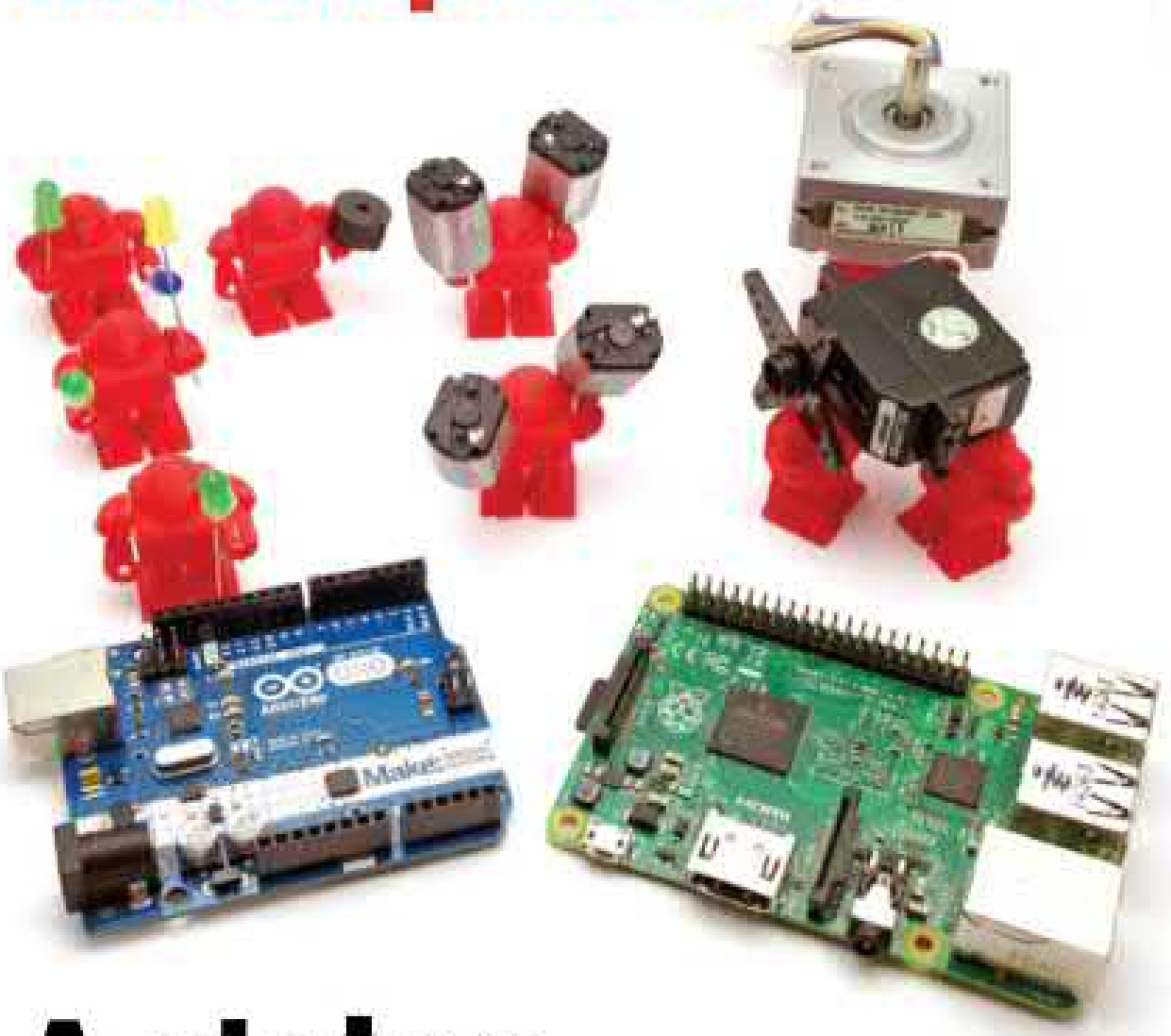
Мейкерство



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*

Подписано в печать 28 04 17
Формат 70×100$^{1}/_{16}$. Печать офсетная Усл. печ л 27,09
Тираж 1500 экз Заказ №4329
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ООО "Печатное дело",
142300, МО, г Чехов, ул. Полиграфистов, д 1

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 Raspbery 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*.

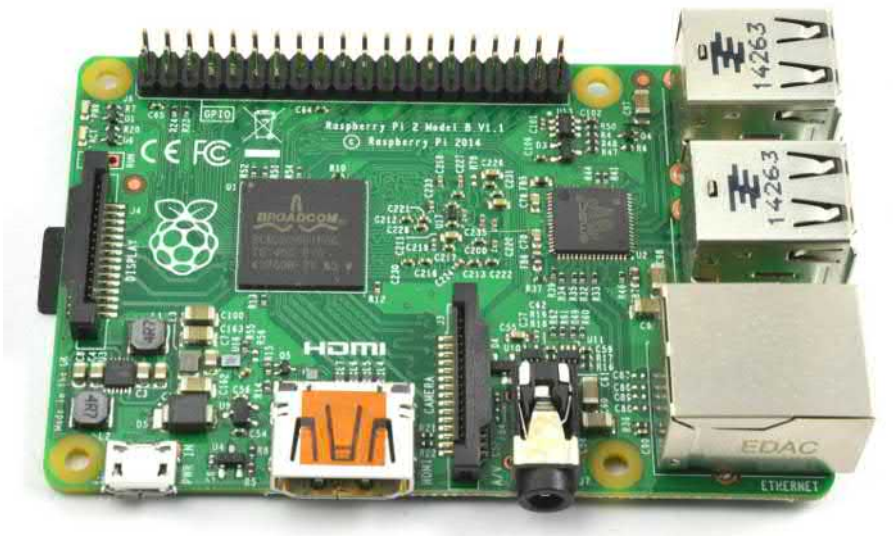


Рис. 1.1. Raspberry Pi 2

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.

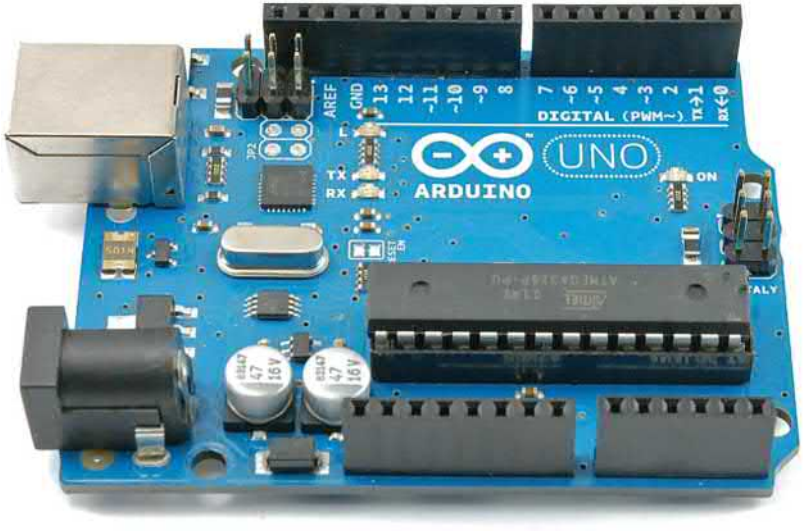


Рис. 1.2. Arduino Uno, версия 3

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.

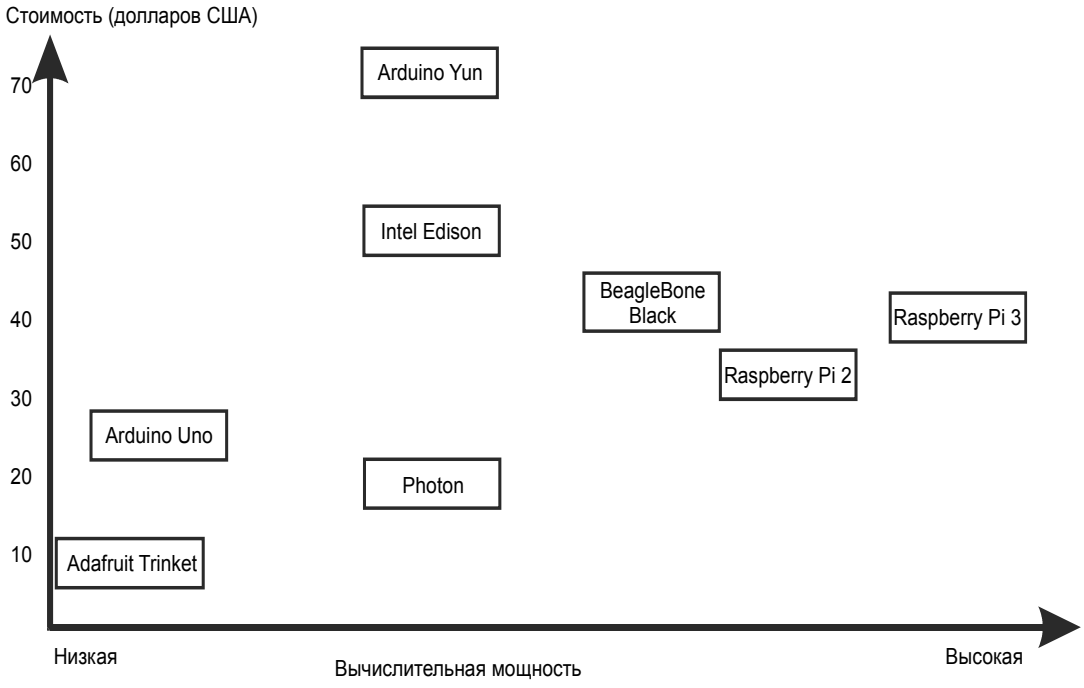


Рис. 1.3. Встраиваемые платформы

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).

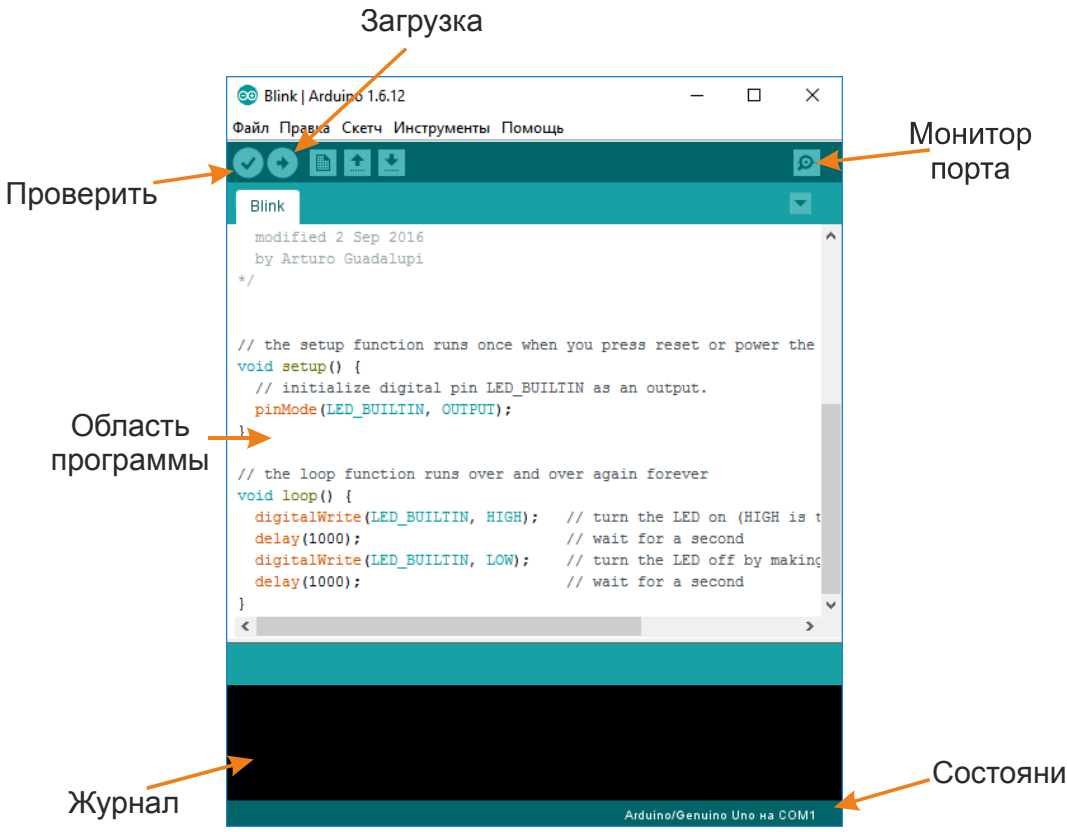


Рис. 2.2. Arduino IDE

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.

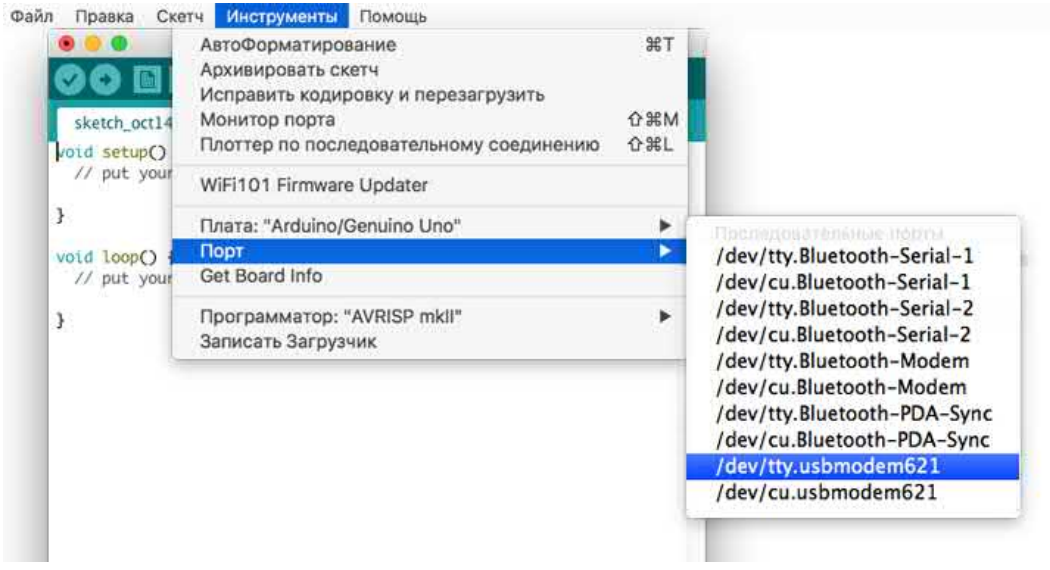


Рис. 2.3. Выбор последовательного порта Arduino

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).

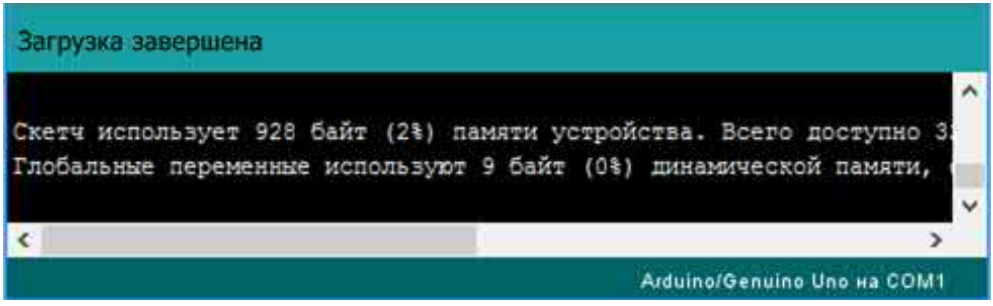


Рис. 2.4. Загрузка произведена успешно

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).

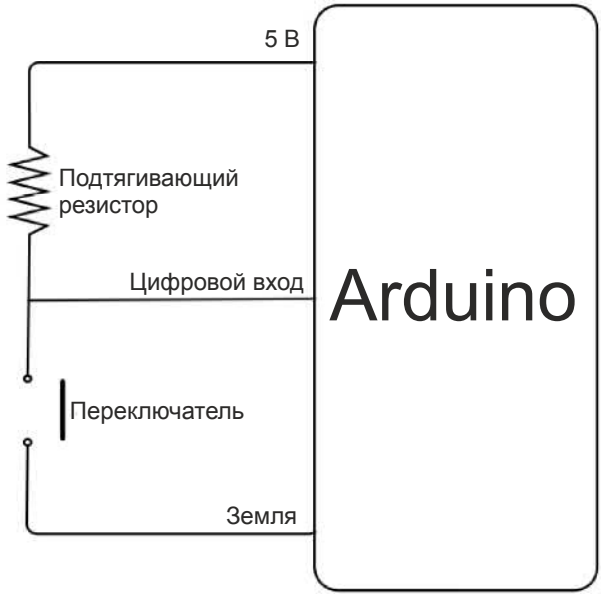


Рис. 2.5. Использование подтягивающего резистора при работе с цифровым входом

Когда переключатель не замкнут (как показано на рис. 2.5), резистор «подтягивает» входной контакт до напряжения 5 В. Когда мы, нажав на кнопку, переключатель замыкаем, то слабое подтягивание входа перекрывается переключателем, подсоединяющим цифровой вход к заземлению.

Вы, конечно, можете пользоваться отдельными подтягивающими резисторами, однако на входах Arduino имеются встроенные подтягивающие резисторы со значением около 40 кОм, которые включаются, если установить для контакта режим работы `INPUT_PULLUP`, а не просто `INPUT`. Следующий код демонстрирует, как задать для контакта цифрового входа режим для работы с переключателем без необходимости использования внешнего подтягивающего резистора, показанного на рис. 2.5:

```
pinMode(switchPin, INPUT_PULLUP);
```

Аналоговые входы

Аналоговые входы позволяют измерять напряжение в диапазоне от 0 до 5 В на любом из специальных аналоговых контактов Arduino, пронумерованных от A0 до A5. В отличие от работы с цифровыми выводами и входами, задействуя аналоговые входы, не нужно использовать функцию `pinMode()` на этапе `setup`.

Для считывания значения аналогового входа служит функция `analogRead()`, одним из параметров которой является имя того контакта, который вы собираетесь считывать. В отличие от функции `digitalRead()`, функция `analogRead()` возвращает число, а не просто значения «истина» или «ложь». Число, которое вы получите от `analogRead()`, будет находиться в диапазоне от 0 до 1023. Результат 0 соответствует 0 В, а результат 1023 — 5 В. Чтобы преобразовать число, полученное от функ-

ции `analogRead()`, в фактическое значение напряжения, нужно умножить число на 5, а затем разделить на 1023, либо просто разделить его на 204,6. Вот как это делается на языке C для платформы Arduino:

```
int raw = analogRead(A0);
float volts = raw / 204.6;
```

Переменная `raw` относится к типу `int` (целое число), поскольку при считывании с аналогового входа мы всегда получаем целое число. Однако для преобразования значения `raw` в десятичное число переменная должна относиться к типу `float` (число с плавающей точкой).

К аналоговым входам можно подключать различные датчики — например, далее в книге будет показано, как использовать аналоговый вход совместно с фоторезисторным датчиком (см. *разд. «Проект: домашняя поливальная установка на Arduino» главы 7*) и как подключать переменный резистор (см. *разд. «Проект: термостатический охладитель безалкогольных напитков» главы 12*).

Аналоговые выводы

Цифровые выводы позволяют подключенные к ним электронные компоненты (например, светодиоды) только включать и выключать. Но если вы хотите постепенно изменять напряжение, подаваемое на устройство, то вам понадобится *аналоговый вывод*. Именно через аналоговый вывод удобно управлять яркостью светодиода или, например, скоростью работы двигателя. Как вы уже догадываетесь, такая возможность будет использоваться в книге очень часто.

В качестве аналоговых выводов способны функционировать не все контакты Arduino Uno, а только D3, D5, D6, D9, D10 и D11. На плате Arduino (см. рис. 2.2) эти контакты помечены символом тильды (~).

Для управления аналоговым выводом используется функция `analogWrite()`, принимающая в качестве параметра число в диапазоне от 0 до 255. Значение 0 означает 0 В (полностью выключено), а значение 255 — включено на полную мощность.

Можно было бы предположить, что аналоговый вывод просто принимает напряжение в диапазоне от 0 В до 5 В, и, если подключить вольтметр между контактом, используемым в качестве аналогового вывода, и заземлением, то при изменении значения параметров функции `analogWrite()` от 0 до 255 действительно покажется, что напряжение меняется между 0 и 5 В. Тем не менее, на самом деле все обстоит несколько сложнее. На рис. 2.6 показано, что фактически происходит с таким выводом — это явление называется *широтно-импульсной модуляцией* (ШИМ).

Каждый контакт, работающий в режиме аналогового вывода, генерирует 490 импульсов в секунду (кроме контактов D5 и D6, которые выдают 980 импульсов в секунду). Ширина импульсов, т. е. их длительность, с помощью параметров функции `analogWrite()` может быть изменена. При этом, чем больше время, в течение которого импульс остается высоким, тем больше мощность, подводимая к выходу, и, следовательно, тем ярче́ горит светодиод или быстрее вращается двигатель.

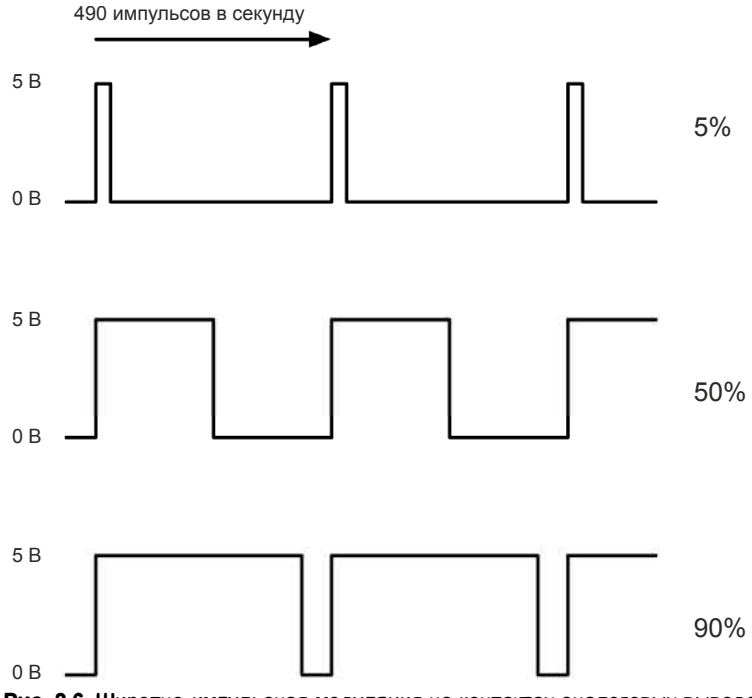


Рис. 2.6. Широтно-импульсная модуляция на контактах аналоговых выводов

Вольтметр фиксирует этот процесс как изменение напряжения лишь потому, что не может реагировать достаточно быстро, поэтому он как бы усредняет получаемое напряжение, и нам кажется, что оно меняется постепенно.

Оператор *If...Else*

Если вы помните, в *разд. «Цифровые входы»* (в этой главе ранее) мы использовали оператор `if` для выполнения указанного действия, если определенное условие соблюдается. Чтобы еще лучше контролировать выполнение кода, можно использовать оператор `if...else`, который выполняет один код, если условие соблюдается, и другой — если не соблюдается.

В следующем примере светодиод будет включен или выключен, в зависимости от того, каково аналоговое значение: больше `500` либо меньше или равно `500`:

```
if (analogRead(A0) > 500)
{
   digitalWrite(led, HIGH);
}
else
{
   digitalWrite(led, LOW);
}
```

До сих пор нам встречались два оператора сравнения: == (равно) и > (больше). На самом деле, можно выполнять и другие сравнения:

- <= (меньше или равно)
- >= (больше или равно)
- != (не равно)

Можно также сравнивать не только пары значений, но и делать более сложные сравнения, пользуясь операторами && (И) и || (ИЛИ). Например, если вы хотите включать светодиод только при значении между 300 и 400, то могли бы написать следующее:

```
int reading = analogRead(A0);
if ((reading >= 300) && (reading <=400))
{
  digitalWrite(led, HIGH);
}
else
{
  digitalWrite(led, LOW);
}
```

Циклы

Цикл позволяет повторить код заданное количество раз либо повторять его до тех пор, пока определенное условие не изменится. Для этого можно пользоваться циклами `for` или `while`: циклы `for` удобнее для выполнения того или иного действия фиксированное число раз, а `while` — для выполнения операции до тех пор, пока не будет соблюдено конкретное условие.

В следующем примере цикл `for` заставит светодиод 10 раз (обратите внимание: цикл `for` находится в функции `setup()`, а не в `loop()`, поскольку, будучи в `loop()`, он заставил бы индикатор мигнуть еще 10 раз после первых 10, а мы добиваемся не такого эффекта):

```
for (int i = 0; i < 10; i++)
{
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Если бы мы хотели, чтобы все светодиоды мигали до тех пор, пока нажата кнопка, подключенная к цифровому входу, то воспользовались бы циклом `while`:

```
while (digitalRead(9))
{
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Этот код предполагает, что контакт D9 подсоединен к переключателю (как показано на рис. 2.5).

Функции

Функции могут серьезно запутать начинающего программиста. Пожалуй, удобнее всего представить функцию как способ сгруппировать некоторое количество строк кода и дать им общее имя, чтобы этот фрагмент кода было легко использовать снова и снова.

Если бы вы попробовали ознакомиться с внутренним устройством Arduino, то заметили бы, что встроенные функции, например, `digitalWrite()`, мягко говоря, сложноваты. Далее приведен код функции `digitalWrite()` (не пытайтесь разобраться, что он делает, просто радуйтесь, что не приходится набирать весь этот код всякий раз, когда вы хотите перевести контакт из высокого состояния в низкое):

```
void digitalWrite(uint8_t pin, uint8_t val)
{
   uint8_t timer = digitalPinToTimer(pin);
   uint8_t bit = digitalPinToBitMask(pin);
   uint8_t port = digitalPinToPort(pin);
   volatile uint8_t *out;

   if (port == NOT_A_PIN) return;

   // Если контакт поддерживает ШИМ-вывод, то его нужно отключить,
   // прежде чем выполнять цифровую запись
   if (timer != NOT_ON_TIMER) turnOffPWM(timer);

   out = portOutputRegister(port);

   uint8_t oldSREG = SREG;
   cli();

   if (val == LOW) {
      *out &= ~bit;
   } else {
      *out |= bit;
   }

   SREG = oldSREG;
}
```

Присвоив имя такому большому фрагменту кода, нам достаточно будет обратиться к нему по имени, — и использовать.

Наряду со встроенными функциями (например, той же `digitalWrite`), можно создавать собственные функции для объединения операций, которыми вы пользуетесь.

Так, можно написать функцию, которая заставит светодиод мигнуть столько раз, сколько указано в параметре. Контакт, на который нужно передать команду «мигай», также можно указать как параметр. Это показано в следующем скетче: мы пишем функцию `blink` и вызываем ее на этапе `startup()`, так что светодиод **L** на плате Arduino мигнет 5 раз после сброса:

```
const int ledPin = 13;
void setup()
{
  pinMode(ledPin, OUTPUT);
  blink(ledPin, 5);
}
void loop() {}
void blink(int pin, int n)
{
  for (int i = 0; i < n; i++)
  {
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
  }
}
```

Функция `setup()` здесь задает `ledPin` в качестве вывода, а затем вызывает функцию `blink`, сообщая ей имя контакта, который должен мигать, и сколько раз он должен мигнуть. Функция `loop()` пуста и ничего не делает, но Arduino IDE настаивает, что она должна здесь быть.

Сама функция `blink` начинается со слова `void`, указывающего, что функция ничего не возвращает. Иными словами, результат вызова этой функции нельзя присвоить переменной, что, возможно, мы захотели бы сделать, если бы функция выполняла какие-либо вычисления. Затем мы видим имя функции (`blink`) и принимаемые ею параметры, заключенные в круглые скобки и разделенные запятыми. Определяя функцию, нужно задать тип каждого параметра (т. е. `int`, `float` или др.). В нашем случае как сам контакт, который будет мигать (`pin`), так и количество проблесков (`n`) являются целыми числами (`int`).

В языке C, как и в большинстве других языков программирования, различаются локальные и глобальные переменные. *Глобальные переменные* (как `ledPin` в предыдущем примере) могут использоваться в любой точке программы. С другой стороны, *локальные переменные*, например, параметры функций (в нашем случае `pin` и `n`), и даже переменная `i` в цикле `for` — все они доступны лишь в рамках той функции, в которой определены.

Итак, в функции `setup()` строка `blink(ledPin, 5)` передает глобальную переменную `ledPin` в функцию `blink`, где она присваивается локальной переменной `pin`. Логично спросить: а зачем это делается? Дело в том, что, передавая `pin` в `blink`,

мы делаем функцию `blink` универсальной, и можем с ее помощью зажигать любой указанный контакт, а не только `ledPin`.

В теле функции `blink` есть цикл `for`, который будет повторять вложенные в него функции `digitalWrite()` и `delay()` 5 раз. Впрочем, если `n` сделать равным 3, то светодиод мигнет 3 раза.

Заключение

В этой главе мы научились устанавливать среду разработки Arduino IDE и изучили несколько простейших команд для программирования Arduino. Мы также разобрались, как код позволяет управлять входными и выходными контактами Arduino.

Если ранее вы не занимались программированием, то на освоение некоторых понятий, представленных в этой главе, вам потребуется время. Один из наиболее удобных способов изучать программирование — рассматривать готовые примеры, а затем пытаться их изменить, чтобы понять, что происходит в коде.

Как уже упоминалось ранее, весь код к этой книге можно скачать из специально созданного репозитория на сайте GitHub (**github.com/simonmonk/make_action**), поэтому для работы с книгой нет необходимости писать какие-либо программы. Хотя, пожалуй, эксперименты и проекты из этой книги могут послужить вам хорошей основой для реализации собственных идей.

В следующей главе вас ждет похожий «курс молодого бойца», но уже по Raspberry Pi.

Raspberry Pi 3

В этой главе вы познакомитесь с компьютером Raspberry Pi. Если вы уже имеете опыт работы с Raspberry Pi и пробовали программировать для него на языке Python, можете сразу перейти к *главе 4*.

Что есть Raspberry Pi?

Raspberry Pi — это одноплатный компьютер с операционной системой Linux, USB-разъемами для подключения мыши и клавиатуры, а также с видеовыходом HDMI для подключения монитора.

Raspberry Pi существует в нескольких вариантах, некоторые из них устарели и более не производятся, однако все модели Raspberry Pi обладают широкой совместимостью, и вы сможете без проблем выполнить любые примеры из этой книги, даже если у вас устаревшая версия Raspberry Pi.

На рис. 2.1 показана плата Raspberry Pi 2 модели B. С правой стороны платы расположены четыре USB-порта. К ним удобно подключать клавиатуру и мышь, а также другие периферийные устройства, — в частности, принтеры, сканеры и флэш-накопители.

Под USB-портами находится гнездо RJ45 Ethernet, через которое можно при помощи соответствующего кабеля подсоединить Raspberry Pi к домашнему сетевому маршрутизатору. Подключение Raspberry Pi к сети бывает нужно, чтобы выходить с него в Интернет и устанавливать программы. В некоторых случаях удобнее обойтись без кабеля и воспользоваться подключаемым по USB адаптером Wi-Fi. Стоит такой адаптер недорого, однако вам придется подобрать адаптер, совместимый с Raspberry Pi. Список совместимых устройств приводится по адресу **http://elinux.org/RPi_VerifiedPeripherals**.

Продолжим знакомство с Raspberry Pi, обходя его по часовой стрелке. Сначала мы увидим разъем для работы со стерео-аудио и композитным видео. В это гнездо обычно подключаются наушники или вспомогательный кабель, ведущий к колонкам, но в разъеме предусмотрена и дополнительная возможность подключения мониторов по специальному кабелю для композитного видео. Впрочем, монитор

или телевизор чаще подключаются к Raspberry Pi через расположенный чуть далее видеовыход HDMI, т. к. он позволяет достичь гораздо более высокого качества изображения, чем на композитном видео. Между HDMI-выходом и аудиоразъемами размещен разъем для плоского кабеля, через который можно подключить к Raspberry Pi специальную видеокамеру.

Рядом с гнездом HDMI находится разъем Micro-USB, который служит только для запитывания Raspberry Pi через 5-вольтовый адаптер.

Над разъемом Micro-USB на обратной стороне платы расположен разъем, в который вставляется карта памяти MicroSD. Как уже отмечалось ранее, компьютер Raspberry Pi не имеет обычного жесткого диска, а операционная система и все пользовательские файлы хранятся на этой карте памяти.

В верхней части платы размещена колодка универсальных контактов ввода/вывода (GPIO). С их помощью можно подключать к Raspberry Pi всевозможные электронные схемы, чтобы управлять смонтированными на них устройствами.

До выпуска Raspberry Pi версии B+ на колодке GPIO имелось всего 26 контактов, а не 40, как показано на рис. 3.1. Однако во всех проектах этой книги используются только 26 контактов, поэтому, даже если у вас старый Raspberry Pi, все проекты все равно должны работать.

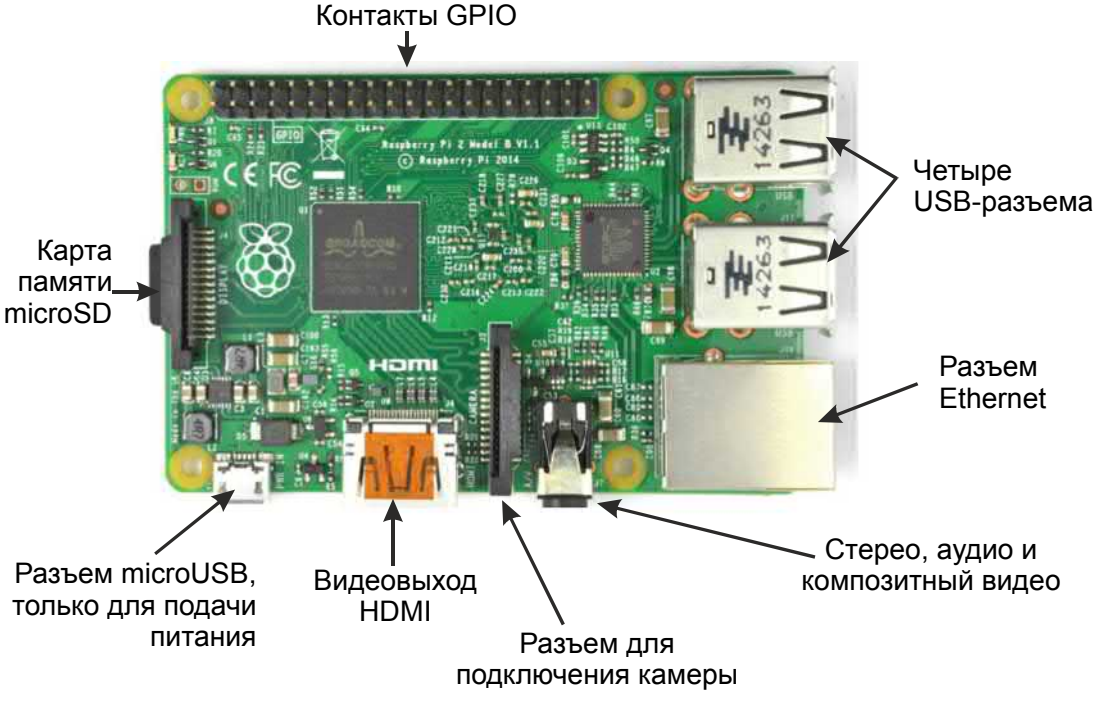


Рис. 3.1. Raspberry Pi 2 модель B

Настройка Raspberry Pi

Чтобы настроить Raspberry Pi, к нему нужно подключить мышь, клавиатуру и монитор. После завершения настройки клавиатуру, мышь и монитор можно будет отключить, а для управления Raspberry Pi подключиться к нему с другого компьютера по безопасному сетевому протоколу SSH. Однако, прежде чем вы дойдете до этого этапа, периферия должна быть к Raspberry Pi подключена, — иначе вам его не настроить.

Для настройки Raspberry Pi понадобится следующее оборудование:

- клавиатура и мышь с USB-разъемами (стандартная периферия для ПК вполне подойдет);
- монитор или телевизор с HDMI-видеовыходом и HDMI-кабелем;
- сетевой адаптер питания с напряжением 5 В и током не менее 1 А, оснащенный разъемом Micro-USB для подключения к Raspberry Pi;

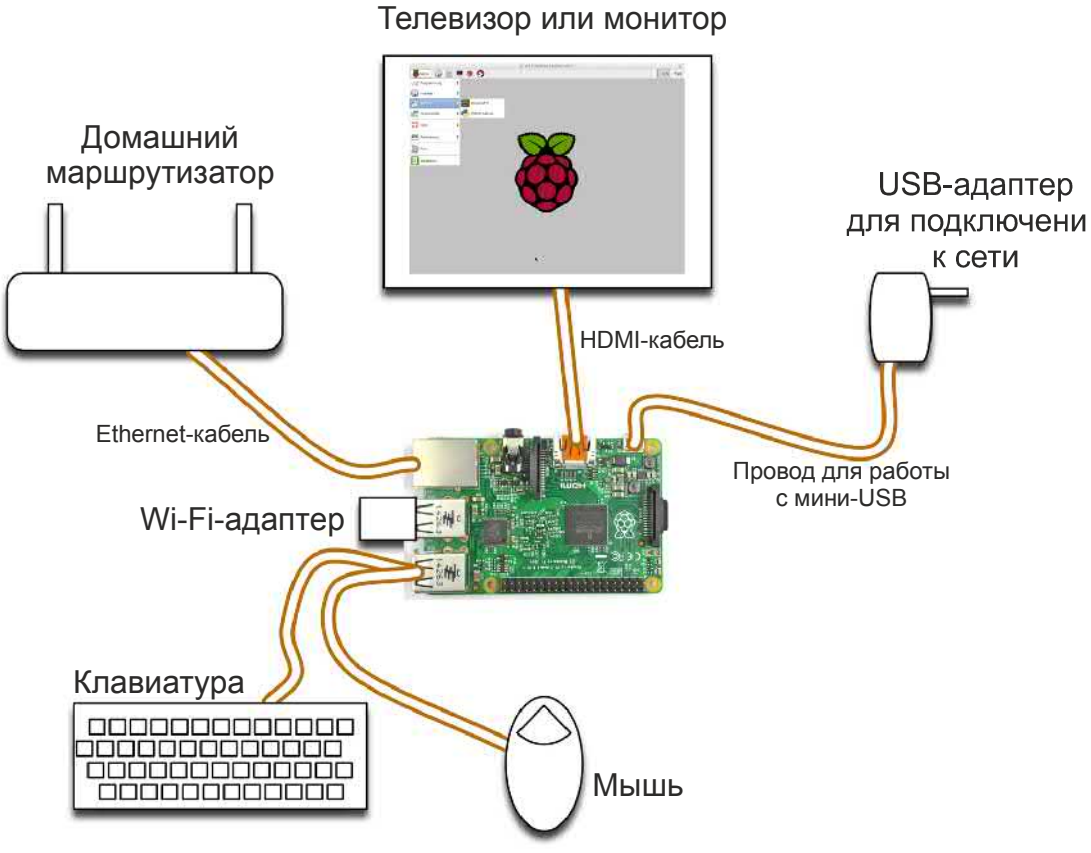


Рис. 3.2. Подготовка Raspberry Pi к настройке

- Ethernet-кабель для подключения к маршрутизатору или подключаемый по USB адаптер Wi-Fi;
- карта памяти MicroSD. Карты объемом 4 Гбайт вполне достаточно, однако на карте объемом 8 Гбайт будет больше места для ваших файлов и любых программ, которые вы захотите на Raspberry Pi установить. В любом случае выбирайте карту памяти класса 10 — это поможет повысить производительность системы;
- второй компьютер и адаптер MicroSD, чтобы подготовить карту памяти, помещаемую в Raspberry Pi. В качестве альтернативы можно приобрести специальную карту памяти NOOBS, на которой уже предустановлено все необходимое ПО.

На рис. 3.2 показано типичное оснащение для настройки Raspberry Pi.

Как минимум на этапе установки операционной системы Raspberry Pi нужно включать там, где можно напрямую выйти на маршрутизатор, подключенный к Интернету. Когда операционная система будет установлена, вы сможете сконфигурировать USB-адаптер Wi-Fi и, если желаете, продолжать работу в беспроводном режиме.

Подготовка карты памяти MicroSD с предустановленным программным обеспечением

Когда компьютер Raspberry Pi только появился на рынке, чтобы подготовить к работе его карту памяти приходилось пользоваться специальным ПО для записи образов операционных систем. Однако технология NOOBS эти сложности отменила — теперь вся подготовка карты сводится к копированию на нее нужных файлов, и никакого специального форматирования не требуется.

Самые свежие инструкции по настройке карты MicroSD при помощи NOOBS вы найдете по адресу **https://www.raspberrypi.org/help/noobs-setup/**.

Загрузив Raspberry Pi с карты NOOBS, вы увидите предложение выбрать операционную систему (рис. 3.3). Обязательно выберите рекомендуемую опцию **Raspbian**.

После длительной загрузки и многих перезагрузок ваш Raspberry Pi будет готов к работе. Теперь, если у вас есть USB-адаптер Wi-Fi, можно его сконфигурировать и подключиться к беспроводной сети при помощи утилиты Wi-Fi Config, которую вы найдете в разделе **Preferences** (Настройки) меню рабочего стола загруженной на Raspberry Pi операционной системы Raspbian.

Настройка SSH

Для многих проектов и экспериментов, описанных в этой книге, клавиатура, мышь и монитор, подключенные к Raspberry Pi, скорее помешают, чем помогут. Протокол SSH открывает сетевой доступ к Raspberry Pi через командную строку с компьютера, подключенного к той же сети.

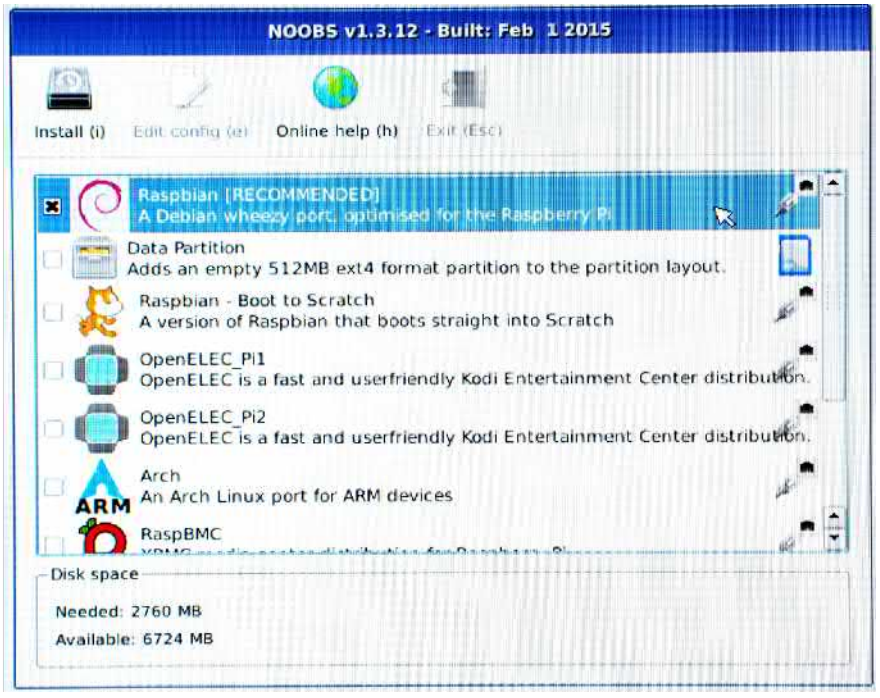


Рис. 3.3. Система NOOBS предлагает выбрать операционную систему для установки

Это означает, что как только вы закончите настройку Raspberry Pi, к нему должны быть подключены лишь провод электропитания и либо сетевой кабель, либо USB-адаптер Wi-Fi.

Щелкните на значке терминала в верхней части рабочего стола Raspberry Pi (рис. 3.4) и в открывшемся окне LXTerminal введите следующую команду (здесь и далее: символ `$` не вводите — это приглашение командной строки):

```
$ sudo raspi-config
```

В результате откроется инструмент **raspi-config** для настройки Raspberry Pi. При помощи клавиш со стрелками перейдите в его меню к пункту **Advanced** (Дополнительно) и нажмите клавишу <Enter>. Затем вновь воспользуйтесь клавишами со стрелками и выберите пункт **SSH** (рис. 3.5).

Выберите затем пункт **Enable** (Активировать), после чего выполните в меню команду **Finish** (Готово).

Теперь на вашем Raspberry Pi активирован протокол SSH для удаленного доступа. Настало время подключаться с компьютера.

Если у вас на компьютере установлена операционная система Linux или Mac OS, то там доступна собственная программа для работы с окном терминала, способная подключаться к Raspberry Pi. Поэтому вы можете сразу переходить к *разд. «SSH в Mac OS или Linux»* далее в этой главе.

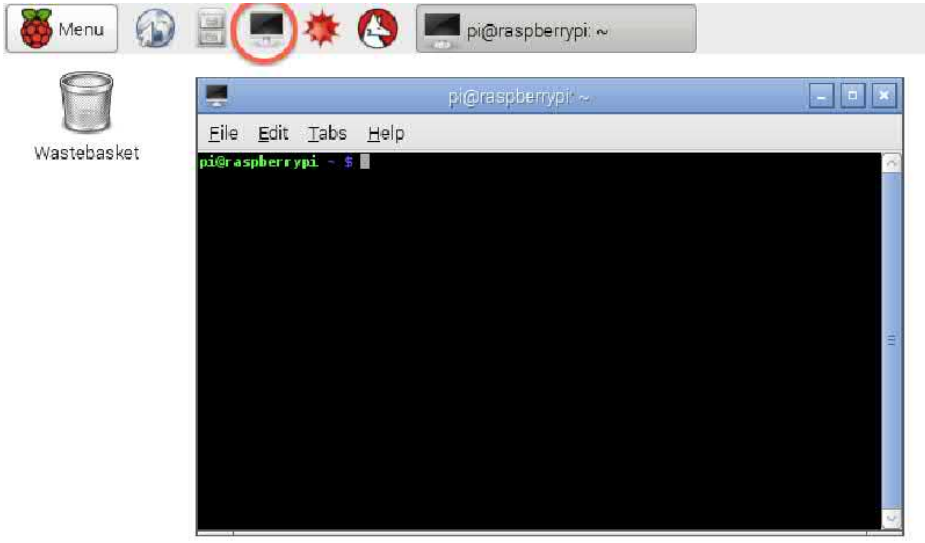


Рис. 3.4. Запуск LXTerminal

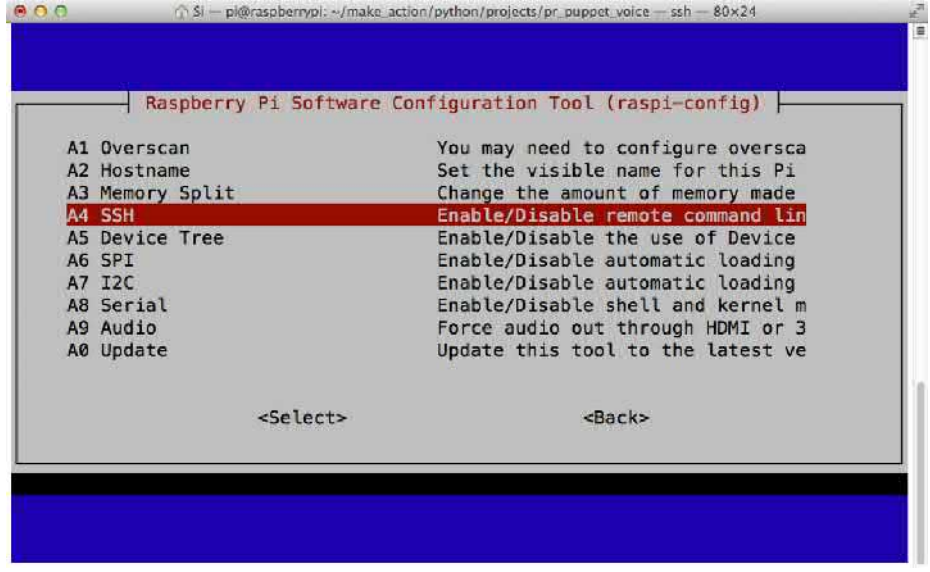


Рис. 3.5. Активация **SSH** при помощи **raspi-config**

Поиск IP-адреса Raspberry Pi

Прежде чем вы сможете подключиться к Raspberry Pi по SSH с другого компьютера своей сети, вам понадобится определить *IP-адрес* Raspberry Pi.

Для этого выполните на Raspberry Pi в окне LXTerminal следующую команду:

```
$ hostname -I
```

Она вернет состоящий из 4-х групп цифр номер, построенный по следующему принципу: 192.168.1.23.

Это и есть IP-адрес вашего Raspberry Pi.

