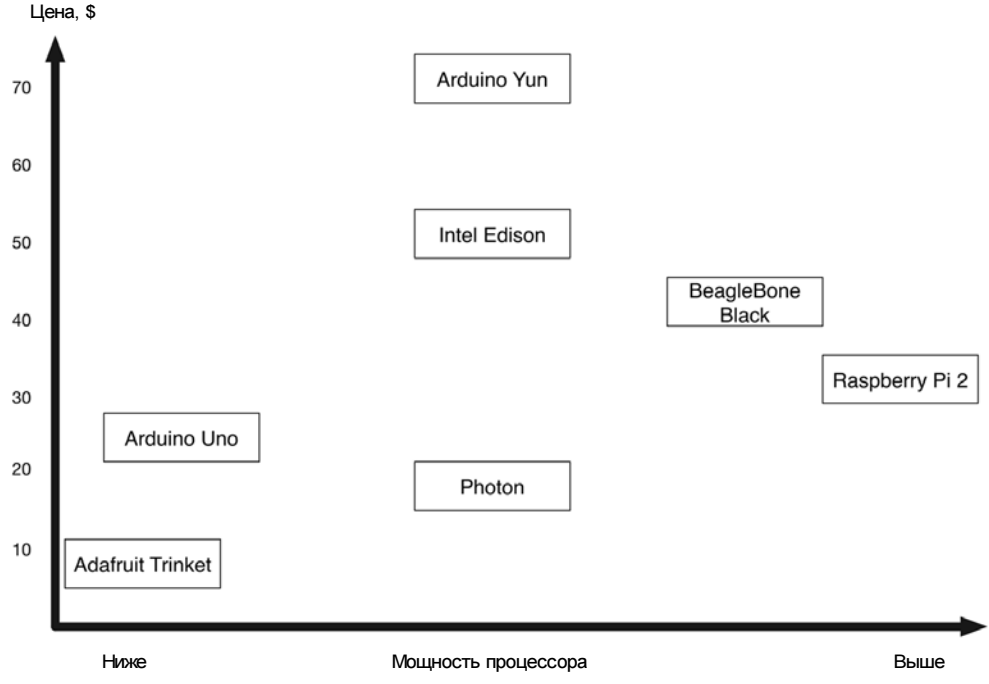


Рис. 1.3. Встраиваемые платформы

Существуют и промежуточные продукты, к которым относятся Arduino Yun, Edison и Photon. Все они обладают встроенными возможностями Wi-Fi и предназначены для реализации проектов, связанных с Интернетом вещей (см. *главу 16*). Пожалуй, наиболее полезна из них плата Photon. Все три указанных устройства программируются при помощи Arduino C, поэтому тот материал об использовании Arduino, который вы изучите, пригодится и при работе с этими платформами.

Плата BeagleBone Black концептуально очень близка Raspberry Pi. Она также представляет собой одноплатный компьютер, и хотя современная версия BeagleBone Black уступает Raspberry Pi по части чистой мощности, в некоторых отношениях BeagleBone у Raspberry Pi выигрывает. В частности, на BeagleBone больше GPIO-контактов, в том числе есть и такие, которые могут служить аналоговыми вводами, — Raspberry Pi 2 лишен такой возможности. При этом BeagleBone Black можно программировать либо на Python (примерно так же, как и Raspberry Pi), либо на JavaScript.

Заключение

В этой главе мы кратко познакомились с Arduino и Raspberry Pi. Обсудили достоинства и недостатки каждой из этих плат, рассмотрели некоторые альтернативы. В следующих двух главах вы узнаете, как начать работу и приступить к программированию сначала на Arduino, а затем на Raspberry Pi.

Если ранее вы уже работали с Arduino и Raspberry Pi, то можете перейти сразу к *главе 4* и приступить к практическому использованию этих устройств. При необходимости же вы всегда сможете вернуться к *главам 2* или *3* соответственно.

2 Arduino

Эта глава — доработанный вариант «курса молодого бойца» по Arduino, который был впервые опубликован в приложении к моей книге «The Maker's Guide to the Zombie Apocalypse» издательства NoStarch Press. Материал используется здесь с любезного разрешения издательства.