SSH на компьютере с Windows

Если ваш компьютер работает под операционной системой Microsoft Windows, то для работы с Raspberry Pi вам потребуется программа PuTTY. Устанавливать программу не требуется — с адреса **http://www.putty.org/** вы скачиваете исполняемый файл программы putty.exe. Можете просто сохранить его где-нибудь (например, на рабочем столе) и запустить двойным щелчком мыши. Откроется окно конфигурации PuTTY (рис. 3.6).

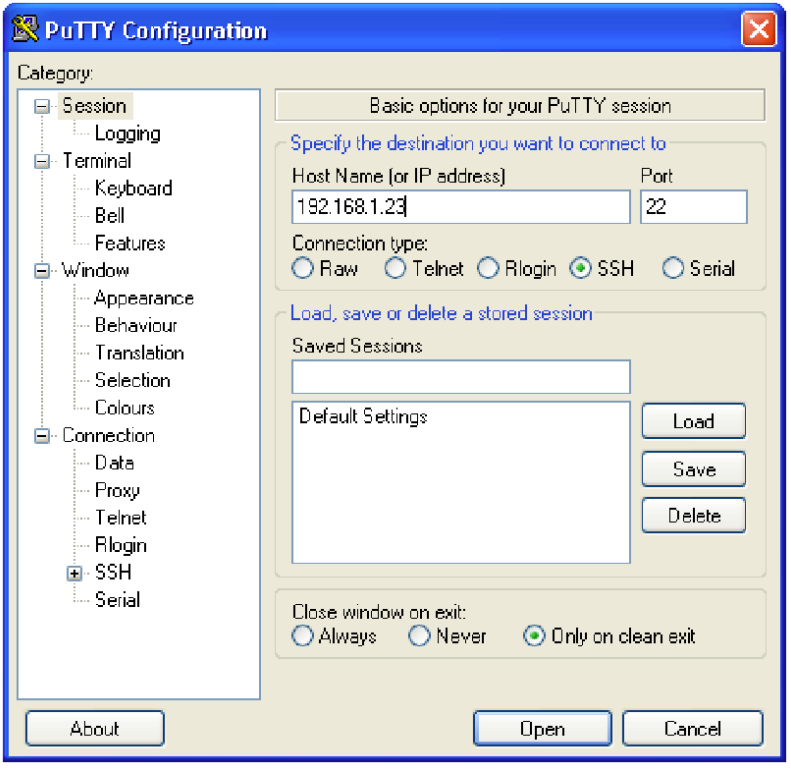


Рис. 3.6. Конфигурационное окно PuTTY

Введите в поле **Host Name (or IP address)** (Хост-имя или IP-адрес) IP-адрес вашего Raspberry Pi (см. ранее врезку *«Поиск IP-адреса Raspberry Pi»*) и нажмите кнопку **Open** (Открыть). Система предложит вам зайти на **pi** (рис. 3.7). Введите имя пользователя: `pi` и пароль: `raspberry` — все, вы вошли. Теперь вы можете вводить в PuTTY команды на вашем основном компьютере — и они будут выполняться на Raspberry Pi.

SSH в Mac OS или Linux

Если вы работаете на компьютере под управлением операционной системы Mac OS или Linux, то программы для подключения к Raspberry Pi на вашем компьютере уже предустановлены. Откройте сеанс работы с терминалом и введите следующую

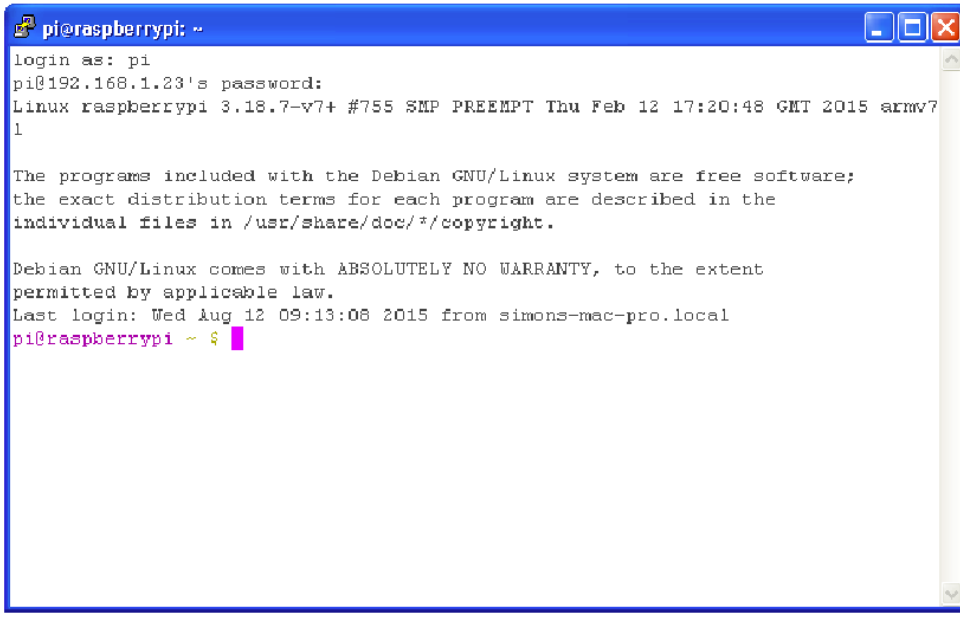


Рис. 3.7. Удаленное управление Raspberry Pi по протоколу SSH

команду, заменив в ней IP-адрес моей платы Raspberry Pi (192.168.1.23) адресом своей платы Raspberry Pie:

```
$ ssh 192.168.1.23 -l pi
```

Сделав это в первый раз, вы увидите следующее сообщение:

```
The authenticity of host '192.168.1.23 (192.168.1.23)' can't be established.
RSA key fingerprint is 48:8f:c3:07:c2:04:9e:8b:59:ed:53:2b:0b:d0:aa:e5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.23' (RSA) to the list of known hosts.
pi@192.168.1.23's password:
```

Чтобы продолжить работу, подтвердите аутентичность того компьютера, к которому подключаетесь, — введите `yes`. С этого момента все вводимые вами команды на самом деле будут выполняться на Raspberry Pi, а не на вашем компьютере.

Командная строка Linux

Если вы обычно работаете в операционной системе Windows или Mac OS, то, возможно, вам никогда не приходилось взаимодействовать с компьютером через командную строку. Однако Linux — операционная система Raspberry Pi — требует вводить команды в командную строку при установке программ, копировании и переименовании файлов, запуске программ или редактировании файлов.

Мы уже воспользовались командной строкой Raspberry Pi при работе с LXTerminal — когда настраивали SSH-соединение на Raspberry Pi, и теперь команды Raspberry Pi можно выполнять по SSH или непосредственно через LXTerminal.

Вы, вероятно, заметили, что как только LXTerminal или SSH-сеанс будут готовы к приему команды, в конце строки появится символ `$`. Он называется *приглашением командной строки* — именно так Linux сообщает, что готова к получению следующей команды.

Работа с командной строкой Linux основана на концепции *текущего* каталога. Это каталог, с которым вы работаете в настоящий момент. Это значит, что если вы хотите выполнить программу на Python, расположенную в конкретном каталоге, то обычно перед запуском этой программы нужно перейти в тот каталог, в котором она находится. Предназначенная для такого перехода команда называется `cd` (от англ. `change directory`, изменить каталог).

Сразу после запуска сеанса LXTerminal вы оказываетесь в текущем каталоге /home/pi. Он называется *домашним каталогом*. Если в домашнем каталоге есть папка make_action, то перейдите в нее, введя следующую команду (изменения каталога относительно текущего):

```
$ cd make_action
```

Также можно ввести и весь путь к каталогу:

```
$ cd /home/pi/make_action
```

Весь код для Raspberry Pi в этой книге написан на языке программирования Python. Чтобы запустить программу на Python, называемую test.py, надо выполнить следующую команду:

```
$ python test.py
```

Другая команда, с которой вам доведется часто иметь дело, называется `sudo`. Она применяется для запуска следующей за ней команды в режиме *суперпользователя*. Дело в том, что Linux пытается защититься от случайного удаления важных файлов операционной системы и выполнения других критических действий и не дает такого права обычным пользователям. При этом имеется в виду, что пользователь, выступающий от имени суперпользователя, знает, что делает. Тем не менее, работа в режиме суперпользователя нужна для получения доступа к контактам GPIO, к чему в этой книге потребуется прибегать довольно часто.

Например, если программа test.py использует контакты GPIO, то запускать ее потребуется следующей командой:

```
$ sudo python test.py
```

Хотя весь код на Python к этой книге доступен для скачивания (см. *разд. «Код к книге»* далее в этой главе), иногда вам понадобится изменить программу или какой-либо другой файл. Это можно сделать в редакторе nano, который отлично работает и по SSH. Чтобы отредактировать файл, просто введите команду `nano`, а затем укажите файл для редактирования. Если файла с таким именем не существует, `nano` создаст его, когда вы решите сохранить отредактированное:

```
$ nano test.py
```

На рис. 3.8 редактор nano показан в действии.

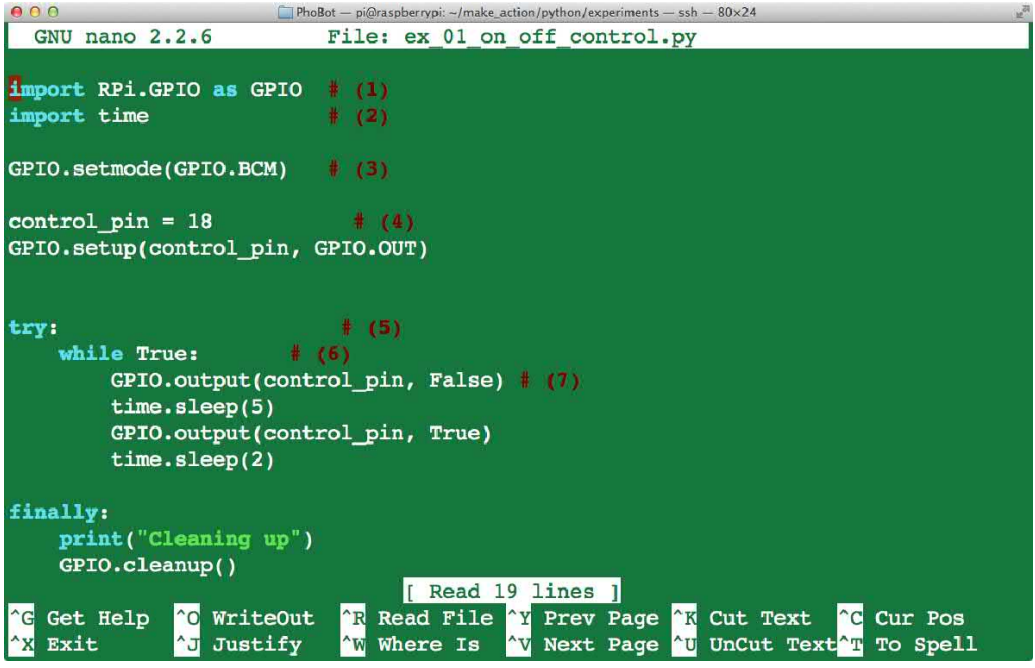


Рис. 3.8. Редактор nano

Поскольку nano предназначен для работы в среде с командной строкой (например, в LXTerminal или по SSH), для перехода по файлу приходится пользоваться клавишами со стрелками, а не мышью. Когда вы соберетесь сохранить файл, нажмите сочетание клавиш <Ctrl>+<X>, затем клавишу <Y> и, наконец, клавишу <Enter>, чтобы подтвердить сохранение.

Код к книге

Raspberry Pi программируется без участия отдельного компьютера, поэтому все программные файлы, используемые в книге, нужно загрузить прямо на Raspberry Pi.

Хотя файлы и можно ужать, сохранив их в виде ZIP-архива, как это делалось с кодом Arduino (см. *главу 2*), лучше воспользоваться Git — специальным инструментом для работы с программным кодом.

Git отслеживает вносимые в код изменения и даже позволяет объединять код, написанный разными программистами. В популярном общедоступном репозитории кода GitHub как раз используется Git. Система Git предустановлена и в дистрибутиве Raspbian, поэтому для извлечения всего кода на Raspberry Pi вам всего лишь потребуется открыть сеанс LXTerminal и выполнить следующую команду:

```
$ git clone https://github.com/simonmonk/make_action.git
```

после чего на вашем Raspberry Pi будет создан каталог make_action. В нем вы найдете каталог python с двумя подкаталогами: experiments и projects.

Работать с Git удобнее потому, что вы можете извлекать командный код всякий раз, когда хотите проверить, не внес ли автор в него какие-либо изменения и дополнения. Чтобы получить эти изменения, просто запустите команду:

```
$ git pull
```

Руководство по программированию

Чтобы научиться программировать, удобнее всего попробовать изменить какие-нибудь простые программы и постепенно разобраться, что в них к чему. Все программы, которые понадобятся вам для этой книги, доступны для скачивания, поэтому работать можно и без опыта программирования. Однако полезно представлять себе, что и как устроено.

Hello, World

Обычно первая программа, которую вы пишете на новом языке, просто выводит на экран слова **Hello, world**. Чтобы познакомиться с ней, запустите редактор nano и введите следующую команду:

```
$ nano hello.py
```

Расширение `py` указывает, что файл с программой написан на языке Python. Далее введите в редактор nano следующий текст и сохраните файл:

```
print('Hello, World')
```

Затем запустите программу:

```
$ python hello.py
Hello, World
```

Сравнение Python 2 и Python 3

Сейчас пользователи Python переживают тяжелые времена — приходится уходить со старой версии Python. И хотя выпущена новейшая версия Python 3, многие по-прежнему работают с Python 2.

В Raspbian используется как Python 2, так и Python 3. Если вы хотите выполнить программу Hello, world! на Python 3, то вводите команду:

```
$ python3 hello.py
```

Результат будет совершенно одинаков.

Почему же в таком случае многие пользователи продолжают работать с Python 2, и на Python 2 также основаны все программы этой книги?

Дело в том, что не все библиотеки Python удалось успешно портировать на Python 3. Приведу пример: библиотека последовательного интерфейса, с которой мы будем работать в *главе 14*, на момент подготовки книги не работала с Python 3.

Табуляция и отступы

Программисты привыкли оформлять код особым образом, чтобы он легче воспринимался. В языке C для Arduino блоки кода внутри функции или инструкции `if` пишутся с отступами, чтобы вы сразу видели, к какой функции или команде относится код. В Python такой стиль — не дело вкуса, а обязательное требование.

В Python нет скобок `{` или `}`, которые указывали бы начало и конец блока кода. Чтобы показать, какие строки кода связаны друг с другом, используются отступы.

Рассмотрим пример кода:

```
while True :
   GPIO.output(control_pin, False)
   time.sleep(5)
   GPIO.output(control_pin, True)
   time.sleep(2)
print("Завершено")
```

Это цикл `while` (точно такого же вида, как цикл `while` в языке C для Arduino, описанный в *разд. «Циклы» главы 2*). В нашем случае условие выполняется (`True`). Значение `True` всегда равно `True`, поэтому цикл получается бесконечным. В конце первой строки стоит двоеточие (`:`). Оно указывает, что далее идет блок кода. Как можно видеть, двоеточие в Python аналогично `{` в C для Arduino, только закрывающий символ отсутствует.

Строки в блоке кода должны быть отделены от предыдущей строки. Пока вы придерживаетесь оформления, не важно, сколько именно пробелов вы оставляете между строками кода. Большинство программистов используют на каждом уровне отступа четыре пробела.

В конце блока кода никакой символ не ставится, просто прекращаются отступы. Обратите внимание: в приведенном примере кода последняя команда `print` находится за пределами цикла. Поскольку цикл `while` бесконечен, последняя строка так и не будет исполнена.

Переменные

Переменные в Python напоминают переменные в C на Arduino, но отличаются тем, что при первом использовании переменной не требуется указывать, относится ли она к типу `int`, `float` или к какому-либо другому. Переменной просто присваивается значение, и ничто (кроме здравого смысла) не мешает вам присвоить одной и той же переменной целочисленное значение, а через минуту — уже строковое. Например, следующий код вполне допустим, но не слишком удачен:

```
a = 123.45
a = "message"
```

При работе со строками можно использовать либо двойные кавычки (как в приведенном примере), либо одиночные.

Инструкции *if*, *while* и пр.

В Python есть структура `if...else`, как и в C на Arduino, но блоки кода внутри инструкции `if` обозначаются при помощи двоеточий и отступов:

```
if x > 10 :
  print("x - больше!")
else:
  print("x - меньше")
```

Работая далее с книгой, вы подробнее познакомитесь с циклами и другими программными структурами.

Библиотека RPi.GPIO

Как и в Arduino C, в Python поддерживаются библиотеки — их просто потребуется импортировать, а потом можно будет использовать в программе.

В этой книге мы будем много работать с колодкой GPIO, расположенной на Raspberry Pi. Самая популярная библиотека Python для управления контактами GPIO, которая и используется в этой книге, называется RPi.GPIO.

Эта библиотека предустановлена на Raspbian, поэтому ее установкой заниматься не придется, — можно просто импортировать ее в программу и начинать использовать.

Существует небольшая путаница с именованием контактов на колодке GPIO в Raspberry Pi: иногда указывается позиция контакта на колодке (1, 2, 3, 4 и т. д.), а иногда — функция контакта. На заре существования Raspberry Pi оба варианта были достаточно распространены. Сегодня большинство пользователей предпочитают именовать контакты по их функциям. Библиотека RPi.GPIO поддерживает оба способа именования контактов, но ей нужно сообщить, какой вариант использовать. Соответствующий код вы встретите в начале почти любой программы Python из этой книги:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
```

Первая строка импортирует библиотеку RPi.GPIO, а вторая определяет Broadcom (BMC) — это название чипа в Raspberry Pi.

На Raspberry Pi, в отличие от Arduino, нет аналоговых входов. Однако библиотека RPi.GPIO поддерживает аналоговые ШИМ-входы.

Колодка GPIO

На рис. 3.9 показаны варианты подключения к колодке GPIO. Если у вас старая модель Raspberry Pi, где всего 26 контактов, то тех контактов, которые находятся под пунктирной линией, у вас на плате не будет. Проекты, описанные в этой книге, можно выполнять на любых моделях Raspberry Pi, поэтому мы будем работать лишь с первыми 26 контактами.

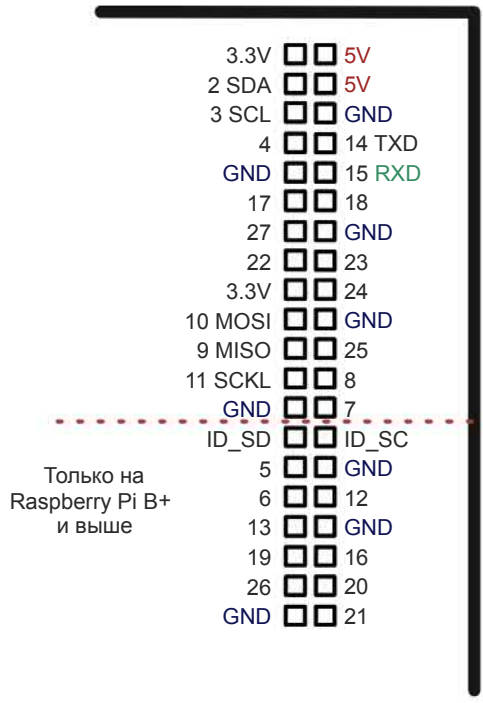


Рис. 3.9. Колодка GPIO

Схема для справки продублирована в конце книги (см. *приложение 2*). В отличие от Arduino, на самой схеме Raspberry Pi имена контактов не нанесены на печатную плату (PCB), поэтому сложно определить, где какой контакт. Чтобы было проще, можно купить или самостоятельно изготовить трафарет, накладываемый поверх контактов, — например, Raspberry Leaf (**https://www.adafruit.com/products/2196**). Обратите внимание, что некоторые контакты пронумерованы (допустим, **2**), но рядом также указана функция (например, **SDA**). Когда вы будете упоминать в коде контакты GPIO, просто ссылайтесь на них по номеру.

Цифровые выходы

В приводимом далее примере показано, как сконфигурировать контакт `18` с колодки GPIO (см. рис. 3.9) в качестве цифрового выхода, а затем установить высокий уровень выходного сигнала:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.output(18, True)
```

Когда мы устанавливаем выход при помощи команды `GPIO.output`, на нем можно задать высокий уровень выходного напряжения, применив значение `True` или `1`, либо низкий, применив `False` или `0`.

Цифровые входы

Цифровые входы на Raspberry Pi функционально во многом похожи на цифровые входы Arduino:

```
GPIO.setup(18, GPIO.IN)
value = GPIO.input(18)
print(value)
```

Указать, что на входе работает подтягивающий резистор (см. *разд. «Цифровые входы» главы 2*), можно вот так:

```
GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Аналоговые выходы

Работа с аналоговыми выходами на Raspberry Pi протекает в два этапа. Сначала нужно задать контакт в качестве выхода, а затем определить ШИМ-канал, который будет использовать этот выход.

Подробный пример, в котором используется такая возможность, приведен в *разд. «Эксперимент: смешивание цветов» главы 6*.

Заключение

В этой главе мы разобрались с тем, как настроить Raspberry Pi, и освоились с командной строкой Linux. Мы также познакомились с основами языка Python, который понадобится для управления устройствами через контакты GPIO.

Теперь давайте попрактикуемся — в *главе 4* мы рассмотрим основы использования Arduino и Raspberry Pi.

Первое знакомство 4

Всегда лучше — и уж точно интереснее — попробовать что-либо самому, а не просто прочитать об этом. И сейчас я расскажу вам, как обращаться с беспаечной макетной платой, на которую устанавливаются различные электронные компоненты. Вы также узнаете, как с помощью Arduino и Raspberry Pi управлять светодиодом и двигателем.

Беспаечная макетная плата

Как правило, подключая к Raspberry Pi или Arduino внешние электронные устройства (те же двигатели), не удается сделать это напрямую — требуется задействовать дополнительные электронные компоненты. В любом случае, даже если вы просто зажигаете светодиод, вам нужно его каким-то образом к Raspberry Pi или Arduino подсоединить.

Чтобы соединять устройства, не прибегая к пайке, удобно пользоваться *беспаечными макетными платами*. Такие платы дают электронщикам возможность создавать и отрабатывать прототипы моделируемых ими схем до того, как воплотить их в более стабильных вариантах, — с надежно пропаянными контактами.

На рис. 4.1 показана беспаечная макетная плата с установленными на ней компонентами из *разд. «Эксперимент: управление электродвигателем»*, который мы рассмотрим в этой главе чуть позже. Как видите, на беспаечной макетной плате можно и компоненты размещать, и подключать к ней провода.

Беспаечная макетная плата, с которой мы будем иметь дело, часто именуется *малой* или *400-точечной* (т. к. в ней 400 отверстий). Бывают беспаечные макетные платы и других размеров, но именно такой размер лучше всего подходит для проектов и экспериментов, описанных в этой книге.

По бокам беспаечной макетной платы такого типа обычно имеются два ряда отверстий. Как правило, один из них маркирован красной линией, а другой — синей. Все отверстия каждого такого ряда электрически соединены между собой перемычками, расположенными на нижней стороне платы. Хотя эти ряды отверстий можно использовать для чего угодно, чаще всего они служат для подачи на схему положительного и отрицательного питания.

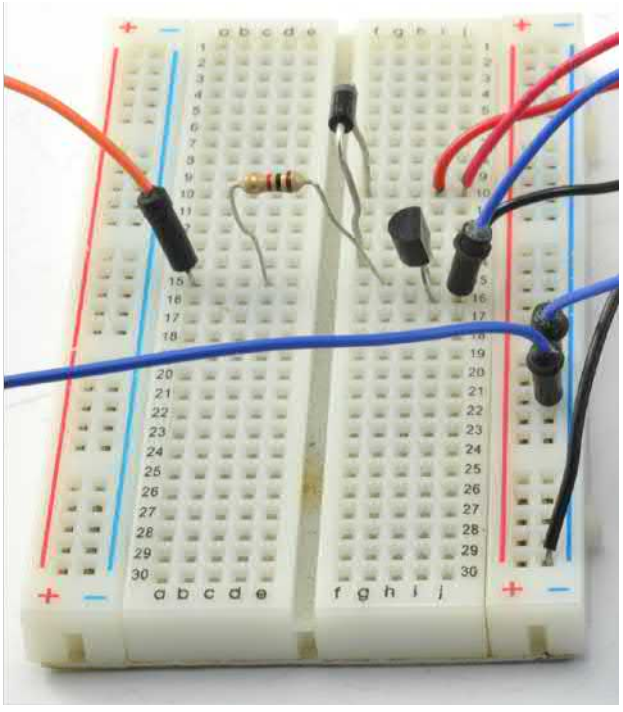


Рис. 4.1. Беспаечная макетная плата

Основное пространство платы поделено на два блока рядов по пять отверстий в ширину на каждом. В этих рядах любые пять отверстий, расположенные на одном уровне, соединены перемычками, расположенными под платой. Чтобы подключить вывод одного компонента к другому, оба провода от них нужно просто вставить в отверстия, расположенные на макетной плате в одном ряду.

Не разбирайте макетную плату!

Под отверстиями, расположенными на лицевой панели макетной платы, находятся металлические перемычки, снабженные зажимами, способными фиксировать провода и выводы компонентов. Не рекомендую разбирать макетную плату (как показано на рис. 4.2), поскольку собрать ее в прежнем виде обычно уже не удается.

Подключение к макетной плате Arduino

Коннекторы GPIO платы Arduino (которые иногда неправильно называют «пинами») на самом деле представляют собой *разъемы*. Чтобы соединить такой разъем с одним из отверстий на макетной плате, можно воспользоваться соединительными проводами (монтажными перемычками), имеющими штыревые контакты на обоих концах, — на жаргоне электронщиков их называют «папа-папа» (рис. 4.3).

Полезно иметь хороший запас таких соединителей разной длины, чтобы подключать что угодно к чему угодно. Обзавестись ими несложно — в продаже имеются наборы монтажных перемычек всевозможных размеров и цветов (см. *приложе-*

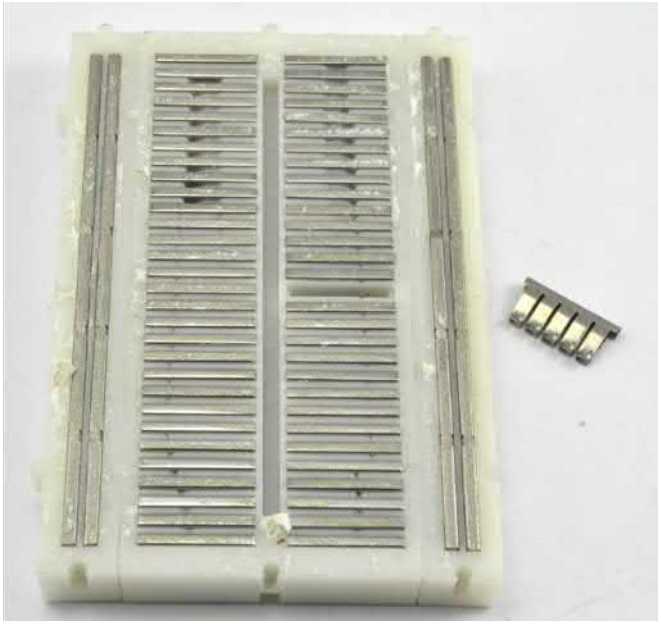


Рис. 4.2. Нижняя сторона макетной платы с одной снятой перемычкой

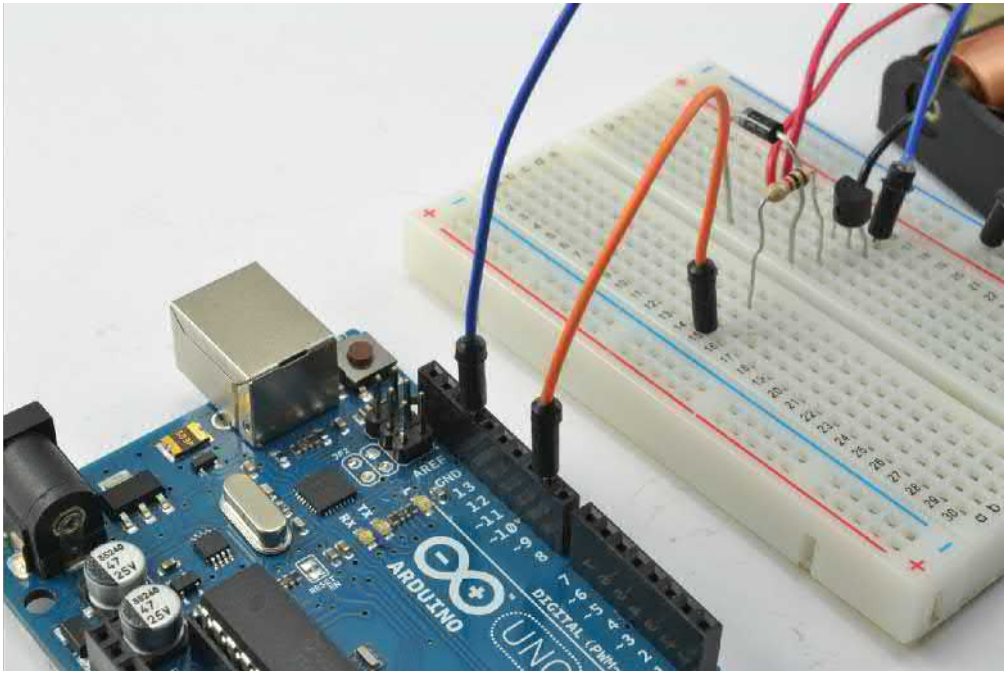


Рис. 4.3. Подключение Arduino к макетной плате

ние 1). Имеет смысл сразу приобрести комплект в составе макетной платы с прилагающимися к ней перемычками. Кстати, когда провода разноцветные, лучше видно, какой из них к чему подключен, особенно если ваша плата опутана множеством проводов.

Подключение к макетной плате Raspberry Pi

Коннекторы GPIO на Raspberry Pi — в отличие от Arduino — это *штифты*, а не разъемы. Соответственно, с ними нельзя использовать перемычки типа «папа-папа», как в случае с Arduino. Вместо них при работе с Raspberry Pi вам понадобятся соединительные провода типа «мама-папа». На одном конце такого провода находится разъем, насаживаемый на штифт колодки GPIO Raspberry Pi, а на другом — штыревой контакт, вставляемый в макетную плату.

На рис. 4.4 показано, как подключать контакты GPIO Raspberry Pi к конкретному ряду на макетной плате, пользуясь перемычками «мама-папа».

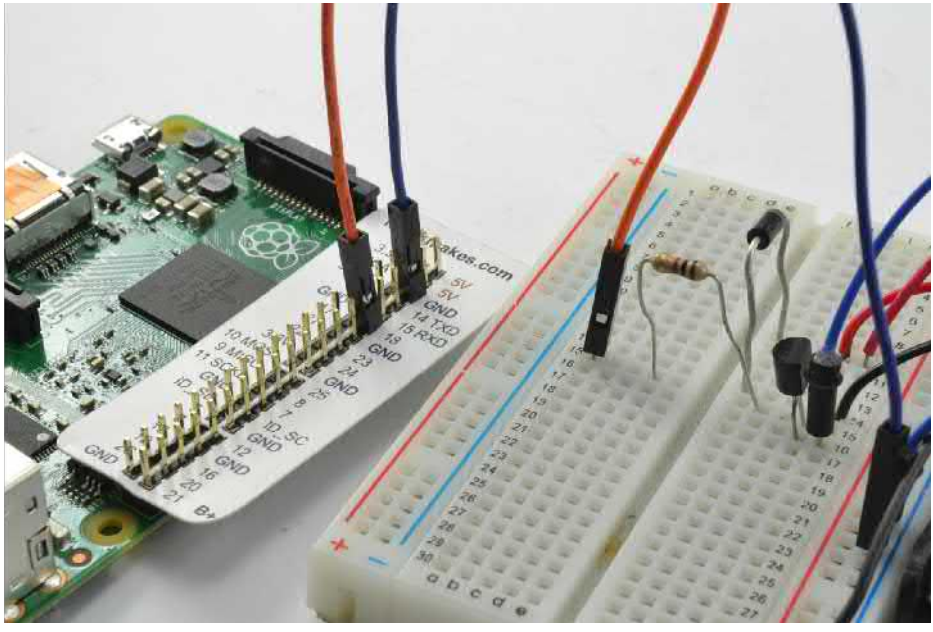


Рис. 4.4. Подключение Raspberry Pi к макетной плате

Идентификация контактов GPIO на Raspberry Pi

Контакты GPIO на плате Raspberry Pi никак не обозначены. Поэтому, чтобы вам не приходилось тщательно пересчитывать штырьки, обзаведитесь GPIO-трафаретом, накладываемым на контакты. Трафарет, показанный на рис. 4.4, называется Raspberry Leaf, его можно купить в Adafruit и на сайте **MonkMakes.com** (см. *приложение 1*). Есть там в наличии и другие трафареты.

Скачивание программ

Все программы к этой книге — и скетчи (так называются программы для Arduino), и программы для Raspberry Pi, написанные на языке Python, доступны в репозитории GitHub к этой книге (**github.com/simonmonk/make_action**).

Инструкции по переносу скетчей Arduino с обычного компьютера на плату Arduino приведены в *разд. «Код к книге» главы 2*.

Чтобы закачать на Raspberry Pi рассматриваемые в книге программы на Python, следуйте инструкциям из *разд. «Код к книге» главы 3*.

Эксперимент: управление светодиодом

В сообществе Arduino сложилась традиция, следуя которой, в качестве первого эксперимента мы заставим мигать светодиод. Прежде всего мы это сделаем на Arduino, а затем — на Raspberry Pi.

Для начала работы — это приятный и простой проект. Нам потребуется подключить к макетной плате всего два компонента: светодиод и резистор. Резистор здесь упомянут потому, что он требуется всем светодиодам для ограничения проходящего через них тока. Более подробно об этом рассказано в *главе 6*, а сами резисторы рассмотрены в *главе 5*.

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 4.1).

Таблица 4.1. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению светодиодом*

Компонент схемы	Источники
Красный светодиод	Adafruit: 297 Sparkfun: COM-09590
Резистор 470 Ом 0,25 Вт	Mouser. 291-470-RC
400-точечная беспаечная макетная плата	Adafruit: 64
Перемычки «папа-папа» (только для Arduino)	Adafruit 758
Перемычки «мама-папа» (только для Pi)	Adafruit: 826

Примечание

В графе **Источники** указаны поставщики и коды товара для каждой детали. Подробнее обо всех комплектующих, используемых в этой книге, и об их возможных поставщиках, включая и российские, рассказано в *приложении 1*.

Компоновка макетной платы

Компоновка макетной платы для нашего проекта показана на рис. 4.5. Она одинакова, работаете ли вы с Arduino или с Raspberry Pi, но подключение макетной платы к Arduino и к Raspberry Pi происходит по-разному.

На рис. 4.5 показано, что ток с выходного контакта Arduino или Raspberry Pi должен поступать сначала на резистор, а затем на светодиод.

Независимо от того, каким образом подключен резистор, плюсовой вывод светодиода (он чуть длиннее минусового) должен быть подключен к верхнему из двух задействованных рядов макетной платы. Определить минусовой вывод светодиода вам поможет и то обстоятельство, что край светодиода, расположенный ближе к минусовому выводу, имеет плоскую кромку.

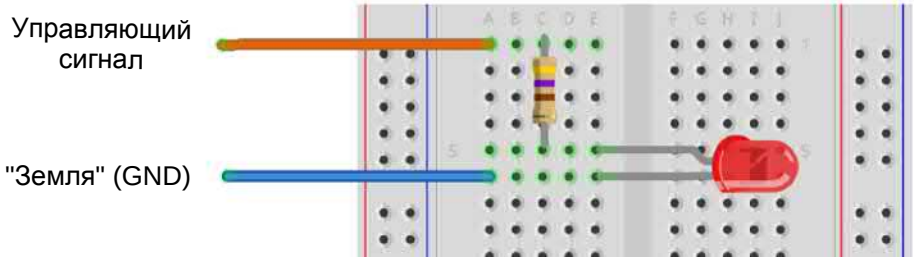


Рис. 4.5. Компоновка макетной платы для управления светодиодом

Экспериментируем с Arduino

Подключение Arduino

Подключите разъемы Arduino **GND** и **D9**, как показано на рис. 4.6. Соединение платы Arduino с платой для макетирования показано на рис. 4.7.

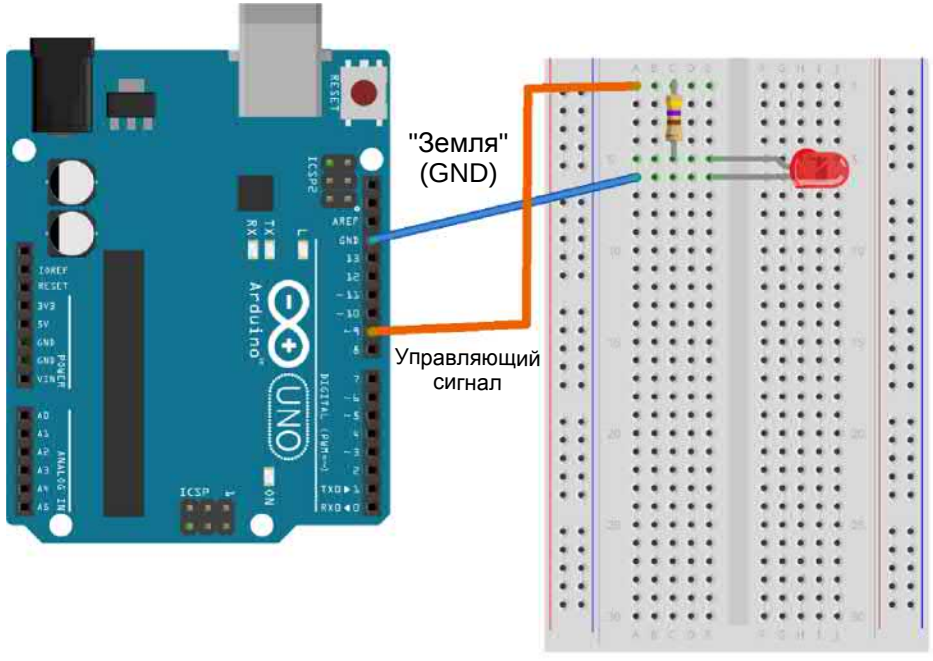


Рис. 4.6. Компоновка макетной платы для управления светодиодом с Arduino

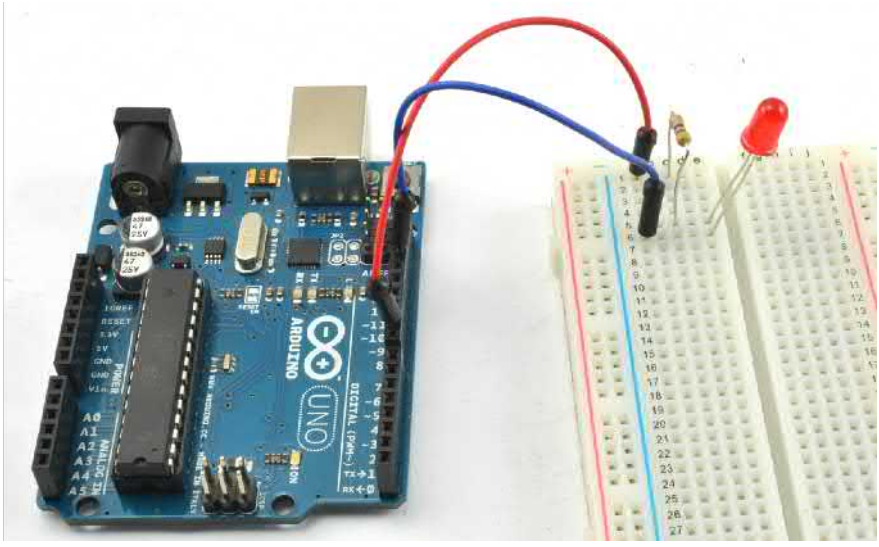


Рис. 4.7. Подключение макетной платы для управления светодиодом с Arduino

Программа для Arduino

Скетч Arduino для этого эксперимента находится в каталоге arduino/experiments/on_off_control (см. *разд. «Скачивание программ»* в этой главе ранее).

Программа включает светодиод на 5 секунд, потом отключает на 2 секунды, затем весь процесс повторяется. Вот ее полный код:

```
const int controlPin = 9; // 1

void setup() { // 2
   pinMode(controlPin, OUTPUT);
}

void loop() { // 3
   digitalWrite(controlPin, HIGH);
   delay(5000);
   digitalWrite(controlPin, LOW);
   delay(2000);
}
```

1. В первой строке (она обозначена комментарием 1) определяется константа `controlPin` для контакта 9. Хотя в Arduino Uno имеются как цифровые контакты от 0 до 13, так и аналоговые от 0 до 5, существует соглашение, что когда код Arduino ссылается просто на число как таковое (в нашем случае: 9), это относится к цифровому контакту. Если вы хотите обратиться к одному из шести аналоговых контактов, то должны поставить перед номером контакта букву `A`.
2. Во второй строке (комментарий 2) функция `setup()` определяет этот контакт как цифровой вывод, использующий функцию `pinMode`.

3. В блоке, помеченном комментарием 3, бесконечная функция `loop()` сначала задает для `controlPin` (контакт 9) значение `HIGH` (5 В), чтобы включить светодиод. Затем она делает задержку в `5000` мс (5 секунд) и задает для `controlPin` значение `LOW` (0 В), чтобы погасить светодиод. В следующей строке мы делаем задержку еще на 2 секунды, после чего цикл запускается снова.

Загружаем и выполняем программу

Загрузите программу на Arduino. Как только плата Arduino в процессе загрузки перезапустится, на ней станет выполняться ваш код, и светодиод замигает.

Если светодиод не мигает, то проверьте все соединения и убедитесь, что светодиод установлен правильно (более длинный вывод светодиода должен идти к верхнему из двух задействованных рядов макетной платы).

Попробуйте поменять числа в функциях `delay`, чтобы варьировать длительность работы светодиода в каждом цикле. Учтите, что после каждого изменения программы ее потребуется загружать заново.

Экспериментируем с Raspberry Pi

Подключение Raspberry Pi

В отличие от Arduino, контакты GPIO на Raspberry Pi никак не помечены, и сходу определить, где какой находится, трудно. Поэтому есть два варианта: либо использовать схему расположения контактов GPIO (см. *приложение 2*) и посчитать, какой

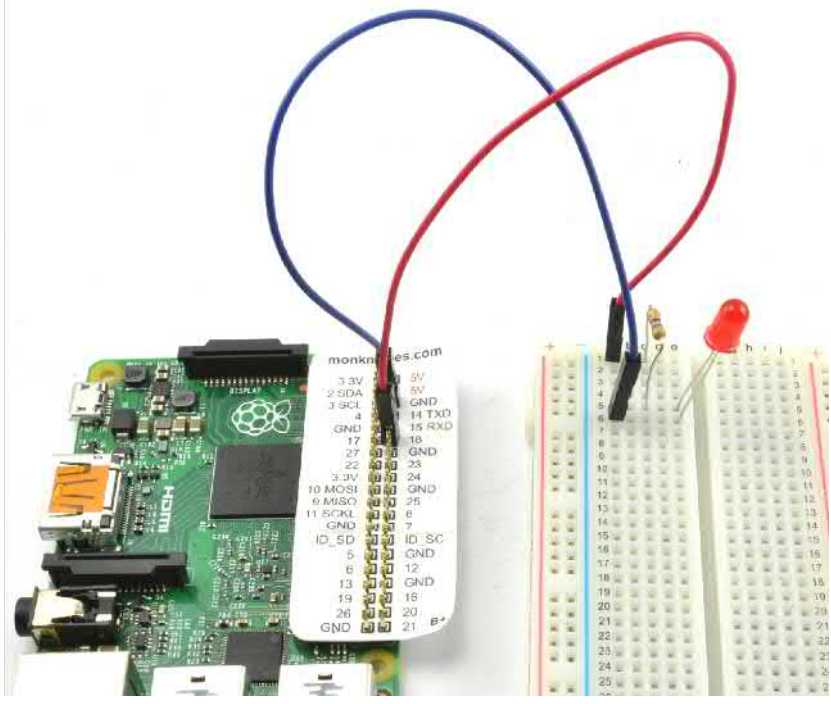


Рис. 4.8. Подключение макетной платы для управления светодиодом с Raspberry Pi

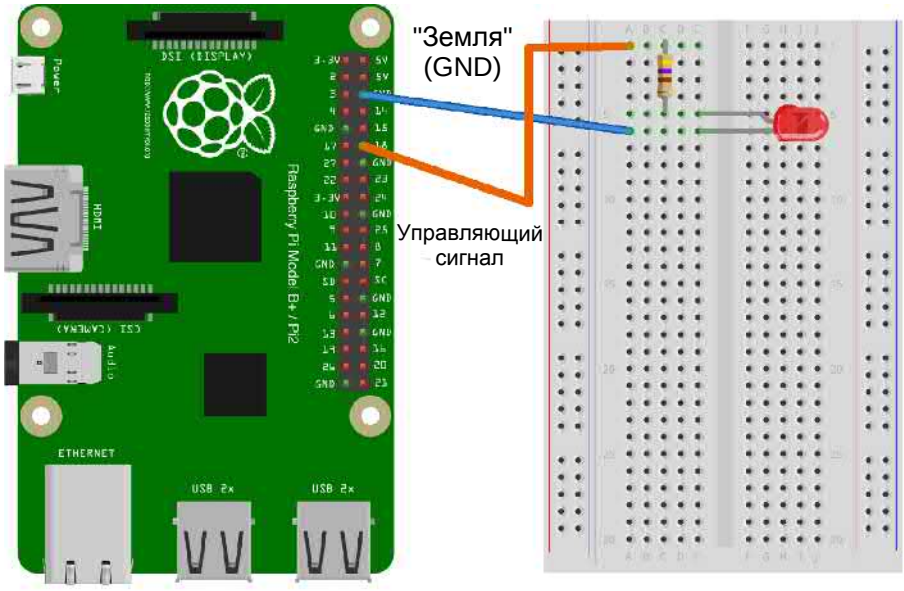


Рис. 4.9. Компоновка макетной платы для управления светодиодом с Raspberry Pi

контакт вам нужен, либо работать с трафаретом для разметки контактов, накладываемым на GPIO, — например, с Raspberry Leaf, как показано на рис. 4.8. Компоновка макетной платы и соединения ее с Raspberry Pi показаны на рис. 4.9.

Программа для Raspberry Pi

Для программирования Raspberry Pi не нужен отдельный компьютер — программу можно записать и запустить на самом Raspberry Pi. Итак, загрузим на Raspberry Pi программу, которая находится в файле on_off_control.py каталога python/experiments:

```
import RPi.GPIO as GPIO // 1
import time // 2

GPIO.setmode(GPIO.BCM) // 3

control_pin = 18 // 4
GPIO.setup(control_pin, GPIO.OUT)

try: // 5
   while True: // 6
      GPIO.output(control_pin, False) // 7
      time.sleep(5)
      GPIO.output(control_pin, True)
      time.sleep(2)

finally:
   print("Сброс")
   GPIO.cleanup()
```

Эта программа представляет собой практически двойника программы для Arduino. Рассмотрим ее также по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Чтобы получить доступ к контактам GPIO Raspberry Pi, вам понадобится библиотека RPi.GPIO на языке Python, которую написал энтузиаст Raspberry Pi Бен Кростон (Ben Croston). Так что, в первой строке кода мы эту библиотеку импортируем. Библиотека RPi.GPIO предустановлена во всех последних версиях стандартного дистрибутива Raspbian, поэтому вам не придется ее устанавливать, если только вы не работаете со старой версией Raspbian. В случае же старой версии удобнее всего установить библиотеку, просто обновив систему, что вы, скорее всего, так и сделаете. Для этого нужно выполнить в консоли следующую команду:

```
$ sudo apt-get upgrade
```

2. Вам также потребуется импортировать библиотеку времени, поскольку именно она обеспечивает задержки при включении и выключении светодиода.

3. Строку `GPIO.setmode(GPIO.BCM)` следует включать в любую программу на Python, которую вы пишете для управления контактами GPIO, прежде чем задавать режим контактов и вообще как-либо их использовать. Команда сообщает библиотеке GPIO, что контакты будут идентифицироваться по их Broadcom-наименованиям (BCM), а не по положению. Библиотека RPi.GPIO поддерживает обе схемы именования, но именование по Broadcom более популярно, и именно им мы будем пользоваться в этой книге.

   Примечание

   Обратите внимание, что при программировании под Raspberry Pi не задействуются отдельные функции `setup()` и `loop()`, как мы это делали при программировании под Arduino. Определения, включаемые в Arduino в функцию `setup()`, просто прописываются здесь ближе к началу программы, а бесконечный цикл `while` играет тут роль, которую в Arduino обычно поручается функции `loop()`.

4. Переменная `control_pin` идентифицирует GPIO-контакт `18` как именно тот, к которому вы собираетесь подключать светодиод. Он и определяется в качестве вывода при помощи команды `GPIO.setup`.

5. Переходим к функции, которая эквивалентна циклу Arduino. Она заключена в конструкции `try...finally`. Суть ее в том, что если в программе возникнет какая-либо ошибка, то вы либо просто остановите программу, нажав комбинацию клавиш <Ctrl>+<C> в окне терминала, где она выполнялась, либо запустится код очистки, содержащийся в блоке `finally`.

   Можно обойтись без этого кода и попросту использовать цикл `while`, но код очистки автоматически сбрасывает все GPIO-контакты в безопасное состояние (на вход), снижая вероятность того, что случайное короткое замыкание или ошибка монтажа на макетной плате повредит Raspberry Pi.

6. В цикле `while` действует условие `True`. Это может показаться странным, но в Python мы именно так обеспечиваем бесконечное выполнение какого-либо

кода. На самом деле, программа просто продолжит циклически выполнять команды в цикле `while`, пока вы не остановите ее комбинацией клавиш <Ctrl>+<C> либо не отключите Raspberry Pi.

7. Код внутри цикла очень похож на его аналог из Arduino. Контакт GPIO имеет значение `True` (high), затем следует задержка 5 секунд, после чего GPIO-контакт получает значение `False` (low), происходит еще одна 2-секундная задержка, и цикл повторяется.

Загружаем и выполняем программу

Для доступа к контактам GPIO в Linux требуются привилегии суперпользователя. Поэтому, чтобы выполнить программу, перейдите в каталог с файлом on_off_control.py и запустите ее следующей командой:

```
$ sudo python on_off_control.py
```

Напомню, что, начиная команду с `sudo`, вы запускаете программу от имени суперпользователя.

Когда светодиод достаточное количество раз помигает, нажмите комбинацию клавиш <Ctrl>+<C> для завершения программы.

Сравнение кода

Общая структура кода на Arduino C и на Python весьма похожа, но стиль кода заметно отличается. Кроме того, при именовании переменных или функций в C используется соглашение, при котором каждое слово после первого начинается с заглавной буквы, а в Python используется «змеиный» стиль snake_case, при котором слова разделяются нижними подчеркиваниями.

В табл. 4.2 приведены ключевые отличия между двумя программами.

Таблица 4.2. *Arduino C и Python*

Команда	Код Arduino C	Код Python
Определяем константу для контакта	`const int controlPin = 9;`	`control_pin = 18`
Задаем контакт в качестве вывода	`pinMode(controlPin, OUTPUT)`	`GPIO.setup(control_pin, GPIO.OUT)`
Задаем для вывода значение high	`digitalWrite(controlPin, HIGH);`	`GPIO.output(control_pin, True)`
Задаем для вывода значение low	`digitalWrite(controlPin, LOW);`	`GPIO.output(control_pin, False)`
Делаем задержку на 1 секунду	`delay(1000);`	`time.sleep(1);`

Эксперимент: управление электродвигателем

Итак, теперь вы знаете, как при помощи Arduino и Raspberry Pi включать и выключать светодиодный индикатор. Давайте, вооружившись этими знаниями, попробуем запускать и останавливать электродвигатель. Для этого мы воспользуемся теми же программами, что описаны ранее в *разд. «Эксперимент: управление светодиодом»* этой же главы, но для управления двигателем постоянного тока применим переключатель.

Гораздо подробнее мы рассмотрим двигатели постоянного тока в *главе 7*. Сейчас же нам достаточно знать, что мы используем в этом эксперименте небольшие моторы, наподобие тех, которые устанавливают, например, в ручном вентиляторе или в игрушечной машинке. Эти двигатели наиболее просты в использовании: через два имеющихся у них контакта на них подается напряжение, и вал начинает вращаться.

Поскольку почти всем двигателям требуется ток, более сильный, чем могут выдержать цифровые выводы Raspberry Pi или Arduino, мы не можем запитать двигатели непосредственно с них, поэтому в схему включается транзистор, позволяющий малому току с Raspberry Pi или Arduino управлять гораздо более сильным током, питающим электродвигатель.

При работе и с Raspberry Pi, и с Arduino используется одно и то же электронное оборудование, которое в обоих случаях устанавливается на одинаковых беспаечных макетных платах.

Эта глава представляет собой лишь первое знакомство с вопросами управления нашими платами, поэтому некоторые детали проводимых экспериментов станут вам более ясны лишь в последующих главах. Имейте также в виду, что на макетных платах будет установлено гораздо больше компонентов, чем в первом эксперименте, поэтому убедитесь, что все их выводы вставлены в нужные отверстия, а также исправны и правильно установлены сами компоненты.

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 4.3).

Таблица 4.3. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению электродвигателем*

Компонент схемы	Источники
Составной транзистор (пара Дарлингтона) MPSA14	Mouser: 833-MPSA14-AP
Диод 1N4001	Adafruit: 755 Sparkfun: COM-08589 Mouser: 512-1N4001
Резистор 470 Ом 0,25 Вт	Mouser: 291-470-RC

Таблица 4.3 (окончание)

Компонент схемы	Источники
Небольшой двигатель постоянного тока с напряжением питания 6 В	Adafruit: 711
Батарейный отсек 4 AA (6 В)	Adafruit: 830
400-точечная беспаечная макетная плата	Adafruit: 64
Перемычки «папа-папа»	Adafruit: 758
Перемычки «мама-папа» (только для Pi)	Adafruit: 826

Напомню, что перемычки «мама-папа» понадобятся только для подключения к макетной плате контактов GPIO Raspberry Pi (если вы планируете провести этот эксперимент с Raspberry Pi тоже).

Компоновка макетной платы

Компоновка макетной платы для этого проекта показана на рис. 4.10. После установки на макетную плату всех компонентов нужно убедиться, что транзистор расположен правильно, — его плоская сторона с надписью должна быть ориентирована вправо. Кроме того, надо проверить и расположение диода — на одной стороне у него имеется полоска, и эта сторона должна быть направлена к верхней части платы.

Поскольку экспериментировать зачастую интереснее, чем разбираться в деталях функционирования, мы отложим подробное обсуждение этого эксперимента до *главы 5*.

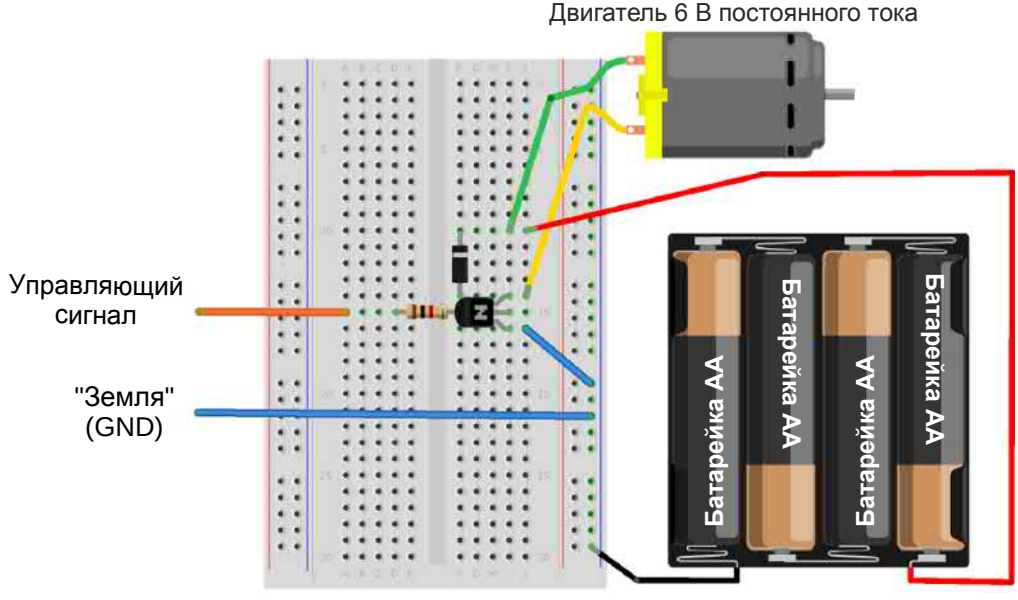


Рис. 4.10. Компоновка макетной платы для управления двигателем

Эксперименты без Arduino или Raspberry Pi

Прежде чем подключать макетную плату к Arduino или Raspberry Pi, можно протестировать ее и поэкспериментировать с ней и без такого подключения.

Транзистор действует здесь в качестве переключателя (подробнее об этом рассказано в *главе 5*). Соответственно, к плате (Arduino или Raspberry Pi) будут подведены два провода: управляющий и заземление. *Заземление* (GND) соответствует нулю вольт, как для схемы макетной платы, так и для плат Arduino и Raspberry Pi. *Управляющее подключение* запускает двигатель, когда он подключен к любому источнику тока с напряжением более 2 В. Когда напряжение окажется ниже этой величины, двигатель выключится.

Потренируйтесь в работе с перемычкой «папа-папа» до того, как приступать к использованию Arduino или Raspberry Pi. Вставьте один ее конец в отверстие того же ряда, куда вставлен левый выход резистора, а другой конец приложите к верхнему контакту диода, который подключен к плюсовому контакту батарейки (рис. 4.11), — двигатель запустится, но как только вы отведете штырек перемычки от диодного вывода, двигатель остановится.

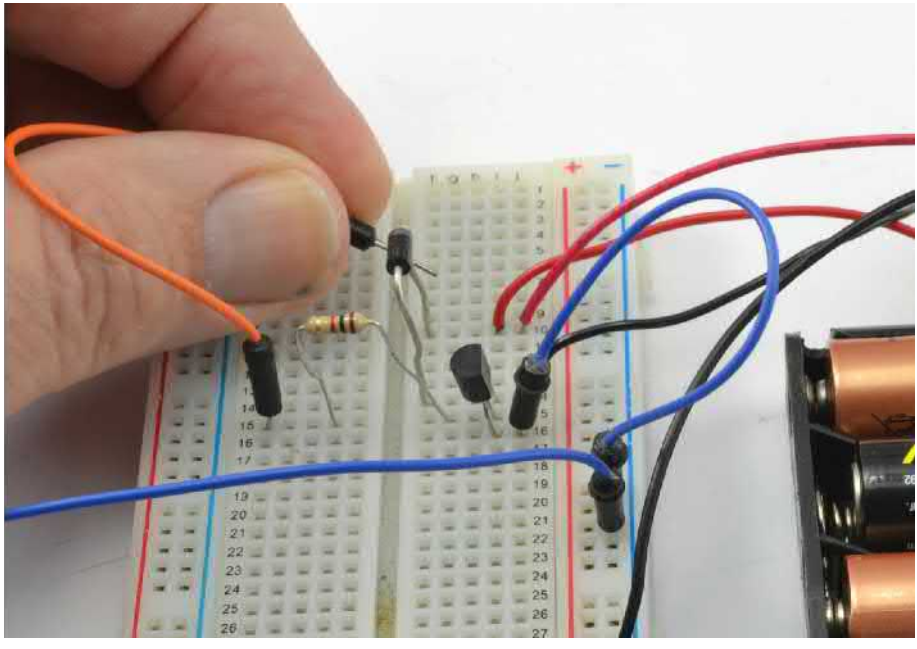


Рис. 4.11. Проверка схемы перед тем, как подключать к ней Arduino или Raspberry Pi

Подключение Arduino

Теперь, когда вы убедились, что управляющий вывод с макетной платы действительно включает и выключает двигатель, можно подсоединить его к одному из GPIO-контактов Arduino при помощи перемычки «папа-папа». Воспользуйтесь для этого контактом под номером **9** (рис. 4.12). Обратите внимание: мы уже имели дело

с этим управляющим контактом ранее (см. *разд. «Эксперимент: управление светодиодом»*).

Кроме того, вам понадобится подключить другой вывод — заземлёние (GND), к выводу **GND** на плате Arduino. Это также показано на рис. 4.12.

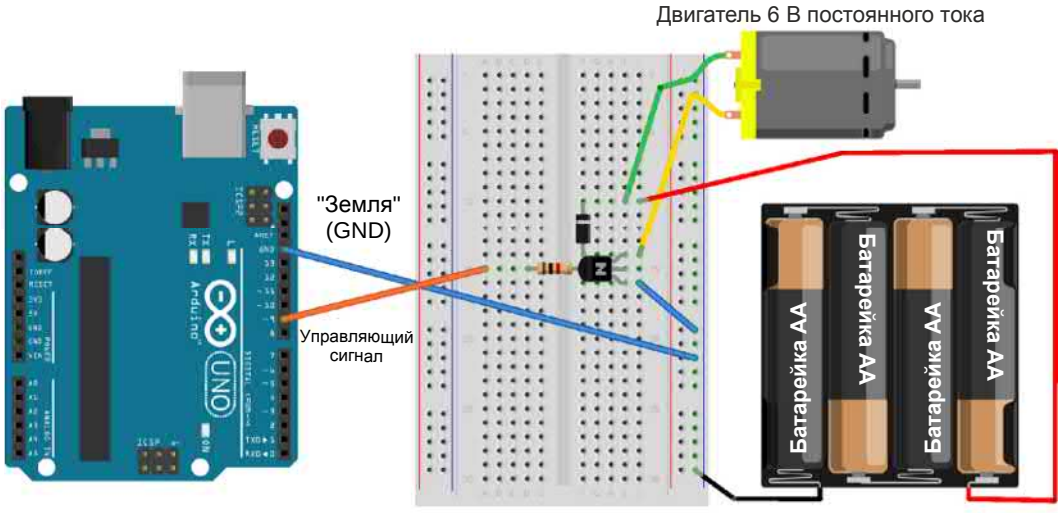


Рис. 4.12. Компоновка макетной платы для управления двигателем при помощи Arduino

Экспериментируем с Arduino

Если у вас на плату Arduino по-прежнему загружена программа из ранее рассмотренного *разд. «Эксперимент: управление светодиодом»*, то загружать больше ничего не надо. Если этой программы на Arduino уже нет, то вернитесь к тому разделу, чтобы вновь ее загрузить.

Как и в эксперименте со светодиодом, поиграйте с числами в функциях `delay`, чтобы изменить длительность работы двигателя в каждом цикле.

Подключение Raspberry Pi

Теперь можно отсоединить макетную плату от Arduino и подключить ее к Raspberry Pi. Подключите GPIO-контакт **18** (управляющий контакт) и **GND** (заземление), как показано на рис. 4.13.

Экспериментируем с Raspberry Pi

Программа для такого эксперимента на Raspberry Pi функционально аналогична программе для работы со светодиодом.

Чтобы выполнить программу, перейдите в каталог с файлом on_off_control.py и запустите ее следующей командой:

```
$ sudo python on_off_control.py
```

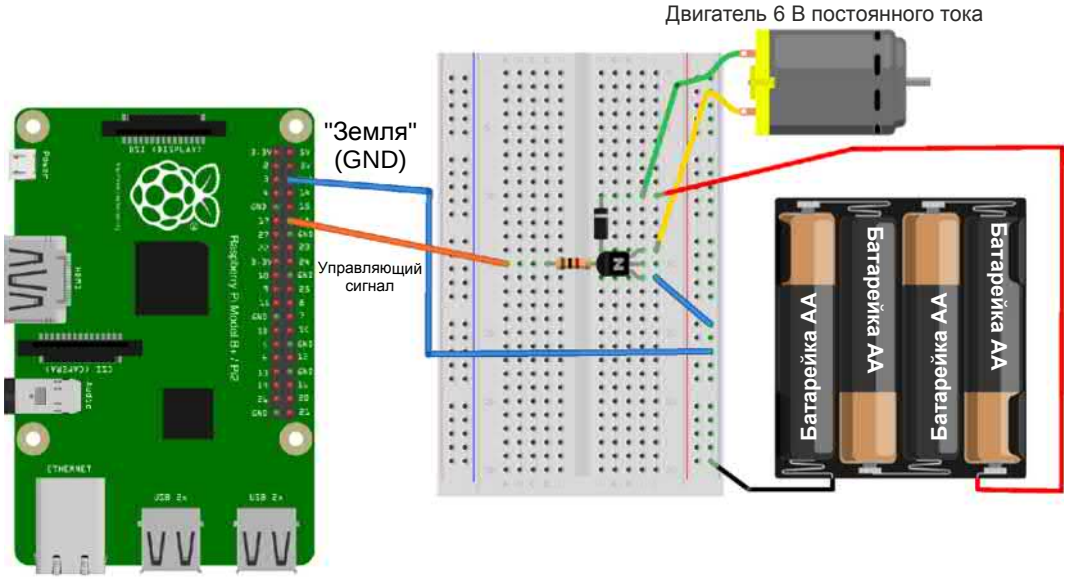


Рис. 4.13. Компоновка макетной платы для управления двигателем при помощи Raspberry Pi

После того как двигатель поработает достаточно долго, нажмите комбинацию клавиш <Ctrl>+<C> для завершения программы.

Заключение

В этой главе мы стремились приобрести первый практический опыт. В следующей главе будет подробнее рассмотрена теоретическая основа представленного здесь проекта. Мы также разберемся, как правильно подобрать транзистор для работы и как различать компоненты.

Основы электроники 5

В *разд. «Эксперимент: управление электродвигателем» главы 4* мы уже использовали транзистор для запуска двигателя, но без каких бы то ни было пояснений, как это работает. Если вы хотя бы немного разбираетесь в электронике, то, вполне возможно, имели дело с транзисторами и можете тогда пропустить большую часть этой главы. Если же вы в электронике новичок, приведенная здесь информация вам весьма пригодится.

Ток, напряжение и сопротивление

Разобраться с этими величинами нам поможет электрическая схема соединений из *разд. «Эксперимент: управление электродвигателем» главы 4*, которую мы собрали на макетной плате (рис. 5.1).

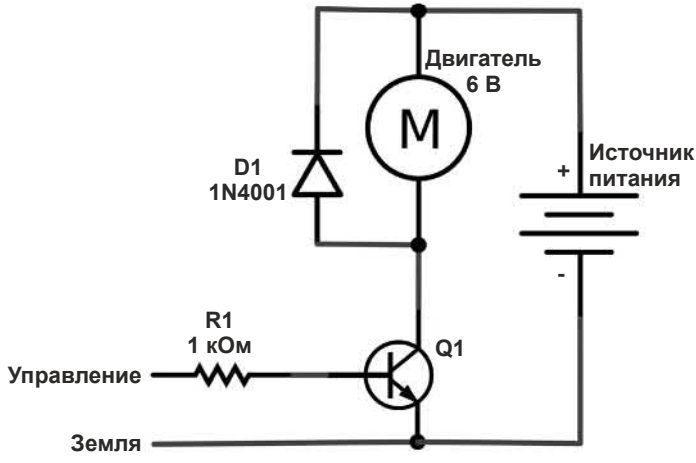


Рис. 5.1. Схема управления двигателем

Такое схематическое представление — просто еще один, более абстрактный, способ показать, что находится у нас на макетной плате. На схеме компоненты изображаются не так, как в реальности, а так, чтобы было понятно, как они работают.

Ток

Ломаная линия, изображающая резистор R1, подсказывает, что этот элемент ограничивает ток. В электронике *током* называется поток электронов, идущий через провода или элементы схемы. Можно сказать, что электроны движутся из одной части вашей схемы в другую. Например, ток может выходить с GPIO-вывода Arduino или Raspberry Pi и поступать на резистор R1. Пройдя через резистор R1, он попадает на центральное соединение (*базу*) транзистора Q1.

Малый ток, идущий через базу транзистора, позволяет управлять протеканием гораздо более сильного тока через два соединения транзистора, показанные справа: *коллектор* (сверху) и *эмиттер* (снизу). Именно за счет этого эффекта слабый ток с GPIO-контакта позволяет управлять сильным током, питающим, например, двигатель. Такую конструкцию удобно сравнить с цифровым переключателем, который можно включать и выключать при помощи малого тока.

Единица измерения силы тока называется *ампер* (сокращенно *А*). Поскольку ток с силой 1 А слишком мощный, чтобы работать с ним на Arduino или Raspberry Pi, чаще для удобства используется единица *миллиампер* (мА). При этом 1 мА = $= {}^{1}/_{1000}$ А.

Ограничивающий силу тока резистор R1 включается в нашу схему потому, что GPIO-контакты Arduino или Raspberry Pi недостаточно мощны, чтобы можно было управлять двигателем прямо с них. И если вы все же попытаетесь это сделать, то, скорее всего, повредите или вовсе выведете из строя Pi или Arduino. Как уже отмечалось ранее, Raspberry Pi может работать с током силой не более 16 мА, а Arduino — около 40 мА.

Напряжение

Вода всегда течет сверху вниз. Аналогично, ток в электрической схеме всегда поступает от компонентов, находящихся под более высоким *напряжением*, на компоненты, находящиеся под напряжением более низким. И если GPIO-контакт, от которого ток должен поступать на нашу схему (см. рис. 5.1) по линии **Управление**, находится под напряжением 0 В, никакого тока не потечет от него на резистор, транзистор и далее на заземление, т. к. напряжение и у GPIO-контакта, и у заземления будет одинаковым, — 0 В.

Однако, когда GPIO-контакт находится в высоком состоянии (high) — т. е. на него подано 3,3 В (Raspberry Pi) или 5 В (Arduino), ток от него пойдет на резистор и далее, через транзистор, в заземление.

Здесь важно отметить, что во всех точках, соединенных на схеме линией, напряжение одинаково.

Напряжение измеряется в *вольтах* (сокращенно *В*). И, как уже отмечалось ранее, GPIO-контакты Raspberry Pi, работающие на вывод, имеют напряжение 3,3 В (high) или 0 В (low), а на Arduino соответственно: 5 В и 0 В.

Заземление

Линия **«Земля»** в нижней части рис. 5.1 — это *заземление*. На подобных схемах она часто обозначается аббревиатурой GND (от англ. ground, земля). Величина напряжения на заземлении всегда равна 0 В и является базовым значением, от которого отмериваются все остальные показатели напряжения на схеме. Так, верхний положительный зажим батарейки обозначен **6 В**, поскольку его напряжение на 6 В выше, чем у выхода **«Земля»**.

При подключении друг к другу различных компонентов какого-либо проекта все заземления этих компонентов должны быть соединены между собой. В нашем случае, когда мы соберемся присоединить модуль управления двигателем к плате Arduino или Raspberry Pi, его заземляющая линия будет подключена к одному из контактов заземления (**GND**) этой платы.

Сопротивление

Каждый резистор обладает своим номинальным *сопротивлением*, которое измеряется в *омах* (*ом*), на схемах омы иногда обозначаются греческой буквой *омега* (Ω). Разброс значений сопротивления резисторов весьма велик, бывают такие, сопротивление которых составляет несколько килоом (тысяч ом), а иногда и мегаом (миллионов ом).

Резистор, включенный в схему, показанную на рис. 5.1, имеет значение 1 кОм, а определить, насколько можно ограничить ток таким резистором, позволяет *закон Ома*. Этот закон гласит, что сила тока, проходящего через резистор, равна разнице напряжения по обе стороны резистора (в вольтах), разделенной на сопротивление резистора (в омах). В случае Raspberry Pi максимальный перепад напряжения между GPIO-контактом и заземлением возможен в момент, когда GPIO-контакт находится в высоком (high) состоянии, т. е. значение напряжения на нем составляет 3,3 В. Отсюда максимальная возможная сила тока при этом:

$$3{,}3\ \text{В} / 1000\ \text{Ом} = 3{,}3\ \text{мА}.$$

Ток на GPIO-контактах Raspberry Pi

До сих пор нет общего мнения о том, с каким максимальным током может работать GPIO-контакт Raspberry Pi. Обычно я исхожу из значения, официально указанного изготовителем Raspberry Pi (т. е., 3 мА на 1 контакт GPIO). Значение 3 мА появилось потому, что на первой модели Raspberry Pi было всего 14 контактов, и его трехвольтовый регулятор напряжения позволял подать на GPIO-контакты всего 50 мА тока, что давало 3 мА на 1 контакт GPIO при условии задействования всех таких контактов платы.

Эти показатели уже не актуальны для новых моделей Raspberry Pi (A+, B+ и Pi 2), имеющих по 24 контакта, доступных для использования в качестве GPIO, а также обладающих трехвольтовым регулятором напряжения, который теоретически может подавать до 1 А. Однако даже если вы работаете с одной из этих новых моделей Raspberry Pi, то все равно не пытайтесь подавать 1 А / 24 = 41 мА на контакт GPIO, поскольку применяемая в Raspberry Pi однокристальная система (system on a chip, SoC) Broadcom также ограничивает предельный ток на один контакт GPIO величиной 16 мА.

Теоретически на Raspberry Pi A+, B+ или Pi 2 можно подавать до 16 мА на какое угодно количество GPIO-контактов, однако есть и другие факторы, от которых зависит, сколько тока можно подать на Pi без риска повредить устройство, — например, частота модуляции и общий ток на всех контактах GPIO, который может выдержать кристалл Broadcom.

Итак, нужно придерживаться следующих рекомендаций:

- на базовом Raspberry Pi можно использовать 16 мА на каждый контакт при общем токе на всех контактах 40 мА;
- на Raspberry Pi A+, B+ или Pi 2 можно без опаски подавать не более 16 мА на контакт и не более 100 мА на все задействованные контакты.

Показанный на рис. 5.1 резистор R1 защищает GPIO-контакт, ограничивая силу тока, который может быть получен с этого контакта, и если каждый цифровой выход с Raspberry Pi всегда будет защищен резистором сопротивлением 1 кОм, мы можем не беспокоиться за наш Pi. Кстати, довольно часто, особенно при работе со светодиодами, можно будет применять резистор и с меньшим значением, поскольку на некоторых элементах (тех же светодиодах) будет происходить частичное падение напряжения, что приведет к снижению силы тока.

Мощность

Когда ток проходит через резистор, тот нагревается. Электрическая энергия превращается в тепловую, и измеряется такое преобразование *мощностью* — количеством энергии, преобразуемой при этом за секунду. Единица мощности — *ватт* (Вт), и чтобы вычислить мощность, выделяемую компонентом при нагревании его протекающим через него током, нужно умножить напряжение, под которым он находится (в вольтах), на силу проходящего через него тока (в амперах).

Вернемся к резистору из рис. 5.1, имеющему сопротивление 1 кОм. Если он находится под напряжением 2,2 В, и через него идет ток силой 2,2 мА, то он будет выдавать мощность около 4,8 мВт. Это очень небольшая мощность. Однако транзисторы также генерируют теплоту, определяемую умножением силы проходящего через них тока на напряжение. И если бы вы работали с достаточно мощным двигателем — потребляющим, скажем, ток 800 мА, — то перепад напряжения между коллектором и эмиттером в целом на транзисторе составил бы около 1,2 В. Тогда мощность, преобразованная в теплоту, равнялась бы:

$$800 \text{ мА} \times 1{,}2 \text{ В} = 960 \text{ мВт}.$$

В этом случае транзистор бы серьезно разогрелся. А если транзистор перегреется, то что-нибудь в нем расплавится, и он выйдет из строя. Поэтому ограничение максимально допустимого значения силы тока существует не только для Arduino или Raspberry Pi, но и для транзистора. Сравнительно крупные по размерам транзисторы обычно рассчитаны на более сильный ток, чем небольшие, и этот фактор необходимо учитывать, подбирая транзистор для управления исполнительным механизмом.

Составной транзистор (пара Дарлингтона) MPSA14, использованный в нашем эксперименте по управлению электродвигателем (см. *разд. «Эксперимент: управление электродвигателем» главы 4*), выдерживает ток силой до 1 А.

Распространенные компоненты

Здесь мы рассмотрим некоторые компоненты, применяемые в этой книге, и объясним, как их использовать и как подбирать.

Резисторы

На вид резисторы — маленькие элементы, несущие на себе разноцветные полоски. Если вы хотите узнать значение сопротивления резистора, можно измерить его мультиметром, либо определить это значение по цветам полосок. Каждому цвету полоски соответствует число, указанное в табл. 5.1.

Таблица 5.1.
Цветовые обозначения резисторов

Цвет полоски	Число
Черный	0
Коричневый	1
Красный	2
Оранжевый	3
Желтый	4
Зеленый	5
Синий	6
Фиолетовый	7
Серый	8
Белый	9
Золотистый	$^1/_{10}$
Серебристый	$^1/_{100}$

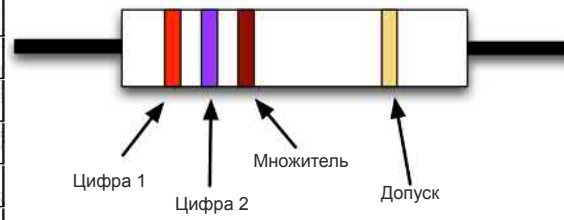


Рис. 5.2. Расшифровка цветовой маркировки резистора

Примечание

Золотистый и серебристый цвет указывают не только доли, соответственно, $^1/_{10}$ и $^1/_{100}$, но и точность резистора. Так, золотистый цвет означает ±5%, а серебристый — ±10%.

Как правило, на одном конце резистора присутствуют три такие полоски, далее идут пробел и еще одна полоска на другом конце резистора, котораая означает точность значения резистора.

На рис. 5.2 показано расположение цветных полос. Значение резистора определяется по трем первым полоскам: первая соответствует первой цифре, вторая — второй, а третья — «множитель», указывает, сколько нулей нужно поставить после первых двух цифр.

Исходя из сказанного, резистор, изображенный на рис. 5.2, имеет сопротивление 270 Ом: первая цифра — 2 (красная полоска), вторая — 7 (фиолетовая полоска), а

множитель — 1 (коричневая полоска), добавляющий 0 после цифр 27. Аналогично, на резисторе сопротивлением 1 кОм будут нанесены коричневая, черная и красная полосы (1, 0, 00).

Резисторы также различаются и по номинальной мощности. Практически все обычные резисторы того типа, что используются в схемах этой книги (для монтажа в сквозные отверстия), обладают мощностью 0,25 Вт. Другие распространенные значения номинальной мощности: 0,5 Вт, 1 Вт и 2 Вт, причем, чем выше номинальная мощность резистора, тем крупнее он сам.

Транзисторы

Как правило, любой поставщик комплектующих предлагает немыслимое количество самых разных *транзисторов*. Поэтому для использования в этой книге я, чтобы не усложнять поиск, подобрал всего четыре транзистора, на основе которых можно создавать практически все электронные схемы, способные управлять различными устройствами, подключенными к платам Arduino и Raspberry Pi.

Транзистор из *разд. «Эксперимент: управление электродвигателем» главы 4* — это как раз тот самый компонент, при помощи которого слабый ток силой несколько миллиампер позволяет управлять сотнями миллиампер, подаваемыми на двигатель. Хотя транзисторы могут использоваться и для других целей, в этой книге они послужат нам переключателями. Малый ток поступает на *базу* транзистора и далее на заземляющий вывод (GND) через *эмиттер* транзистора. Этот ток и будет переключать гораздо более сильный ток, поступающий с *коллектора* на *эмиттер*.

На рис. 5.3 показана подборка транзисторов различных типов, позволяющих работать с разными мощностями.

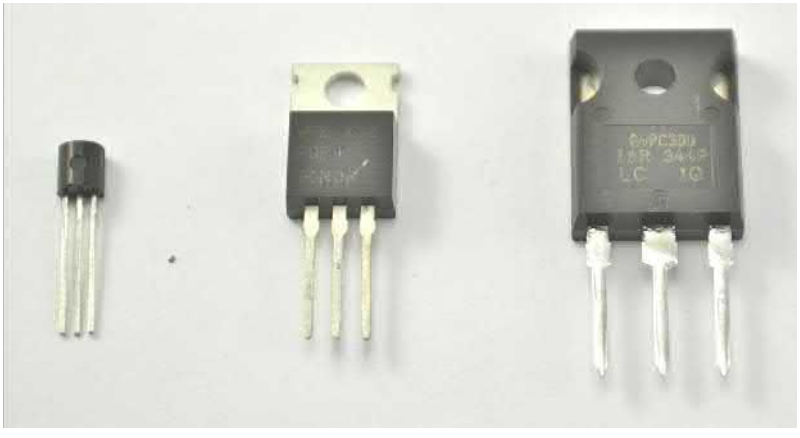


Рис. 5.3. Подборка транзисторов

Существует всего несколько разновидностей корпусов транзисторов. При этом по внешнему виду транзистора определить его свойства невозможно — нужно прочитать, что на нем написано.

Наиболее распространенные варианты корпусов: TO-92 (рис. 5.3, *слева*) и TO-220 (рис. 5.3, *в центре*). Иногда, для работы на очень сильных токах, могут использо-

ваться транзисторы с более крупным корпусом — как вариант TO-247, показанный на рис. 5.3, *справа.*

Модели TO-220 и TO-247 предназначены для монтажа на теплоотводах. Впрочем, если вы используете эти транзисторы при токах, которые значительно меньше указанного для них максимума, то устанавливать их на теплоотводах необязательно.

Биполярные транзисторы

Транзисторы изготавливаются по разным технологиям, каждой из которых присущи свои достоинства и свои недостатки. Поэтому транзистор может подойти вам в одной ситуации и не пригодиться в другой.

Начиная работать с транзисторами, вы, скорее всего, столкнетесь именно с *биполярной* моделью. Они практически не изменились со времен зарождения транзисторной отрасли. Преимущество таких транзисторов — в их дешевизне и в том, что их очень легко использовать при малых нагрузочных токах. Есть у них и недостаток — хотя малый ток, проходящий через базу к эмиттеру транзистора, позволить прохождение более сильного тока, идущего от коллектора к эмиттеру, ток коллектора ограничивается множителем тока базы, и этот множитель (называемый *коэффициентом усиления*, или *hFE*), как правило, находится в диапазоне от 50 до 200. Соответственно, если Raspberry Pi подает на базу ток силой 2 мА, то через коллектор может идти ток не более 100 мА. Эта величина гораздо меньше той, на которую вы, возможно, рассчитывали, — поскольку транзистор вполне может выдерживать и более сильный ток (скажем, 500 мА), но она никогда не достигнет такого предела — ведь тока на базе просто не хватит. Как правило, при работе с Arduino такая проблема не возникает, поскольку Arduino позволяет подавать на базу больший ток (до 40 мА), если вместо резистора сопротивлением 1 кОм (см. рис. 5.1) применить более слабый резистор. Так, если взять резистор со значением 150 Ом, то ток базы увеличится до:

$$I = V / R = (5 - 0{,}5) / 150 = 30 \text{ мА}.$$

И даже в худшем случае — если транзистор имеет коэффициент усиления 50 — ток базы силой 30 мА даст на коллекторе ток силой 1,5 А.

Тут надо пояснить: в только что рассмотренном примере напряжение рассчитывается как (5 – 0,5), поскольку напряжение между базой и эмиттером биполярного транзистора, когда транзистор включен, составляет около 0,5 В.

В этой книге мы будем работать всего с одной моделью биполярного транзистора — весьма распространенным 2N3904. Хотя в наличии есть биполярные транзисторы, выдерживающие более сильный ток, при увеличении силы тока лучше работать с более высокотехнологичными транзисторами.

Составные транзисторы

Когда вам требуется более значительное усиление — если, к примеру, вы управляете небольшим электродвигателем с платы Raspberry Pi, позволяющей подать на базу всего несколько миллиампер, то вместо обычного биполярного транзистора

удобно взять *составной транзистор* (так называемую *пару Дарлингтона*), который обычно имеет коэффициент усиления не менее 10 000.

Составной транзистор содержит два биполярных транзистора в одном корпусе (рис. 5.4), и именно благодаря такой двухкомпонентной структуре составные транзисторы получают столь высокий коэффициент усиления.

Рис. 5.4. Составной транзистор

Поскольку в составном транзисторе имеются два компонента с базой и эмиттером в каждом, при включенном транзисторе в каждой такой паре перепад напряжения составит как минимум 0,5 В, что дает общий перепад напряжения 1 В, а не 0,5 В, как в обычном биполярном транзисторе. На самом деле, этот перепад касается и напряжения коллектора и увеличивается с повышением нагрузочного тока. Таким образом, когда мы управляем током в 1 А, составной транзистор MPSA14 на практике способен дать нам всего 9 В, питая нагрузку в 12 В. В одних случаях это имеет значение, в других — нет.

Транзистор из *разд. «Эксперимент: управление электродвигателем» главы 4* — это составной транзистор MPSA14. Перепад напряжения на резисторе R1 при использовании Raspberry Pi на самом деле составит не 3,3 В, а 3,3 В – 1 В = 2,2 В. Соответственно, Raspberry Pi потребуется подавать ток 2,2 В / 1 кОм = 2,2 мА.

Кроме маломощного транзистора MPSA14, который удобен для управления нагрузками до 0,5 В, также рекомендую вам работать с более мощным составным транзистором TIP120, который является стандартным устройством и обязательно должен присутствовать в вашем ящике с комплектующими.

МОП-транзисторы

В принципе, биполярный транзистор представляет собой управляемое током устройство: малый базовый ток усиливается и превращается в большой ток коллектора. Однако существует еще один вид транзисторов — так называемые *полевые*, или *МОП-транзисторы* (сокращение от «металл-оксид-полупроводник»), требующие для переключения очень малого тока, но остающиеся во включенном состоянии, пока напряжение на их входном затворе превышает некоторое пороговое значение.

На рис. 5.5 схематически представлен такой транзистор — как вы можете догадаться, на этой схеме входной его компонент (затвор) электрически не соединен с остальными компонентами транзистора напрямую.

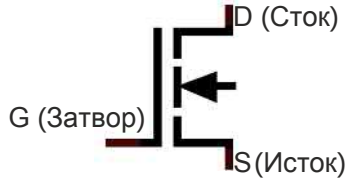


Рис. 5.5. Схема МОП-транзистора

Обратите внимание: в отличие от биполярного транзистора, компоненты которого назывались базой, коллектором и эмиттером, компоненты МОП-транзистора называются затвор (**G**, от англ. gate), сток (**D**, от англ. drain) и исток (**S**, от англ. source). Напрашивается мысль, что исток эквивалентен коллектору, но на самом деле коллектор из биполярного транзистора эквивалентен стоку полевого.

МОП-транзисторы при работе с Arduino или Raspberry Pi весьма целесообразно применять для включения/отключения устройств схемы, поскольку в таком случае можно обойтись минимальным током. Нужно просто убедиться, что напряжение на затворе транзистора выше его порогового значения. Пороговым значением затвора является такое, при котором МОП-транзистор включается, пропуская ток от стока к истоку. На рис. 5.6 показано, как подключить МОП-транзистор для управления нагрузкой. Как можно видеть, на самом деле подключение точно такое же, что и в случае биполярного транзистора.

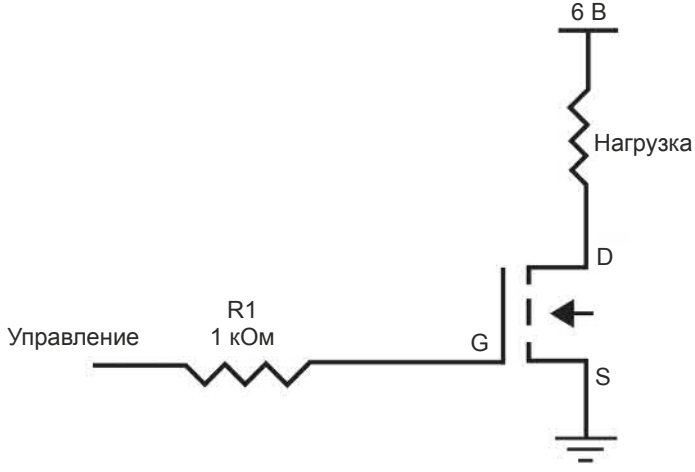


Рис. 5.6. Схема включения МОП-транзистора

В схеме, показанной на рис. 5.6, используются два символа, с которыми мы ранее не сталкивались. От истока транзистора (**S**) отходит линия, несущая три параллельные линии, которые постепенно укорачиваются. Это символ заземления — применяя его на схеме, мы избавляемся от лишних соединительных линий.

Второй символ расположен в верхней части схемы — это просто горизонтальная черта, которой отмечены **6 В**. Она означает, что напряжение в этой части схемы составит 6 В, что избавляет нас от необходимости рисовать батарею.

> ***Цоколевка транзисторов различается***
>
> Хотя в этой книге мы и стараемся использовать транзисторы со взаимно совместимыми *цоколевками* (расположением выводов), это правило не универсально. Не все транзисторы в схожих корпусах имеют одинаковую цоколевку, поэтому перед применением нового транзистора сверяйтесь с его паспортом.
>
> Цоколевки разнообразных устройств, используемых в книге, приведены в *приложении 1*.

Возможно, после рассмотрения рис. 5.6 у вас возникнет вопрос — а зачем нам по-прежнему нужен резистор R1, ведь затвор не принимает никакого тока? Дело в том, что иметь этот резистор все равно целесообразно, ведь стоит нам увеличить напряжение на затворе — и на долю секунды возникнет молниеносный бросок тока. Резистор гарантирует, что такой бросок не сожжет контакт GPIO.

Сложность работы с МОП-транзисторами в том, что иногда их пороговое значение слишком высоко, и его не удается переключить при помощи напряжения в 3,3 В с Raspberry Pi или 5 В с Arduino. МОП-транзисторы, чье пороговое значение затвора достаточно низкое, чтобы их можно было переключать непосредственно с GPIO-контактов наших плат, называются *МОП-транзисторами с логическим уровнем входа* (logic-level). Для этой книги выбраны два стандартных МОП-транзистора: на низких мощностях используется 2N7000, а на сравнительно высоких — FQP30N06L. Пороговое значение на затворе у обоих гарантированно не превышает 3 В, поэтому их можно использовать как с Arduino, так и с Raspberry Pi.

В целом МОП-транзисторы при переключении нагрузок нагреваются не так сильно, как биполярные. Один из основных параметров МОП-транзисторов, на который нужно обращать внимание при покупке компонента, называется *сопротивлением открытого канала* (on resistance). МОП-транзисторы с очень низким сопротивлением открытого канала будут переключать большие токи, даже не нагреваясь. Как вы догадываетесь, чем ниже сопротивление открытого канала у МОП-транзистора, тем он дороже.

В этой книге мы будем достаточно много работать с МОП-транзисторами. В основном я буду рассказывать о FQP30N06L (токи до 30 А), но для работы на таких токах вам понадобится большой теплоотвод.

Кстати, коллектор, база и эмиттер составного транзистора TIP120 и исток, затвор и сток МОП-транзистора FQP30N06L расположены одинаково, поэтому вы можете просто извлечь из макетной платы TIP120 и поставить на его место FQP30N06L не меняя конфигурации — при этом схема все равно должна работать.

PNP-транзисторы и транзисторы с p-каналом

Транзисторы каждого из всех ранее описанных типов на самом деле существуют в двух разновидностях. До сих пор мы рассматривали всего одну их разновидность: отрицательно-положительно-отрицательные транзисторы (NPN, от англ. negative-

positive-negative), они же транзисторы с n-каналом (в случае МОП-транзисторов) Такие транзисторы — наиболее распространенные, и обычно ими вполне можно обойтись.

Транзисторы другого вида — это положительно-отрицательно-положительные (PNP, от англ. positive-negative-positive) устройства, они же транзисторы с p-каналом. Если устройства N-типа применяются для переключения нагрузки на заземление, то устройства P-типа переключают нагрузку на источник положительного питания. МОП-транзисторы с p-каналом используются в мостовых схемах управления двигателем, рассматриваемых в *главе 8*.

Как подбирать транзистор?

Подобрать правильный транзистор порой бывает непросто. С помощью табл. 5.2 проблема выбора у вас сведена всего к пяти транзисторам.

При работе с Raspberry Pi имеется в виду, что между GPIO-контактом и базой или затвором транзистора стоит резистор сопротивлением 1 кОм. В случае с Arduino предполагается, что сопротивление такого резистора будет 150 Ом. Приведенные значения получены по результатам тестирования реальных устройств, а максимальные значения напряжения взяты из спецификаций изделий.

Таблица 5.2. *Полезный набор транзисторов*

Транзистор	Тип	Вариант корпуса	Макс. ток (Pi = 3,3 В)	Макс. ток (Arduino = 5 В)	Макс. напряжение
2N3904	Биполярный	TO-92	100 мА	200 мА	40 В
2N7000	МОП	TO-92			60 В
MPSA14	Составной	TO-92	1 А	1 А	30 В
TIP120	Составной	TO-220	5 А	5 А	60 В
FQP30N06L	МОП	TO-220	30 А	30 А	60 В

Приобретая транзистор FQP30N06L, убедитесь, что это МОП-транзистор версии L (логический) — название модели оканчивается именно на L, в противном случае пороговое напряжение затвора может быть слишком высоким.

Транзистор MPSA14 является практически универсальным для работы с токами до 1 А, хотя при таких токах перепад напряжения составляет почти 3 В, и транзистор разогревается до 120 °С! При токе 500 мА перепад напряжения уже не такой страшный — всего 1,8 В, и температура транзистора 60 °С.

Итак, если вам требуется переключать токи всего лишь около 100 мА, то вам вполне хватит транзистора 2N3904. Если нужен 1 А, используйте транзистор MPSA14. При более сильных токах наилучший вариант, пожалуй, это транзистор FQP30N06L, если, конечно, цена вас не смущает, поскольку транзистор TIP120 существенно дешевле.

Диоды

Диод D1, включенный в схему, показанную на рис. 5.1, нужен для защиты плат Raspberry Pi или Arduino и транзистора.

Дело в том, что электродвигатели создают скачки напряжения и всевозможные электрические помехи, которые могут устроить настоящий хаос в такой тонкой электронике, как Raspberry Pi или Arduino. Диод гарантирует, что всплески тока, спровоцированные двигателем, не приведут к мгновенному изменению направления тока, что может сразу сжечь транзистор. Суть в том, что диод пропускает ток только в одном направлении — в этом он похож на обратный клапан. Ток может идти через диод лишь в том направлении, куда указывает сам диод, по своей форме напоминающий стрелку.

Из-за таких свойств диодов их довольно часто ставят на выводах двигателя. Обычно ток идет через двигатель в направлении, обратном тому, которое допускает диод, но если случится отрицательный скачок напряжения, то диод вступает в дело, проводит ток и обнуляет его, фактически гася тем самым короткое замыкание.

Светодиоды

Гораздо подробнее мы обсудим *светодиоды* (LED, от англ. Light Emitting Diode, светоизлучающий диод) в *главе 6*.

Как вы уже догадываетесь, светодиод работает как самый обычный диод, однако когда через него проходит ток, он излучает свет. Схемный символ светодиода точно такой же, что и у обычного диода, но добавленные справа стрелки означают световое излучение (рис. 5.7).

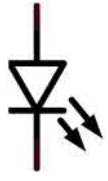


Рис. 5.7. Символ диода

Светодиоды бывают самых разных цветов и размеров. Ими можно управлять непосредственно с GPIO-контакта Arduino или Raspberry Pi, однако, как и при работе с транзистором, необходимо использовать резистор для ограничения силы тока. О том, как это делается, рассказано в *главе 6*.

Конденсаторы

Можно сказать, что *конденсатор* предназначен для временного запасания электричества — примерно, как очень малоемкие батарейки, сохраняющие небольшой резервный заряд. Конденсаторы мы станем использовать в проектах книги в различных качествах — с их помощью мы будем подавлять электрические помехи, а также запасать небольшие резервы электроэнергии, чтобы при резкой необходимости добавить энергию, ее можно было взять с конденсатора.

На рис. 5.8 показаны символы конденсаторов. Если конденсатор имеет очень высокое значение емкости, то он обычно поляризуется. Малоемкие конденсаторы не имеют положительного и отрицательного полюса.

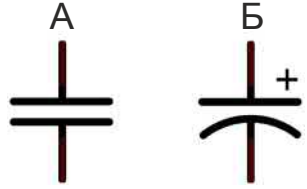


Рис. 5.8. Символы конденсатора неполяризованный (А) и поляризованный (Б)

Иногда встречаются немного иные, но тем не менее узнаваемые символы конденсатора: в них положительный полюс поляризованного конденсатора обозначается пустой рамкой, а отрицательный — закрашенной. В этой книге конденсаторы обозначаются символикой, принятой в США, как и показано на рис. 5.8.

Интегральные схемы

Интегральные схемы (ИС), часто именуемые просто ***чипами*** или *микросхемами*, состоят из множества транзисторов, диодов и других электронных компонентов, размещенных на кремниевом кристалле и соединенных между собой в сложные схемы. На печатных платах и Raspberry Pi, и Arduino расположено множество таких микросхем и прочих электронных элементов.

Существуют интегральные микросхемы специального назначения, предназначенные для работы в любых типах электронных устройств, с которыми бы вам только захотелось иметь дело. Но в этой книге нам особенно важны микросхемы, которые позволяют управлять электронными устройствами. В корпусе таких микросхем часто объединяются сильноточные транзисторы и логические схемы управления.

Подробнее о соединениях

Рассмотрев основы электроники, нужно хотя бы кратко упомянуть и о том, как те или иные электронные устройства подключаются к Raspberry Pi или Arduino. Ранее, в *главах 2* и *3*, эти вопросы были рассмотрены весьма подробно, особенно все, что связано с программированием, так что, если вам нужны какие-либо уточнения, вы можете вернуться к этим главам.

Цифровые выходы

Как было показано в *главе 4*, *цифровые выходы* нужны для включения и выключения устройств. Если вы работаете с Raspberry Pi, то напряжение на цифровом выходе будет равно либо 0 В (low), либо 3,3 В (high). В Arduino высокое напряжение составляет 5, а не 3,3 В, но принцип аналогичен — на цифровых выходах не может быть каких-либо иных значений напряжения, кроме high и low.

Еще одно различие между Arduino и Raspberry Pi заключается в том, что Arduino может работать под более сильным током (40 мА, а не 16 мА).

Цифровые входы

Цифровые входы часто подсоединяются к переключателям или цифровым выходам других устройств. Пороговое напряжение цифрового входа обычно равно среднему значению в диапазоне между high и low, поэтому у Arduino этот порог составит около 2,5 В, а у Raspberry Pi — около 1,65 В. Когда программа, работающая на Arduino или Raspberry Pi, считывает значение на цифровом входе, она проверяет, выше или ниже порогового значения это напряжение: в первом случае вход считается high, во втором — low.

Как и цифровые выходы, цифровые входы не могут иметь никаких половинчатых значений, — они находятся в состоянии либо high, либо low.

О том, как работать с цифровыми входами на Arduino, рассказано в *разд. «Цифровые входы» главы 2*, а на Raspberry Pi — в *разд. «Цифровые входы» главы 3*.

Аналоговые входы

Аналоговые входы позволяют измерять на своих контактах напряжение, находящееся между значениями high и low. На Raspberry Pi аналоговых входов нет, а на Arduino их шесть, и они обозначены от A0 до A5.

В Arduino любое напряжение от 0 В до 5 В соответствует числовому значению в диапазоне от 0 до 1023. Так, 0 В соответствует 0, а 5 В — 1023. Среднее напряжение 2,5 В будет соответствовать показателю около 511.

Подробнее об аналоговых входах Arduino рассказано в *разд. «Аналоговые входы» главы 2*.

Аналоговые выходы

Хотя можно предположить, что аналоговый выход позволяет задать любое значение напряжения между low и high, но на самом деле ситуация несколько сложнее. Эти выводы используют так называемую *широтно-импульсную модуляцию* (ШИМ), позволяющую управлять средней мощностью, поступающей на обычный цифровой выход.

ШИМ служит для управления скоростью работы электродвигателей и яркостью светодиода. В *разд. «Управление скоростью (ШИМ)» главы 7* содержатся необходимые сведения о принципе действия ШИМ, и поясняется, как с ее помощью можно управлять скоростью вращения электродвигателя.

Соединения по последовательным интерфейсам

Ранее были описаны самые простые, низкоуровневые приемы подключения электронных устройств к платам Arduino или Raspberry Pi. Однако некоторые устройства, которыми, вы, вероятно, попробуете управлять с Arduino или Raspberry Pi, ис-

пользуют последовательные интерфейсы, побитово передающие двоичные данные с цифрового выхода одного устройства на цифровой вход другого.

Например, в *главе 14* мы встретим дисплеи, данные для которых должны поступать по последовательному интерфейсу.

Существуют различные стандарты последовательных интерфейсов, решающие очень схожие задачи, но слегка по-разному. К ним относятся собственно последовательный интерфейс (он же TTL serial), интерфейс I^2C (по совету автора произносится как «ай квадрат си») и последовательный периферийный интерфейс SPI.

Заключение

В этой главе мы подробно рассмотрели фундаментальные основы электроники, связанные с силой тока, сопротивлением и напряжением, а также обсудили некоторые электронные компоненты, используемые в этой книге. Но довольно теории! В следующей главе мы узнаем, как с помощью Arduino и Raspberry Pi управлять разными типами светодиодов.

6 Светодиоды

Управление светодиодными индикаторами (LED) — это задача, которая, вероятно, будет в самом начале списка дел у любого электронщика. Разнообразных светодиодов существует невероятное множество: от самых простых до высокомощных, от инфракрасных до ультрафиолетовых.

На рис. 6.1 показана подборка светодиодов с самыми разными требованиями к силе тока, причем некоторые вполне нормально работают от цифрового выхода, а для управления другими требуются транзисторы или целые электронные схемы.

Рис. 6.1. Разнообразие светодиодов

Обычные светодиоды

Обычные светодиоды — это те самые разноцветные изделия диаметром 5 мм (реже — 10 мм или 3 мм), которые зажигаются от не очень сильного тока, поэтому ими можно управлять непосредственно с выхода Arduino или Raspberry Pi.

На рис. 6.2 показано, как такие светодиоды обычно подключают к цифровому выходу Arduino или Raspberry Pi.

Резистор здесь нужен для ограничения тока, проходящего через светодиод, и на то есть две причины: во-первых, чтобы не превышать максимальное значение силы

тока, на которое рассчитан светодиод (в противном случае он прослужит недолго), и во-вторых, чтобы не превышать предельную величину выходного тока на отдельном контакте платы или суммарно на всех ее выходных контактах.

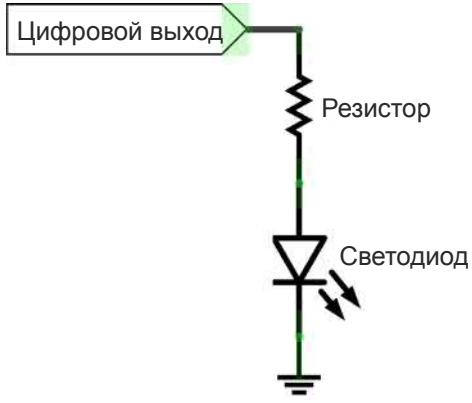


Рис. 6.2. Подключение светодиода к цифровому выходу

Ограничение тока

Когда светодиод подключен так, как показано на рис. 6.2, то во всей цепи установится более или менее постоянное напряжение. Оно называется *прямым напряжением* светодиода. В зависимости от цвета светодиода прямое напряжение может быть различным: у красных светодиодов оно обычно минимальное, а у синих и белых — наивысшее среди всех светодиодов, дающих видимый свет (табл. 6.1).

Существуют также *инфракрасные* (ИК) светодиоды, наподобие тех, что устанавливаются в телевизионных пультах, а также ультрафиолетовые (УФ) светодиоды, которые часто используют на вечеринках для придания фиолетового оттенка белым одеждам либо для проверки подлинности банкнот.

Кроме прямого напряжения у светодиода есть еще одна важная характеристика, которую нужно знать, — это прямой ток, который вы собираетесь через него пропускать. Большинство светодиодов хотя бы немного светятся от тока силой 1 мА или чуть менее, но, как правило, достигают оптимальной яркости при токе около 20 мА. Это достаточно широкий диапазон, поэтому из соображений безопасности используйте резистор на 470 Ом при работе с любым светодиодом, который подключаете к Arduino или Raspberry Pi, хотя светодиод и будет светить не столь ярко, как мог бы.

Для расчета сопротивления последовательного резистора удобно воспользоваться специальным веб-сервисом, который все за вас рассчитает. Подобный сервис находится по адресу **led.linear1.org/1led.wiz**, его окно показано на рис. 6.3.

Как можно видеть, за напряжение источника питания здесь принято 3,3 В (очевидно, расчет ведется для Raspberry Pi), а максимальная сила тока, допустимая для любого контакта Raspberry Pi, составляет 16 мА. Калькулятор сообщает, что при этих

исходных данных для светодиода с прямым напряжением 2,2 В нужно использовать резистор сопротивлением 82 Ом.

Если вы хотите произвести такой расчет сами, то сначала вычтите значение прямого напряжения (2,2 В) светодиода из напряжения источника питания 3,3 В. Получится 1,1 В. Затем рассчитайте сопротивление резистора по закону Ома:

$$R = V / I = 1{,}1 \text{ В} / 16 \text{ мА} = 68{,}75 \text{ Ом}.$$

При подборе резистора также можно воспользоваться данными табл. 6.1. В ней указаны и примерные уровни прямого напряжения для светодиодов разного цвета. Обратите внимание: значения сопротивления откорректированы и приближены к стандартным значениям таких резисторов, которые имеются в свободном доступе.

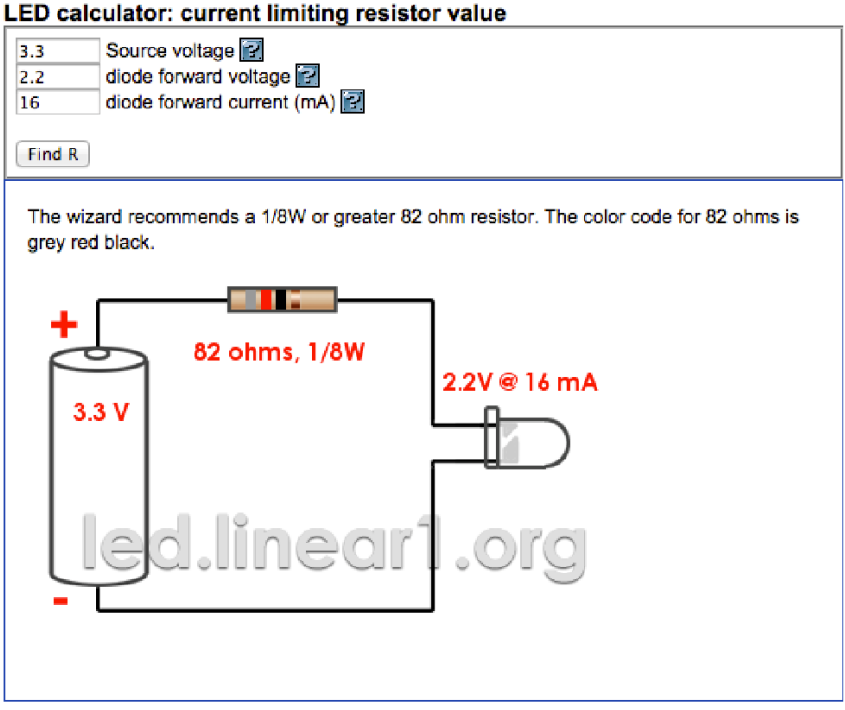


Рис. 6.3. Онлайн-калькулятор для работы с последовательными резисторами

Таблица 6.1. *Резисторы, ограничивающие силу тока, для разных светодиодов*

	ИК	Красный	Оранжевый / желтый / зеленый	Голубой / белый	Фиолетовый	УФ
Прямое напряжение, В	1,2–1,6	1,6–2	2–2,2	2,5–3,7	2,7–4	3,1–4,4
Pi (3,3 В, 3 мА), Ом	×	680	470	270	220	68
Pi (3,3 В, 16 мА), Ом	150	120	82	56	39	15
Arduino (5 В; 20 мА), Ом	220	180	150	150	120	100

Примечание

В столбце ИК для устройства Pi (3,3 В; 3 мА) указан символ ×. Дело в том, что инфракрасные светодиоды для пультов дистанционного управления обычно требуют для работы как минимум 10 мА, причем проектируются с расчетом на силу тока 100 мА или более, — только при таких значениях они получаются достаточно «дальнобойными».

Проект: светофор

Имея такой отличный выбор разноцветных светодиодов, возьмем красный, желтый и зеленый излучатели и сделаем светофор, управляемый с Arduino или Raspberry Pi (рис. 6.4).

Светодиоды должны зажигаться в такой последовательности:

1. Красный.
2. Красный с желтым.
3. Зеленый.
4. Желтый.

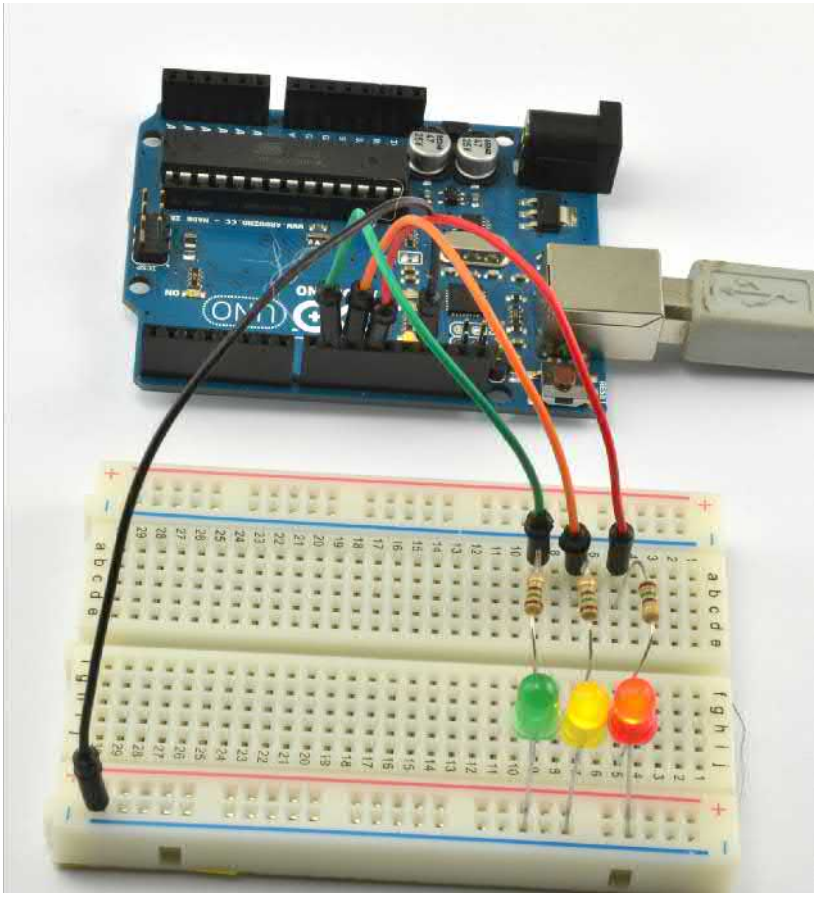


Рис. 6.4. Светофор на Arduino

Комплектующие

В этом проекте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 6.2).

Таблица 6.2. *Комплектующие для работы с Arduino и Raspberry Pi в проекте «Светофор»*

Обозначение	Компонент схемы	Источники
LED1	Красный светодиод	Adafruit: 297 Sparkfun: COM-09590
LED2	Желтый светодиод	Sparkfun. COM-09594
LED3	Зеленый светодиод	Adafruit: 298 Sparkfun. COM-09650
R1-3	Резисторы на150 Ом	Mouser. 291-150-RC
	400-точечная беспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758
	Перемычки «мама-папа» (только для Pi)	Adafruit: 826

Со временем вы, возможно, начнете подбирать резисторы с самыми разными значениями, однако в этом проекте я предлагаю компромиссный вариант — резисторы на 150 Ом и для Arduino, и для Raspberry Pi, причем для светодиодов всех трех цветов. Если вы захотите добиться максимальной яркости, то подберите к ним оптимальные резисторы по табл. 6.1.

Общая конструкция

Каждый из трех светодиодов подключен к отдельному выходному контакту на Arduino или Raspberry Pi.

Подключение к Arduino

На рис. 6.5 показана компоновка макетной платы и подключение ее к Arduino.

Не забывайте, что длинные выводы светодиодов — положительные. Светодиоды устанавливаются в макетную плату этими выводами влево, и концы их подключаются каждый к своему резистору. Более короткие выводы — отрицательные, идут от каждого светодиода к ряду отрицательного питания, идущего по правому краю макетной платы.

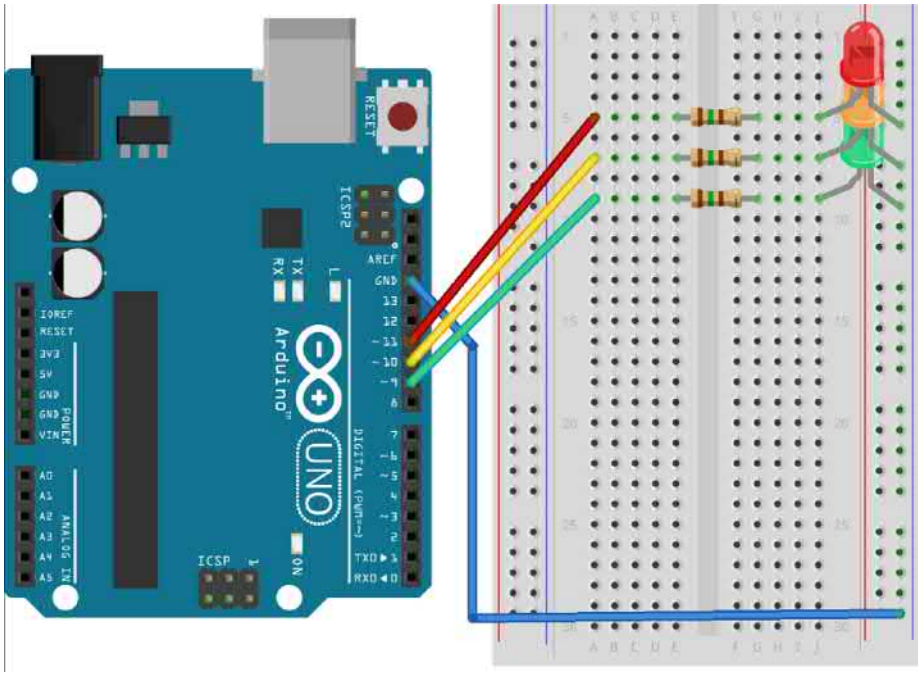


Рис. 6.5. Компоновка макетной платы для светофора под управлением Arduino

Программа для Arduino

Скетч Arduino для этого проекта находится в каталоге arduino/projects/traffic_signals (см. *разд. «Код к книге» главы 2*). Рассмотрим этот скетч:

```
const int redPin = 11; // 1
const int orangePin = 10;
const int greenPin = 9;

void setup() { // 2
   pinMode(redPin, OUTPUT);
   pinMode(orangePin, OUTPUT);
   pinMode(greenPin, OUTPUT);
}

void loop() { // 3
   setLEDs(1, 0, 0);
   delay(3000);
   setLEDs(1, 1, 0);
   delay(500);
   setLEDs(0, 0, 1);
   delay(5000);
   setLEDs(0, 1, 0);
   delay(500);
}
```

```
void setLEDs(int red, int orange, int green) { // 4
  digitalWrite(redPin, red);
  digitalWrite(orangePin, orange);
  digitalWrite(greenPin, green);
}
```

Этот скетч во многом похож на исходный скетч с мигалкой (см. *главу 4*), за тем исключением, что здесь мы управляем не одним светодиодом, а сразу тремя. Рассмотрим его по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Определяются константы для каждого из контактов Arduino, подключенных к светодиоду.
2. В функции `setup()` контакты устанавливаются на выход.
3. Функция `loop()` вызывает функцию `setLEDs()`, позволяющую включать или выключать три светодиода (задавать для них значения `1` или `0`). Задержка перед каждым вызовом `setLEDs()` определяет, как долго светодиод пребывает в той или иной фазе.
4. Благодаря функции `setLEDs()` циклическая функция всего лишь укорачивается и становится более удобочитаемой, при этом мы можем уложить все три оператора `digitalWrites()` в одну строчку.

Подключение к Raspberry Pi

В версии проекта, выполняемой на основе Raspberry Pi, перемычки «папа-папа» заменяются на перемычки «мама-папа», и макетная плата в той же компоновке подключается к контактам GPIO **18**, **23** и **24** (рис. 6.6).

Программа для Raspberry Pi

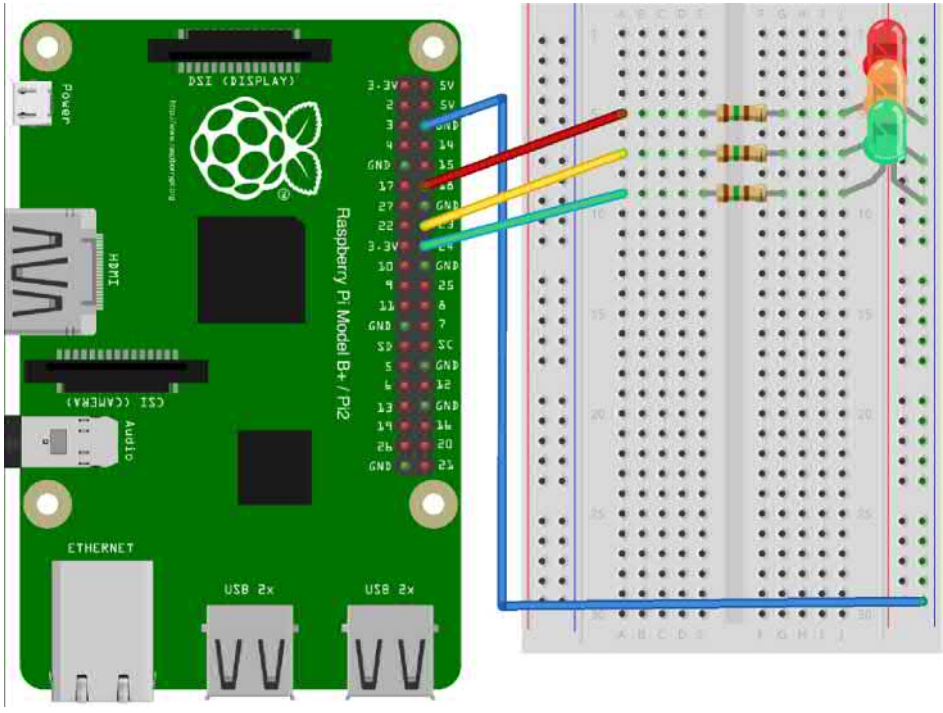
Программа на языке Python для этого проекта находится в файле traffic.py каталога python/projects/ (подробнее об установке программ Python на Raspberry Pi рассказано в *разд. «Код к книге» главы 3*).

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

red_pin = 18
orange_pin = 23
green_pin = 24

GPIO.setup(red_pin, GPIO.OUT)
GPIO.setup(orange_pin, GPIO.OUT)
GPIO.setup(green_pin, GPIO.OUT)
```



Компоновка макетной платы для светофора под управлением Raspberry Pi

```
def set_leds(red, orange, green):

GPIO.output(red_pin, red)
GPIO.output(orange_pin, orange)
GPIO.output(green_pin, green)

try:
   while True:
set_leds(1, 0, 0)
time.sleep(3)
set_leds(1, 1, 0)
time.sleep(0.5)
set_leds(0, 0, 1)
time.sleep(5)
set_leds(0, 1, 0)
time.sleep(0.5)

finally:
   print("Сброс")
   GPIO.cleanup()
```

Функция `set_leds`**, определяемая в строке, обозначенной комментарием** `1`**, точно так же, как и в случае с Arduino, используется для того, чтобы не захламлять главный цикл множеством функций** `GPIO.output`**.**

ШИМ и светодиоды

Попытка управления яркостью светодиода регулированием напряжения на нем обычно не приносит удовлетворительных результатов, поскольку в начале диапазона напряжений на светодиоде имеет место обширная мертвая зона, и лишь после ее преодоления напряжение достигает такой величины, которое позволяет светодиоду излучать хоть какой-то свет.

Для управления яркостью светодиода лучше всего подходят аналоговые выходы с ШИМ (см. далее врезку *«Широтно-импульсная модуляция»*). Светодиоды могут включаться и выключаться очень быстро — гораздо быстрее, чем за миллионную долю секунды, при этом с использованием ШИМ светодиод и мерцает на частоте ШИМ, однако зрительно это воспринимается просто как изменение яркости, пропорциональное тому, как долго светодиод фактически остается во включенном состоянии.

Широтно-импульсная модуляция

До сих пор мы управляли устройствами очень «по-цифровому» — т. е. включали, а затем выключали. Но что, если требуется организовать управление более «аналоговым» образом? Допустим, вы хотите управлять скоростью двигателя или яркостью светодиода. Для этого нужно научиться управлять мощностью, подаваемой на интересующее вас устройство.

Прием, лежащий в основе управления по такому принципу, называется *широтно-импульсной модуляцией* (ШИМ). Для ее использования задействуются цифровые выходы, дающие серию высоких и низких импульсов. Управляя долей времени, в течение которого подаются высокие импульсы, можно управлять общей подачей мощности на двигатель или светодиод.

На рис. 6.7 показано, как действует ШИМ. При этом предполагается, что напряжение на GPIO-контакте Raspberry Pi составляет 3,3 В. Вам также понадобится транзистор,

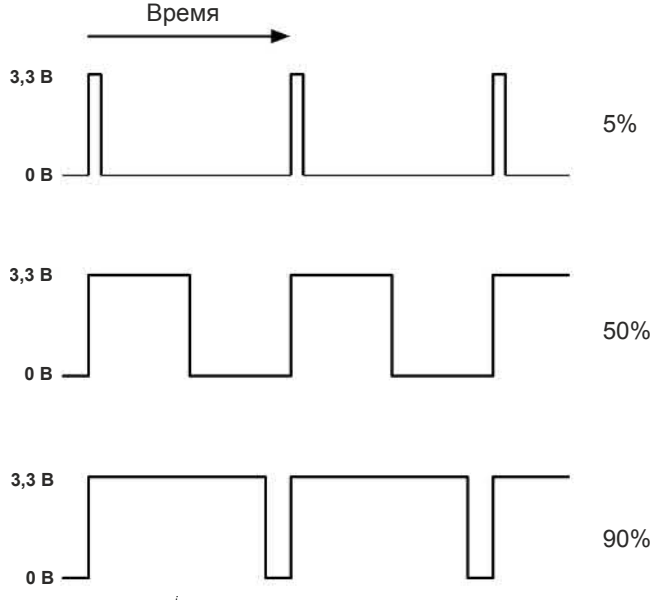


Рис. 6.7. Широтно-импульсная модуляция

подключенный к контакту GPIO, — он нужен для подачи тока, достаточного для управления двигателем.

Отношение длительности импульса к периоду его следования называется *коэффициентом заполнения*. Если импульсы соответствуют включенному состоянию на протяжении лишь 5% всего времени цикла (коэффициент заполнения равен 5%), то на двигатель будет поступать не слишком много энергии, и он станет вращаться очень медленно (либо светодиод будет светить тускло). Стоит повысить коэффициент заполнения до 50% — и двигатель или светодиод получат половину мощности, соответственно, скорость вращения или яркость составят примерно половину от максимума. Если коэффициент заполнения возрастет до 90%, то двигатель будет вращаться почти с полной скоростью, а светодиод — светить в полную силу.

Чтобы выключить двигатель или светодиод, достаточно лишь снизить коэффициент заполнения до 0, а для разгона на полную мощность — поднять его до 100%. Удержание контакта GPIO постоянно в состоянии high и обеспечит коэффициент заполнения 100%.

Как Arduino, так и Raspberry Pi позволяют использовать на выходных контактах широтно-импульсную модуляцию. На Arduino Uno она может включаться только на контактах D3, D5, D6, D9, D10 и D11 — эти контакты на самой плате Arduino помечены тильдой (~). Частота ШИМ на Arduino Uno для большинства контактов составляет 490 Гц (импульсов в секунду). Исключение представляют контакты 5 и 6, работающие с частотой 980 Гц.

Для управления яркостью светодиода или скоростью двигателя вполне достаточно частоты ШИМ 490 Гц, которая установлена в Arduino по умолчанию.

RGB-светодиоды

RGB-светодиод — это единая сборка, в которой на самом деле находятся три светодиода: один синего, один красного и один зеленого цвета. Используя ШИМ для управления яркостью каждого из этих светодиодов, можно придать RGB-светодиоду любой из трех цветов.

Хотя в RGB-светодиоде содержится три обычных двухвыводных светодиода, это не означает, что корпус светодиода должен иметь шесть выводов, — по одному выводу от каждого светодиода можно соединить вместе, придав ему вид обычного вывода.

Если подключить друг к другу отрицательные выводы всех светодиодов, то получившийся вывод будет называться *общим катодом* (рис. 6.8), а если проделать то же самое с положительными выводами, получится *общий анод*.

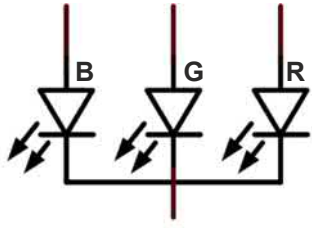


Рис. 6.8. Схема общего катода RGB-светодиода

RGB-светодиод может излучать как направленный свет, так и рассеянный. Если светодиод направленный, то в сборке будут четко различаться красный, зеленый и синий светодиоды, и их цвета почти не станут смешиваться друг с другом. В рассеянных сборках цвета трех светодиодов смешиваются друг с другом гораздо лучше.

В *главе 14* мы будем работать с дисплеями, основанными на RGB-светодиодах, имеющих собственную управляющую микросхему, которая ограничивает поступление тока на красный, зеленый и синий светодиоды, и в то же время служит им последовательным интерфейсом передачи данных. Arduino или Raspberry Pi могут управлять через этот интерфейс множеством таких светодиодов, и для этого задействуется всего один их выходной контакт.

Эксперимент: смешивание цветов

В этом эксперименте мы будем управлять цветом RGB-светодиода с Arduino и с Raspberry Pi. В той версии эксперимента, которая выполняется на Raspberry Pi, для управления цветом мы воспользуемся графическим пользовательским интерфейсом с тремя ползунками. На рис. 6.9 показан вариант эксперимента для Raspberry Pi, а схема эксперимента — на рис. 6.10.

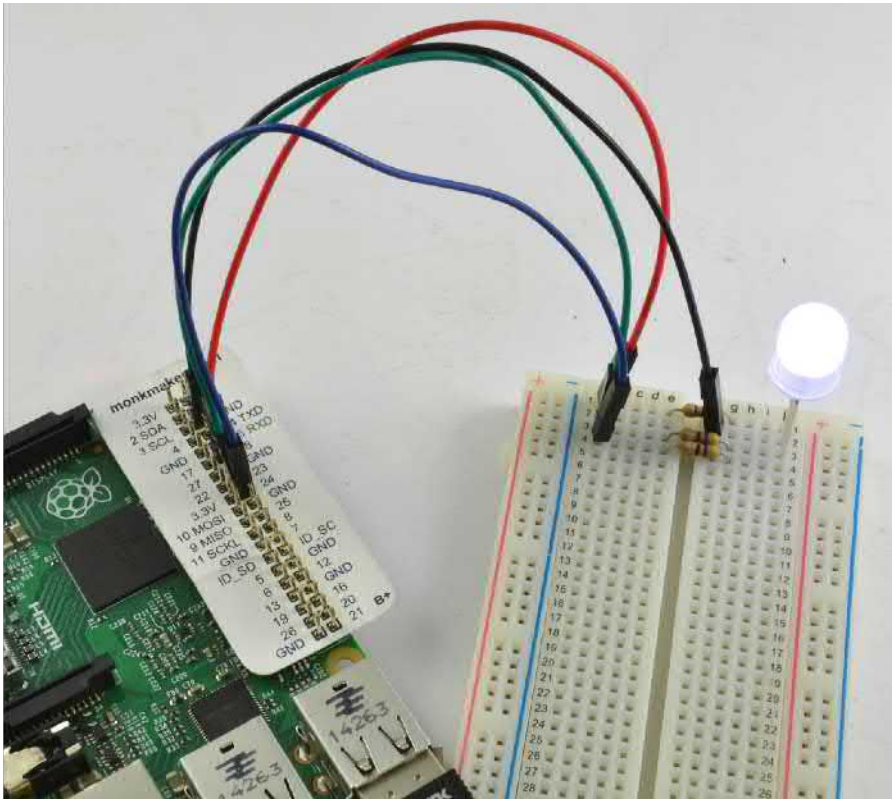


Рис. 6.9. Подключение макетной платы для смешивания цветов при помощи Raspberry Pi

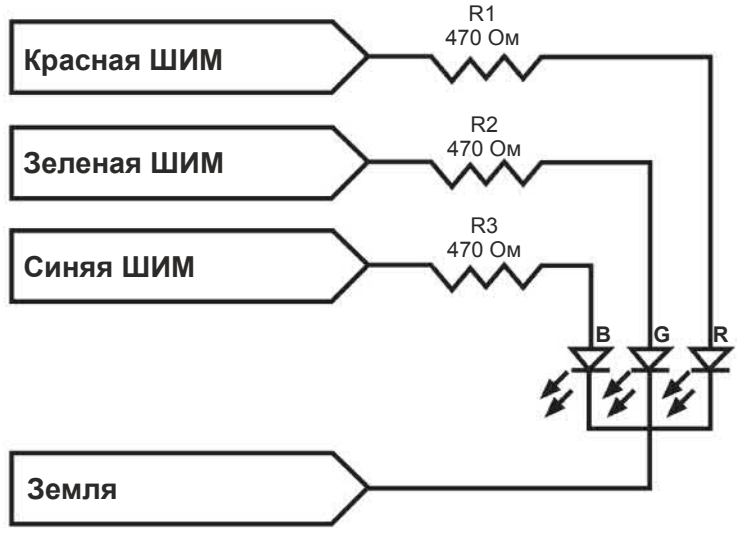


Рис. 6.10. Схема эксперимента с RGB-светодиодом

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 6.3).

Таблица 6.3. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по смешиванию цветов*

Обозначение	Компонент схемы	Источники
LED1	RGB-светодиод	Sparkfun: COM-11120
R1-3	Резистор 470 Ом 0,25 Вт	Mouser: 291-470-RC
	400-точечная беспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758
	Перемычки «мама-папа» (только для Pi)	Adafruit. 826

Примечание

Все эти компоненты входят в набор MonkMakes Electronic Starter Kit для Raspberry Pi (см. *приложение 1*).

Напомню, что перемычки «мама-папа» понадобятся только для подключения к макетной плате контактов GPIO Raspberry Pi (если вы планируете провести этот эксперимент с Raspberry Pi тоже).

Достичь оптимальной яркости и наилучшего цветового баланса можно лишь при внимательном подборе резисторов соответствующих сопротивлений. Однако приобретать комплектующие проще, если на всех трех каналах использовать резисто-

ры с одним и тем же значением. В нашем случае я рекомендую в качестве универсальных применить резисторы на 470 Ом — они будут работать и с Raspberry Pi, и с Arduino.

Кстати, яркость и эффективность RGB-светодиода таковы, что даже при токе силой всего 3 мА (на Arduino — 6 мА) светодиод будет сиять весьма ярко.

Экспериментируем с Arduino

Подключение к Arduino

В варианте этого эксперимента для Arduino используется монитор последовательного интерфейса, через который задаются пропорции красного, зеленого и синего цветов.

На рис. 6.11 показана компоновка макетной платы и подключение ее к Arduino.

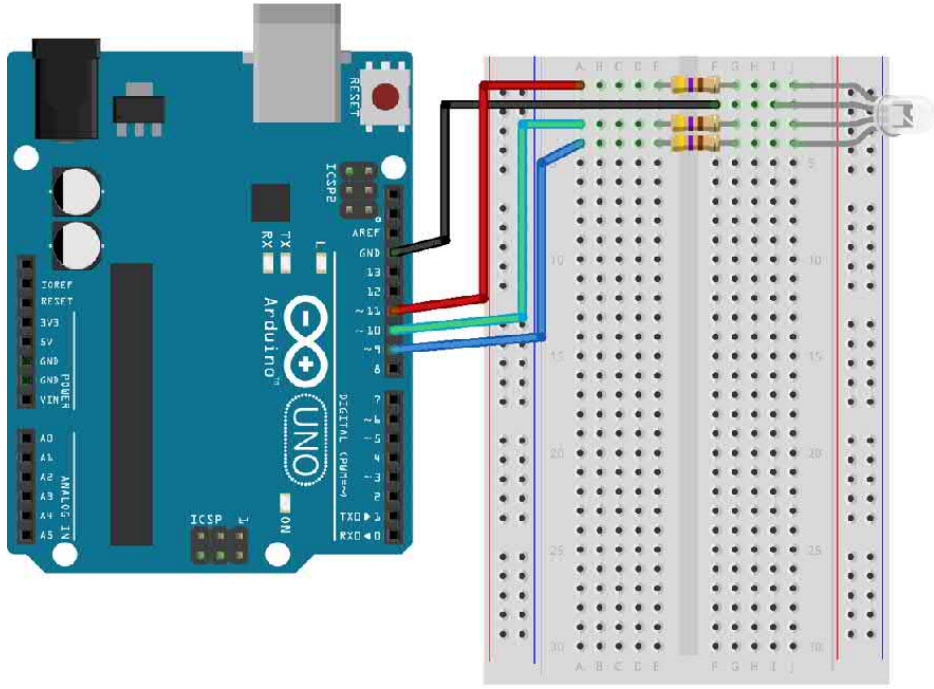


Рис. 6.11. Компоновка макетной платы для смешивания цветов RGB-светодиода с помощью Arduino

Самый длинный вывод RGB-светодиода — это общий катод, который на рис. 6.11 подключается к разъему **GND** платы Arduino. Порядок подключения других выводов может быть не таким, как показано у меня, поэтому, возможно, вам придется переставить некоторые провода, если ваши цвета станут смешиваться неправильно.

Программа для Arduino

Скетч Arduino для этого эксперимента находится в каталоге arduino/experiments/mixing_colors (см. *разд. «Код к книге» главы 2*). В скетче организуются три ШИМ-

канала, при помощи которых осуществляется управление яркостью каждого из трех цветов.

```
const int redPin = 11;
const int greenPin = 10;
const int bluePin = 9;

void setup() {
   pinMode(redPin, OUTPUT);
   pinMode(greenPin, OUTPUT);
   pinMode(bluePin, OUTPUT);
   Serial.begin(9600);
   Serial.println("Введите значение R G B (Например, 255 100 200)");
}

void loop() {
   if (Serial.available()) {
      int red = Serial.parseInt(); // 1
      int green = Serial.parseInt();
      int blue = Serial.parseInt();
      analogWrite(redPin, red); // 2
      analogWrite(greenPin, green);
      analogWrite(bluePin, blue);
   }
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Каждое из трех значений ШИМ, которые должны находиться в диапазоне от 0 до 255, считывается в переменные.
2. Затем для каждого канала задается ШИМ-вывод.

Загружаем и выполняем программу

Загрузив скетч, откройте в Arduino IDE окно монитора порта. Затем введите три числа в диапазоне от 0 до 255 каждое, разделив их пробелами, и нажмите кнопку **Отправить** (Send) — светодиод должен изменить цвет так, чтобы значения красного, зеленого и синего соответствовали тем числам, которые вы только что ввели.

Можно по отдельности проверить каждый из каналов, введя по отдельности `255 0 0` (красный), затем `0 255 0` (зеленый) и, наконец, `0 0 255` (синий).

Экспериментируем с Raspberry Pi

Подключение к Raspberry Pi

В версии эксперимента, ориентированной на Raspberry Pi, мы не просто будем вводить команды для изменения цвета, а воспользуемся небольшим окном с пользова-

тельским интерфейсом, содержащим три ползунка — по одному на каждый цветовой канал. Когда вы будете менять положение ползунков, станет меняться и цвет RGB-светодиода.

Поскольку в этой программе организуется графический пользовательский интерфейс, а в SSH графический интерфейс отсутствует, вам потребуется подключить к Raspberry Pi клавиатуру, мышь и монитор.

Компоновка макетной платы для работы с Raspberry Pi показана на рис. 6.12 — она точно такая же, как и в случае с Arduino, за тем исключением, что при работе с Raspberry Pi вам понадобятся перемычки «мама-папа». В качестве ШИМ-выходов здесь задействуются GPIO-контакты 18, 23 и 24.

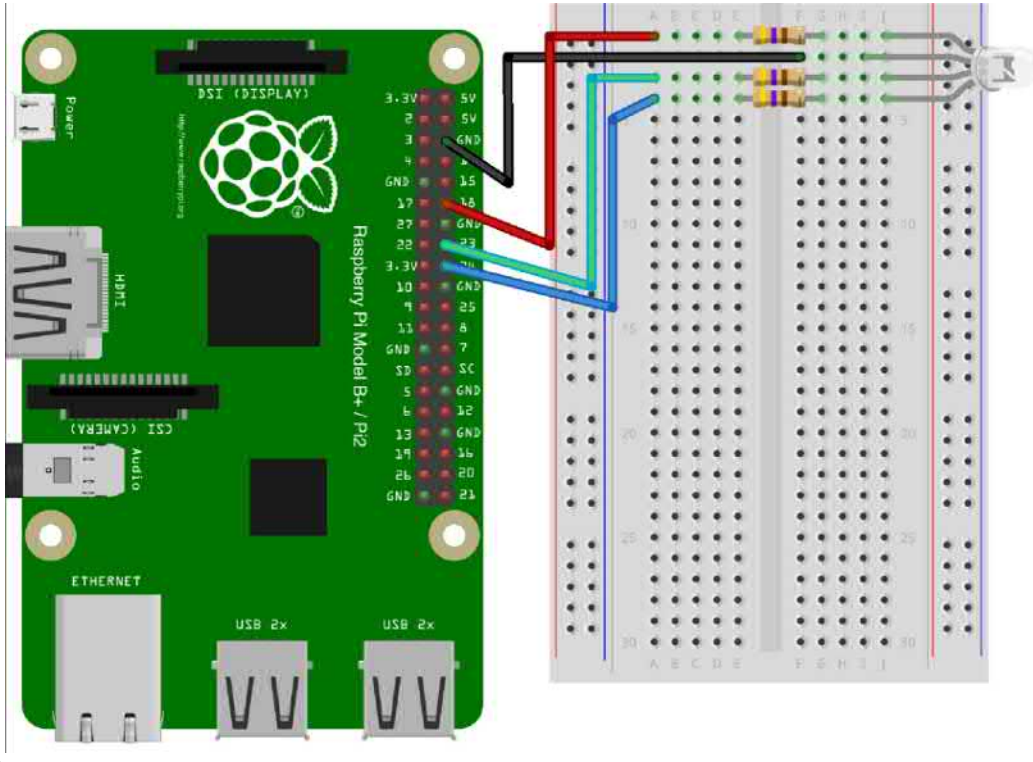


Рис. 6.12. Компоновка макетной платы для смешивания цветов RGB-светодиода с помощью Raspberry Pi

Программа для Raspberry Pi.

В программе для этого эксперимента, написанной на языке Python, применяется фреймворк `Tkinter`. С его помощью можно создавать оконные приложения, имеющие свои элементы управления, что дает возможность не ограничиваться примитивной командной строкой, с которой вы работали до сих пор. Соответственно и код программы получается немного длиннее, чем обычно. Кроме того, местами в нем используется сравнительно сложное программирование.

Программа для этого эксперимента находится в файле mixing_colors.py каталога python/experiments. Рассмотрим ее код:

```
from Tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM) // 1
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)

pwmRed = GPIO.PWM(18, 500) // 2

pwmRed.start(100)

pwmGreen = GPIO.PWM(23, 500)
pwmGreen.start(100)

pwmBlue = GPIO.PWM(24, 500)
pwmBlue.start(100)

class App:

   def __init__(self, master): // 3
      frame = Frame(master) // 4
      frame.pack()

      Label(frame, text='Red').grid(row=0, column=0) // 5
      Label(frame, text='Green').grid(row=1, column=0)
      Label(frame, text='Blue').grid(row=2, column=0)

      scaleRed = Scale(frame, from_=0, to=100, // 6
         orient=HORIZONTAL, command=self.updateRed)
      scaleRed.grid(row=0, column=1)
      scaleGreen = Scale(frame, from_=0, to=100,
         orient=HORIZONTAL, command=self.updateGreen)
      scaleGreen.grid(row=1, column=1)
      scaleBlue = Scale(frame, from_=0, to=100,
         orient=HORIZONTAL, command=self.updateBlue)
      scaleBlue.grid(row=2, column=1)

   def updateRed(self, duty): // 7
      # change the led brightness to match the slider
      pwmRed.ChangeDutyCycle(float(duty))

   def updateGreen(self, duty):
      pwmGreen.ChangeDutyCycle(float(duty))

   def updateBlue(self, duty):
      pwmBlue.ChangeDutyCycle(float(duty))
```

```
root = Tk()// 8
root.wm_title('RGB LED Control')
app = App(root)
root.geometry("200x150+0+0")
try:
   root.mainloop()
finally:
   print("Сброс")
   GPIO.cleanup()
```

Уточним некоторые моменты этой программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Сконфигурируем Pi так, чтобы можно было работать с именами контактов по системе Broadcom, а не просто с обозначениями позиций.
2. Запускаем широтно-импульсную модуляцию (ШИМ) на красном, зеленом и синем канале, чтобы управлять яркостью светодиодов.
3. Эта функция вызывается при создании приложения.
4. В этом фрейме содержатся различные элементы управления из графического интерфейса.
5. Создаем надписи и располагаем каждую в своей ячейке сетки.
6. Создаем ползунки и располагаем каждый в своей ячейке сетки. Атрибут `command` указывает, какой метод нужно вызывать при перемещении ползунка.
7. Этот метод и подобные методы для других цветов вызываются при перемещении ползунка.
8. Запускаем графический пользовательский интерфейс, задаем для окна название, размер и положение.

Загружаем и выполняем программу

Запустим программу от имени администратора при помощи следующей команды:

```
$ sudo python mixing_colors.py
```

Спустя пару секунд появится окно, показанное на рис. 6.13. При перемещении ползунков цвет RGB-светодиода будет меняться.

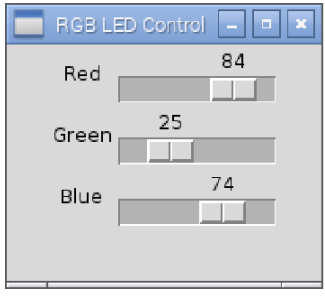


Рис. 6.13. Перемещаем ползунки для изменения цвета RGB-светодиода

Заключение

В этой главе мы научились включать и выключать светодиод, а также управлять его яркостью с помощью Arduino и Raspberry Pi. Теперь, когда вы умеете управлять освещением, в следующей главе поговорим о движении и подробнее познакомимся с двигателями постоянного тока. Электродвигатели такого типа наиболее распространены — и мы научимся ими управлять.

Двигатели, насосы и исполнительные механизмы

7

В *главе 4* мы уже провели эксперимент с двигателем постоянного тока. Те основные положения, которые мы усвоим, имея дело с двигателями постоянного тока, применимы и при обращении со многими другими механизмами, которыми вы, возможно, захотите управлять с Arduino или Raspberry Pi. Именно электродвигатели позволяют работать различным полезным выходным устройствам — в частности, насосам и линейным исполнительным механизмам, с которыми мы познакомимся в этой главе далее.

На рис. 7.1 показана подборка двигателей постоянного тока. Как видите, эти устройства могут иметь всевозможные размеры и форму.

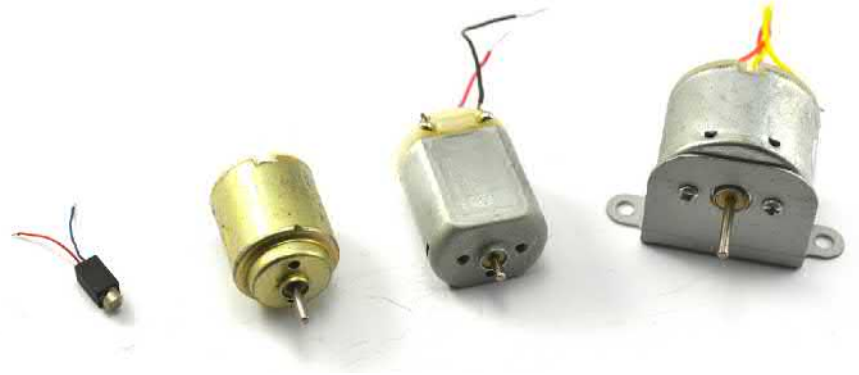


Рис. 7.1. Разновидности двигателей постоянного тока

Как мы узнали из *разд. «Эксперимент: управление электродвигателем» главы 4*, управлять двигателем постоянного тока непосредственно с контакта Arduino или Raspberry Pi нельзя — для этого нужен весьма сильный ток, поэтому мы и использовали транзистор, чтобы с его помощью получить возможность включать и выключать двигатель. Теперь нам предстоит научиться управлять скоростью его вращения.

Как работают двигатели постоянного тока?

Как правило, двигатель постоянного тока состоит из трех основных частей, которые показаны на рис. 7.2: внешнюю (неподвижную) часть двигателя представляет стационарный магнит (*статор*), а на валу внутри статора располагается подвижная часть двигателя, состоящая из *ротора* и *коллектора*.

Ротор несет на себе проволочные катушки (*обмотки*) — на рис. 7.2 показаны три такие катушки, намотанные вокруг трех фасонных фрагментов ротора и подсоединенные к коллектору. Назначение коллектора — поочередно возбуждать по мере вращения ротора в его обмотках магнитное поле с тем, чтобы следующая обмотка всегда подтягивалась к постоянным магнитам статора, в результате чего ротор бы вращался.

Коллектор представляет собой сегментированное кольцо (на рис. 7.2 показаны три сегмента), а *щетки* служат для подключения питающих контактов к отдельным сегментам коллектора по мере того, как коллектор вращается вместе с ротором.

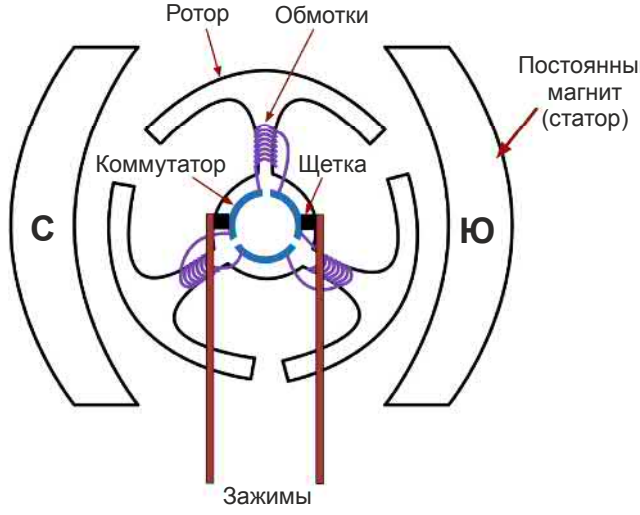


Рис. 7.2. Внутреннее устройство двигателя постоянного тока

Рассмотренная здесь модель электродвигателя с тремя наборами обмоток весьма типична для таких небольших двигателей постоянного тока, что показаны на рис. 7.1.

Полезное свойство двигателей постоянного тока заключается в том, что изменение полярности напряжения на контактах заставляет двигатель вращаться в противоположном направлении.

Управление скоростью (ШИМ)

В *разд. «Эксперимент: смешивание цветов» главы 6* для управления яркостью светодиода мы воспользовались широтно-импульсной модуляцией (см. в *главе 6* врезку *«Широтно-импульсная модуляция»*). Точно таким же способом можно управлять и скоростью электродвигателя.

Эксперимент: управление скоростью двигателя постоянного тока

Оборудование

В этом эксперименте используется то же самое оборудование, что и в *разд. «Эксперимент: управление электродвигателем» главы 4*, но здесь вы не просто включаете и выключаете двигатель, а еще и управляете его скоростью.

Если вы ранее еще не сделали этого, соберите схему из *разд. «Эксперимент: управление электродвигателем» главы 4*. В качестве напоминания компоновка макетной платы для этого эксперимента показана на рис. 7.3.

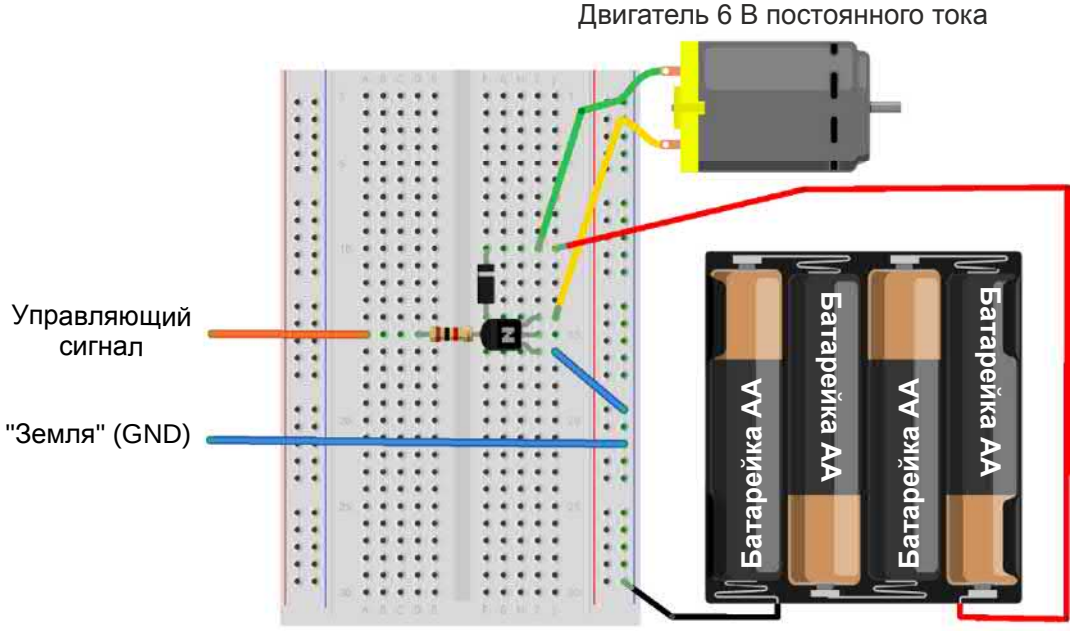


Рис. 7.3. Компоновка макетной платы для экспериментов с электродвигателем

Экспериментируем с Arduino

Подключение Arduino

Подключите макетную плату к Arduino, как показано на рис. 7.4: подсоедините заземление с макетной платы к разъему **GND**, а управляющий вывод с макетной платы — к разъему **D9**.

Программа для Arduino

Скетч Arduino для этого эксперимента находится в каталоге arduino/experiments/pwm_motor_control (см. *разд. «Код к книге» главы 2*).

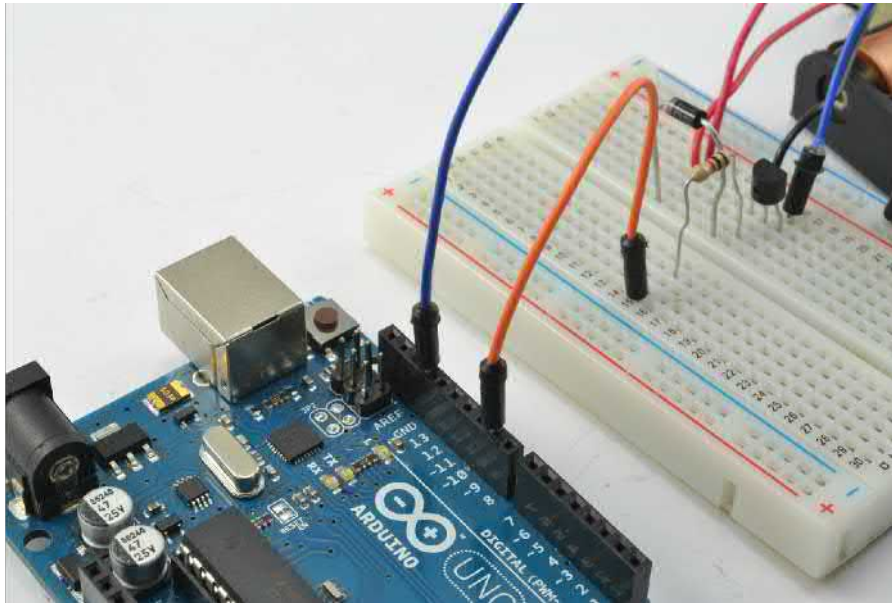


Рис. 7.4. Подключение макетной платы к Arduino

Для ввода коэффициента заполнения и задания нужной скорости вращения электродвигателя воспользуемся монитором последовательного интерфейса Arduino IDE:

```
const int controlPin = 9;

void setup() { // 1
   pinMode(controlPin, OUTPUT);
   Serial.begin(9600);
   Serial.println("Укажите рабочий цикл (0 to 100)");
}

void loop() { // 2
   if (Serial.available()) { // 3
      int duty = Serial.parseInt();
      if (duty < 0 || duty > 100) { // 4
         Serial.println("0 - 100");
      }
      else {
         int pwm = duty * 255 / 100;
         analogWrite(controlPin, pwm);
         Serial.print("рабочий цикл установлен на ");
         Serial.println(duty);
      }
   }
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Функция `setup` определяет `controlPin` как выход, а также при помощи функции `Serial.begin` запускает последовательное соединение — теперь вы можете через USB отправлять значения коэффициента заполнения со своего компьютера на Arduino.
2. Функция `loop()` при помощи `Serial.available` проверяет, есть ли в очереди какие-либо последовательные сообщения, которые могли поступить через USB.
3. Если есть сообщение, представляющее собой число, то это число считывается из потока символов и преобразуется в тип `int` при помощи `parseInt`.
4. Проверяем, находится ли полученное значение в диапазоне от `0` до `100`:
   - если оно выходит за его пределы, то напоминание об этом передается вам через USB на монитор последовательного интерфейса;
   - если же число находится в диапазоне от 0 до 100, то оно преобразуется в число от 0 до 255, а затем функция `analogWrite()` выдает значение ШИМ. Такое преобразование необходимо, поскольку функция `analogWrite()` Arduino принимает значение коэффициента заполнения в диапазоне от 0 до 255, где 0 = 0%, а 255 = 100%.

Текст и числа

В этой книге мы довольно часто будем считывать числа через монитор последовательного интерфейса, поэтому стоит разобраться, что именно делает функция `parseInt` и что происходит, когда мы передаем информацию на Arduino по USB.

В *разд. «Соединения по последовательным интерфейсам главы 5* я упоминал, что один из способов взаимодействия с Arduino (и, если уж на то пошло, с Raspberry Pi) — работа через последовательный интерфейс. Arduino Uno обладает встроенным последовательным интерфейсом, подключенным к цифровым контактам D0 и D1. Эти контакты не следует использовать для обычного ввода/вывода, т. к. они предоставляют последовательный интерфейс между вашим компьютером и Arduino через интерфейсную микросхему USB, взаимно преобразующую последовательный вывод USB и прямой последовательный ввод, понятный Arduino.

Когда вы вводите сообщение в монитор последовательного интерфейса и отправляете его в Arduino, текст сообщения преобразовывается в поток битов (высоких и низких сигналов), которые на месте назначения вновь собираются в группы по восемь битов (т. е. в байты). Каждый из этих байтов — соответствующий определенной букве латинского алфавита или цифре числовой код ASCII (Американский стандартный код обмена информацией).

Когда работающая программа Arduino получает сообщение в виде такой последовательности, она может либо считывать по одному байту (букве) в единицу времени, либо использовать функцию `parseInt`, которая считывает символы и, если оказывается, что символ — это цифра, составляет из цифр число. Например, число 154 было бы передано как три символа: 1, 5 и 4. Если после символа следует знак перехода на новую строку, пробел или символ, не являющийся цифрой, то функция `parseInt` «понимает», что все цифры из переданного числа уже получены, и возвращает значение 154 типа `int`. Впрочем, даже если после отправки последней цифры наступит небольшая задержка, этого хватит, чтобы `parseInt` зафиксировала конец числа.

Загружаем и выполняем программу

Загрузите программу на Arduino. На компьютере, с которого вы программируете Arduino, откройте в Arduino IDE монитор последовательного интерфейса. Для этого щелкните на значке с лупой в верхнем правом углу окна Arduino IDE (на рис. 7.5 он обведен кружочком).

Далее вам будет предложено ввести значение коэффициента заполнения от 0 до 100. Попробуйте вводить разные значения и посмотрите, как это отразится на скорости электродвигателя. Когда вы решите остановить двигатель, просто обнулите коэффициент заполнения.

Обратите внимание: если спуститься до значений 10–20, то двигатель может не вращаться, а натужно выть, поскольку поступающей энергии будет недостаточно, чтобы преодолеть силу трения и начать вращение.

Если вы собираетесь использовать двигатель для какой-либо практической работы, то с помощью этого скетча весьма удобно определить для него минимальное полезное значение коэффициента заполнения.

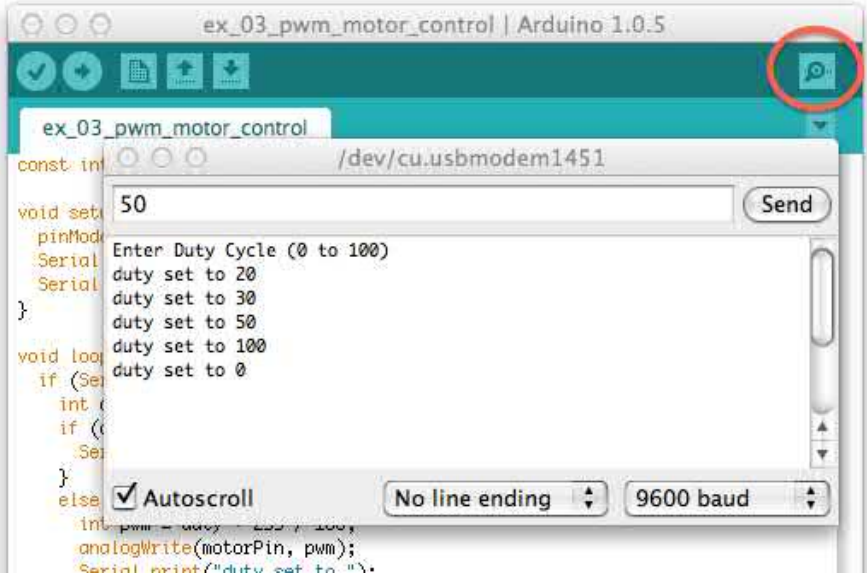


Рис. 7.5. Использование монитора последовательного интерфейса для управления скоростью электродвигателя

Экспериментируем с Raspberry Pi

Подключение Raspberry Pi

Подключите макетную плату к Raspberry Pi, как показано на рис. 7.6: при помощи перемычек «мама-папа» подсоедините заземление с макетной платы к одному из контактов **GND** колодки GPIO, а управляющий провод макетной платы — к контакту **18** Raspberry Pi.

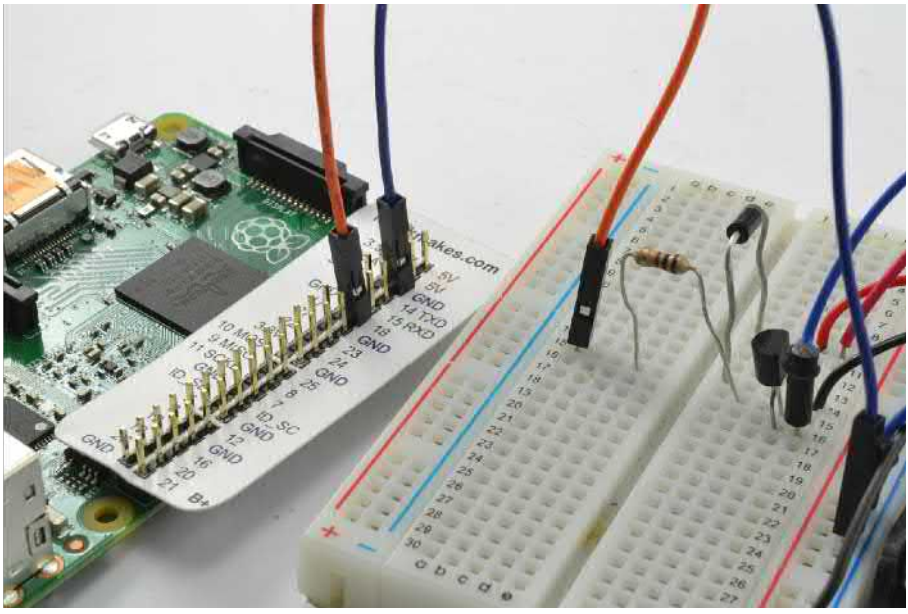


Рис. 7.6. Подключение макетной платы для управления двигателем с Raspberry Pi

Программа для Raspberry Pi

Программа для Raspberry Pi принципиально похожа на код Arduino. В нашем случае программа предложит вам вводить значения коэффициента заполнения и тем самым управлять контактом **18**.

Программа для этого эксперимента находится в файле pwm_motor_control.py каталога python/experiments (подробнее об установке программ Python на Raspberry Pi рассказано в *разд. «Код к книге» главы 3*). Рассмотрим ее код:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

control_pin = 18 // 1
GPIO.setup(control_pin, GPIO.OUT)
motor_pwm = GPIO.PWM(control_pin, 500) // 2
motor_pwm.start(0) // 3

try:
   while True: // 4
      duty = input('Enter Duty Cycle (0 to 100): ')
      if duty < 0 or duty > 100:
         print('0 to 100')
      else:
         motor_pwm.ChangeDutyCycle(duty)
```

```
finally:
   print("Сброс")
   GPIO.cleanup()
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Начальная часть кода точно такая же, как и код из *разд. «Эксперимент: управление электродвигателем» главы 4*, но затем контакт `18` определяется в качестве вывода.
2. В качестве ШИМ-вывода контакт `18` задается в этой строке (библиотека `RPi.GPIO` позволяет в качестве ШИМ-вывода установить любой из контактов GPIO). Параметр `500` означает, что частота ШИМ равна 500 Гц (импульсов в секунду).
3. ШИМ-вывод не начнет работать, пока не будет вызвана функция `start`. Ее параметр — это исходное значение коэффициента заполнения, и поскольку мы начинаем работу с выключенным двигателем, это значение равно `0`.
4. В основном цикле у вас при помощи `input` запрашивается значение `duty`, а затем проверяется, в каком оно диапазоне. Если оно находится в диапазоне от 0 до 100, то используется в качестве коэффициента заполнения в функции `ChangeDutyCycle`.

Загружаем и выполняем программу

Запустите программу в режиме администратора при помощи команды `sudo`, а затем попробуйте вводить различные значения коэффициента заполнения. Скорость двигателя должна меняться, точно так же, как и в случае с Arduino:

```
pi@raspberrypi ~/make_action/python $ sudo python pwm_motor_control.py
Enter Duty Cycle (0 to 100): 50
Enter Duty Cycle (0 to 100): 10
Enter Duty Cycle (0 to 100): 100
Enter Duty Cycle (0 to 100): 0
Enter Duty Cycle (0 to 100):
```

Управление двигателями постоянного тока при помощи реле

Если вы собираетесь с помощью Arduino или Raspberry Pi лишь эпизодически включать и выключать двигатель, то можно воспользоваться реле. Хотя такой прием считается довольно-таки старомодным, у него есть ряд преимуществ:

- он прост в реализации, и для него требуется минимальное количество компонентов;,
- между двигателем, работающим под сильным током и создающим массу помех, и «нежной» конструкцией Pi и Arduino образуется исключительно хорошая изоляция;

- он позволяет работать с сильными токами (если реле подобрано правильно);
- существуют готовые релейные модули, которые можно использовать непосредственно с Raspberry Pi или Arduino.

Тем не менее, в работе с релейными модулями имеются и недостатки:

- они относительно громоздкие;
- позволяют лишь включать или выключать двигатель, но не позволяют управлять его скоростью;
- поскольку реле — это электромеханические устройства, они, как правило, выдерживают не более 10 млн операций срабатывания, а затем в них что-то ломается.

Электромеханические реле

На рис. 7.7 показан, пожалуй, самый распространенный тип реле, так называемый *кубик сахара*, на который оно похоже по форме, но не по цвету, — обычно такие реле черные.

Общий принцип работы подобных электромеханических реле таков: когда через катушку реле проходит ток (силой около 50 мА), она действует как электромагнит и притягивает друг к другу два контакта так, что между ними возникает соединение. Эти контакты рассчитаны на высокое напряжение и сильный ток — они могут переключать десятки ампер.

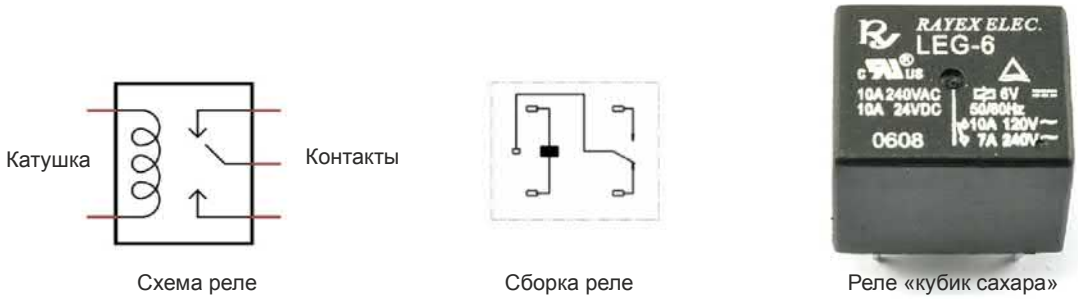


Схема реле Сборка реле Реле «кубик сахара»

Рис. 7.7. Реле

Хотя в этой главе мы будем при помощи реле управлять электродвигателем, само реле действует как обычный переключатель, поэтому с его помощью можно управлять практически чем угодно.

Подобные реле называются *однополюсными переключателями* (от англ. Single Pole Change Over, SPCO), т. к. у них не два контакта, которые могут быть замкнуты либо нет, а целых три: так называемый *общий контакт* (обычно обозначаемый COM) и еще два контакта: *нормально открытый* (NO) и *нормально закрытый* (NC). В данном контексте «нормально» означает «без подачи питания на катушку реле». Соответственно, контакты NO и COM будут открыты (не замкнуты), пока катушка не будет возбуждена. Нормально закрытый контакт действует противоположным образом, т. е. контакты NC и COM в нормальном виде будут замкнуты, но разъединятся, когда катушка окажется возбуждена.

В стандартных ситуациях только два контакта реле — NO и COM — задействуются для включения управляемого им устройства.

Использование реле с Arduino или Raspberry Pi

Если вы используете реле с Raspberry Pi или Arduino, то вам понадобится такое реле, катушка которого рассчитана на напряжение 5 В. Катушки реле потребляют слишком много тока (около 50 мА), поэтому ими нельзя напрямую управлять с Raspberry Pi или Arduino, так что в обоих случаях нам придется включить в схему небольшой транзистор, который позволит работать с катушкой реле на 5 В (рис. 7.8).

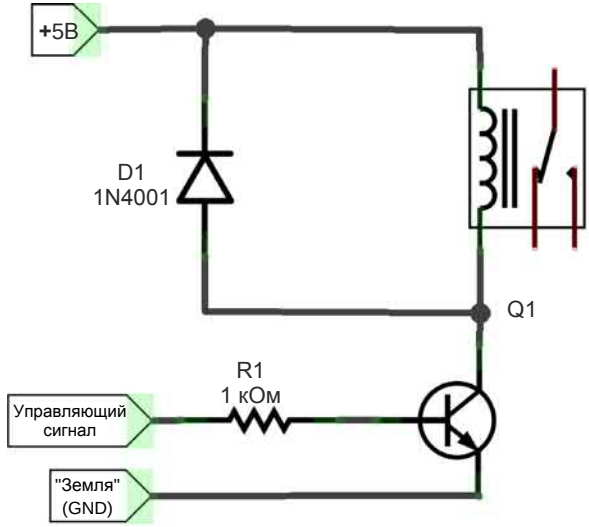


Рис. 7.8. Использование небольшого транзистора для переключения реле

Как уже было сказано, катушка реле рассчитана на работу под напряжением 5 В и требует около 50 мА тока, чего многовато для Arduino и чрезмерно много для GPIO-контакта Raspberry Pi. Поэтому, точно так же, как мы это делали в *разд. «Эксперимент: управление электродвигателем» главы 4*, для управления двигателем мы воспользуемся транзистором, только сейчас катушка реле будет выступать в качестве «нагрузки» вместо двигателя.

Такая структура целесообразна лишь при условии, что двигатель (или любая другая нагрузка, которой вы хотели бы управлять) потребляет достаточно много тока, — настолько, что ею нельзя управлять непосредственно при помощи транзистора.

Подобно двигателю, релейная катушка при включении и выключении может давать всплески напряжения, поэтому в схему требуется включить еще и диод.

Вполне очевидно, что в схеме, показанной на рис. 7.8, коммутирующая часть реле полностью электрически изолирована от части с катушкой. Это снижает вероятность того, что какие-либо электрические помехи, всплески напряжения и прочие пагубные электрические явления доберутся до ваших Arduino или Raspberry Pi.

Поскольку запитанное реле требует всего лишь около 50 мА тока, в нашем случае вполне подойдет бюджетный транзистор 2N3904.

Релейные модули

Если у вас есть несколько устройств, которыми вы хотите управлять, и они удовлетворяют отмеченному ранее ограничению, связанному с управлением от реле (возможно только включение/отключение), то было бы очень удобно обзавестись релейным модулем, — типа такого, что изображен на рис. 7.9.

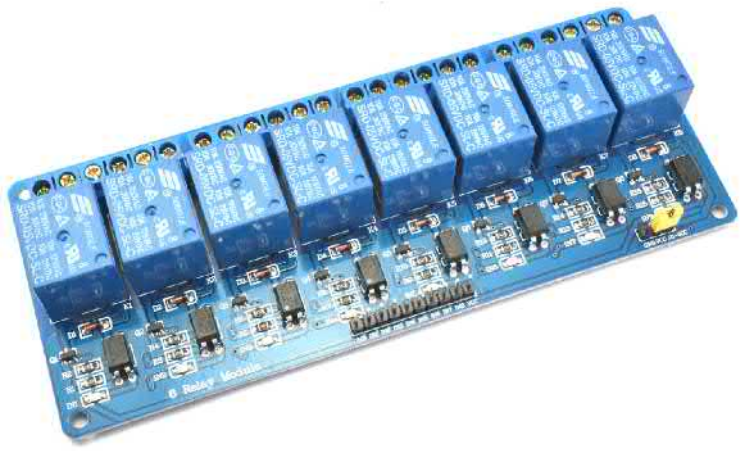


Рис. 7.9. 8-канальный релейный модуль

Эти модули можно очень дешево приобрести на eBay или Amazon. в их состав входят и сами реле, и транзисторы для управления ими, а также небольшие светодиоды, помогающие определить, какое конкретное реле активировано. Учитывая такую их обвязку, подобные релейные модули можно подключать непосредственно к Raspberry Pi или Arduino.

Релейные модули могут содержать различное количество реле: от одного реле до восьми (как показано на рис 7.9) и более.

Как правило, релейные модули имеют такие контакты:

- **GND** — заземление;
- **VCC или 5V** — соедините этот контакт с разъемом 5 В на Raspberry Pi или Arduino, и когда вы его активируете, он сможет подавать напряжение на релейные катушки;
- **Data pins** (информационные контакты) — каждый информационный контакт управляет одним из реле. Иногда эти контакты могут находиться в состоянии active high — это означает, что тот GPIO-контакт, к которому они подключены, будет включать реле при переходе в состояние high. В других случаях информационный контакт может находиться в состоянии active low — при этом катушка реле будет активироваться, когда соответствующий GPIO-контакт находится в состоянии low.

Кроме того, релейные модули несут ряд зажимных винтами клемм, подключенных непосредственно к рабочим контактам реле, замыкающим цепи питания управляемых устройств.

Эксперимент: управление двигателем постоянного тока при помощи релейного модуля

В этом эксперименте мы будем просто включать и выключать электродвигатель при помощи реле.

Здесь задействован готовый релейный модуль на восемь реле, нам же для эксперимента потребуется всего одно реле. Поэтому, если хотите, можете взять модуль и с меньшим количеством реле.

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 7.1).

Таблица 7.1. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению двигателем постоянного тока при помощи релейного модуля*

Компонент схемы	Источники
Небольшой двигатель постоянного тока с напряжением питания 6 В	Adafruit 711
Релейный модуль	eBay
Батарейный отсек 4 AA (6 В)	Adafruit 830
Перемычки, подходящие к релейному модулю	См *приложение 1*

Некоторые релейные модули подключаются к Arduino или Raspberry Pi при помощи разъемов, а другие — при помощи штырьков, поэтому выбирайте подходящие перемычки. Это означает, что для подключения релейного модуля к Arduino при помощи штырьков вам понадобятся перемычки «мама-папа» (Adafruit: 826), а для подключения его к Raspberry Pi — перемычки «мама-мама» (Adafruit: 266).

Схема эксперимента

Схема этого эксперимента показана на рис. 7.10. Релейные контакты действуют в качестве переключателей, каждый из которых может находиться в одном из двух положений. Как уже отмечалось, контакты бывают *общие*, *нормально открытые* и *нормально закрытые*. Когда катушка реле не возбуждена, общий контакт замкнут на нормально закрытый. Когда на катушку подается питание, положение переключателя меняется, и общий контакт замыкается на нормально открытый.

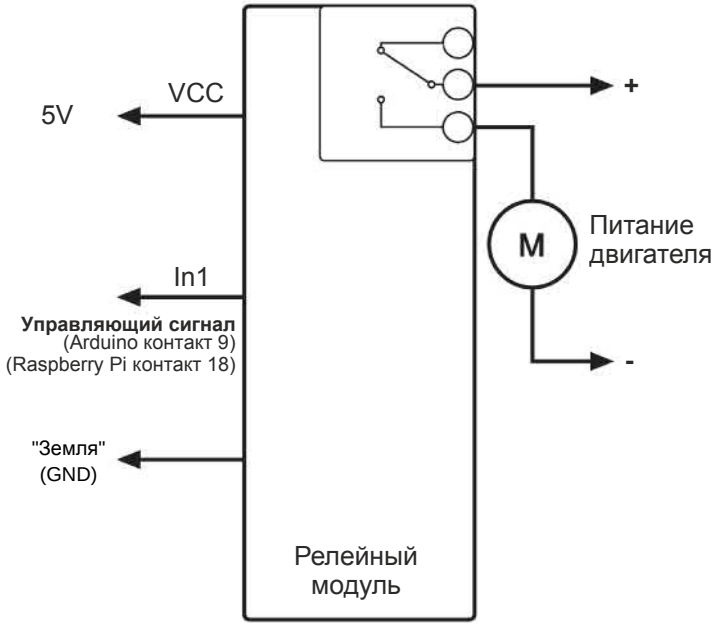


Рис. 7.10. Схема для управления двигателем постоянного тока при помощи релейного модуля

Программы этого эксперимента для Arduino и Raspberry Pi практически идентичны тем, с которыми мы имели дело в *разд. «Эксперимент: управление электродвигателем» главы 4*. Единственное различие возникнет, если у вашего модуля будет логика active low, — как у того, с которым работал я.

Программа для Arduino

Скетч Arduino для этого эксперимента находится в каталоге arduino/experiments/relay_motor_control (см. *разд. «Код к книге» главы 2*).

Программа включит реле (и, следовательно, электродвигатель) на 5 секунд, потом выключит на 2 секунды, затем операция повторится:

```
const int controlPin = 9;

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

void loop() {
  digitalWrite(controlPin, LOW); // 1
  delay(5000);
  digitalWrite(controlPin, HIGH);
  delay(2000);
}
```

В строке, обозначенной комментарием 1, код точно такой же, что и в *разд. «Эксперимент: управление светодиодом» главы 4*, за тем исключением, что в функциях

digitalWrite значения LOW и HIGH поменяны местами. Если вы обнаружите, что при запуске программы двигатель включается на 2 сек. и выключается на 5, то это означает, что логика вашего релейного модуля — active high, и вы должны оставить значения LOW и HIGH в том порядке, в котором они были в *разд. «Эксперимент: управление светодиодом» главы 4*.

Программа для Raspberry Pi

Программа для этого эксперимента находится в файле relay_motor_control.py каталога python/experiments. Рассмотрим ее код:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

control_pin = 18

GPIO.setup(control_pin, GPIO.OUT)

try:
   while True:
      GPIO.output(control_pin, False)
      time.sleep(5)
      GPIO.output(control_pin, True)
      time.sleep(2)

finally:
   print("Сброс")
   GPIO.cleanup()
```

Выбор двигателя

Двигатели постоянного тока бывают всевозможных форм и размеров, и для своего проекта вам важно подобрать достаточно мощный. О двигателе следует знать две основные характеристики: какую силу вращения он позволяет развивать (крутящий момент) и как быстро он вращается.

Если он вращается быстрее, чем вам нужно, но не обладает достаточным крутящим моментом, то можно скомпенсировать этот недостаток, воспользовавшись редуктором.

Крутящий момент

Грубо говоря, *крутящий момент* — это сила вращения двигателя. Чем больше крутящий момент двигателя, тем с большим усилием может вращаться его вал (ротор).

С научной точки зрения крутящий момент определяется как сила, умноженная на расстояние, где сила измеряется в ньютонах (Н), а расстояние — в метрах (м). В более практическом контексте силовая составляющая крутящего момента зачастую выражается как сила, необходимая для подъема определенного веса в килограммах (или унциях) на расстояние, измеряемое в сантиметрах (или дюймах).

Расстояние присутствует в уравнении потому, что с увеличением расстояния от оси ротора двигатель способен прикладывать к грузу все меньшую и меньшую силу. Например, если указано, что крутящий момент двигателя составляет 1 кгс·см (килограмм силы на сантиметр), то на расстоянии 1 см от оси ротора двигатель сможет удерживать вес в 1 кг — т. е. не справится с более значительным весом, но, в то же время, груз с указанным весом не упадет. При этом на расстоянии 10 сантиметров от оси ротора он сможем удерживать вес всего 100 г (10 см × 100 г = 1 кгс·см).

Соотношение между весом и расстоянием от оси двигателя проиллюстрировано на рис. 7.11.

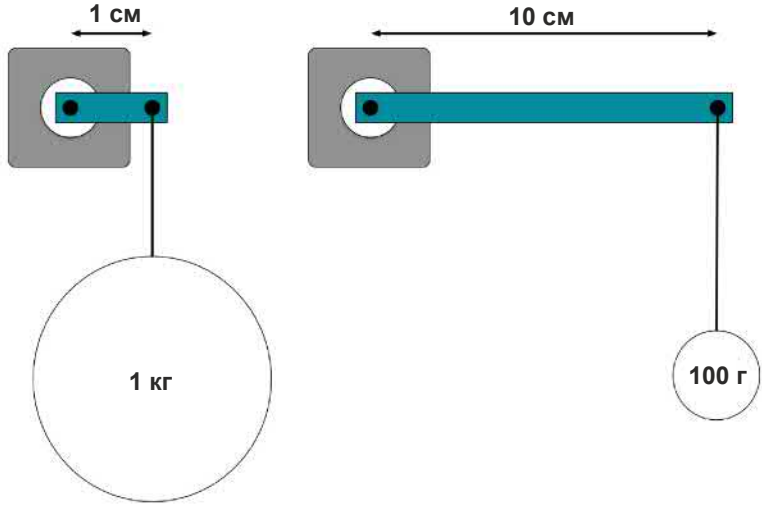


Рис. 7.11. Суть крутящего момента

Скорость вращения

Двигатели постоянного тока вращаются довольно-таки быстро — именно поэтому они часто используются совместно с передачами или в качестве редукторных электродвигателей (см. два следующих раздела). Типичный низковольтный двигатель постоянного тока может вращаться со скоростью 10 тыс. об/мин (оборотов в минуту), или примерно 166 оборотов в секунду.

Передачи

Хотя замедлять скорость двигателя можно с помощью ШИМ, при этом уменьшается подаваемая на двигатель энергия, поэтому генерируемый им крутящий момент также снижается.

Эффективно замедлять вращение двигателей позволяют *передачи*, а в качестве побочного эффекта при этом усиливается крутящий момент. Так, если вы воспользуетесь редуктором 5:1 (рис. 7.12) с 50-ю зубцами на одной шестерне и с 10-ю на другой, то на каждые пять оборотов двигателя будет передаваться всего один оборот из редуктора, но крутящий момент на выходе из редуктора окажется в 10 раз сильнее, чем мы получили бы непосредственно от двигателя.

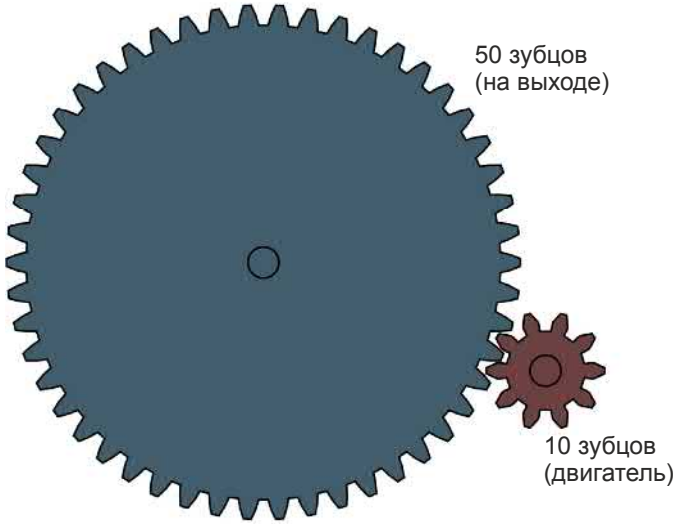


Рис. 7.12. Передачи

Редукторные электродвигатели

Поскольку двигатели так часто и в таких разных случаях используются с передачами, лучше купить *редукторный электродвигатель*, в котором редуктор уже объединен с электромотором.

В некоторых источниках предлагается широкий ассортимент редукторных электродвигателей (см. например, Polulu, **www.pololu.com**).

Самыми дешевыми являются редукторные электродвигатели с пластиковыми шестернями. Они работают, но служат не так долго и не дают столь сильного крутящего момента, как редукторные электродвигатели с металлическими шестернями.

Насосы

Как правило, *насосы* конструируются на основе обычных двигателей постоянного тока или, иногда, на основе бесколлекторных двигателей постоянного тока (они будут рассмотрены в *разд. «Бесколлекторные двигатели постоянного тока» главы 10*). Насос подает энергию на механизм, перекачивающий жидкость из одного резервуара в другой.

Любители часто пользуются насосами двух распространенных типов: шланговыми или динамическими (рис. 7.13).

Рис. 7.13. Шланговый насос (*слева*) и динамический насос (*справа*)

Насосы обоих типов приводятся в движение от двигателей постоянного тока, но свойства их весьма различны: если вам требуется медленный размеренный ход, берите шланговый насос, если быстрый и бурный — то динамический.

Шланговые насосы

Шланговые насосы предназначены для очень размеренного перекачивания жидкости. Они часто используются в медицине и в исследовательской работе, чтобы прокачивать строго определенные объемы жидкости. Для более точного контроля потока шланговые двигатели иногда управляются шаговыми электродвигателями (см. *главу 10*).

Принцип работы шлангового насоса показан на рис. 7.14. В насосе используются двигатель и редуктор, вращающий ось с роликами, сжимающими гибкую трубку и проталкивающими по ней жидкость. Неудивительно, что при постоянном сжатии трубка рано или поздно износится, и ее потребуется заменить. Поэтому шланги в таких насосах обычно сменные.

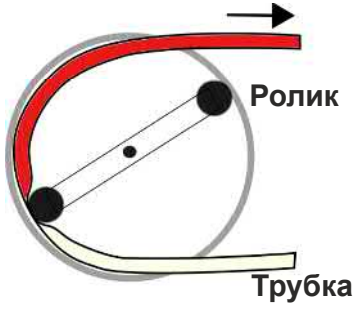


Рис. 7.14. Принцип работы шлангового насоса

Если подавать энергию на редуктор шлангового насоса при помощи ШИМ и Н-моста (мы подробно рассмотрим Н-мосты в *главе 8*), то можно будет управлять не только скоростью, но и направлением потока жидкости.

Шланговый насос является самовсáсывающим — т. е. если он расположен немного выше уровня жидкости, то создает достаточную тягу, чтобы засосать немного жидкости и начать ее качать.

Объемный расход

Объемный расход — это количество жидкости, которое насос может прокачивать в единицу времени. Для измерения этого объема и времени применяются различные единицы. Объемный расход небольшого шлангового насоса может составлять 50 мл/мин (миллилитров в минуту). Динамический насос, предназначенный для полив-ки сада, может иметь объемный расход 5 л/мин (литров в минуту).

Динамические насосы

Если вы стремитесь быстро прокачать большой объем жидкости, то вам понадобится *динамический насос*. Разновидностей таких насосов много, но чаще всего, пожалуй, встречается *центробежный насос* (рис. 7.15).

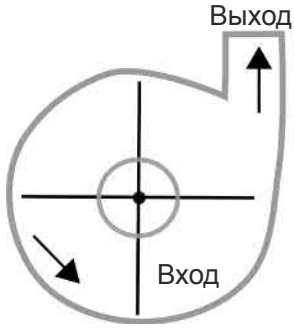


Рис. 7.15. Принцип действия центробежного динамического насоса

Жидкость попадает в насос спереди по оси двигателя, вращающего лопасти. Эти лопасти сообщают центробежную силу жидкости, подступающей к краю насосного корпуса и далее наружу.

Динамические насосы не являются самовсасывающими, и перед перекачиванием жидкости у них на входе уже должна быть жидкость. В отличие от шланговых насосов, динамические могут пропускать через себя жидкость, даже когда не качают ее. Некоторые насосы такого типа предназначены для садового пруда или аквариума и могут быть полностью скрыты под водой. Еще одним отличием динамических насосов от шланговых является то, что они не могут качать жидкость в обратном направлении.

В некоторых насосах динамической конструкции используются бесколлекторные двигатели, прямо в корпусе которых установлена собственная управляющая электроника. За счет этого достигается максимальная сила перекачивания жидкости при небольшом размере насоса.

Проект: домашняя поливальная установка на Arduino

В этом простом проекте для Arduino (рис. 7.16) при помощи шлангового насоса заданный объем воды ежедневно подается на комнатные растения (это удобно, когда вы, например, в отпуске).

Рис. 7.16. Домашняя поливальная установка

Таймер, который позволял бы задать, когда поливать растения, здесь не применяется. Вместо этого устройство измеряет интенсивность дневного освещения, и полив происходит, как только начинает темнеть.

Схема проекта

Принципиальная схема этого проекта приведена на рис. 7.17. Транзистор MPSA14 используется Arduino для включения и выключения двигателя насоса. Диод D1 обеспечивает защиту от отрицательных скачков напряжения.

В левой части схемы мы видим фоторезистор и постоянный резистор — они образуют делитель напряжения для измерения интенсивности света на аналоговом контакте **A0** платы Arduino. Чем больше света попадает в фоторезистор, тем ниже его сопротивление, поэтому на контакте **A0** напряжение возрастает вплоть до 5 В.

Собрать такой проект очень просто. Вряд ли к выходным контактам двигателя вашего насоса будут заранее прикреплены провода, поэтому вся пайка, которую здесь придется выполнить, будет заключаться в припайке к двигателю насоса соответствующих монтажных проводов.

Комплектующие

В этом проекте для работы с Arduino вам понадобятся следующие комплектующие (табл. 7.2).

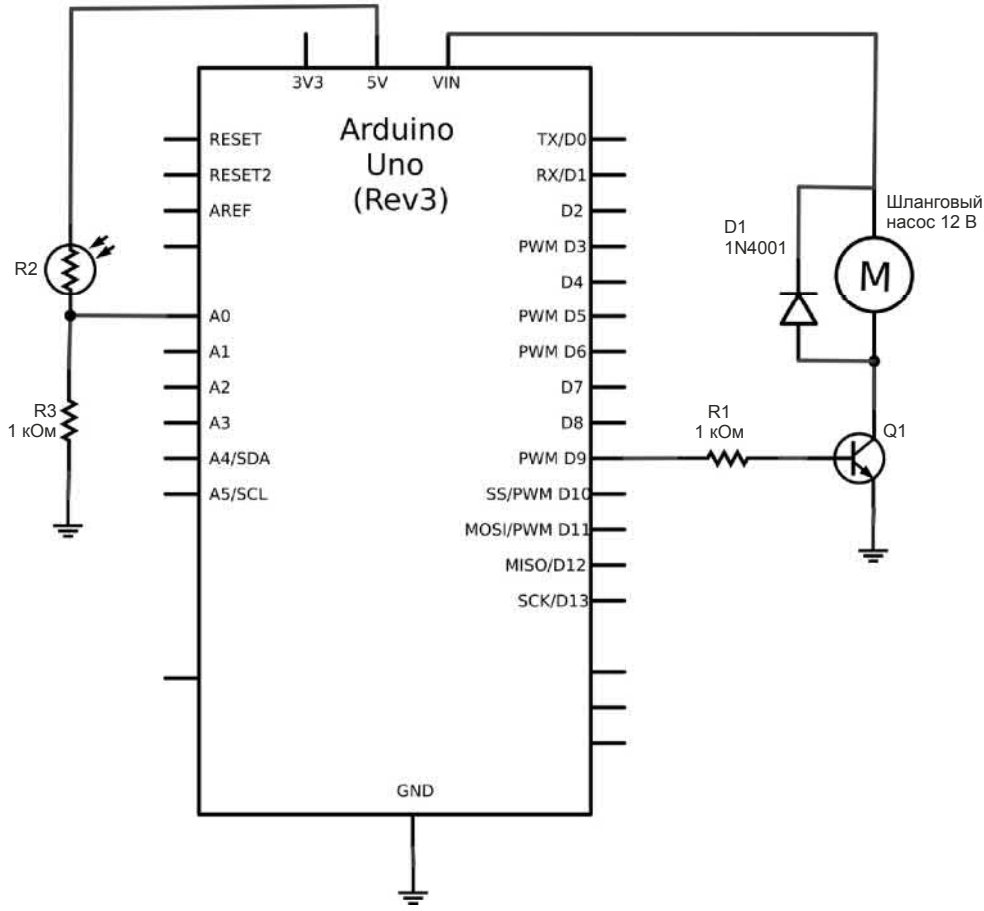


Рис. 7.17. Схема домашней поливальной установки

Таблица 7.2. *Комплектующие для работы с Arduino и Raspberry Pi в проекте домашней поливальной установки*

Обозначение	Компонент схемы	Источники
	Arduino Uno	Магазины электроники
Q1	Составной транзистор MPSA14	Mouser: 833-MPSA14-AP
R1, R3	Резистор 1 кОм	Mouser: 291-1k-RC
R2	Фоторезистор 1 кОм	Adafruit: 161 Sparkfun: SEN-09088
D1	Диод 1N4001	Adafruit: 755 Sparkfun: COM-08589 Mouser: 512-1N4001
	Шланговый насос 12 В	eBay
	400-точечная беспаечная макетная плата	Adafruit: 64

Таблица 7.2 (окончание)

Обозначение	Компонент схемы	Источники
	Перемычки «папа-папа» (только для Arduino)	Adafruit 758
	Трубка для вставки в насос, 1 м	Хозяйственный магазин
	Источник питания 12 В 1 А	Adafruit. 798
	Большая емкость для воды	
	Монтажные провода, припаиваемые к двигателю насоса	Adafruit 1311

Разновидность шлангового насоса, используемого в этом проекте, предназначена для применения в аквариуме, ее можно приобрести за очень небольшие деньги.

Трубка, которую я здесь использовал, досталась мне в комплекте с поливочным инвентарем, который я купил в хозяйственном магазине. Вместе с ней я обзавелся также небольшими пластиковыми соединительными втулками, при помощи которых отрезки трубки соединяются между собой. Соединения трубок должны быть герметичными, иначе насос не станет работать.

Сборка проекта

Чтобы собрать этот проект, потребуется немного поработать с макетной платой, а также вручную надежно закрепить емкость для воды.

Шаг 1. Припаиваем провода к двигателю

Припаиваем монтажные провода к контактам двигателя насоса, если, конечно, их там еще нет. Провода должны быть достаточно длинными и доставать от насоса до макетной платы и Arduino. Вполне хватит полуметровых проводов.

Шаг 2. Собираем макетную плату

Устанавливаем компоненты проекта на макетной плате, как показано на рис. 7.18. Убедитесь, что транзистор и диод установлены правильно.

Шаг 3. Прикрепляем трубку к насосу

Нам понадобятся два отрезка трубки. Один должен погружаться в емкость с водой на всю ее глубину и доставать от посадочного отверстия для насоса, расположенного в верхней части емкости, до самого ее дна. Второй должен доставать от выходного отверстия насоса до того растения, которое вы собираетесь поливать. На рис. 7.19 крупным планом показаны насос и трубки.

Вход и выход насоса обычно никак не помечаются, но шланговые насосы могут качать жидкость в обоих направлениях. Поэтому, если вы заметите, что насос всасывает воду, тогда как должен ее нагнетать, то лучше перепаять монтажные провода на двигателе, чем переставлять трубки.

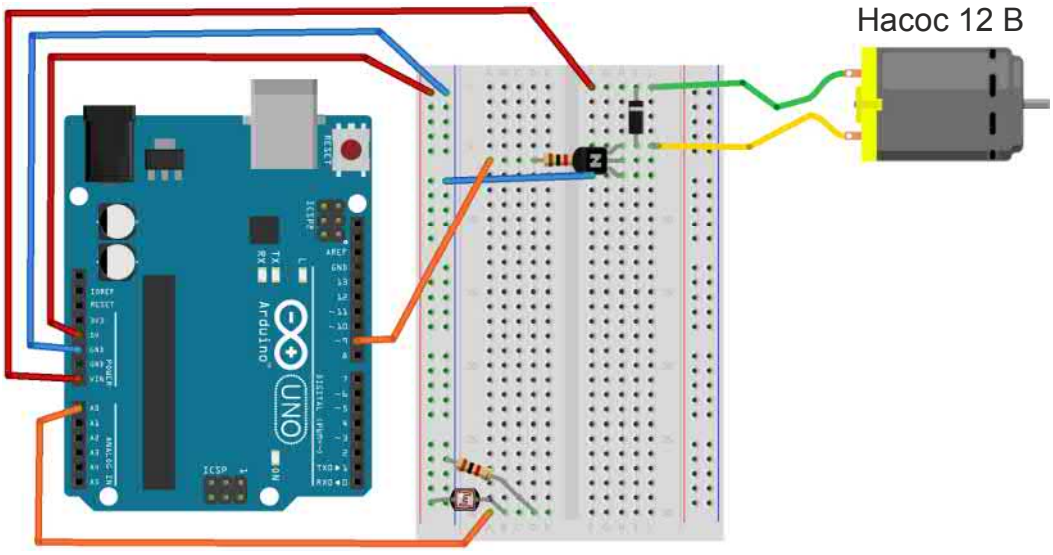


Рис. 7.18. Компоновка макетной платы для сборки домашней поливальной установки

Рис. 7.19. Подсоединение трубок к насосу

Шаг 4. Окончательная сборка

Я решил, что удобно вставить насос в емкость с водой сверху, а двигатель приклеить на насос. Входная трубка тогда опускается прямо в емкость, а выходная торчит сбоку и направлена на растение. Чтобы вставить насос, мне пришлось немного подрезать горлышко бутыли для молока, которой я воспользовался.

На рис. 7.16 показана конструкция, которая получилась у меня, но вы вполне можете установить насос внизу, рядом с макетной платой. Правда, в таком случае вам понадобится более длинная трубка.

Программа

Скетч Arduino для этого проекта (файл pr_01_waterer.ino) находится в каталоге arduino/projects/waterer. В этом же каталоге вы найдете скетч waterer_test, с помощью которого я откалибровал датчик освещенности.

Рассмотрим скетч из файла pr_01_waterer.ino:

```
const int motorPin = 9;       // 1
const int lightPin = A0;

const long onTime = 60 * 1000; // 60 секунд 2
const int dayThreshold = 200; // 3
const int nightThreshold = 70;

boolean isDay = true; // 4

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

void loop() { //  5
  int lightReading = analogRead(lightPin);
  if (isDay && lightReading < nightThreshold) { // стемнело // 6
    pump();
    isDay = false;
  }
  if (!isDay && lightReading > dayThreshold) { // 7
    isDay = true;
  }
}

void pump() { // 8
  digitalWrite(motorPin, HIGH);
  delay(onTime);
  digitalWrite(motorPin, LOW);
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Скетч начинается с определения констант для двух используемых контактов Arduino: контакта для управления двигателем и аналогового входа (`lightPin`), на котором для измерения интенсивности света используется фоторезистор.
2. Константа `onTime` указывает, как долго насос должен каждую ночь оставаться включен. Когда вы тестируете проект, здесь можно задать краткий период, скажем, 10 секунд, чтобы не пришлось слишком долго ждать.