Если вы — новичок в области Arduino, то предлагаемая глава поможет вам освоиться с этим великолепным миниатюрным устройством.

Что есть Arduino?

Существуют различные типы плат Arduino, но, пожалуй, наиболее распространена та, которая используется во всех проектах этой книги, — Arduino Uno. Плата Arduino Uno несколько раз перерабатывалась, и на рис. 2.1 показана плата третьей версии (R3) — новейшая на момент подготовки книги.

Наше знакомство с Arduino мы начнем с *USB-разъема*, обеспечивающего несколько функций: он позволяет подавать на Arduino питание, программировать Arduino с компьютера и, наконец, поддерживать линию связи платы с внешними устройствами.

Маленькая красная кнопка рядом с USB-разъемом служит в качестве *кнопки сброса* — если ее нажать, Arduino перезапустится и выполнит программу, которая в нее записана.

На верхнем и нижнем краях платы Arduino расположены *соединительные разъемы*, к которым могут быть подключены различные электронные устройства и компоненты. Пронумерованные от 0 до 13 контакты, показанные в верхней части рис. 2.1, представляют собой *цифровые входы и выходы*. На выход или на вход станет работать такой контакт, определяется в программе, загружаемой в плату. К цифровому входу можно подключить переключатель, и этот вход будет в состоянии определить, нажата кнопка переключателя или нет. Точно так же можно подсоединить светодиод к цифровому выходу, и, переведя выход от низкого уровня к высокому, включить этот светодиод. Кстати, один такой светодиод встроен прямо в плату (он обозначен на ней буквой **L**) и подключен к цифровому контакту 13.