   Самая интересная часть скетча — та, в которой мы определяем, что на улице стемнело. Поскольку плата Arduino не содержит встроенных часов, она не мо-

жет определять время, если, конечно, не добавить к проекту RTC (часы реального времени). В этом проекте я собираюсь поливать растение раз в сутки, поэтому наступление сумерек — самый подходящий фактор, от которого срабатывал бы насос. После полива вы больше не собираетесь поливать растение до следующих сумерек, т. е., пока не закончится светлое время суток (этот период называется «день»).

3. Чтобы было проще различать ночь и день, определяем две константы: `dayThreshold` и `nightThreshold`. Вероятно, эти значения потребуется откорректировать в зависимости от того, где стоит вазон с растением, и насколько чувствителен ваш фоторезистор. Принципиальная идея такова: если текущее значение освещенности выше `dayThreshold`, то сейчас день, а если ниже `nightThreshold`, то сейчас ночь. Может возникнуть вопрос: почему здесь две константы, а не всего одна? Дело в том, что в сумерках, когда только начинает темнеть, уровень освещенности может некоторое время колебаться возле порогового значения, и тогда аппарат сработает несколько раз.
4. Булева переменная `isDay` содержит ответ на вопрос, день сейчас или нет. Если `isDay` равно `true`, то с точки зрения поливальной установки сейчас день.
5. Логика решения о том, наступило ли время для полива, заключена в функции `loop`. Она принимает значение освещенности.
6. Если сейчас день, но значение освещенности оказалось ниже `nightThreshold`, это означает, что только что стемнело — поэтому вызывается функция `pump`, чтобы полить растение. После этого переменная `isDay` получает значение `false`, и это означает: наступила ночь, полив пока следует прекратить.
7. Второй оператор `if` в цикле `loop` проверяет, ночь ли сейчас (`!isDay`), и не превышает ли уровень освещения значение `dayThreshold`. Если оба условия соблюдены, то `isDay` получает значение `true`.
8. Наконец, функция `pump` включает насос, выжидает период, указанный в `onTime`, а затем выключает насос.

Загружаем и выполняем программу

Перед запуском основной части проекта нужно загрузить на Arduino тестовую программу waterer_test.ino — с ее помощью удобно подобрать подходящие значения для `dayThreshold` и `nightThreshold`. Так что, загрузите этот скетч на Arduino и откройте монитор последовательного интерфейса.

Должен появиться ряд значений, которые соответствуют текущему уровню освещенности, — они будут выводиться на мониторе последовательного интерфейса каждые полсекунды. Пометьте, каково значение освещенности днем, когда очень пасмурно. Примерно половина этого значения должна соответствовать `dayThreshold` — тогда наш прибор будет правильно работать и в самые хмурые дни.

Далее дождитесь, когда около вазона стемнеет, и вновь снимите показания. Можно, конечно, попытаться просто угадать либо прикрыть датчик освещения пальцем.

Удвойте полученное значение и задайте результат в качестве `nightThreshold`. Обратите внимание: `nightThreshold` должен быть существенно ниже, чем `dayThreshold`. Возможно, при оценке двух этих значений потребуются какие-то компромиссы.

Теперь можно изменить значения `dayThreshold` и `nightThreshold` в реальном скетче (файл **pr_01_waterer.ino**) и загрузить его в Arduino.

Сымитировать наступление сумерек можно снова прикрыв фоторезистор пальцем, — насос должен сработать и действовать в течение заданного периода.

Насос, с которым я работал, нагнетал около 90 мл/мин. Чтобы определить, сколько времени должен длиться полив, воспользуйтесь секундомером и мерным ковшиком. Так вы узнаете объемный расход вашего насоса и откорректируете значение `onTime`, чтобы растение получало ровно столько воды, сколько нужно.

Линейные исполнительные механизмы

Линейные исполнительные механизмы преобразуют вращение двигателя постоянного тока в линейное движение. Такие устройства часто применяются для дистанционного открытия и закрытия окон и дверей.

В линейных исполнительных механизмах основным элементом является резьбовой ведущий стержень — на него навинчена своеобразная гайка, которая не может прокручиваться, но свободно движется вверх-вниз по стержню, когда он вращается, перемещая рабочую часть исполнительного механизма (вал) вперед-назад. На рис. 7.20 показано, как это происходит, а на рис. 7.21 продемонстрирован типичный линейный исполнительный механизм.

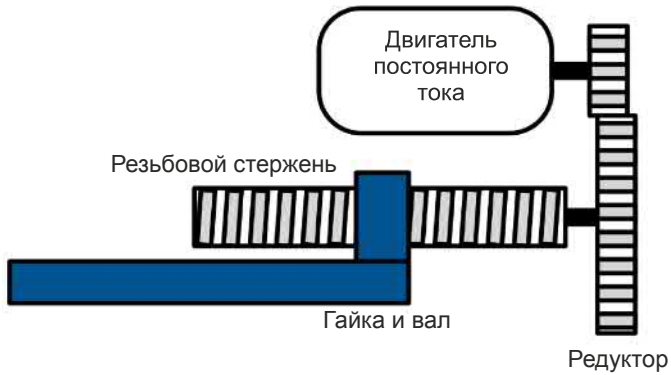


Рис. 7.20. Принцип работы линейного исполнительного механизма

Линейные исполнительные механизмы двигают вал довольно медленно, поскольку резьбовой стержень и гайка фактически действуют как редуктор, а со стороны двигателя обычно устанавливается и обычный редуктор. Поскольку скорость невелика, а двигатель постоянного тока весьма мощный, линейный исполнительный механизм, как правило, дает довольно сильную тягу или толчок. Механизм, показанный

на рис. 7.21, развивает силу 1500 Н (ньютонов) — это аналогично подъему груза в 150 кг. Под полной нагрузкой двигатель подобного линейного исполнительного механизма вполне может принимать ток 5 А при напряжении 12 В.

Рис. 7.21. Линейный исполнительный механизм

Как правило, двигатель линейного исполнительного механизма получает энергию через Н-мост с максимальной силой тока не менее 5 А, благодаря чему может вращаться в обоих направлениях. Чтобы линейный исполнительный механизм не повредился, когда гайка дойдет до одного из концов резьбового стержня, такие устройства обычно оснащены концевыми выключателями, которые автоматически отключают питание, как только гайка упрется в один из концов. При этом управление двигателем упрощается, т. к. можно просто настроить Н-мост так, чтобы он запитывал двигатель при ходе в любом направлении лишь на некоторое ограниченное время — достаточное, чтобы механизм мог пройти весь путь назад или вперед.

В *разд. «Проект: пресс для расплющивания банок из-под газировки на Arduino» главы 8* задействован именно такой исполнительный механизм, как показан на рис. 7.21.

Соленоиды

Соленоиды часто используются совместно дверными щеколдами и несложными водяными клапанами. Подобно линейным исполнительным механизмам, они обеспечивают линейное движение находящегося внутри них штока (плунжера), но соленоид — гораздо более простое устройство. Фактически, это электромагнит, который и обеспечивает поступательное движение штока. Путь его в соленоиде обычно очень короток, — доли сантиметра.

Под действием тока катушка соленоида возбуждается, и магнитные силы выталкивают шток, преодолевая сжатие пружины. Когда же на катушку питание поступать перестает, шток под действием пружины свободно возвращается в исходное положение. Принцип работы соленоида показан на рис. 7.22.

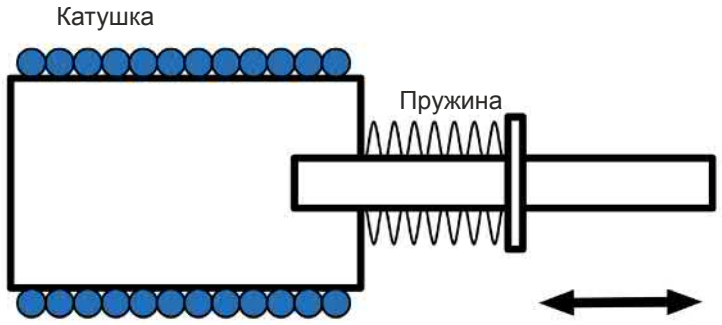


Рис. 7.22. Принцип работы соленоида

Сконструированный на основе соленоида регулирующий водяной клапан, показанный на рис. 7.23, предназначен для переключения подачи водопроводной воды, нагнетаемой под давлением. Когда на клапан не подается питание, вода через него не течет, но при возбуждении катушки плунжер отходит назад, и вода свободно течет через клапан, пока он запитан.

Рис. 7.23. 12-вольтовый регулирующий водяной клапан

Это 12-вольтовое устройство обычно применяется в бытовых стиральных машинах. Можно встретить модели таких клапанов для работы и с переменным током 120 В или 220 В, они также используются в хозяйстве. Работать с большим напряжением при контакте с водой опасно, поэтому в ваших проектах я настоятельно рекомендую ограничиться 12-вольтовыми насосами и клапанами.

Заключение

В этой главе мы разобрались, как работают двигатели постоянного тока, как включать и выключать двигатель при помощи Arduino или Raspberry Pi, как управлять скоростью работы двигателя.

В следующей главе мы рассмотрим работу с электрической цепью под названием *H-мост* — она позволяет управлять направлением вращения электродвигателя.

Расширенное управление электродвигателями

8

В предыдущей главе мы научились управлять скоростью вращения электродвигателя, но пока не умеем менять направление его вращения. В этой главе исследованы различные варианты управления направлением вращения двигателей. В частности, мы рассмотрим ряд специальных интегральных схем и модулей, предназначенных для более простого управления как скоростью, так и направлением вращения двигателей постоянного тока.

Уметь обращать направление вращения двигателя часто бывает весьма удобно. Например, линейные исполнительные механизмы открывают и закрывают окна и двери так: открывая створку двигатель постоянного тока вращается в одном направлении, а закрывая — в противоположном. Точно так же, если вы конструируете небольшого робота на колесиках, то, вероятно, хотели бы, чтобы он катался и вперед, и назад.

Допустим, у нас есть двигатель с двумя выходными контактами: А и Б (рис. 8.1). Когда на контакт А подается положительный потенциал, а на Б — отрицательный, двигатель вращается в одном направлении. Если же поменять полярность соединений, двигатель станет вращаться в противоположном направлении.

Таким образом, если вы хотите управлять направлением вращения электродвигателя, то нужно каким-то образом менять полярность тока, на него поступающего. Этой цели служит специальная электрическая схема, называемая *Н-мостом.*

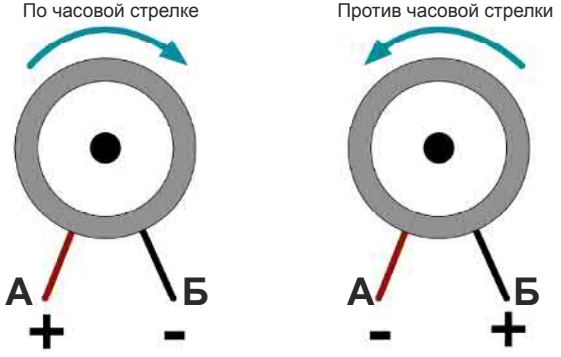


Рис. 8.1. Управление направлением вращения двигателя

Н-мосты

Принцип работы Н-моста показан на рис. 8.2. В этом его варианте разберемся сначала, как для переключения полярности тока используются переключатели, а уж затем перейдем к работе с транзисторами и интегральными схемами.

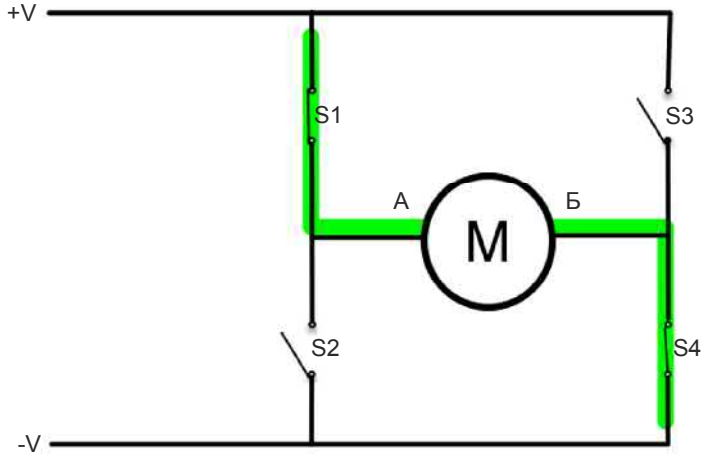


Рис. 8.2. Н-мост, в котором используются переключатели

Итак, если все четыре переключателя разомкнуты, то на двигатель не подается никакого тока. Но если переключатели S1 и S4 — замкнуты, а S2 и S3 — разомкнуты (как показано на рис. 8.2), ток пойдет от положительного полюса источника питания через переключатель S1 к контакту А двигателя, далее через сам двигатель и через переключатель S4 к отрицательному полюсу источника питания, и двигатель станет вращаться в одном направлении.

Если переключатели S1 и S4 разомкнуть, а S2 и S3 — замкнуть, то на этот раз ток от положительного полюса источника питания будет поступать уже на контакт Б двигателя и далее — через сам двигатель и переключатель S2 к отрицательному полюсу источника питания, обратив направление вращения.

В табл. 8.1 показано, как будет работать эта схема при различных положениях ее переключателей: 0 означает, что переключатель разомкнут, 1 — что замкнут (пропускает ток), а × — что состояние переключателя не имеет значения.

Как с помощью переключателей Н-моста изменять направление вращения двигателя мы разобрались. Однако есть еще некоторые комбинации положений переключателей, о которых нужно знать.

- Во-первых, что достаточно очевидно, если все переключатели разомкнуты, то никакого тока через двигатель не пойдет, и он вращаться не будет.
- Особенно важны такие комбинации переключателей, при которых источник положительного питания напрямую соединяется с источником отрицательного. Такая ситуация называется *коротким замыканием* и может привести к катастрофическим последствиям, поскольку в цепи возникнет очень сильный ток.

- Иная ситуация — когда короткого замыкания нет, но контакты двигателя фактически соединены друг с другом. В результате возникает интересный эффект — *торможение двигателя*, когда он быстро замедляется, если только что был в движении, или отказывается вращаться, если уже был в состоянии покоя. И если при этом двигатель вращал колеса, например, игрушечного вездехода, то при помощи торможения этому вездеходу можно не дать скатиться со склона.

Таблица 8.1. *Комбинации переключателей*

Переключатели				Двигатель
S1	S2	S3	S4	
1	0	0	1	Вращается по часовой стрелке
0	1	1	0	Вращается против часовой стрелки
0	0	0	0	Остановлен
1	1	×	×	КОРОТКОЕ ЗАМЫКАНИЕ
×	×	1	1	КОРОТКОЕ ЗАМЫКАНИЕ
1	0	1	0	Торможение
0	1	0	1	Торможение

H-мост на интегральной микросхеме L293D

Среди конструкторов-любителей популярна содержащая H-мост интегральная микросхема L293D (ею мы воспользуемся чуть далее — в *разд. «Эксперимент: управление направлением и скоростью вращения двигателя»*). Эта микросхема отлично подходит для работы с небольшими двигателями, рассчитанными на силу тока не более 600 мА и напряжение до 36 В. Полную информацию о ней можно найти в ее спецификации по адресу: **http://www.ti.com/lit/ds/symlink/l293.pdf**.

Микросхема L293D содержит два H-моста, а также некоторые дополнительные компоненты, позволяющие автоматически отключить интегральную схему, если она начнет перегреваться. И хоть L293D вполне можно сломать, если неправильно использовать, сделать это все-таки весьма непросто.

Основные параметры этой микросхемы таковы:

- напряжение двигателя варьируется от 4,5 до 36 В;
- длительный ток двигателя — 600 мА:
- пиковая сила тока в двигателе — 1,2 А;
- на всех выходах стоят диоды, защищающие от эпизодических всплесков напряжения, производимых двигателем;
- наличие теплозащиты;
- совместимость с 3- и 5-вольтовой логикой (Pi и Arduino).

На рис. 8.3 приведена схема внутреннего устройства и контактов этой микросхемы и показано, как с ее помощью управлять двумя двигателями постоянного тока. Как можно видеть, схема содержит четыре Н-полумоста, а не два полноценных Н-моста. Каждый Н-полумост можно считать мощным цифровым выводом, способным подавать и отводить токи силой до 600 мА. Поэтому работа с этой микросхемой отличается большой гибкостью.

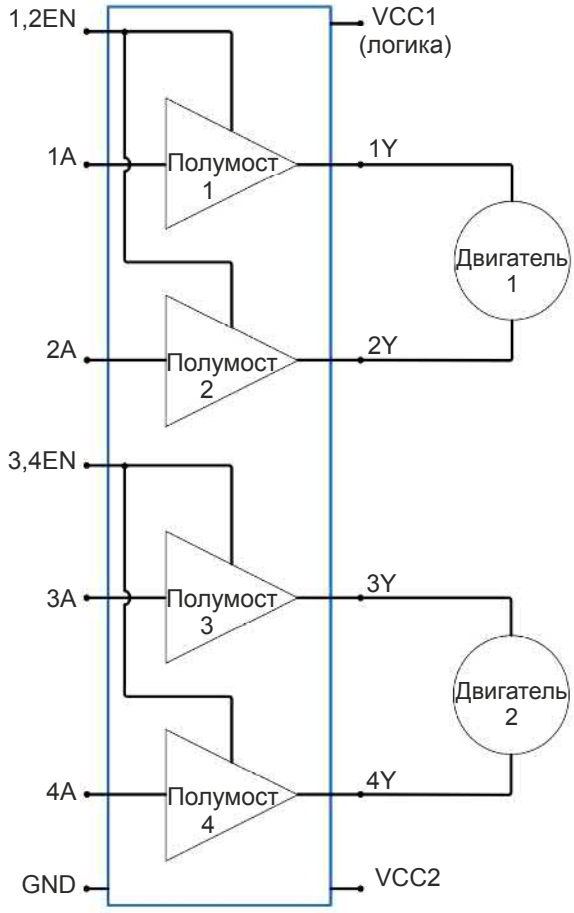


Рис. 8.3. Схема внутреннего устройства и контактов микросхемы L293D

Микросхема оснащена отдельными контактами для приема логических (управляющих) сигналов и для питания двигателей. Это дает возможность, скажем, управлять 6-вольтовым двигателем с помощью логических сигналов 3,3 В от Raspberry Pi (в этом мы убедимся в *разд. «Эксперимент: управление направлением и скоростью вращения двигателя»* в этой главе далее).

Функции всех контактов, показанных на рис. 8.3, представлены в табл. 8.2.

Как и обещано ранее, мы воспользуемся этой микросхемой в *разд. «Эксперимент: управление направлением и скоростью вращения двигателя»* для управления как скоростью, так и направлением вращения двигателя постоянного тока.

Таблица 8.2. Функции контактов микросхемы L293D

Номер контакта	Наименование контакта	Описание
1	1,2EN	Контакт активирует выходы полумостов 1 и 2 (т. е. если он не находится в состоянии high, то выходы будут бездействовать). Он часто применяется с ШИМ-сигналом для управления общей скоростью двигателя
2	1A	Элемент управления вводом для полумоста 1 — если он находится в состоянии high, то в состоянии high будет и выходной контакт 3 (верхний транзистор включен)
3	1Y	Вывод полумоста 1
4, 5, 12, 13	GND	Заземление (на печатной плате все эти контакты припаивались бы к большой контактной площадке, выполняющей роль теплоотвода)
6	2Y	Выход полумоста 2
7	2A	Вход полумоста 2
8	VCC2	Источник напряжения для двигателей, вплоть до 36 В. Разделение логических и питающих контактов обеспечивает более высокую стабильность всей системы
9	3,4EN	Активирует полумосты 3 и 4
10	3A	Элемент управления вводом для полумоста 3
11	3Y	Выход для полумоста 3
14	4Y	Выход для полумоста 4
15	4A	Элемент управления вводом для полумоста 4
16	VCC1	Источник напряжения для логических схем. Здесь напряжение может быть ниже, чем подаваемое на двигатель (контакт 8) и обычно ограничивается 5 В

Эксперимент: управление направлением и скоростью вращения двигателя

В этом эксперименте мы воспользуемся микросхемой L293D, разместив ее на макетной плате. Схема эксперимента в сборе: макетная плата с микросхемой L293D, Raspberry Pi, двигатель постоянного тока, источник питания — показана на рис. 8.4.

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 8.3).

Схема эксперимента будет работать и без конденсаторов, но я рекомендую вам приобрести хорошую привычку работать с конденсаторами, — на случай, если вы решите перейти от экспериментов к реальным проектам.

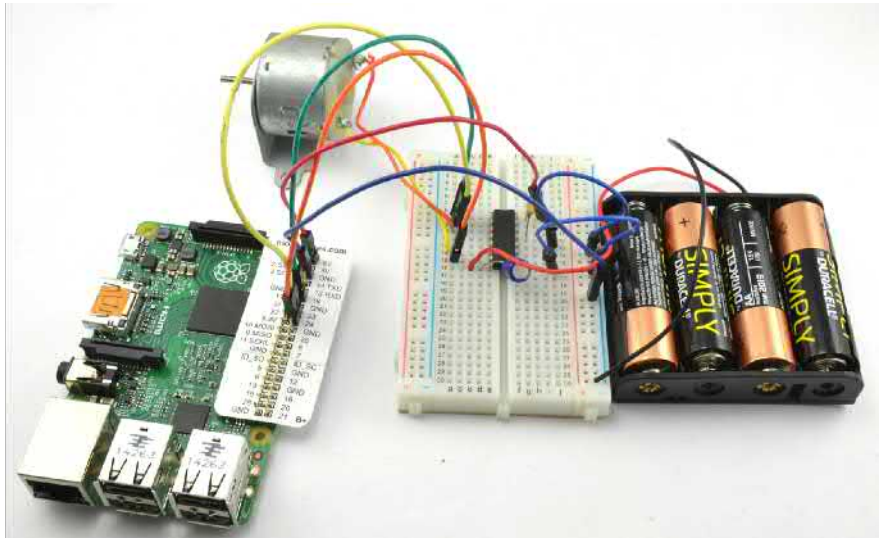


Рис. 8.4. Схема эксперимента по управлению скоростью и направлением вращения электродвигателя с помощью Raspberry Pi в сборе

Таблица 8.3. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению скоростью и направлением вращения электродвигателя с помощью Raspberry Pi*

Обозначение	Компонент схемы	Источники
IC1	Микросхема с H-мостом L293D	Adafruit: 807
		Mouser: 511-L293D
C1	Конденсатор 100 нФ	Adafruit: 753
		Mouser: 810-FK16X7R2A224K
C2	Конденсатор 16 В 100 мкФ	Adafruit: 2193
		Sparkfun: COM-00096
		Mouser: 647-UST1C101MDD
M1	Небольшой двигатель постоянного тока с напряжением питания 6 В	Adafruit 711
	Батарейный отсек 4 AA (6 В)	Adafruit. 830
	400-точечная беспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758
	Перемычки «мама-папа» (только для Pi)	Adafruit: 826

Напомню, что перемычки «мама-папа» понадобятся только для подключения к макетной плате контактов GPIO Raspberry Pi.

Схема эксперимента

Схема этого эксперимента показана на рис. 8.5. Raspberry Pi или Arduino подают управляющее напряжение 5 В на логический контакт 16 микросхемы L293D. Питание для двигателя поступает от 6-вольтового комплекта батареек на ее контакт 8.

Практически здесь используется всего один из H-мостов микросхемы, поэтому ее контакт EN2 подключается к заземлению, чтобы деактивировать неиспользуемую ее часть.

Контакты EN1, IN1 и IN2 подключаются к цифровым выходным контактам Raspberry Pi или Arduino.

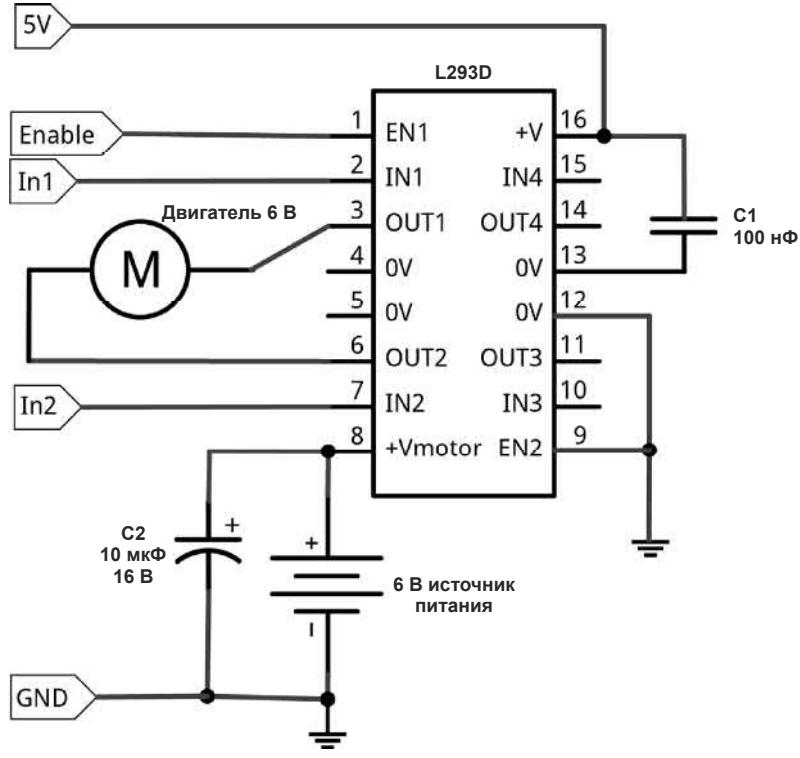


Рис. 8.5. Электрическая схема подключения H-моста микросхемы L293D

Конденсаторы

Как уже отмечалось, использовать в эксперименте конденсаторы не обязательно — если мы собираемся всего лишь поэкспериментировать с этой цепью несколько часов, то нас не слишком интересует та надежность, которую могли бы обеспечить конденсаторы.

Показанное на схеме (см. рис. 8.5) положение конденсаторов весьма типично для интегральной схемы с H-мостом. Конденсатор C1 называется *развязывающим*. Его следует располагать максимально близко к микросхеме — между логическим сигналом и заземлением. Емкость конденсатора C1 может не превышать 100 нФ (это очень мало), однако он поможет избавиться от всяческих электрических помех, которые могут повредить логике микросхемы.

Конденсатор C2 представляет собой как бы резервуар энергии, которого может хватить на какое-то время, — но эта энергия питает именно двигатель, а не тратится на логику переключения. Емкость этого конденсатора обычно гораздо выше, чем у C1, — как правило, 100 мкФ или более.

Компоновка макетной платы

Прежде чем подключать H-мост к Arduino или Raspberry Pi, можно поэкспериментировать со схемой в автономном режиме: проверить и двигатель, и логику переключения, взяв для них питание от одного и того же 6-вольтового комплекта батареек. Такой вариант отлично подходит, если вы хотите опробовать работу микросхемы без подключения к Arduino или Raspberry Pi, но когда дело доходит до использования H-моста с Arduino или Raspberry Pi, лучше разграничить питание — двигатель должен получать энергию от комплекта батареек, а логика микросхемы обеспечиваться энергией от Arduino или Raspberry Pi.

На рис. 8.6 продемонстрирована компоновка макетной платы для автономного эксперимента. Для работы с Arduino или Raspberry Pi эту компоновку переделывать практически не придется — вам потребуется просто переставить некоторые перемычки.

Размещая на макетной плате компоненты, особое внимание уделите микросхеме. Ее обязательно нужно расположить правильно: маленький вырез с одной ее стороны должен быть направлен к верхней части макетной платы в ряду 10. Левее и выше этого выреза должен располагаться контакт 1.

Теперь можно подключить батарею. Изначально двигатель вращаться не должен.

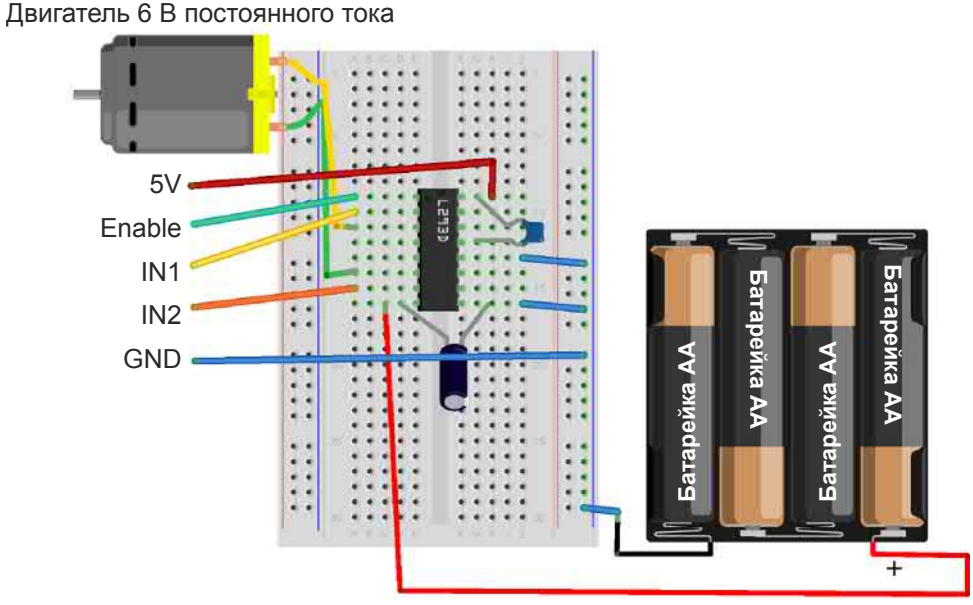


Рис. 8.6. Компоновка макетной платы для автономной проверки H-моста

По часовой стрелке и против часовой стрелки

Когда у меня написано, что мой двигатель вращается по часовой стрелке или против часовой стрелки, ваш двигатель может вращаться как раз в противоположном направлении. И если ваш двигатель вращается против часовой стрелки, когда у меня сказано, что он должен вращаться по часовой, то это не проблема, т. к. он будет вращаться по часовой, когда у меня написано «против часовой».

Если вы хотите, чтобы вращение вашего двигателя совпадало с описываемым, можете поменять местами подводящие провода.

Иногда бывает сложно определить, в каком направлении вращается двигатель, — ведь это просто голый металлический вал. Вы можете его просто пощупать — если аккуратно зажать вал большим и указательным пальцами, то станет понятно, куда он крутится.

Можно также отрезать короткую полоску цветной клейкой ленты и приклеить ее на шпиндель, чтобы она послужила своеобразным флажком.

Автономный эксперимент

Независимо от того, в каком направлении должен вращаться двигатель, — по часовой стрелке или против нее, — контакт Enable должен быть подключен к +V. Чтобы двигатель начал вращаться, например, по часовой стрелке, подключите к +V свободный конец той перемычки, которая уже подсоединена к IN1, а IN2 подключите к ряду GND, расположенному по правому краю макетной платы (рис. 8.7).

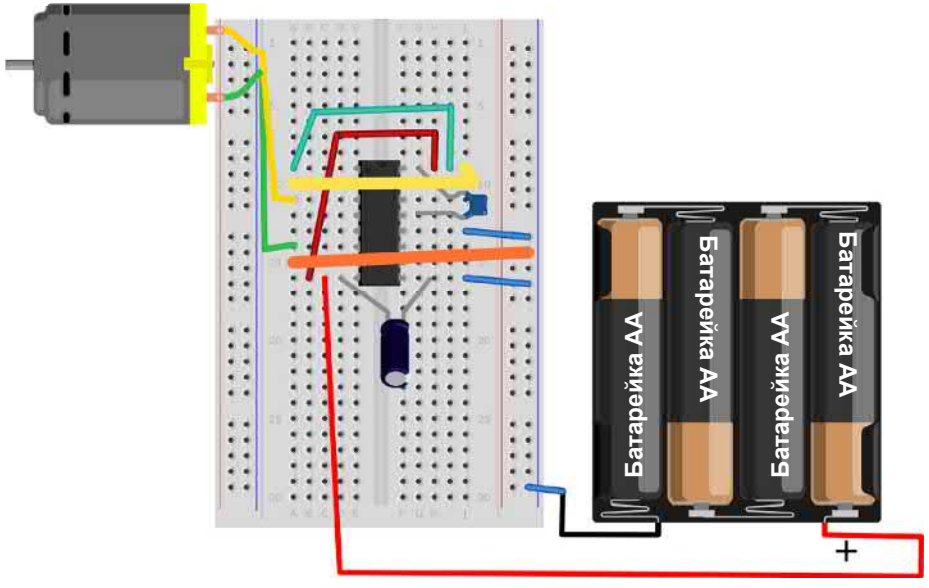


Рис. 8.7. Запускаем двигатель по часовой стрелке

Теперь мы собираемся обратить направление вращения двигателя. Соединение с Enable можно не трогать, но нужно поменять подключение к IN1 и IN2, так, чтобы IN1 теперь соединялся с GND, а IN2 — с +V, как показано на рис. 8.8.

Обратите внимание: на рис. 8.7 и 8.8 перемычки IN1 и IN2 показаны более толстыми, чтобы их было проще отличать от других перемычек.

Теперь, когда мы убедились, что H-мост как таковой работает, можно подключить нашу схему к Arduino (либо, если вы предпочитаете работать с Raspberry Pi, можете сразу перейти к *разд. «Экспериментируем с Raspberry Pi»* в этой главе далее).

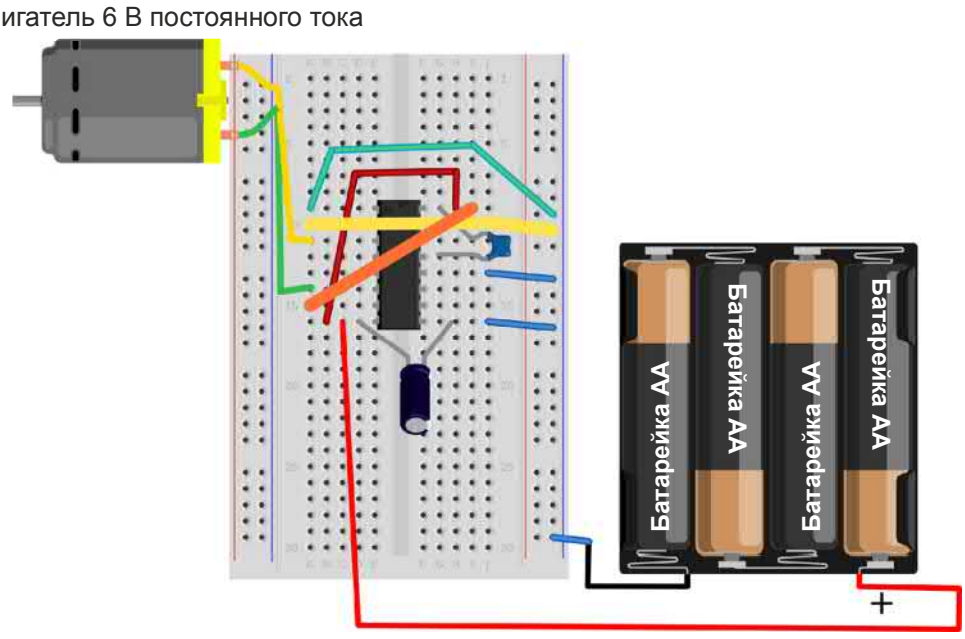


Рис. 8.8. Запускаем двигатель против часовой стрелки

Экспериментируем с Arduino

Подключение Arduino

На рис. 8.9 показано, как следует с помощью перемычек подключать макетную плату к Arduino Uno.

Контакт Arduino под номером **11**, используемый в качестве Enable, поддерживает ШИМ, и мы воспользуемся контактом Enable микросхемы L293D, чтобы управлять скоростью двигателя. К IN1 и IN2 можно подключать любые цифровые контакты Arduino — контакты **10** и **9** были выбраны по той простой причине, что они расположены рядом с контактом **11**, и когда все провода лежат рядом, это выглядит аккуратнее.

Arduino подает напряжение 5 В на логическую часть L293D, но, что важно, при этом мы не запитываем от Arduino двигатель, — питание двигателя по-прежнему осуществляется от батареек.

Макетная плата, подключенная к Arduino перемычками «папа-папа» и готовая к работе, показана на рис. 8.10.

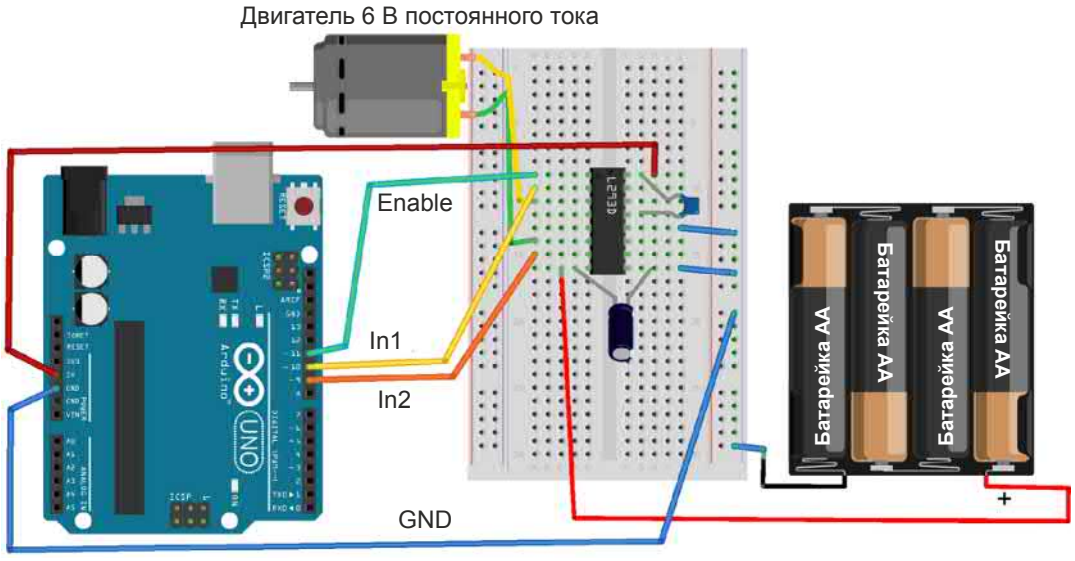


Рис. 8.9. Подключение макетной платы с микросхемой H-моста к Arduino

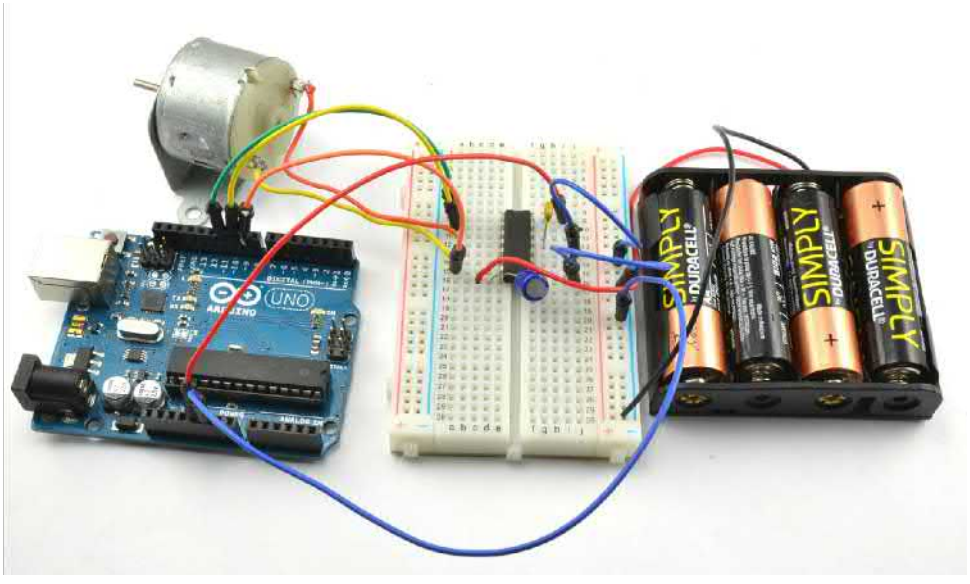


Рис. 8.10. Схема управления двигателем с помощью Arduino в сборе

Программа для Arduino

В той программе для Arduino, которую мы использовали в *разд. «Эксперимент: управление скоростью двигателя постоянного тока» главы 7* (на основе ШИМ), мы пользовались монитором последовательного интерфейса, позволявшим отправлять двигателю команды на изменение скорости. В этом эксперименте та программа дополнена — теперь она отправляет команды, регулирующие не только скорость, но и направление вращения.

Скетч Arduino для этого эксперимента находится в каталоге /arduino/experiments/full_motor_control (см. *разд. «Код к книге» главы 2*). Рассмотрим этот скетч:

```
const int enablePin = 11;    // 1
const int in1Pin = 10;
const int in2Pin = 9;

void setup() { // 2
   pinMode(enablePin, OUTPUT);
   pinMode(in1Pin, OUTPUT);
   pinMode(in2Pin, OUTPUT);
   Serial.begin(9600);
   Serial.println("Введите s (стоп) или f или r для изменения рабочего цикла
                                                    (0 - 255). Например, f120");
}

void loop() { // 3
   if (Serial.available()) {
      char direction = Serial.read();       // 4
      if (direction == 's') {        // 5
         stop();                             // 6
         return;
      }
      int pwm = Serial.parseInt();   //7
      if (direction == 'f') {        // 8
         forward(pwm);
      }
      else if (direction == 'r') {
         reverse(pwm);
      }
   }
}

void forward(int pwm)         // 9
{
   digitalWrite(in1Pin, HIGH);
   digitalWrite(in2Pin, LOW);
   analogWrite(enablePin, pwm);
   Serial.print("Вперед ");
   Serial.println(pwm);
}

void reverse(int pwm)         // 10
{
   digitalWrite(in1Pin, LOW);
   digitalWrite(in2Pin, HIGH);
   analogWrite(enablePin, pwm);
```

```
  Serial.print("Назад ");
  Serial.println(pwm);
}

void stop()          // 11
{
  digitalWrite(in1Pin, LOW);
  digitalWrite(in2Pin, LOW);
  analogWrite(enablePin, 0);
  Serial.println("Стоп");
}
```

Этот скетч относительно длинный, но он хорошо структурирован с помощью функций, на основе которых код удобно видоизменять и заново использовать в собственных проектах. Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Скетч начинается с определения констант для трех управляющих контактов.
2. Функция `setup()` задает эти контакты в качестве выводов, а затем запускает последовательную передачу со скоростью 9600 бод и посылает сообщение-инструкцию монитору последовательного интерфейса, напоминая вам о формате сообщений, применяемых для управления двигателем.
3. Функция `loop()` бездействует, пока `Serial.available` не «доложит», что поступило новое входящее сообщение от монитора последовательного интерфейса.
4. Первый символ сообщения считывается как буква, обозначающая направление, это может быть `s` (стоп), `f` (вперед) или `r` (обратно).
5. Если первая буква — `s`, то вызывается функция `stop`, и команда `return` гарантирует, что никакой последующий код из функции `loop` выполняться не будет.
6. Команда `s` не требует параметра `pwm`, поэтому и мы не пытаемся прочитать его из сообщения — его там просто не будет.
7. Если код направления — `f` или `r` (не `s`), то параметр `pwm` извлекается из оставшейся части сообщения при помощи `parseInt`.
8. Затем вызывается одна из функций: `forward` или `reverse`, в зависимости от того, каков символ `direction`.
9. Функция `forward` устанавливает `in1Pin` в значение `HIGH`, а `in2Pin` — в значение `LOW`, чтобы указать направление вращения двигателя, а затем использует функцию `analogWrite` для управления скоростью двигателя — при этом задействуется `enablePin` и параметр `pwm`, сообщаемый `forward`. Наконец, сообщение с подтверждением действия отправляется обратно на монитор последовательного интерфейса.
10. Функция `reverse` практически аналогична `forward`, за исключением того, что `in1Pin` получает значение `LOW`, а `in2Pin` — `HIGH`, и поэтому двигатель вращается в противоположном направлении.
11. Функция `stop` сбрасывает все управляющие контакты в состояние `LOW`.

Загружаем и выполняем программу

Чтобы проделать эксперимент с Arduino, подключите к Arduino USB-кабель и загрузите скетч. Откройте монитор порта (рис. 8.11), введите команду `f100` в верхней части окна и нажмите кнопку **Send** (Отправить) — двигатель должен начать очень медленно вращаться в каком-то направлении.

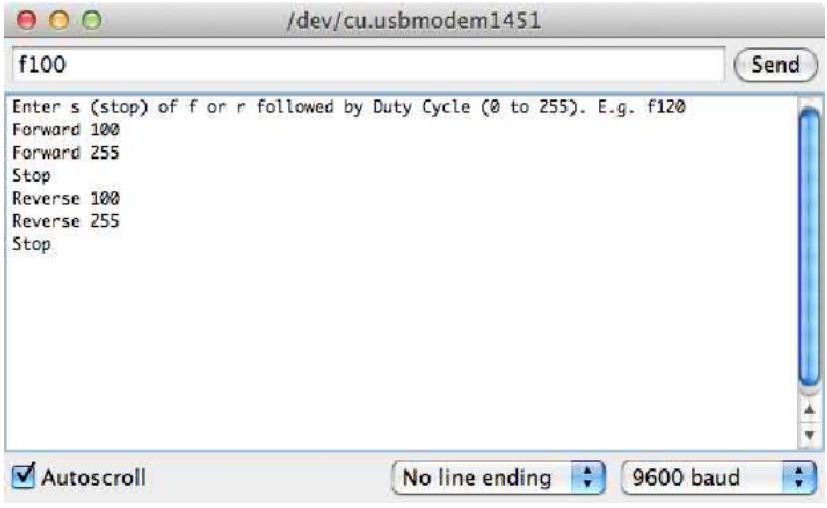


Рис. 8.11. Управление двигателем при помощи команд последовательного интерфейса

Затем попробуйте ввести команду `f255` — двигатель начнет вращаться на полной скорости. Команда `s` остановит двигатель, команда `r100` медленно запустит его в противоположном направлении, а команда `r255` разгонит двигатель в обратном направлении на полную скорость.

Экспериментируем с Raspberry Pi

Подключение Raspberry Pi

Преимущество использования интегральной микросхемы с Н-мостом — такой, как L293D, — в том, что ее управляющим контактам для регулирования работы двигателя требуется очень слабый ток. В ее спецификации указано, что ток неизменно должен оставаться ниже 100 мкА (0,1 мА), а это значит, что можно безо всяких проблем использовать выводы Raspberry Pi, рассчитанные на малые токи.

Если у вас есть и Raspberry Pi, и Arduino, и вы только что закончили часть эксперимента, связанную с Arduino, то, чтобы подготовить макетную плату для работы с Raspberry Pi, потребуется всего лишь заменить перемычки «папа-папа» (ими мы подключали к макетной плате Arduino) на провода «мама-мама» и подключать их к GPIO-колодке Raspberry Pi.

На рис. 8.12 показано, как подключить Raspberry Pi к макетной плате, а готовая к работе система управления двигателем с помощью Raspberry Pi была показана на рис. 8.4.