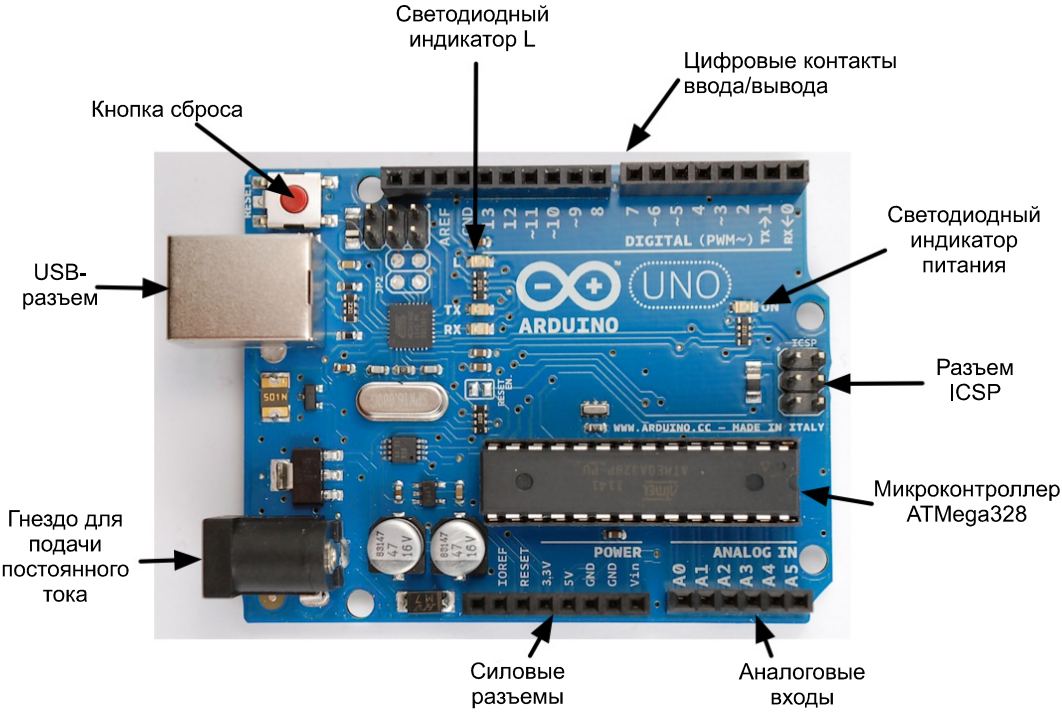


Рис. 2.1. Arduino Uno R3

Под цифровыми контактами входа/выхода расположен *светодиодный индикатор питания*, просто указывающий, что на плату подано напряжение. *ICSP-разъем*, предназначенный для внутрисхемного программирования по последовательному протоколу, позволяет осуществлять «продвинутое» программирование Arduino без подключения по USB. Большинству пользователей Arduino никогда не приходилось работать с ICSP-разъемом.

Следует отдельно упомянуть «мозг» Arduino — микроконтроллерную интегральную схему *ATMega328*. Именно в ее флэш-памяти (напомню, что ее там 32 Кбайт) хранится программа, которую будет выполнять Arduino.

Ниже ATMega328 расположен ряд *аналоговых входных контактов*, помеченных номерами от A0 до A5. И если цифровые входы могут лишь сообщить о том, включен компонент или выключен, аналоговые входы позволяют измерять поданное на контакт напряжение (в пределах от 0 до 5 В), — например, от какого-либо датчика. Впрочем, если вам станет не хватать цифровых входов и выходов, то эти аналоговые входы и выходы также можно будет сконфигурировать как цифровые.

Левее аналоговых входов выстроились *силовые разъемы*, которые могут быть использованы в качестве альтернативных входов для запитывания Arduino. Через них также можно будет подавать питание на электронные устройства и компоненты собираемых вами схем.

Имеется на плате Arduino также и *гнездо для подачи постоянного тока*, через которое на плату может быть подан постоянный ток напряжением от 7 до 12 В. Регуля-

тор напряжения, встроенный в плату, позволяет преобразовать его в напряжение 5 В, под которым работает Arduino. Плата Arduino способна автоматически принимать питание либо от USB-разъема, либо от гнезда с постоянным током, — в зависимости от того, куда сделано подключение.

Установка интегрированной среды разработки Arduino IDE

Arduino не слишком походит на обычный компьютер. Плата Arduino не управляется какой-либо операционной системой, к ней нельзя подключить монитор, клавиатуру или мышь. На Arduino всегда выполняется единственная программа (скетч), и эту программу требуется загрузить во флэш-память микроконтроллера при помощи компьютера. Перепрограммировать Arduino можно столько раз, сколько вам угодно (в принципе, много тысяч раз).

Чтобы получить возможность программировать Arduino, нужно установить на свой компьютер *интегрированную среду разработки* (Arduino IDE). Эта среда представляет собой кроссплатформенное приложение — Arduino IDE может работать в операционных системах Windows, Mac OS и Linux — и в этом одна из причин огромной популярности Arduino. Кроме того, Arduino IDE позволяет запрограммировать Arduino по USB без какого-либо специального оборудования для программирования.

Чтобы установить Arduino IDE на компьютер, скачайте, следуя инструкциям с сайта Arduino (**arduino.cc/en/Guide/HomePage**), программу для вашей платформы и запустите установщик.

> ***Внимание!***
>
> Пользователям Windows и Mac OS потребуется установить специальные USB-драйверы, чтобы среда Arduino IDE могла взаимодействовать с платой Arduino.

Когда все будет установлено, запустите Arduino IDE (рис. 2.2). Кнопка **Загрузка** (Upload), как понятно из ее названия, *загружает* скетч на плату Arduino. Перед загрузкой скетча она преобразует его текстовый программный код в исполняемый код для Arduino. Если в коде обнаружатся ошибки, информация о них будет выведена в *области журнала*. Кнопка **Проверить** (Verify) работает аналогичным образом, но не выполняет последнего шага, — т. е. не загружает программу на плату.

Кнопка **Монитор порта** (Serial Monitor) открывает окно *монитора последовательного интерфейса*, служащее для связи с Arduino. Вы постоянно будете пользоваться монитором последовательного интерфейса, выполняя эксперименты, описанные в этой книге, т. к. он очень удобен для отправки команд на Arduino прямо с компьютера. Монитор последовательного интерфейса обеспечивает двунаправленную связь — т. е., вы можете посылать на Arduino текстовые сообщения и получать от платы ответы.