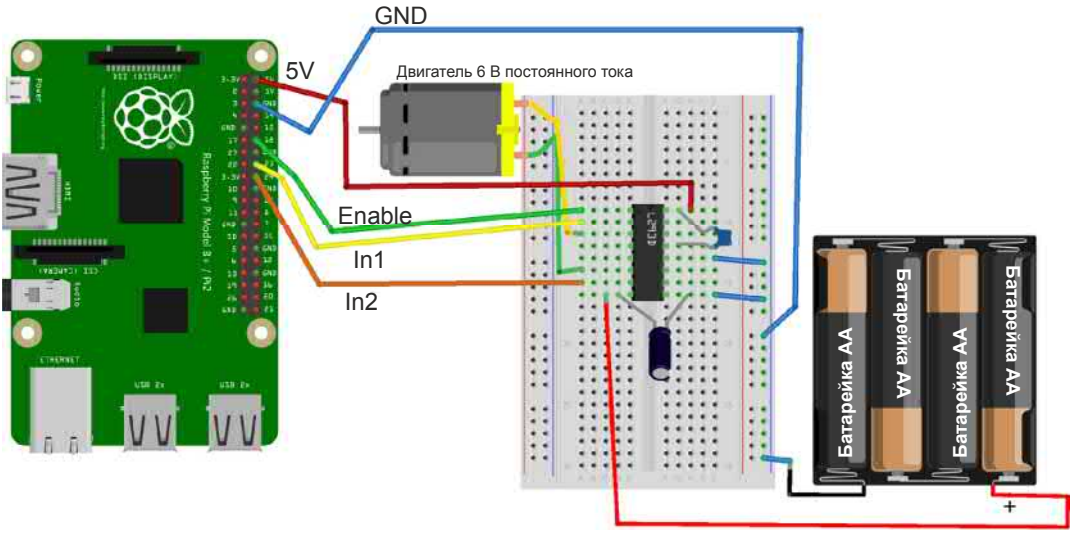


Рис. 8.12. Подключение макетной платы с микросхемой H-моста к Raspberry Pi

Программа для Raspberry Pi

Программа для этого эксперимента находится в файле full_motor_control.py каталога python/experiments. Рассмотрим ее код:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

enable_pin = 18 // 1
in_1_pin = 23
in_2_pin = 24

GPIO.setup(enable_pin, GPIO.OUT)
GPIO.setup(in_1_pin, GPIO.OUT)
GPIO.setup(in_2_pin, GPIO.OUT)
motor_pwm = GPIO.PWM(enable_pin, 500)
motor_pwm.start(0)

def forward(duty): // 2
GPIO.output(in_1_pin, True)
GPIO.output(in_2_pin, False)
motor_pwm.ChangeDutyCycle(duty)

def reverse(duty): // 3
   GPIO.output(in_1_pin, False)
   GPIO.output(in_2_pin, True)
   motor_pwm.ChangeDutyCycle(duty)
```

```
def stop():
   GPIO.output(in_1_pin, False)
   GPIO.output(in_2_pin, False)
   motor_pwm.ChangeDutyCycle(0)

try:
   while True: // 4
      direction = raw_input('Enter direction letter (f - forward, r - reverse,
                                                                  s - stop): ')
      if direction[0] == 's':
         stop()
      else:
         duty = input('Enter Duty Cycle (0 to 100): ')
         if direction[0] == 'f':
            forward(duty)
         elif direction[0] == 'r':
            reverse(duty)

finally:
   print("Сброс")
   GPIO.cleanup()
```

Этот код в значительной мере основан на примерах кода Python из других экспериментов. Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. В верхней части текста программы находится типичный код с установкой GPIO и определениями контактов. Контакт Enable на L293D служит для управления скоростью двигателя, поэтому подключенный к нему контакт `18` конфигурируется как ШИМ-вывод.
2. Функция `forward` настраивает контакты IN1 и IN2 таким образом, чтобы они контролировали направление движения, а затем устанавливает заполнение для ШИМ-канала.
3. Если сравнить ее с функцией `reverse`, то заметно, что значения контактов IN1 и IN2 поменялись местами. Функция `stop` останавливает работу направляющих контактов (оба переходят в значение `LOW`), а коэффициент заполнения становится равен 0.
4. Главный цикл `while` предлагает пользователю ввести команду, а затем, в зависимости от нее, вызывает функцию `stop`, `forward` или `reverse`.

Загружаем и выполняем программу

Запустите программу в режиме администратора при помощи команды `sudo` — и сможете убедиться, что она функционирует практически как аналог на Arduino, предлагая вам задавать скорость и направление вращения двигателя.

```
$ sudo python full_motor_control.py
Enter direction letter (f - forward, r - reverse, s - stop): f
```

```
Enter Duty Cycle (0 to 100): 50
Enter direction letter (f - forward, r - reverse, s - stop): f
Enter Duty Cycle (0 to 100): 100
Enter direction letter (f - forward, r - reverse, s - stop): s
Enter direction letter (f - forward, r - reverse, s - stop): r
Enter Duty Cycle (0 to 100): 50
Enter direction letter (f - forward, r - reverse, s - stop): r
Enter Duty Cycle (0 to 100): 100
Enter direction letter (f - forward, r - reverse, s - stop): s
Enter direction letter (f - forward, r - reverse, s - stop):
```

Аккуратно обращайтесь с двигателем

Представьте себе, что случится, если машина едет на полной скорости, а затем водитель вдруг дает задний ход. В случае с небольшими двигателями, к которым не подключены никакие массивные детали, это обычно проблем не вызывает. Но вы, возможно, обнаружите, что если работать с Arduino или Raspberry Pi, запитанными от того же источника, что и сам двигатель, то Raspberry Pi может аварийно прекратить работу, а Arduino — сброситься. Это происходит потому, что при резкой смене направлений через устройства идет очень сильный ток, из-за чего электропитание резко падает.

В случае с крупными двигателями резкая смена скорости или направления вращения массивной детали, обладающей сильной инерцией, может привести к крупным неприятностям. Мало того, что резко возросшие токи могут повредить Н-мост, так еще и подшипники двигателя подвергнутся механическому сотрясению.

Это необходимо учитывать при разработке управляющего ПО для сравнительно крупных двигателей. Чтобы бережно обращаться с двигателем, перед каждой сменой направления его лучше останавливать (делать паузу, в течение которой он действительно успеет.остановиться) и лишь затем вновь запускать уже в противоположном направлении.

Если в Arduino вы применяете вспомогательные функции `forward` и `reverse` из *разд. «Эксперимент: управление направлением и скоростью вращения двигателя»* этой главы, у вас может получиться примерно такой код:

```
forward(255);
delay(200);
reverse(255);
```

Другие интегральные микросхемы для работы с Н-мостом

На рынке существует множество интегральных микросхем для управления двигателем. В этом разделе мы рассмотрим некоторые из наиболее распространенных.

Интегральная микросхема L298N

Рассмотренная нами ранее интегральная микросхема L293D выдерживает максимальный ток силой 600 мА — это весьма немного. Если вам нужно что-то посильнее, то, к счастью, у L293D есть «старший брат» — интегральная микросхема L298N (**www.sparkfun.com/datasheets/Components/General/L298N.pdf**).

Основные достоинства этой микросхемы таковы:

- напряжение двигателя может достигать 50 В;
- на двигатель постоянно подается ток силой 2 А;
- пиковая сила тока, подаваемого на двигатель, — 3 А;
- совместимость с 3- и 5-вольтовой логикой (Pi и Arduino).

На рис. 8.13 показана схема контактов этой микросхемы. Конфигурация ее очень напоминает L298D, но L298N позволяет, кроме всего прочего, отслеживать силу тока, проходящего через двигатель, — если добавить в схему пару маломощных резисторов (см. далее).

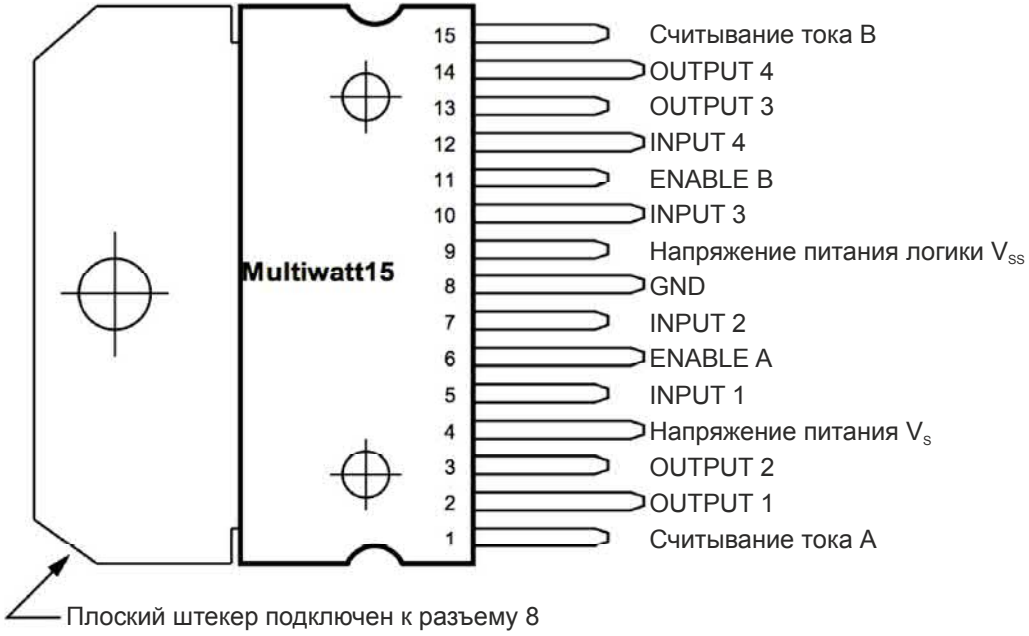


Рис. 8.13. Интегральная схема L298N с двойным H-мостом

Как показано на рис. 8.13, интегральная схема заключена в кожух, привинченный к теплоотводу, — чтобы можно было переключать более высокие мощности. Контакты микросхемы (табл. 8.4) выстроены в два ряда со смещением — таким образом, микросхема не встанет на стандартную макетную плату. Чтобы работать с ней, проще всего купить недорогой готовый модуль для H-моста, в котором эта микросхема как раз установлена, — он стоит всего несколько долларов (см. *разд. «Модули с H-мостом»* в этой главе далее).

Большинство контактов микросхемы L298N точно соответствуют аналогичным контактам микросхемы L293D. Точно так же, как и на L293D, полумосты можно активировать попарно, чтобы управлять мощностью двигателя, когда задействован H-мост, при помощи ШИМ.

Возможность измерять ток, идущий через двигатель, может пригодиться, если, например, вы хотите зафиксировать, что двигатель заклинило (что-то мешает ему

Таблица 8.4. *Схема контактов микросхемы L298N*

Номер контакта	Наименование контакта	Описание
1	CURRENT SENSE A	Контакт А для измерения величины тока на двигателе (см. пояснение под этой таблицей)
2	OUTPUT 1	Выход полумоста 1
3	OUTPUT 2	Выход полумоста 2
4	Vs	Питание для двигателей
5	INPUT 1	Элемент управления вводом для полумоста 1
6	ENABLE A	Активирует полумосты 1 и 2
7	INPUT 2	Элемент управления вводом для полумоста 2
8	GND	Заземление
9	Vss	Питание для логики. Может быть ниже, чем питание для двигателя на контакте 8, обычно составляет 5 В
10	INPUT 3	Элемент управления вводом для полумоста 2
11	ENABLE B	Активирует полумосты 1 и 2
12	INPUT 4	Элемент управления вводом для полумоста 4
13	OUTPUT 3	Выход полумоста 3
14	OUTPUT 4	Выход полумоста 4
15	CURRENT SENSE B	Контакт В для измерения величины тока на двигателе (см пояснение под этой таблицей)

крутиться). Когда такое случается, ток, поступающий на двигатель, резко усиливается. Этот феномен имеет название *ток при заторможенном роторе*. Если вам не требуется измерять ток, идущий через каждый из двух Н-мостов, то контакты 1 и 15 следует подключить к заземлению.

А вот чтобы измерить этот ток, надо подключить между контактом 1 и заземлением, а также между контактом 15 и заземлением резисторы с малым сопротивлением, но рассчитанные на высокую силу тока. Сопротивление резисторов выбирается невысоким, чтобы они не вносили ощутимых помех в работу двигателя. Мощность же резисторов должна быть достаточной, чтобы можно было справиться с той теплотой, которую они будут генерировать. В такой схеме напряжение на резисторе будет пропорционально току, который через него проходит, и это напряжение можно измерить на аналоговом входе Arduino (см. *разд. «Аналоговые входы» главы 2*).

Допустим, мы работаем с двигателем на 12 В, который обычно использует ток силой 500 мА, но при заторможенном роторе через него идет ток силой 2 А. Вполне можно пожертвовать напряжением и снизить его до 0,5 В, чтобы иметь возможность отследить затормаживание двигателя.

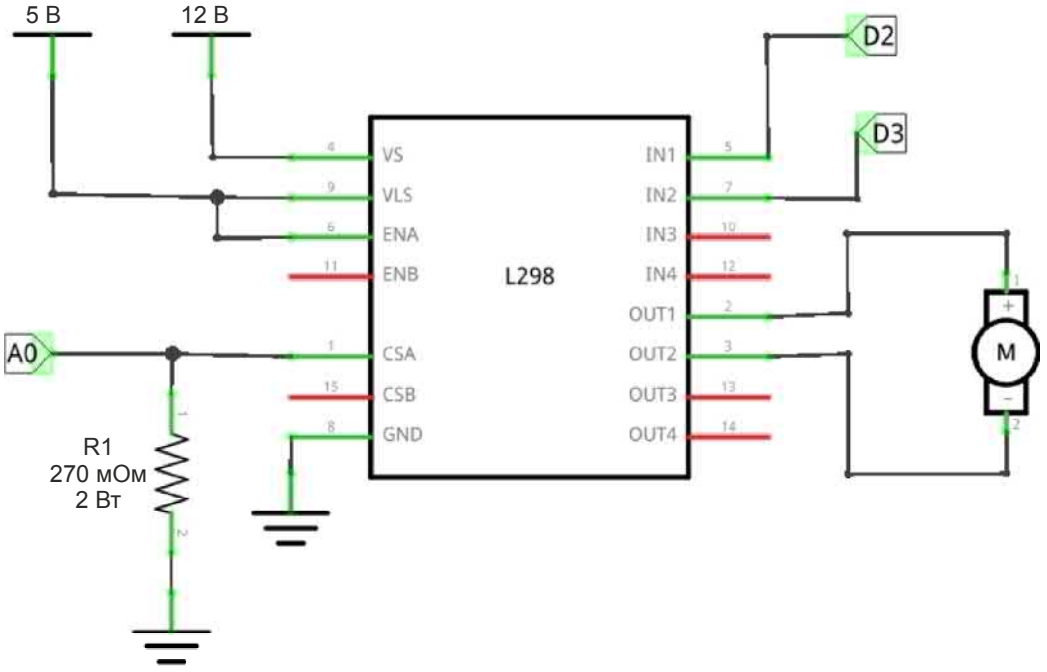


Рис. 8.14. Измерение тока двигателя

На рис. 8.14 показаны микросхема L298N, резистор и двигатель в схеме замера тока для одного из двух H-мостов.

Контакт CURRENT SENSE A микросхемы подключен к аналоговому входу Arduino (допустим, A0), а два контакта: IN1 и IN2 — к цифровым выходам Arduino D2 и D3 и управляют направлением хода двигателя.

Максимальный ток составляет 2 А, а мы стремимся, чтобы напряжение на резисторе составляло 0,5 В. Для этого нам понадобится резистор со значением:

$$R = V/I = 0{,}5/2 = 0{,}25 \text{ Ом}.$$

Скорректировав эту величину до ближайшего стандартного значения, получим резистор с сопротивлением 0,27 Ом или 270 мОм (миллиом! — только не путайте с МОм, мегаом). Таким образом, при силе тока 2А фактическое напряжение составит:

$$V = I \times R = 2 \text{ А} \times 0{,}27 \text{ Ом} = 0{,}54 \text{ В}.$$

Чтобы рассчитать класс мощности резистора, умножим напряжение на нем (0,54 В) на ток (2 А) и получим класс мощности чуть выше 1 Вт. Пожалуй, в таком случае стоит работать с резистором 2 Вт, чтобы у нас был некоторый допуск на ошибку.

Контакт CURRENT SENSE A можно подключить непосредственно к аналоговому входу Arduino (у Raspberry Pi нет аналоговых входов), после чего мы сможем измерить напряжение. Поскольку напряжение ниже тех 5 В, с которыми может работать аналоговый вход Arduino, приблизительное значение `analogInput` при силе 2 А (0,5 В) будет около 100, т. е. по-прежнему сохраняется довольно высокая точность.

В следующем скетче для Arduino показано, как можно запрограммировать Arduino для автоматического отключения двигателя в случае, если сила тока превысит 1,5 А, и произойдет затормаживание двигателя:

```
const float R = 0.27;
const int in1pin = 2;
const int in2pin = 3;
const int sensePin = A0;

void setup() {
   pinMode(in1pin, OUTPUT);
   pinMode(in2pin, OUTPUT);
   // двигатель будет вращаться вперед
   digitalWrite(in1pin, HIGH);
   digitalWrite(in2pin, LOW);
}

void loop() {
   int raw = analogRead(sensePin);
   float v = raw / 204.6; // 204.6 = 1024 / 5V
   float i = v / R;
   if (i > 1.5) {
      // остановить двигатель
      digitalWrite(in1pin, LOW);
      digitalWrite(in2pin, LOW);
   }
}
```

Функция цикла здесь принимает аналоговое значение. Это значение с аналогового входа имеет максимальную величину 1023 при 5 В, поэтому для преобразования приблизительного аналогового значения в значение напряжения его нужно разделить на 204,6, т. е. на $^{1023}/_{5}$. Затем силу тока можно вычислить по закону Ома.

Интегральная микросхема TB6612FNG

Интегральные микросхемы L293D и L298N присутствуют на рынке уже много лет, и сегодня они считаются весьма старомодными устройствами, в которых используются биполярные, а не МОП-транзисторы. Поэтому такие микросхемы обычно разогреваются даже при сравнительно слабом токе.

Гораздо более современной интегральной микросхемой, похожей на L293D по классу мощности, является микросхема TB6612FNG (**www.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf**). Она также оснащена двумя H-мостами.

Максимальное напряжение двигателя при работе с микросхемой TB6612FNG составляет всего 15 В, а общий ток с обоих H-мостов должен держаться ниже 1,2 А, хотя пиковые значения могут достигать 3,2 А. Микросхема также оснащена встроенным механизмом защитного отключения при перегреве.

Интегральная микросхема TB6612FNG доступна только в корпусе для поверхностного монтажа, но можно купить готовые модули, которые используют эту микросхему и при этом легко встают на макетную плату.

Модули с H-мостами

Можно и не конструировать H-мост самостоятельно, а просто воспользоваться готовым модулем со встроенным H-мостом. Такие модули бывают самых разных форм и размеров и рассчитаны на разные токи двигателя. На рис. 8.15 показана подборка модулей с H-мостами для управления двигателями:

- *слева* изображен самый дешевый модуль, который можно купить на eBay (там он стоит пару долларов). В нем используются две микросхемы L9110S, каждая из которых представляет собой единичный H-мост. Четыре зажимные клеммы подключаются к двум двигателям постоянного тока, а колодка на шесть контактов — это заземление (GND), VSS (слабый ток напряжением 5 В) и четыре контакта, задающие направление движения, точно так же, как и на L293D;
- в *центре* показан модуль, использующий микросхему TB6612FNG компании Sparkfun (код товара ROB-09457). К соединительным контактам модуля можно припаять собственные штыревые контакты, чтобы он обрел возможность устанавливаться на макетную плату, или, как показано здесь, воспользоваться колодками с гнездами и перемычками «папа-папа»;
- *справа* представлен модуль, содержащий микросхему L298N, оснащенную теплоотводом. По обеим сторонам теплоотвода в модуле присутствуют защитные диоды, есть даже регулятор напряжения, позволяющий подавать на Arduino напряжение до 5 В. Правда, для использования на Raspberry Pi силы подаваемого тока может не хватить. Резисторов для считывания величины тока, проходящего через двигатель, в этом модуле нет.

Встречаются и более мощные контроллеры для двигателей — искать их удобно на сайте Pololu (**www.pololu.com**). Здесь вы найдете H-мосты, которые могут справить-ся с десятками ампер. Разумеется, чем выше допустимая сила тока, тем дороже прибор.

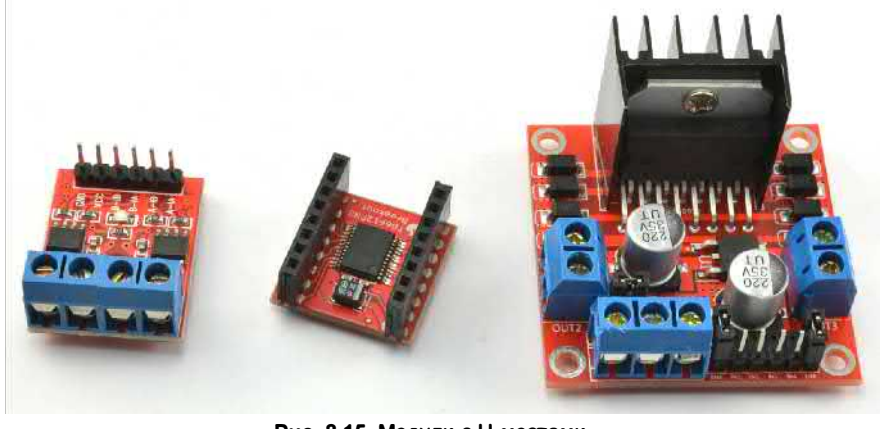


Рис. 8.15. Модули с H-мостами

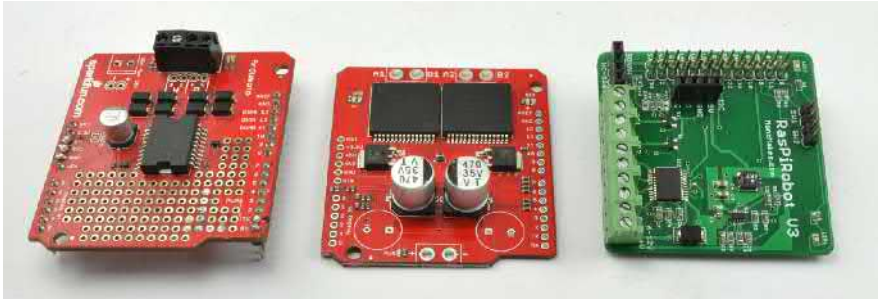


Рис. 8.16. Шилды с H-мостами для Arduino и Raspberry Pi

Можно найти не только отдельные модули с H-мостами, но и полноценные шилды, вставляемые в Arduino, а также платы, подсоединяемые к Raspberry Pi. На рис. 8.16 показаны три такие платы:

- плата *слева* — это шилд Arduino от Sparkfun для управления двигателем (код товара DEV-09815) на основе микросхемы L298P (это разновидность микросхемы L298N, но в корпусе для поверхностного монтажа). На этой плате имеется монтажное поле, к которому можно подсоединять другие компоненты;
- в *центре* изображен мощный шилд с H-мостом, который продается на Polulu, — вас удивит, как легко он управляется с токами до 30 А;
- *справа* показана плата RasPiRobot V3, накладываемая на колодочные контакты Raspberry Pi и использующая микросхему TB6612FNG для управления двумя двигателями постоянного тока.

Проект: пресс для расплющивания банок из-под газировки на Arduino

Помните механизм, о котором шла речь в *разд. «Линейные исполнительные механизмы» главы 7*? Да, его можно подключить к модулю с H-мостом и к Arduino, добавить кое-какие детали из дерева и сделать себе автоматический пресс для расплющивания банок из-под газировки (рис. 8.17).

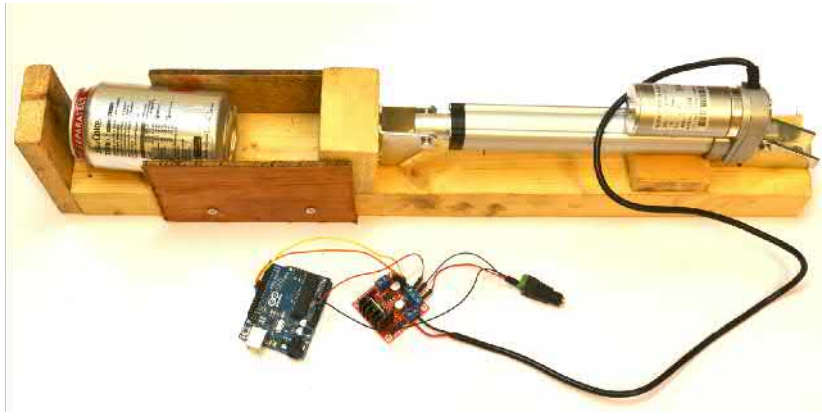


Рис. 8.17. Пресс для расплющивания банок, работающий на Arduino

Прототип такого устройства в действии показан в ролике по адресу **youtu.be/qbWMEIFnq2I**.

> ***Линейные исполнительные механизмы очень сильные!***
>
> В проекте описан прибор для расплющивания жестяных банок, но этот механизм с тем же успехом раздавит и руку, и что угодно еще, что можно по небрежности сунуть под пресс. Поэтому будьте осторожны, особенно когда только начинаете использовать пресс и хотите что-то в нем подправить.

Комплектующие

Кроме Arduino Uno для сборки этого проекта понадобятся следующие комплектующие (табл. 8.5).

Таблица 8.5. *Комплектующие для сборки баночного пресса*

Компонент схемы	Источники
6-дюймовый линейный исполнительный механизм на 12 В	eBay
Модуль L298 с H-мостом	eBay Mouser: 511-L298
4 перемычки «папа-папа»	Adafruit: 758
2 перемычки «мама-папа»	Adafruit: 826
Переходник с круглым гнездом и винтовыми зажимами	Adafruit 368
Источник питания (12 В и не менее 3 А)	Adafruit 352
Небольшие дощечки и немного фанеры	Хозяйственный магазин
Шурупы для дерева и столярные инструменты	Хозяйственный магазин

Приобретая линейный исполнительный механизм, имейте в виду, что стоимость их в зависимости от эксплуатационных характеристик может существенно различаться. Для баночного пресса вам понадобится устройство примерно с 6-дюймовым рабочим ходом. Уточните максимальный ток линейного исполнительного механизма, который берете, а также подберите модуль с H-мостом, который с таким током справится. Линейный исполнительный механизм, с которым работал я, принимает максимальный ток 3А, поэтому я воспользовался H-мостом на основе микросхемы L298.

Подключение

На рис. 8.18 показана схема подключений для этого проекта, а на рис. 8.19 — крупным планом изображены Arduino и модуль с H-мостом.

Модуль L293 имеет соединительные контакты, которые по умолчанию оставляют активными оба H-моста. Таким образом, нам потребуется всего два выхода Arduino, которые будут подключены к контактам IN1 и IN2 модуля.

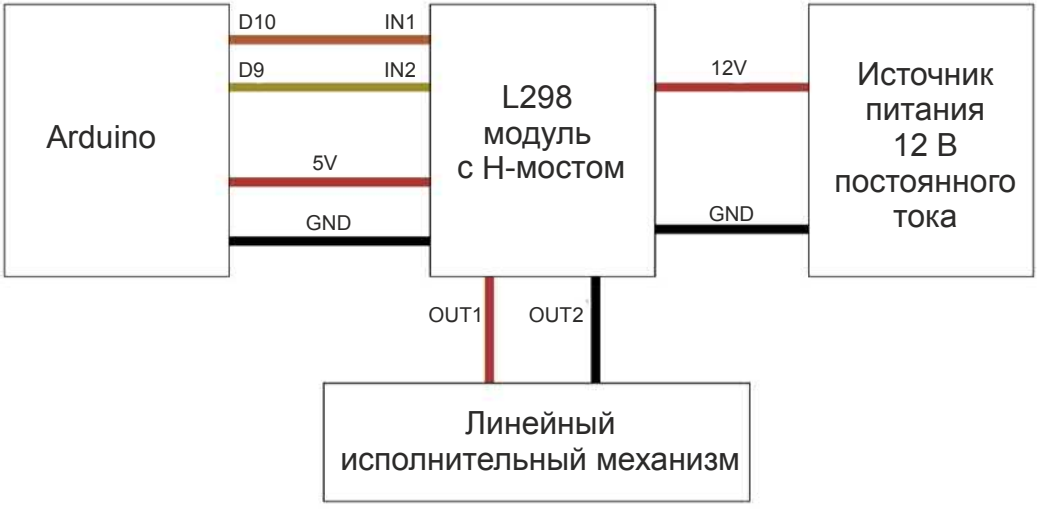


Рис. 8.18. Схема подключения для баночного пресса

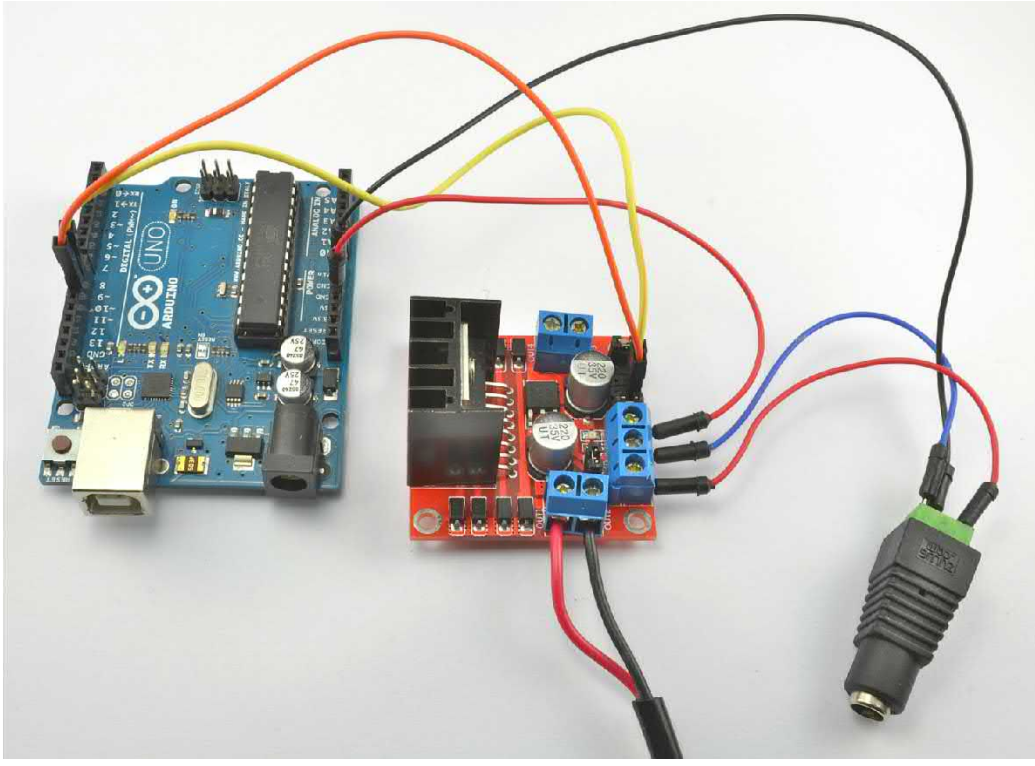


Рис. 8.19. Схема управления баночным прессом на основе Arduino и модуля с Н-мостом в сборе

Удобно, что в модуле с Н-мостом есть и регулятор напряжения, обеспечивающий на выходе 5 В. Его можно подключить к 5-вольтовому контакту Arduino и запитать плату от него.

Механическая конструкция

Как можно видеть на рис. 8.17, проект представляет собой небольшую деревянную конструкцию. На одном конце конструкции установлен исполнительный механизм (при помощи креплений, которые поставляются вместе с ним). Шток исполнительного механизма крепится к деревянной «давилке», расплющивающей банку при движении до упора. Две боковые ограничивающие фанерные планки нужны, чтобы в процессе расплющивания банка не выскочила из пресса.

Я не даю точных размеров, поскольку, вероятно, ваш исполнительный механизм будет немного отличаться по размеру от моего. Лучше всего установить исполнительный механизм на деревянную платформу, а затем замерить все расстояния. Не забудьте оставить небольшой зазор между полностью выдвинутой «давилкой» и упором — в противном случае пресс может просто высадить этот упор.

Программа для Arduino

Пресс срабатывает после нажатия кнопки сброса — т. е. по команде сброса Arduino автоматически запускает исполнительный механизм. Скетч Arduino для этого проекта находится в каталоге arduino/projects/pr_02_can_crusher (см. *разд. «Код к книге» главы 2*). Рассмотрим этот скетч:

```
const int in1Pin = 10;
const int in2Pin = 9;

const long crushTime = 30000; // 1

void setup() { // 2
   pinMode(in1Pin, OUTPUT);
   pinMode(in2Pin, OUTPUT);
   crush();
   stop();
   delay(1000);
   reverse();
   stop();
}

void loop() {
}

void crush()
{
   digitalWrite(in1Pin, LOW);
   digitalWrite(in2Pin, HIGH);
   delay(crushTime);
}
```

```
void reverse()
{
  digitalWrite(in1Pin, HIGH);
  digitalWrite(in2Pin, LOW);
  delay(crushTime);
}

void stop()
{
  digitalWrite(in1Pin, LOW);
  digitalWrite(in2Pin, LOW);
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Хотя линейный исполнительный механизм автоматически останавливается после того, как заканчивает рабочий ход, этот период (в моем двигателе — 30 секунд) определяет, как долго двигатель должен оставаться включенным, прежде чем отвести «давилку» обратно и дать возможность положить следующую банку.
2. Функция `setup` контролирует работу всего проекта. Задав оба управляющих контакта в качестве выводов, программа сразу же запускает процесс. Расплющивание выполняется при помощи функции `crush`.

Заключение

В этой главе вы научились изменять направление вращения двигателя постоянного тока при помощи Н-моста. Н-мосты могут применяться и для управления двигателями других типов, в том числе шаговыми (они рассмотрены в *главе 10*). Н-мосты также позволяют переключать питание при работе с иными устройствами — например, с нагревательными элементами Пельтье и охлаждающими устройствами (они рассмотрены в *главе 11*).

Серводвигатели 9

Серводвигатели (их не следует путать с *шаговыми двигателями*, которые мы рассмотрим в *главе 10*) включают в свой состав небольшой двигатель постоянного тока, редуктор и схему управления, содержащую переменный резистор, дающий возможность установить выходной вал серводвигателя под определенным углом.

Поэтому серводвигатели очень удобны для проектов, где требуется осуществлять весьма быстрое и относительно точное перемещение какого-либо рабочего органа.

Типы серводвигателей

Хотя можно встретить серводвигатели непрерывного типа, способные вращаться постоянно, большинство серводвигателей (их еще часто называют *сервоприводами или сервомашинками*) могут поворачивать выходной вал только в пределах примерно 180°. Они обычно используются в радиоуправляемых моделях автомобилей для поворота рулевых колес или в моделях радиоуправляемых самолетов — для поворота управляющих поверхностей (рулей). На рис. 9.1 показаны два серводвигателя разных размеров.

Серводвигатель *справа* представляет собой так называемый *стандартный* серводвигатель. Это наиболее распространенный тип серводвигателя. Такие серводвига-

Рис. 9.1. Сервопривод 9g (*слева*) и стандартный серводвигатель (*справа*)

тели довольно часто имеют одинаковые размеры и монтажные расстояния между отверстиями. Намного меньший (и более легкий) серводвигатель *слева* предназначен для летательных аппаратов. Эти серводвигатели называются *сервоприводами 9g*.

Сервоприводы с более высоким качеством исполнения и более высоким крутящим моментом имеют редуктор с шестернями из металла, а не из нейлона. Большинство серводвигателей работают на номинальном напряжении питания около 5 В при допустимом диапазоне питающих напряжений от 4 до 7 В. Подключение любительских сервоприводов обычно осуществляется через провода, заканчивающиеся 3-контактным разъемом: питание +, питание – и управляющий сигнал.

Большие и иногда весьма мощные серводвигатели также доступны для использования, но они не так стандартизированы, как любительские маломощные сервомашинки.

Устройство сервопривода

Сервопривод (рис. 9.2) состоит из электродвигателя постоянного тока, приводящего в действие редуктор, уменьшающий скорость вращения двигателя и, в то же время, увеличивающий крутящий момент на валу. Для контроля положения выходного вала он соединен с датчиком положения (как правило, это переменный резистор). Для управления мощностью и направлением, в котором поворачивается двигатель сервопривода, схема управления использует входной сигнал от датчика положения в сочетании с сигналом управления, задающим требуемое положение.

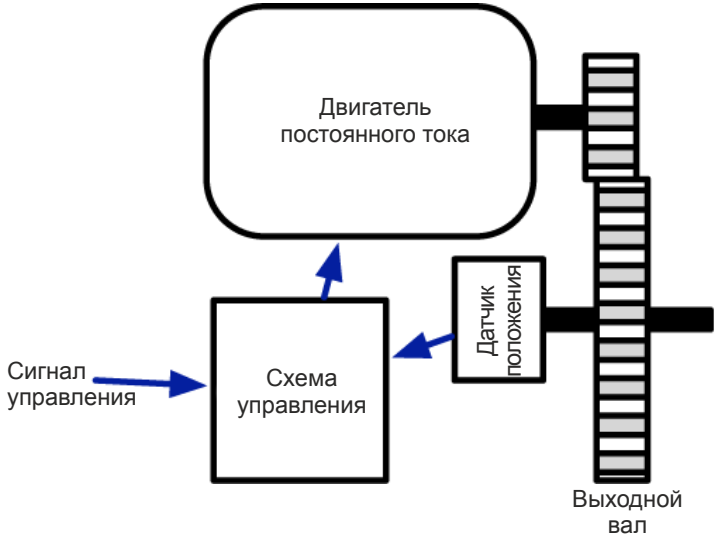


Рис. 9.2. Устройство серводвигателя

Блок управления, получив через сигнал управления величину желаемого положения вала, вычитает из него величину действительного его положения и вырабатывает «сигнал ошибки», который может быть положительным или отрицательным. Этот «сигнал ошибки» подается на питание двигателя, заставляя его изменить положение вала в нужном направлении. Чем больше разница между желаемым и действительным положением выходного вала, тем быстрее двигатель будет поворачиваться к желаемой позиции. Чем ближе к нулю становится значение ошибки (рассогласования), тем меньше становится питание двигателя.

Управление серводвигателем

Управляющий сигнал на серводвигатель — это не напряжение, как можно было бы ожидать, а сигнал широтно-импульсной модуляции (ШИМ). Этот сигнал является стандартным для всех любительских сервомашинок и выглядит так, как показано на рис. 9.3.

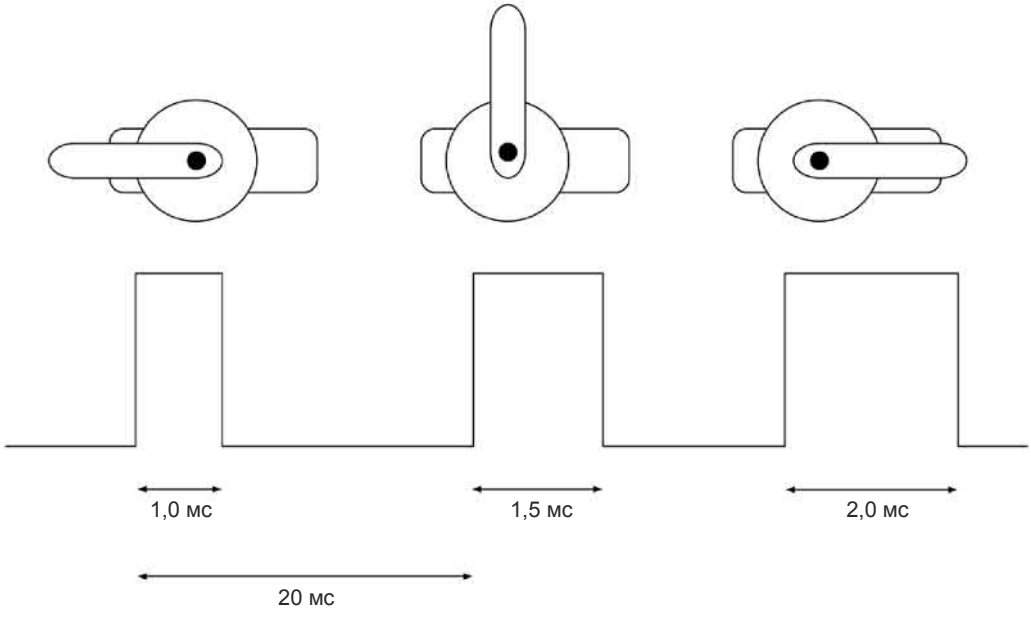


Рис. 9.3. Управление серводвигателем

Серводвигатель ожидает прихода импульса управления каждые 20 мс. Импульс длительностью 1,5 мс установит серводвигатель в центральное положение, соответствующее повороту выходного вала на 90°. Более короткие импульсы в 1,0 мс установят выходной вал в начальное положение — 0°, а импульсы в 2,0 мс — в крайнее положение — 180°. В реальности этот диапазон может быть немного меньше, чем полные 180°, без укорочения импульсов на одном конце и удлинения на другом. Не редкость и ситуация, когда для 0° нужен импульс 0,5 мс, а для 180° — 2,5 мс.

Эксперимент: управление положением серводвигателя

В этом эксперименте мы с помощью сначала Arduino, а затем и Raspberry Pi научимся поворачивать выходной вал серводвигателя на заданный угол.

Для платы Arduino существует библиотека управления серводвигателями, которая способна генерировать импульсы на любом из ее контактов, так что вам не придется задействовать специальный контакт, помеченный как ШИМ-выход. Управление сервоприводом производится через окно монитора последовательного интерфейса

(минитора порта), откуда на плату посылается значение угла, на который нужно повернуть вал двигателя.

Генерация точных по длительности импульсов на Raspberry Pi осуществляется гораздо сложнее, чем на Arduino. Плата Arduino оснащена аппаратными таймерами, которые генерируют нужные импульсы, а Raspberry Pi приходится генерировать импульсы программным способом. При этом, поскольку Raspberry Pi имеет операционную систему, которая позволяет выполняться множеству процессов, конкурирующих за время процессора, длительность ШИМ-импульсов иногда получается большей, чем нужно, что может приводить к небольшому дрожанию выходного вала серводвигателя. Несмотря на то, что Raspberry Pi можно использовать как есть, в случаях, когда необходима большая точность, следует задействовать внешние устройства ШИМ (см. в этой главе далее *разд. «Проект: танцующая кукла Пепе на Raspberry Pi»*).

Оборудование

Замечательной особенностью сервоприводов является то, что управляющая электроника их двигателей заключена внутри корпуса привода, так что нет никакой необходимости в специальных отдельных Н-мостах или транзисторных блоках управления двигателями. Все, что нужно, — это подключить питание 5 или 6 В к контактам питания двигателя и подать слаботочные импульсы управления с цифрового выхода платы Arduino или Raspberry Pi.

На рис. 9.4 показано подключение сервопривода к Raspberry Pi. При этом к сервоприводу уже присоединена одна из качалок (они поставляются вместе с сервоприводом в маленьком пакетике), так что теперь можно увидеть, в каком положении находится вал сервопривода.

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 9.1).

Таблица 9.1. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению сервоприводом*

Компонент схемы	Источники
Серводвигатель 9g	eBay Adafruit: 169
Перемычки «папа-папа»	Adafruit. 758
Перемычки «мама-папа» (только для Pi)	Adafruit· 826

Напомню, что перемычки «мама-папа» понадобятся только для подключения к макетной плате контактов GPIO Raspberry Pi.

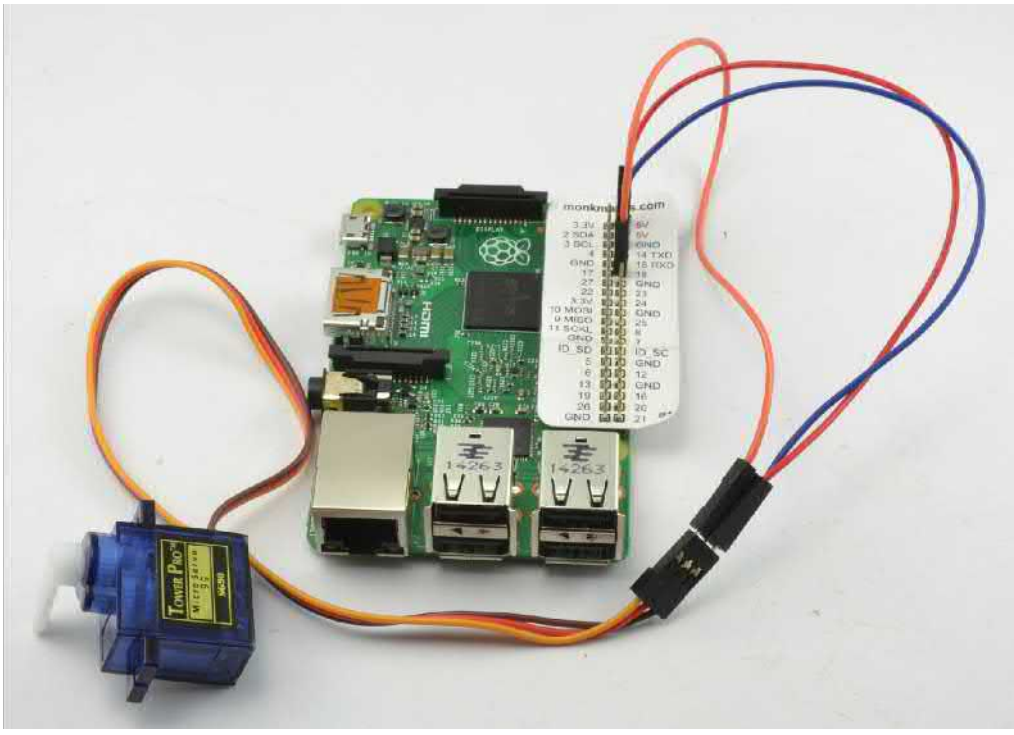


Рис. 9.4. Схема для управления сервоприводом с Raspberry Pi в сборе

Небольшой серводвигатель 9g должен работать просто отлично, получая питание 5 В непосредственно от платы Pi или Arduino. Однако если вы соберетесь управлять более мощным серводвигателем, то вам потребуется внешний источник питания, — например, отсек с батарейками на 6 В, который применялся в предыдущих экспериментах.

Экспериментируем с Arduino

Подключение Arduino

Для подключения 3-контактного разъема серводвигателя к Arduino используются перемычки «папа-папа» (рис. 9.5).

Подключение серводвигателей

На рис. 9.6 крупным планом показан серводвигатель и его разъем. Как и во многих других случаях с серводвигателями, такой разъем является практически стандартом для всех производителей серводвигателей.

Назначение проводов, подключаемых к серводвигателю, определяют по их цвету. Провода почти всегда следуют в таком порядке:

- коричневый (иногда черный) — заземление (GND);
- красный — положительное напряжение источника питания;
- оранжевый (иногда желтый) — управляющий сигнал (Control).

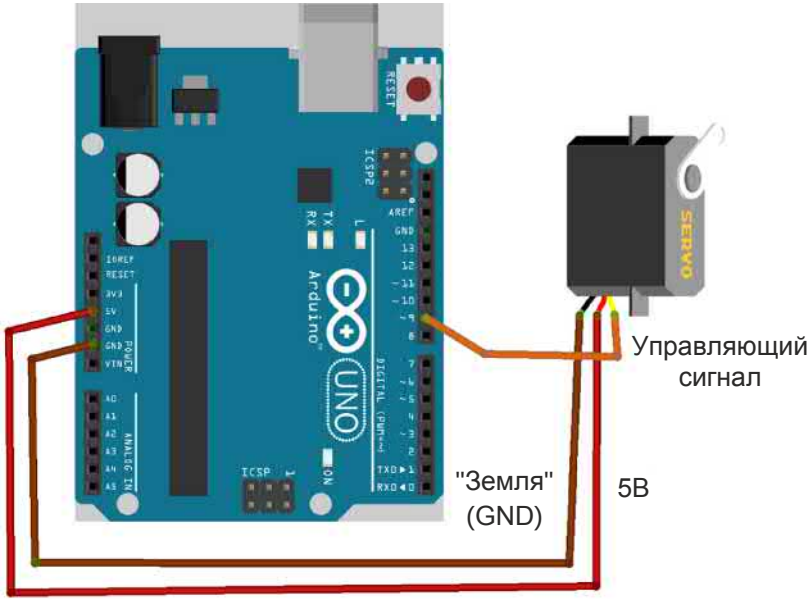


Рис. 9.5. Подключение серводвигателя к Arduino

Рис. 9.6. Серводвигатель и его разъем

Программа для Arduino

Библиотека `servo` для Arduino входит в комплект Arduino IDE и берет на себя всю тяжелую работу по работе с серводвигателями.