В *строке состояния*, расположенной в нижней части окна Arduino IDE, указываются тип Arduino и последовательный порт, через который плата будет программиро-

ваться после нажатия кнопки **Загрузка** (Upload). Если ваш компьютер работает под управлением операционной системы Linux или Mac OS, то название порта будет иметь вид /dev/cu.usbmodem411. Если же вы программируете Arduino через компьютер с ОС Windows, то здесь будет прописан порт COM с тем номером, который Windows выделит Arduino после подключения платы к компьютеру, — например, **COM1** (см. рис. 2.2).

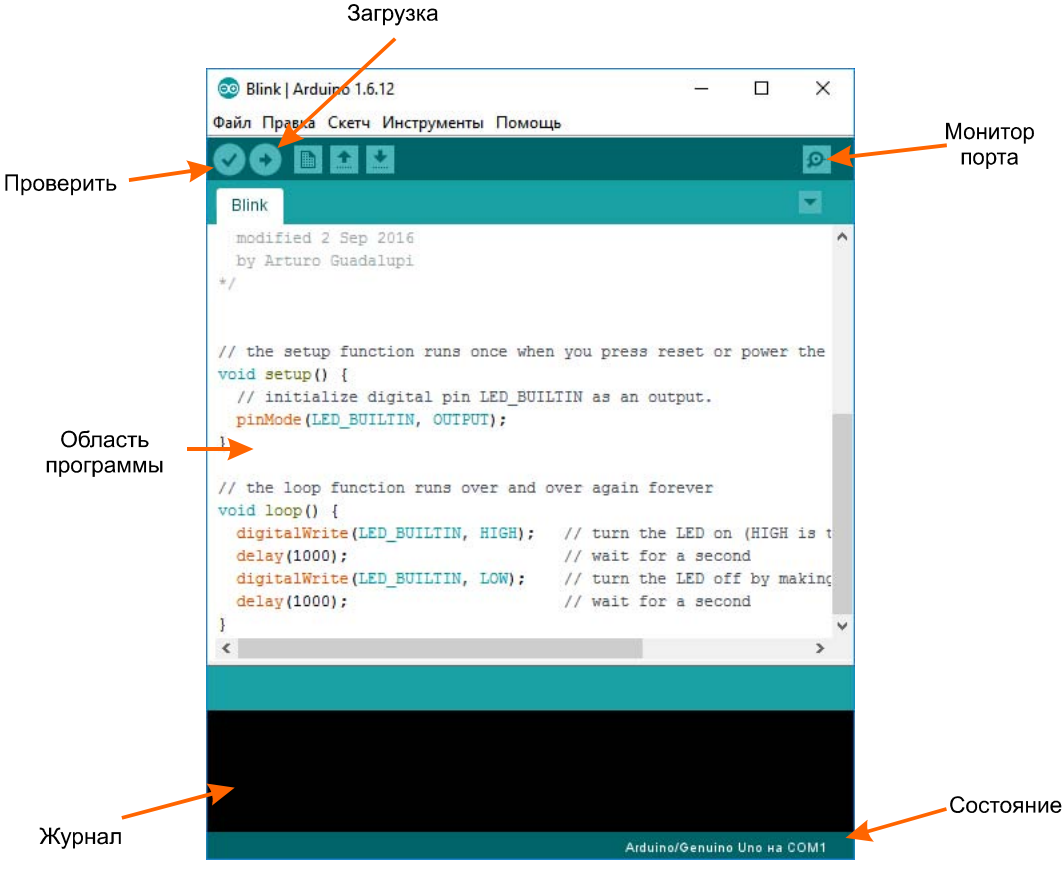


Рис. 2.2. Arduino IDE

Последнее, но важное замечание: основную часть окна Arduino IDE занимает *область программы*, куда вы вводите программный код, который хотите загрузить на Arduino.

В мире Arduino программы, как уже было отмечено ранее, именуются *скетчами*, и через меню **Файл** (File) Arduino IDE можно открывать (**Открыть** (Open)) или сохранять (**Сохранить** (Save)) скетчи точно так же, как открываются и сохраняются документы в любом текстовом редакторе. Меню **File** (Файл) также включает подменю **Examples** (Примеры), с помощью которого можно загрузить в плату встроенные в Arduino IDE скетчи, содержащие полезные примеры программ.

Загрузка скетча

Чтобы протестировать плату Arduino и убедиться, что среда Arduino IDE установлена верно, откройте скетч Blink, предлагаемый в качестве примера. Вы найдете его, выполнив команду меню **Файл | Примеры | 01. Basics** (File | Examples | 01. Basics) — именно скетч Blink и показан на рис. 2.2.

При помощи кабеля USB подключите плату Arduino к тому компьютеру, с которого собираетесь ее программировать, — на Arduino должен загореться светодиод питания и замигать несколько других светодиодов.

Теперь, когда плата Arduino подключена, нужно сообщить среде Arduino IDE как тип программируемой платы (Arduino Uno), так и последовательный порт, к которому мы ее подключаем:

- чтобы задать *тип платы*, выберите команду меню **Инструменты | Плата | Arduino Uno** (Tools | Board | Arduino Uno);
- чтобы задать последовательный порт, выберите команду меню **Инструменты | Порт** (Tools | Serial Port). Если вы работаете на компьютере под управлением ОС Windows, то, вероятно, вариантов вам будет предложено не много. Скорее всего, вы сможете выбрать пункт **COM4**. На компьютерах с Mac OS и Linux обычно перечисляется множество USB-устройств, и порой сложно определить, какое из них — ваша плата Arduino. Как правило, ей будет соответствовать запись **dev/tty.usbmodem*NNN***, где ***NNN*** — некий номер. На рис. 2.3 выбрана плата Arduino, подключенная к моему Макинтошу.

Если Arduino не отображается в списке, это обычно свидетельствует о проблеме с USB-драйверами. В таком случае попробуйте их переустановить.

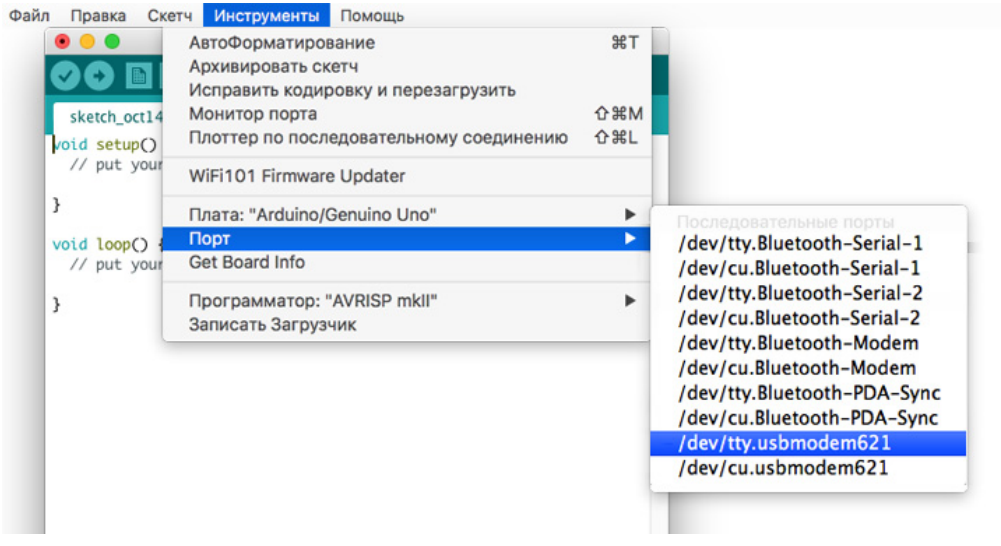


Рис. 2.3. Выбор последовательного порта Arduino

Когда вы будете готовы загрузить скетч на Arduino, нажмите кнопку **Загрузка** (Upload) — в области журнала должны отобразиться сообщения, а затем через несколько секунд светодиоды **TX** и **RX** на плате должны замигать. Это означает, что программа загружается на плату.

Если все пройдет штатно, то по завершении загрузки вы должны увидеть примерно такое сообщение (рис. 2.4).

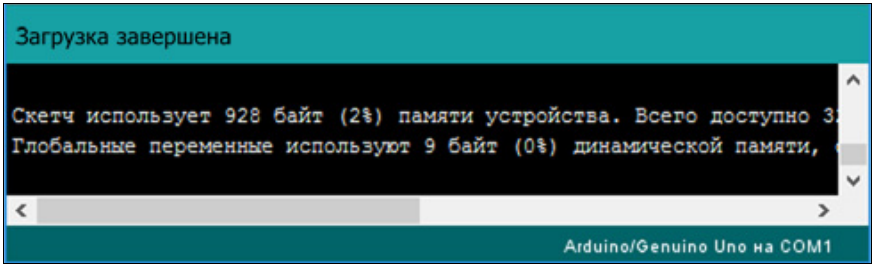


Рис. 2.4. Загрузка произведена успешно

Это сообщение свидетельствует, что скетч загружен, и что он занял 928 байтов из 32 256 байтов, имеющихся на плате.

По завершении загрузки скетча вы заметите, что встроенный светодиод **L** на плате Arduino начнет медленно мигать. Так ваш скетч Blink (Мерцание) оправдывает свое имя.

Код к книге

Весь код к этой книге — и скетчи Arduino, и программы для Raspberry Pi, написанные на языке Python, — выложен на специально созданную для нее страницу GitHub: **https://github.com/simonmonk/make_action**.

Чтобы скачать эти файлы на ваш компьютер — неважно, под управлением какой операционной системы он работает: Mac OS, Linux или Windows, — нажмите на этой странице кнопку **Clone or Download** (Клонировать или скачать), расположенную над списком файлов справа, а затем появившуюся кнопку **Download ZIP** (Скачать ZIP-архив).

В результате скачается ZIP-файл, который вы сможете сохранить у себя на компьютере или в каком-либо другом удобном вам месте. Распаковав ZIP-архив, вы получите каталог под названием make_action-master. Код для Arduino вы найдете в каталоге arduino, внутри которого имеются два подкаталога: experiments и projects.

Каждая программа для эксперимента (подкаталог experiments) или проекта (подкаталог projects) содержится в собственном каталоге — обычно там находится один файл с конкретной программой. Например, в подкаталоге experiments вы найдете каталог ex_01_basic_motor_control с единственным файлом basic_motor_control.ino. Если у вас уже установлена среда Arduino IDE, то этот файл по щелчку на нем откроется именно в Arduino IDE.

Другой способ доступа ко всем этим скетчам — скопировать подкаталоги experiments и projects в ваш каталог с хранилищем скетчей Arduino, который называется Arduino и представляет собой обычную папку с документами (такую, например, как Мои документы в Windows или Documents в Mac OS). Если файлы были скопированы в каталог со скетчами, то их можно будет открыть, выполнив в Arduino IDE команду меню **Файл | Папка со скетчами** (File | Sketchbook).

> ***ЭЛЕКТРОННЫЙ АРХИВ***
>
> Для удобства читателей русского перевода этой книги весь упомянутый здесь автором код уже скачан и выложен на FTP-сервер издательства «БХВ-Петербург» вместе с комплектом цветных иллюстраций книги. Скачать электронный архив с этими материалами можно по ссылке: **ftp://ftp.bhv.ru/9785977537544.zip**, а также со страницы книги на сайте **www.bhv.ru**.

Руководство по программированию

В этом разделе предоставлен обзор основных команд, которые помогут вам понять скетчи из книги. Если ранее вам не приходилось программировать, и вы хотите изучить язык C для Arduino, прочтите мою книгу «Программируем Arduino. Основы работы со скетчами», ссылку на которую вы без труда найдете в Интернете.

Функции *setup* и *loop*

Функции — это блоки программного кода, делающие что-либо. В каждом скетче Arduino должны быть своя функция `setup()` и своя функция `loop()`. Чтобы рассмотреть функции `setup()` и `loop()` в действии, давайте исследуем скетч Blink, который мы уже загрузили в Arduino:

```
int led = 13;
// процедура запуска выполняется один раз
// после того, как вы нажмете кнопку сброса:
void setup() {
// инициализируем цифровой контакт как вывод.
pinMode(led, OUTPUT);
}
// циклическая процедура повторяется снова и снова,
// и так до бесконечности:
void loop() {
   digitalWrite(led, HIGH);  // включить светодиод
                             // (HIGH — это уровень напряжения)
   delay(1000);              // подождать 1 секунду
   digitalWrite(led, LOW);   // выключить светодиод,
                             // переведя напряжение на уровень LOW
   delay(1000);              // подождать 1 секунду
}
```

В скетче вы видите немало текстовых строк, перед которыми стоят символы //. Они означают, что весь текст после // и до конца строки считается *комментарием*. Это не программный код, а просто пояснения, помогающие человеку, читающему код, понять, что происходит.

Как понятно из комментариев, строки кода в функции `setup()` выполняются всего один раз — точнее, всякий раз, как только на Arduino подается питание или после нажатия кнопки сброса. То есть, функция `setup()` используется для выполнения всех однократных операций, которые нужно выполнить при запуске программы. В примере с Blink нам всего лишь требуется указать, что контакт светодиода сконфигурирован как вывод.

Команды внутри функции `loop()` выполняются снова и снова — т. е., как только в функции `loop()` закончится выполнение последней строки, программа возвращается к выполнению ее первой строки.

Здесь я не буду останавливаться на том, что именно делают в скетче Blink команды, находящиеся в функциях `setup()` и `loop()`, но не волнуйтесь — вскоре мы об этом поговорим.

Переменные

При помощи *переменных* можно присваивать значениям имена. Первая строка скетча Blink (не считая комментариев) такова:

```
int led = 13;
```

В ней определяется переменная `led`, получающая исходное значение `13`. Значение `13` выбрано по имени того контакта Arduino, к которому подключен светодиод **L**, а `int` — это тип переменной. Слово `int` является сокращением от *integer*, что в переводе с английского означает «целое число» (без десятичных знаков).

Хотя и не обязательно давать отдельное имя переменной для каждого контакта, с которым вы работаете, лучше все-таки это делать, поскольку так становится проще понять, какой контакт для чего используется. Кроме того, если вы захотите перейти на другой контакт, то вам потребуется изменить имя переменной всего в одном месте — там, где вы ее определили.

Возможно, вы заметили, что в скетчах к этой книге при подобном определении переменных (когда выбирается контакт, с которым мы будем работать) перед строкой стоит слово `const` — вот так:

```
const int led = 13;
```

Ключевое слово `const` сообщает Arduino IDE, что на самом деле это никакая не переменная, а *константа*, — иными словами, ее значение равно 13, и никогда не меняется. В результате скетчи становятся немного компактнее и работают быстрее — рекомендуется, чтобы использование этого слова вошло у вас в привычку.

Цифровые выводы

Скетч Blink — хороший пример цифрового вывода. Ранее мы присвоили переменной `led` значение `13`, а в следующей строке контакт `13` в функции `setup()` был сконфигурирован на вывод:

```
pinMode(led, OUTPUT);
```

Эта операция делается именно в функции `setup()`, поскольку ее нужно выполнить всего один раз. Как только контакт сконфигурирован на вывод, он так и будет работать на вывод, пока вы не отдадите другую команду.

Чтобы светодиод мигал, его нужно постоянно включать и выключать, и соответствующий код приводится здесь:

```
digitalWrite(led, HIGH);  // включить светодиод
                          // (HIGH — это уровень напряжения)
delay(1000);              // подождать 1 секунду
digitalWrite(led, LOW);   // выключить светодиод,
                          // переведя напряжение на уровень LOW
delay(1000);              // подождать 1 секунду
```

У функции `digitalWrite()`, как можно видеть, имеются два параметра — они приводятся в скобках и разделены запятой. Первый параметр — это контакт Arduino, на который пойдет запись, а второй — значение, которое будет туда записано. Итак, значение `HIGH` дает нам 5 В на выходе (и светодиод включается), а значение `LOW` соответствует 0 В (и светодиод выключается).

Функция `delay()` приостанавливает выполнение программы на некоторое время (в миллисекундах) — это время задается в качестве параметра. В одной секунде 1000 миллисекунд, поэтому каждая функция `delay(1000)` приостановит выполнение программы на 1 секунду.

В *разд. «Эксперимент: управление светодиодом» главы 4* мы будем работать с цифровым выводом, подключенным к внешнему светодиоду, и у нас станет мигать именно внешний индикатор, а не встроенный.

Цифровые входы

Поскольку в этой книге рассказывается преимущественно о выводах, а не о входах, здесь чаще используется функция `digitalWrite()`. Однако вы должны знать и о *цифровых входах*, через которые можно подсоединять к Arduino переключатели и датчики.

В качестве цифрового входа контакт Arduino можно определить при помощи функции `pinMode()`:

```
pinMode(7, INPUT)
```

Здесь мы конфигурируем на вход контакт `7`. Разумеется, вместо номера контакта `7` можно использовать имя переменной.

Так же, как и в случае с выводным контактом, входной контакт задается при помощи функции `setup()`, поскольку на этапе выполнения скетча менять режим работы контакта приходится редко.

С контакта, сконфигурированного на вход, потом можно считывать информацию, определяя, ближе к какому значению находится напряжение на контакте: 5 В (`HIGH`) или 0 В (`LOW`). В следующем примере светодиод будет включен, если на момент проверки вход будет иметь значение `LOW` (после этого светодиод так и останется гореть, поскольку в коде нет команды, которая бы его выключала):

```
void loop()
{
   if (digitalRead(7) == HIGH)
   {
      digitalWrite(led, LOW)
   }
}
```

Что ж, наш код стал немного усложняться, поэтому разберем его построчно.

Во второй строке у нас присутствует символ `{`. Иногда он ставится на той же строке, что и `loop()`, а иногда переносится на следующую. Это дело вкуса — оба варианта кода выполняются одинаково. Символ `{` отмечает начало блока кода, а заканчивается этот блок парным символом `}`. Таким образом все строки кода, относящиеся здесь к функции `loop`, группируются вместе.

В первой из этих строк стоит оператор `if`. Сразу после слова `if` идет условие. В нашем случае условие таково: (`digitalRead(7) == HIGH`). Двойной знак равенства (`==`) означает сравнение значений, расположенных слева и справа от него. Итак, если в нашем случае контакт `7` имеет значение `HIGH`, то после оператора `if` выполнится блок кода, записанный между `{ и }`, — в противном случае этого не произойдет. Если выровнять по вертикали открывающие и закрывающие блоки кода символы `{ и }`, становится понятнее, какая `}` замыкает какую `{`.

Мы уже видели, как выполняется код, если условие соблюдается, — тогда срабатывает функция `digitalWrite()`, включающая светодиод.

В этом примере мы предположили, что цифровой вход может иметь строго одно из двух значений: `HIGH` или `LOW`. Если вы используете переключатель, подсоединенный к цифровому входу, то этот переключатель позволяет только лишь закрыть (замкнуть) соединение. Как правило, в таком случае имеется в виду подсоединение цифрового входа к заземлению (0 В). Если соединение на переключателе открыто, то принято говорить, что цифровой вход является неподключенным (незамкнутым). Иными словами, он не имеет никакого электрического соединения. Такой вход будет принимать электрические помехи и зачастую может самопроизвольно переключаться между высоким и низким состояниями. Во избежание такого нежелательного процесса обычно используют так называемый *подтягивающий* резистор (рис. 2.5).