Скетч Arduino для этого эксперимента находится в каталоге arduino/experiments/servo (см. *разд. «Код к книге» главы 2*):

```
#include <Servo.h>

const int servoPin = 9; // 1

Servo servo; // 2

void setup() {
   servo.attach(servoPin); // 3
   servo.write(90); // 4
   Serial.begin(9600); // 5
   Serial.println("Введите значение угла в градусах");
}

void loop() { // 6
   if (Serial.available()) {
      int angle = Serial.parseInt();
      servo.write(angle); // 7
```

> ***Примечание***
>
> Вы, разумеется, можете задействовать ШИМ и команду `analogWrite`, чтобы сгенерировать для управления серводвигателем импульс нужной длительности, однако такой подход повлечет за собой изменение частоты ШИМ и ограничит варианты контактов для управления серводвигателем только теми выходами платы Arduino, которые могут работать в режиме ШИМ. Это метод описан далее, в *разд. «Программа для Raspberry Pi»*, но на Arduino проще и лучше использовать библиотеку `servo`.

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. После подключения библиотеки константа `servoPin` определена как вывод, на который будет выдаваться сигнал управления серводвигателем.
2. Объявляется переменная `servo` типа `Servo`. Затем эта переменная используется всякий раз, когда положение серводвигателя необходимо изменить.
3. Устанавливается соответствие выходов, которые будут использоваться для генерации импульсов с использованием `servo.attach`.
4. Задается угол поворота вала серводвигателя в 90° (среднее положение).
5. Запускается последовательный интерфейс, так что теперь можно задать угол поворота вала серводвигателя с помощью окна монитора порта.

6. В цикле основной программы ожидается приход значения угла поворота серводвигателя, после чего оно с помощью `parseInt` преобразуется в целое число.
7. Задается новый угол поворота с помощью `servo.write`.

Загружаем и выполняем программу

Загрузите скетч в Arduino — качалка на валу серводвигателя немедленно займет среднее положение (90°). Откройте окно монитора порта (рис. 9.7) и попробуйте ввести несколько значений угла поворота между 0° и 180°. Серводвигатель будет перемещаться в новую позицию всякий раз, как только будет введено новое значение угла.

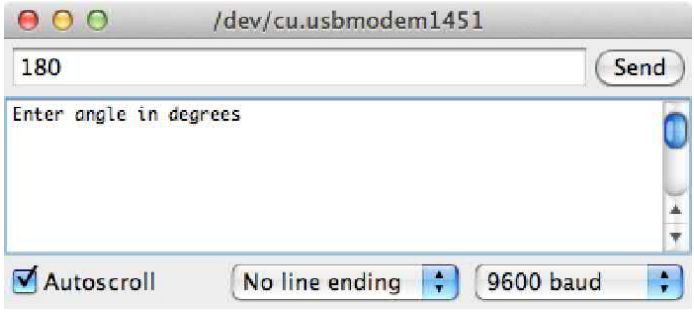


Рис. 9.7. Управление положением серводвигателя с использованием монитора порта

Экспериментируем с Raspberry Pi

Подключение Raspberry Pi

Подключение контактов серводвигателя к GPIO-контактам Raspberry Pi осуществляется с помощью перемычек «мама-папа» (рис. 9.8).

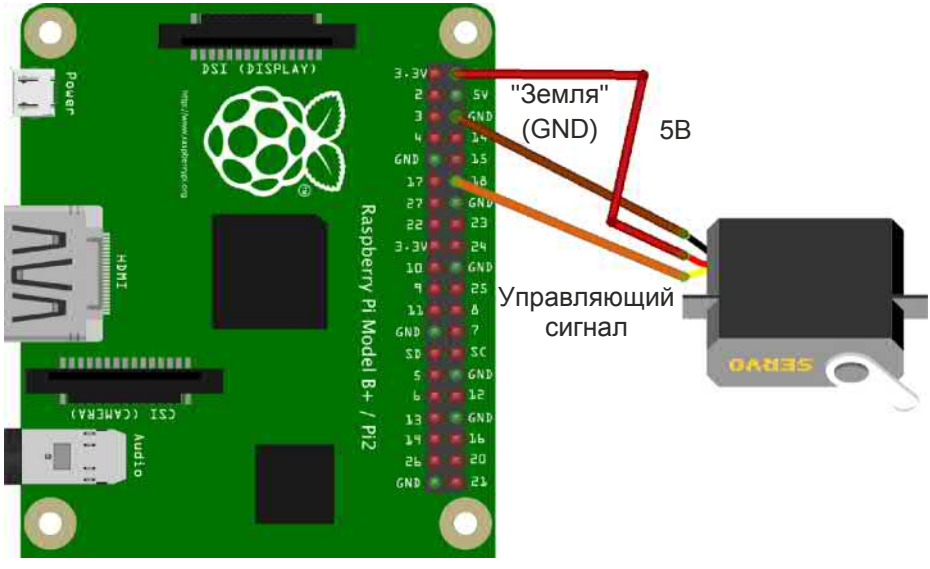


Рис. 9.8. Подключение серводвигателя к Raspberry Pi

Программа для Raspberry Pi

Программа на языке Python для этого эксперимента находится в файле servo.py каталога python/experiments (подробнее об установке программ Python на Raspberry Pi рассказано в *разд. «Код к книге» главы 3*). Программа использует ШИМ-функции библиотеки `RPi.GPIO` для генерации импульсов управления серводвигателем.

Генерация импульса корректной длительности в библиотеке осуществляется с использованием большого количества вычислений. Если у вас нет желания в этом подробно разбираться, но программа нужна для каких-либо ваших проектов, просто скопируйте ее код и вызывайте `set_angle`, чтобы задать угол положения качалки серводвигателя.

```
import RPi.GPIO as GPIO
import time

servo_pin = 18

# Корректируем эти значения, чтобы серводвигатель вращался на полную мощность
deg_0_pulse = 0.5 # ms // 1
deg_180_pulse = 2.5 # ms
f = 50.0 #50Hz = 20ms between pulses // 2

# Вычисляем широту ШИМ
period = 1000 / f # 20ms // 3
k = 100 / period # duty 0..100 over 20ms // 4
deg_0_duty = deg_0_pulse * k // 5
pulse_range = deg_180_pulse - deg_0_pulse
duty_range = pulse_range * k // 6

# Инициализируем контакт GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(servo_pin, GPIO.OUT) // 7
pwm = GPIO.PWM(servo_pin, f)
pwm.start(0)

def set_angle(angle): // 8
   duty = deg_0_duty + (angle / 180.0) * duty_range
   pwm.ChangeDutyCycle(duty)

try:
   while True: // 9
      angle = input("Введите угол (0 - 180): ")
      set_angle(angle)

finally:
print("Сброс")
GPIO.cleanup()
```

Примечание

Каждому серводвигателю нужны немного разные длительности импульсов управления, чтобы обеспечить работу с максимальным диапазоном углов. Для установки импульсов управления на углах поворота 0° и 180° используются две константы: `deg_0_pulse` и `deg_180_pulse` соответственно. Чтобы получить максимальный диапазон движения, эти константы следует подстроить по месту.

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Следующий фрагмент программы производит некоторые вычисления, связанные с длительностью импульса.
2. Подача импульса каждые 20 мс означает, что необходимо установить частоту ШИМ (`f`) равную 50 импульсам в секунду.
3. Период (20 мс) равен 1000, деленной на `f`. Если нужно использовать другую частоту импульсов, следует просто изменить значение `f` . Остальные расчеты будут сделаны автоматически.
4. При изменении цикла ШИМ-сигнала необходимо устанавливать его значения в диапазоне от 0 до 100. Константа `k` изначально определена как 100 за период и может быть использована для масштабирования величины угла до рабочих значений.
5. Чтобы преобразовать величину длительности импульса, соответствующего нулевому углу, к рабочему диапазону от 0 до 100, длительность импульса умножают на `k`.
6. Аналогичным образом, диапазон рабочих значений также вычисляется путем умножения периода следования импульсов `pulse_range` на `k`.
7. Настраиваются выводы GPIO, и ШИМ начинает работать.
8. Функция `set_angle` преобразует величину угла в значение рабочего цикла, а затем вызывает `ChangeDutyCycle` для задания новой длины импульса.
9. Главный цикл очень похож на версию аналогичной программы для Arduino: значение угла сначала запрашивается, а затем пересчитывается и выдается.

Загружаем и выполняем программу

Запустите программу следующей командой:

```
$ sudo python servo.py
```

Используйте окно монитора порта для управления серводвигателем так же, как это было в случае с Arduino, — вводите углы и наблюдайте, как серводвигатель уверенно занимает заданную позицию:

```
$ sudo python servo.py
Enter angle (0 to 180): 90
Enter angle (0 to 180): 0
Enter angle (0 to 180): 180
Enter angle (0 to 180): 90
Enter angle (0 to 180): 0
```

По большей части серводвигатель будет реагировать на команды плавно, но если проследить за ним внимательно некоторое время, то можно заметить небольшую дрожь, особенно если Raspberry Pi в это время выполняет множество каких-либо других задач. Этого следовало ожидать, т. к. чем длиннее импульс, генерируемый Raspberry Pi, тем чаще процессор будет отвлекаться на выполнение чего-нибудь еще.

Если дрожание серводвигателя окажется слишком велико, то альтернативой программной генерации импульса на Raspberry Pi может стать использование аппаратных модулей, — таких как 16-канальная плата-драйвер PWM/Servo компании Adafruit (код товара 815). Этот модуль задействует только два сигнальных контакта Raspberry Pi для взаимодействия с ним и при использовании библиотеки Python, также поставляемой Adafruit, позволяет управлять 16-ю серводвигателями.

Проект: танцующая кукла Пепе на Raspberry Pi

Серводвигатели очень быстро реагируют на команды смены позиции. В этом проекте это их свойство используется, чтобы потянуть марионетку за ниточку и заставить ее танцевать или двигаться в любом желаемом направлении (рис. 9.9).

Управление сервоприводами здесь строится на пошаговом проигрывании списка позиций их рабочих органов. В *разд. «Проект: кукла Пепе обретает голос» главы 15* марионетка получит возможность говорить, а в дальнейшем, благодаря выходу в Интернет, сможет исполнить танец, заданный определенным хештегом в Твиттере.

Комплектующие

Для реализации этого проекта на Raspberry Pi понадобятся следующие комплектующие (табл. 9.2).

Таблица 9.2. *Комплектующие для работы с Raspberry Pi в проекте по управлению танцующей куклой Пепе*

Компонент схемы	Источники
16-канальный 12-битный модуль (драйвер) PWM/Servo	Adafruit, 815
4 серводвигателя 9g	eBay
Перемычки «мама-мама»	Adafruit, 266
4 зубочистки (длиной около 7–8 см каждая)	Универмаг
Небольшая кукла (с бечевкой для каждой конечности)	eBay

Таблица 9.2 *(окончание)*

Компонент схемы	Источники
[redacted]	[redacted]
Лист монтажного картона, формат А4	Канцелярский магазин
Клеевой пистолет или эпоксидный клей и дрель	Хозяйственный магазин

Рис. 9.9. Кукольный Пепе

Схема проекта

В проекте задействована плата Raspberry Pi в основном потому, чтобы ее можно подключить к Интернету, как будет показано в *главе 16*. Тем не менее, можно использовать и Arduino. В этом случае внешние модули PWM/Servo Adafruit нам не

понадобятся, а подключить сервоприводы к Arduino надо будет с помощью макетной платы и перемычек «папа-папа». Однако потребуется целая куча таких перемычек!

Управляются руки и ноги куклы четырьмя сервоприводами 9g — по одному на каждую руку и ногу. Управление сервоприводами осуществляется с помощью 16-канальной контроллерной платы (драйвера) PWM/Servo компании Adafruit. Дополнительное преимущество такой конструкции заключается в том, что разъемы на проводах серводвигателей могут быть напрямую подключены к разъемам контроллерной платы (на рис. 9.10, чтобы не запутать вас, показан только средний провод, идущий к плате от каждого сервопривода).

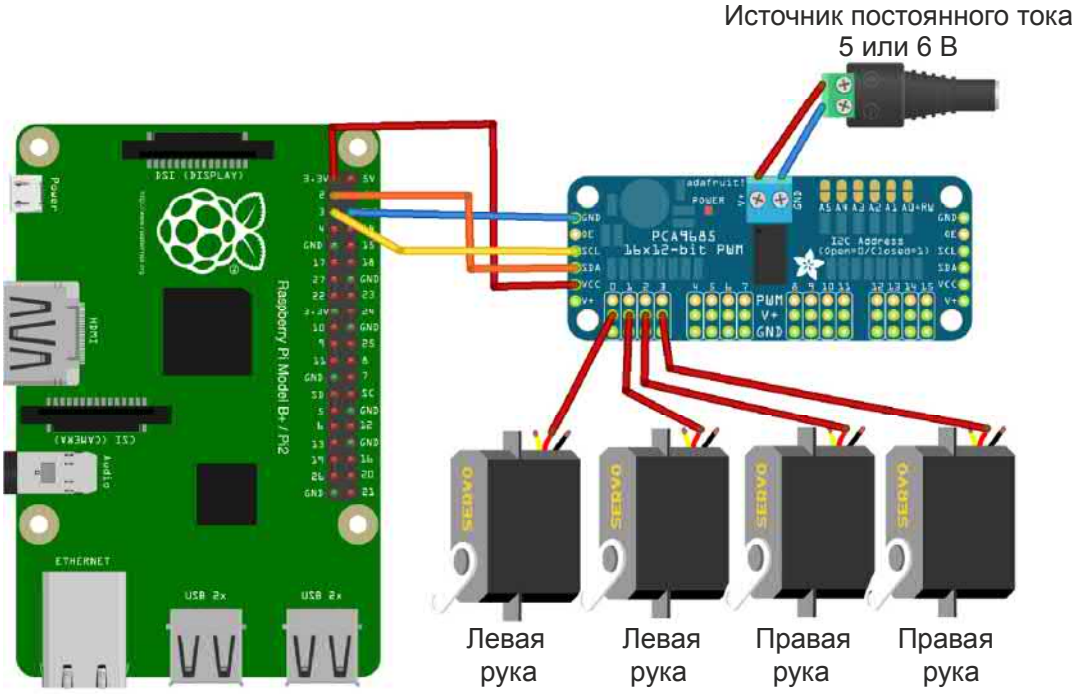


Рис. 9.10. Подключение сервоприводов куклы к Raspberry Pi с помощью контроллерной платы (драйвера) PWM/Servo

Одновременная работа четырех сервоприводов требует применения для их питания отдельного батарейного блока. Это необходимо для уменьшения влияния помех от двигателей на работу Raspberry Pi.

Длина пластиковых качалок, поставляемых вместе с сервоприводами для нашего проекта, недостаточна, поэтому, чтобы обеспечить полные диапазоны движения рук и ног куклы, необходимо удлинить качалки серводвигателей с помощью зубочисток.

Сборка проекта

В этом проекте для создания танцующей куклы требуется много механики, электроники и программирования. Далее приведены шаги, необходимые для реализации проекта.

Шаг 1. Удлинение качалок сервоприводов

Серводвигатели поставляются с небольшим пакетиком разных пластиковых качалок, которые можно установить на выходном валу серводвигателя. Выберите прямую качалку и приклейте к ней зубочистку, как показано на рис. 9.11. Для этой цели отлично подойдут клеевой пистолет или эпоксидный клей.

Рис. 9.11. Удлинение качалок серводвигателей

Обратите внимание, что на конце каждой зубочистки оставлена капля клея. Это необходимо, чтобы предотвратить соскальзывание бечевки, когда она будет привязана к концу зубочистки.

Шаг 2. Изготовление шасси

Чтобы управлять движением конечностей куклы, качалки четырех сервоприводов должны иметь возможность свободно двигаться вверх и вниз. Чтобы разместить все на нужных местах, сделаем вырезы на подходящей монтажной панели (хорошо подходит картон для наклейки фотографий).

Для точного определения мест вырезов воспользуйтесь шаблоном из файла puppet.svg, расположенного в каталоге python/projects/puppet. Разместите на монтажной панели распечатанный из файла шаблон (рис. 9.12) и приклейте его к панели с помощью легкого клея, чтобы его можно было удалить перед приклеиванием к этой панели серводвигателей.

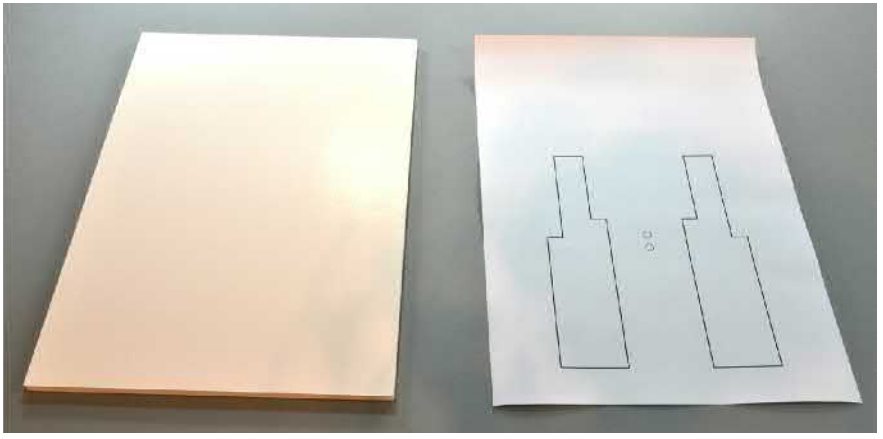


Рис. 9.12. Использование шаблона

Сделав в панели канцелярским ножом два больших выреза согласно шаблону, просверлите также два небольших отверстия, которые нужны для бечевки, идущей к голове куклы и принимающей на себя основной ее вес (рис. 9.13).

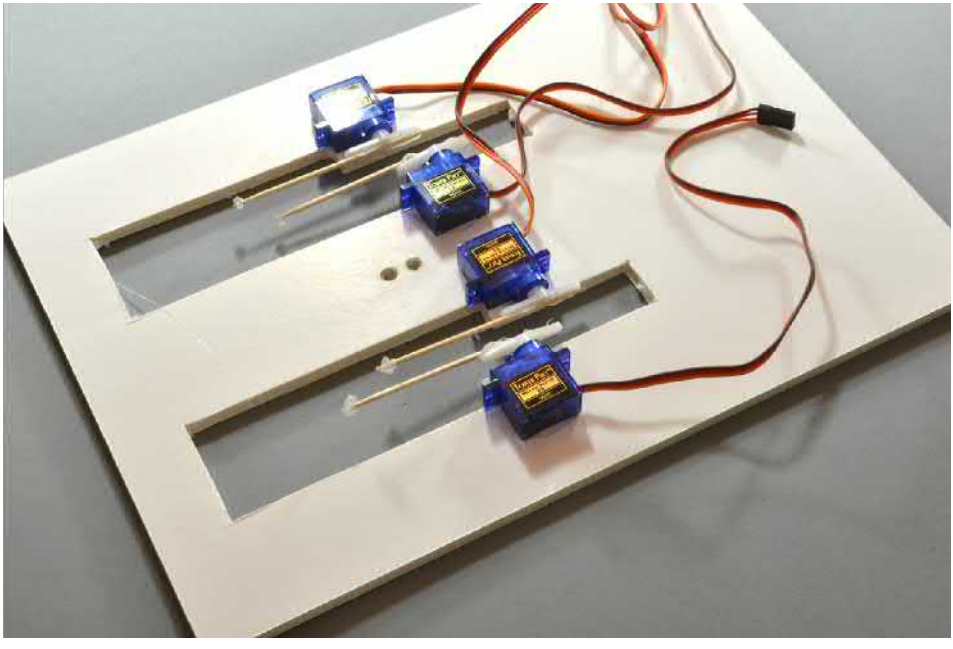


Рис. 9.13. Вырезы, отверстия и примерка размещения сервоприводов

Шаг 3. Приклеивание сервоприводов

Удалите с монтажной панели бумажный шаблон и закрепите удлиненные качалки на выходных валах сервоприводов. При этом крепите качалки не слишком туго, чтобы можно было их снять, если потребуется подобрать правильное положение сервопривода.

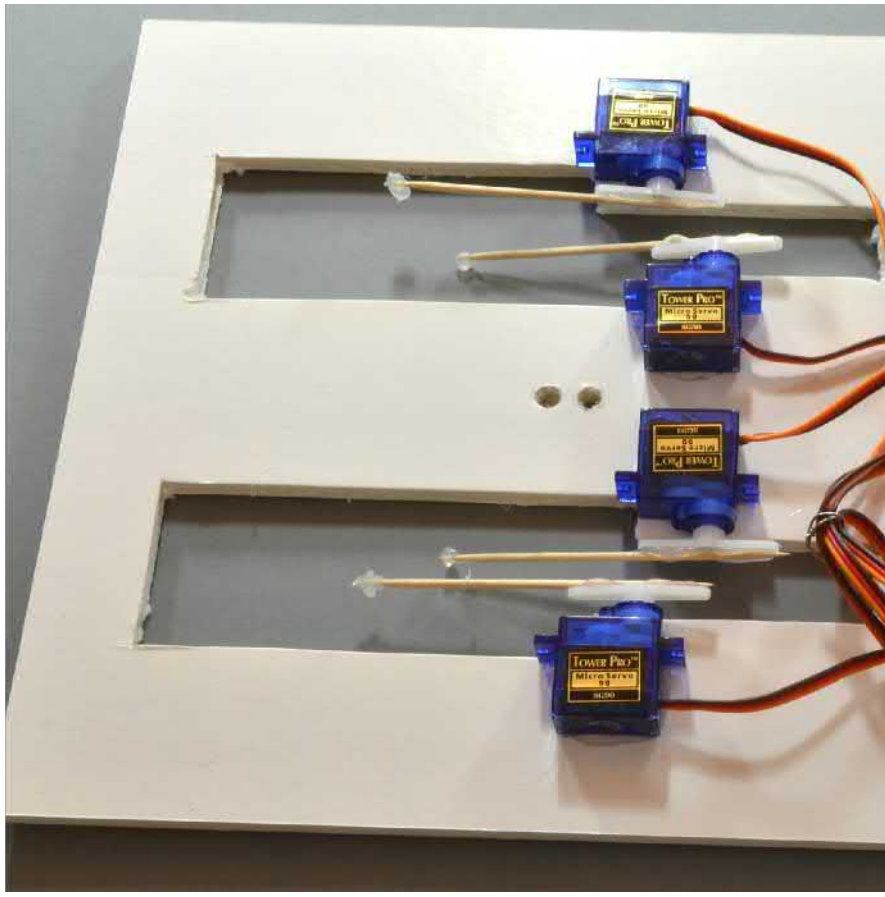


Рис. 9.14. Крепление сервоприводов

Выровняйте сервоприводы так, чтобы все качалки могли свободно перемещаться и не мешали друг другу (рис. 9.14).

Прежде чем приклеить сервоприводы, удалите с них этикетки, — так сервоприводы будут держаться лучше.

Шаг 4. Подготовка куклы

На рис. 9.15 показана кукла, которой мы будем управлять в этом проекте. Одна бечевка закреплена на верхней части головы куклы — она принимает на себя весь вес куклы. Эту бечевку следует пропустить через два отверстия, просверленные в монтажной панели (шасси), и завязать под ней.

К каждой ноге куклы прикреплена отдельная бечевка, а руки соединены вместе бечевкой, которая крепится к одному из плеч деревянной крестовины. Предварительно бечевки, идущие от рук к плечам крестовины, нужно от нее отрезать и, пропустив через отверстие в одном из плеч, связать вместе. Чтобы концы разрезанной нейлоновой бечевки не размочалились, их следует оплавить с помощью спички или

горячего фена. Лучше, если это возможно, не разрезать бечевки, а развязать их концы, закрепленные на деревянной крестовине.

Прежде чем закрепить куклу на новой конструкции, нужно подключить все сервоприводы и запустить нужные программы, чтобы установить качалки всех сервоприводов в правильное положение.

Рис. 9.15. Кукла Пепе

Шаг 5. Подключаем провода

Модуль Adafruit продается в виде собранного комплекта, к которому остается только припаять разъемы. Как это сделать, подробно описывает инструкция на странице Adafruit для этого продукта: **www.adafruit.com/products/815**.

Плату Raspberry Pi лучше сориентировать так, чтобы разъем интерфейса I^2C оказался на другой его стороне по отношению к модулю контроллера сервоприводов. Так будет удобнее при использовании оборудования этого проекта в *главе 15*, когда Пепе обретет голос.

Если хочется уменьшить количество пайки, то можно ограничиться припаиванием лишь четырех разъемов для сервоприводов, поскольку в этом проекте их задействовано только четыре.

После подключения проводов по схеме, показанной на рис. 9.10, все должно выглядеть так, как показано на рис. 9.16.

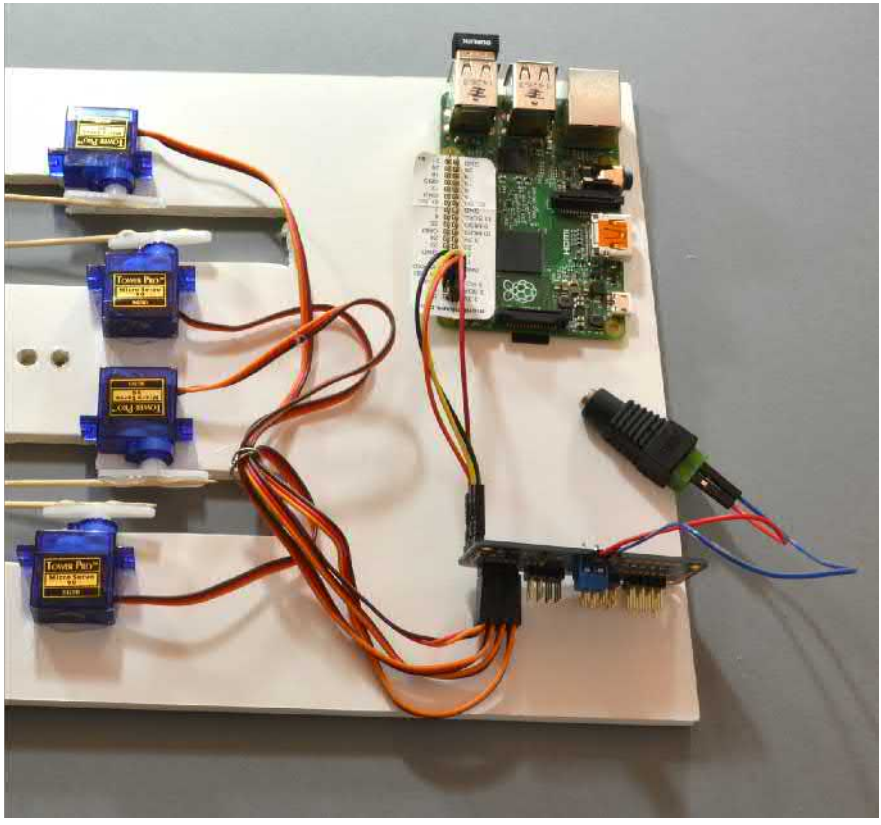


Рис. 9.16. Схема управления пляшущей куклой в сборе

Следует убедиться в правильном подключении разъемов всех сервоприводов: оранжевый провод управления (Control) должен быть вверху, а коричневый или черный общий провод (GND) — внизу.

Теперь можно подсоединить питание 5 В для сервоприводов и отдельное питание для Raspberry Pi через разъем USB.

Шаг 6. Запуск тестовой программы

Чтобы установить сервоприводы в правильное положение, нужно использовать отдельную программу (set_servos.py), которая позволяет установить качалки всех сервоприводов под определенным углом. Эту программу и основную программу кукольного проекта можно найти в каталоге /python/projects/puppet.

Модуль серводвигателей Adafruit использует интерфейс I^2C, который в Raspberry Pi по умолчанию выключен. Настройка интерфейса I^2C описана во врезке *«Настройка I^2C на Raspberry Pi»*, приведенной далее.

Настройка I^2C на Raspberry Pi

Для начала убедитесь, что ваш менеджер пакетов актуален — это делается с помощью следующей команды:

```
$ sudo apt-get update
```

Далее запустите конфигуратор `raspi-config`:

```
$ sudo raspi-config
```

В появившемся меню выберите пункт **Advanced** (Дополнительно), а затем **I2C** — появится запрос **Would you like the ARM I2C interface to be enabled?** (Хотите включить интерфейс I2C?). Выберите вариант **Yes** (Да). Будет задан вопрос **Would you like the I2C kernel module to be loaded by default?** (Загружать модуль I2C по умолчанию?) Это подходящий в данном случае вариант, поэтому снова следует ответить **Yes** (Да). Выберите пункт **Finish** (Готово) для выхода из конфигуратора `raspi-config`.

Выполните следующую команду из вашего домашнего каталога, чтобы установить некоторые полезные инструменты I^2C:

```
sudo apt-get install python-smbus i2c-tools
```

Убедиться, что Raspberry Pi подключается к плате сервоприводов Adafruit и готов к работе, можно, выполнив такую команду:

```
$ sudo i2cdetect -y 1
```

Вы должны увидеть следующий результат (числа `40` и `70` в таблице означают, что плата подключена):

```
$ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:
10:
20:
30:
40:  40
50:
60:
70:  70
```

Настроив интерфейс I^2C, снимите качалки с валов сервоприводов и запустите программу set_servos.py. При появлении запроса введите значение угла `90`:

```
$ sudo python set_servos.py
Angle:90
Angle:
```

Сервоприводы должны с жужжанием повернуться на угол 90°. Теперь можно закрепить качалки настолько близко к горизонтали, насколько это позволят сделать зубцы на валах сервоприводов.

Шаг 7. Подключение куклы

Теперь, когда все сервоприводы заняли положение 90°, подвяжите бечевку, идущую от головы куклы, к отверстиям в центре шасси. Затем привяжите бечевки, идущие от конечностей куклы, к удлиненным качалкам серводвигателей. Бечевки нужно прикрепить так, чтобы руки и ноги оказались в наполовину поднятом положении.

Теперь можно снова запустить программу set_servos.py и, вводя различные углы, проверить, охвачен ли движениями конечностей куклы максимальный диапазон.

Программа для Raspberry Pi

Программу для этого проекта можно считать только отправной точкой. Она использует массив значений положения серводвигателей, в которой можно вписывать свои собственные значения для управления движениями куклы. Основной «танец» в определенном смысле зажигателен, но его едва ли можно назвать элегантным.

Вы можете запустить программу даже до того, как взглянуть на код. Она находится в файле dance.py, расположенном в каталоге /python/projects/puppet. Не забудьте запустить ее с помощью команды sudo:

```
from Adafruit_PWM_Servo_Driver import PWM # // 1
import time

pwm = PWM(0x40)

servoMin = 150 # Min pulse length out of 4095 # // 2
servoMax = 600 # Max pulse length out of 4095

dance = [      // 3
   #lh lf rf rh
   [90, 90, 90, 90],
   [130, 30, 30, 130],
   [30, 130, 130, 30]
]

delay = 0.2 // 4

def map(value, from_low, from_high, to_low, to_high): # // 5
   from_range = from_high - from_low
   to_range = to_high - to_low
   scale_factor = float(from_range) / float(to_range)
   return to_low + (value / scale_factor)

def set_angle(channel, angle): // 6
pulse = int(map(angle, 0, 180, servoMin, servoMax))
pwm.setPWM(channel, 0, pulse)

def dance_step(step): // 7
set_angle(0, step[0])
set_angle(1, step[1])
set_angle(2, step[2])
set_angle(3, step[3])

pwm.setPWMFreq(60) // 8

while (True): // 9
for step in dance:
dance_step(step)
time.sleep(delay)
```

Примечание

Кроме собственно программы *dance.py* каталог */python/projects/puppet* содержит некоторые файлы Adafruit, которые используются этой программой. Оригинальные исходные тексты программ размещены также на GitHub (**github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code**).

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Плата Adafruit может быть использована не только для управления серводвигателями, но и другими устройствами, воспринимающими ШИМ-сигнал, — например, светодиодами. Именно поэтому код импортирует класс под названием `PWM` (ШИМ).
2. Две константы: `servoMin` и `servoMax` задают длительность импульса между `0` и `4095`, где `4095` соответствует 100% включению.
3. Массив `dance` содержит три группы данных для конкретного танца. Сюда можно добавить столько строк, сколько захочется. Каждая строка состоит из массива из четырех значений. Это угол поворота серводвигателей для левой руки, левой ноги, правой ноги и правой руки соответственно. Поскольку серводвигатели рук и ног смонтированы зеркально по отношению друг к другу, угол более 90° означает поднятие руки, но опускание ноги (это необходимо учитывать при составлении своего оригинального танца).
4. Переменная `delay` задает паузу между каждым шагом танца. Чем меньше ее значение, тем быстрее будет двигаться кукла.
5. Функция `map` подробнее объясняется в *главе 12* (см. врезку *«Функция `map`, используемая в Arduino»*) и служит для подтягивания значения угла к правому (максимальному) значению длительности импульса при использовании в функции `set_angle`.
6. Функция `set_angle` имеет два параметра: первый из них задает номер канала серводвигателя (от `0` до `3`), а второй — угол поворота.
7. Функция `dance_step` задает углы для четырех серводвигателей конечностей и приводит каждый серводвигатель к заданному углу.
8. Установка частоты ШИМ `60` раз в секунду дает серию импульсов каждые 17 мс — это примерно то, что нужно серводвигателю.
9. На каждую итерацию цикла главной программы приходится один шаг в танце, серводвигатели устанавливаются на заданные углы, затем выдерживается пауза перед новым шагом. Когда будут выполнены все шаги, заданные в программе, все начнется заново.

Пусть Пепе не только танцует...

Попробуйте изменить содержимое массива данных танца, чтобы создать свои собственные движения куклы. Можно попробовать заставить куклу ходить, совершать волнообразные движения или стоять на одной ноге. Установка несколько большей

задержки между шагами дает больше возможностей для отслеживания движений куклы и внесения коррективов в массив данных танца.

Кукла Пепе понадобится нам снова в *главе 15* (там она обретет голос) и в *главе 16* (где мы научим ее танцевать в ответ на полученные твиты).

Заключение

Серводвигатели весьма интересны, их довольно легко программировать и присоединять к ним всякие механические штучки.

В следующей главе речь пойдет о совершенно другом типе двигателей, называемых *шаговыми*.

Шаговые электродвигатели

10

В любом стандартном принтере (или, если на то пошло, в 3D-принтере) наверняка можно обнаружить один или несколько *шаговых электродвигателей*. На рис. 10.1 показан механизм подачи пластиковой проволоки в экструдер 3D-принтера. Шаговые электродвигатели широко используются в принтерах, поскольку они обеспечивают перемещение очень точным образом, — один шаг за один раз.

Рис. 10.1. Шаговый электродвигатель в 3D-принтере

По сравнению с двигателями постоянного тока для управления шаговыми электродвигателями применяются совершенно иные методы. В этой главе мы рассмотрим оба типа шаговых электродвигателей: униполярные и биполярные, а также различные управляющие ими микросхемы (так называемые *драйверы*), что пригодится нам при изучении шаговых электродвигателей.

Виды шаговых электродвигателей

Как следует из их названия, вращение шаговых электродвигателей представляет собой серию коротких шагов. В этом и состоит их отличие от двигателей свободного вращения — шаговый двигатель сделает столько шагов, сколько ему будет задано, и вы будете точно знать, сколько он их сделал. Это одна из причин, почему шаговые двигатели широко применяются как в обычных, так и в 3D-принтерах. В обычных принтерах они точно позиционируют бумагу, а в 3D — рабочий столик и сопло.

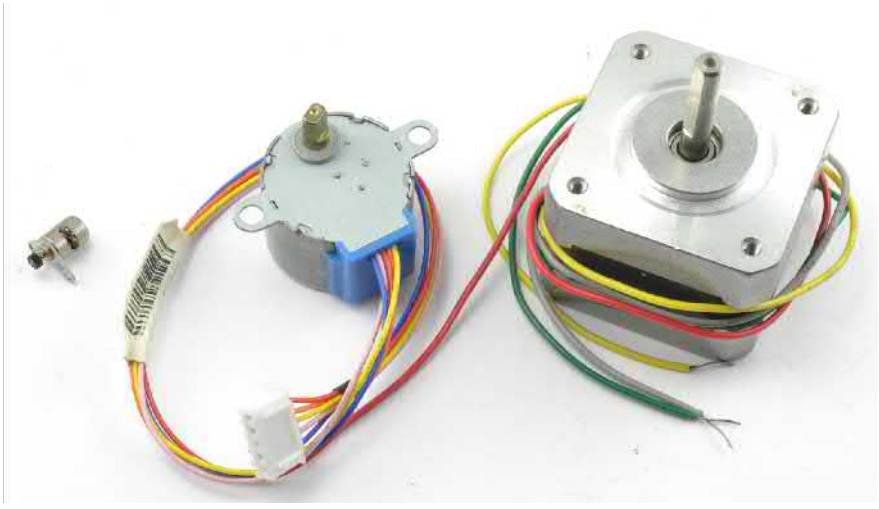


Рис. 10.2. Варианты шаговых электродвигателей

На рис. 10.2 представлены три образца шаговых электродвигателей. Миниатюрные электродвигатели типа тех, что показан *слева*, служат для перемещения элементов объектива в компактной фотокамере или смартфоне. В *центре* находится шаговый мотор-редуктор с питанием 5 В — в его корпусе заключен и шаговый электродвигатель, и редуктор. Справа изображен шаговый электродвигатель, типичный для принтеров.

Биполярные шаговые электродвигатели

На рис. 10.3 показана схема работы шагового электродвигателя — точнее, схема работы *биполярного шагового электродвигателя*. О другом типе шаговых электродвигателей — униполярных — будет рассказано в *разд. «Униполярные шаговые электродвигатели»* далее.

В биполярном шаговом двигателе, как правило, имеются четыре катушки. Катушки, расположенные одна против другой, соединены так, что работают синхронно. Все катушки расположены на неподвижном статоре двигателя, а значит, нет необходимости во вращающемся коллекторе и щетках, как у двигателей постоянного тока.

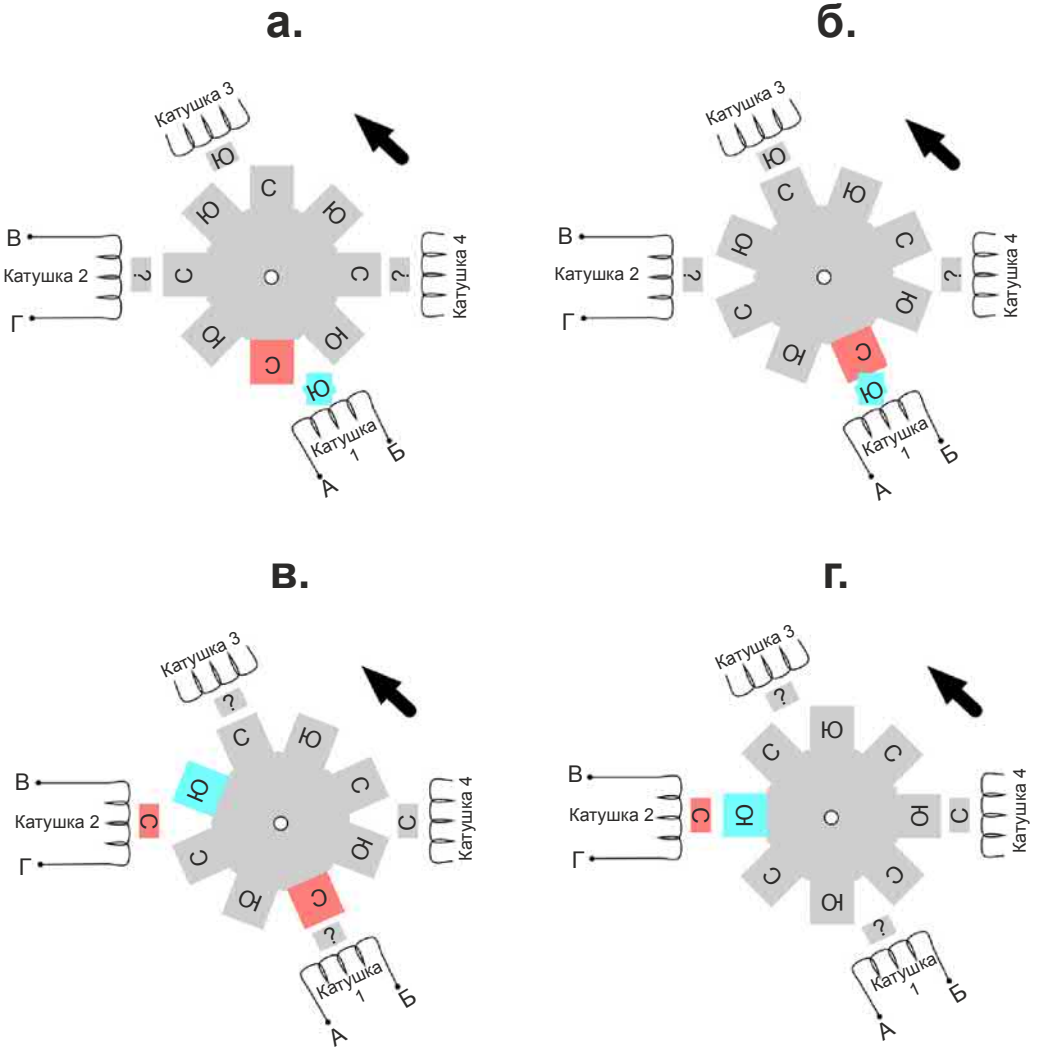


Рис. 10.3. Схема работы биполярного шагового электродвигателя

Ротор шагового электродвигателя выполнен в форме намагниченных зубцов с чередующимися северным (С) и южным (Ю) полюсами (зубцов на роторе обычно гораздо больше, чем показано на рис. 10.3). Каждую катушку можно подключить так, что она будет намагничена или как северный полюс, или как южный, — в зависимости от направления тока в катушке (вспоминаются Н-мосты, не так ли?). Катушки 1 и 3 работают совместно так, что когда катушка 1 будет южным полюсом, катушка 3 также будет южным полюсом. То же самое относится и к катушкам 2 и 4.

Начнем с рис. 10.3, *а* — когда катушка 1, а значит, и катушка 3 запитаны так, что становятся южными полюсами (Ю), вследствие того, что разноименные полюса притягиваются, а одноименные отталкиваются, ротор поворачивается против часовой стрелки до тех пор, пока ближайшие зубцы ротора с намагниченностью север-

ного полюса (С) не поравняются с катушками 1 и 3 (как показано на рис. 10.3, *б*). Чтобы продолжить вращение против часовой стрелки, на следующем шаге (рис. 10.3, *в*) необходимо подать ток в катушки 2 и 4 так, чтобы они стали северными полюсами (С). Тогда ближайшие зубцы ротора с намагниченностью Ю подтянутся к катушкам 2 и 4 (рис. 10.3, *г*).

Каждое такое действие проворачивает ротор электродвигателя на один шаг. Для продолжения вращения против часовой стрелки в катушке 1 снова нужно создать намагниченность С (табл. 10.1).

Таблица 10.1. *Последовательность действий при вращении шагового двигателя против часовой стрелки*

Катушки 1 и 3	Катушки 2 и 4
Ю	—
—	С
С	—
—	Ю

Прочерки в графах табл. 10.1 указывают на то, что катушка в этот момент не оказывает влияния на вращение ротора и должна быть обесточена. Чтобы усилить момент вращения двигателя, на эти обесточенные катушки можно подать такой ток, чтобы полярность их намагниченности совпадала с полярностью стоящего под ней зубца ротора (табл. 10.2).

Таблица. 10.2. *Уточненная последовательность переключения катушек при вращении шагового двигателя*

Катушки 1 и 3	Катушки 2 и 4
Ю	С
С	С
С	Ю
Ю	Ю

Можно задаться вопросом, что произойдет, если начать с катушек 2 и 4, а не с катушек 1 и 3? В этом случае просто другая пара катушек будет способствовать занятию зубцами ротора правильного положения.

А для изменения направления вращения ротора нужно всего лишь изменить порядок переключения катушек, указанный в табл. 10.2, на обратный.

Эксперимент: управление биполярным шаговым двигателем

Поскольку в шаговом двигателе имеются две пары катушек, для управления им придется изменять направление тока на обратное в каждой такой паре, а значит, нам потребуются два H-моста. Похоже, эта работа для микросхемы L293D.

Так и есть — в этом эксперименте и для Arduino, и для Raspberry Pi управление биполярным шаговым электродвигателем будет осуществляться с помощью микросхемы L293D, установленной на макетной плате (рис. 10.4).

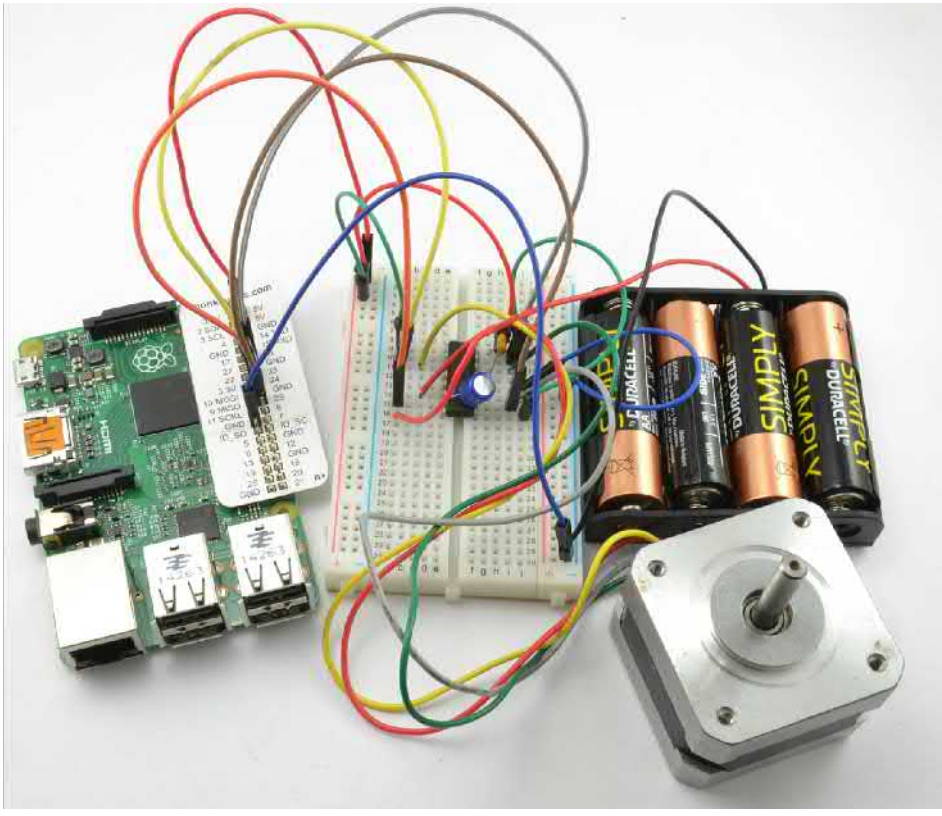


Рис. 10.4. Схема управления биполярным шаговым электродвигателем в сборе (вариант с Raspberry Pi)

Хотя в эксперименте используется шаговый двигатель с номинальным напряжением питания 12 В, он будет работать и от блока батарей с напряжением 6 В. У двигателя упадет момент, но крутиться он будет так же хорошо.

Определение назначения контактов шагового электродвигателя

Каждый новый шаговый двигатель должен иметь технический паспорт, а также еще и табличку на корпусе, с указанием назначения каждого из его четырех выводов. Главное, что здесь нужно знать, это какие выводы образуют пару, подключенную к одним и тем же катушкам.

Для выяснения этого есть один прием. Нужно взять любые два провода и зажать их между большим и указательным пальцами, вращая при этом вал двигателя. Если ощущается сопротивление вращению, то эти два провода и есть пара.

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 10.3).

Таблица 10.3. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению биполярным шаговым электродвигателем*

Обозначение	Компонент схемы	Источники
IC1	Микросхема с Н-мостом L293D	Adafruit: 807 Mouser: 511-L293D
C1	Конденсатор 100 нФ	Adafruit: 753 Mouser: 810-FK16X7R2A224K
C2	Конденсатор 16 В 100 мкФ	Adafruit: 2193 Sparkfun: COM-00096 Mouser. 647-UST1C101MDD
M1	Биполярный шаговый электродвигатель 12 В	Adafruit: 324
	Батарейный отсек 4 AA (6 В)	Adafruit: 830
	400-точечная беспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758
	Перемычки «мама-папа» (только для Pi)	Adafruit: 826

Напомню, что перемычки «мама-папа» понадобятся только для подключения к макетной плате контактов GPIO Raspberry Pi.

Конструкция

На рис. 10.5 показана принципиальная электрическая схема этого эксперимента. В данном случае ШИМ не используется, поэтому два входа Enable микросхемы L293D подсоединены к источнику питания 5 В, чтобы поддерживать оба Н-моста постоянно включенными. Шаговый двигатель управляется четырьмя каналами: In1, In2, In3 и In4, на которые будут подаваться управляющие сигналы от Arduino или Raspberry Pi.

Экспериментируем с Arduino

В версии этого эксперимента для Arduino (рис. 10.6) мы воспользуемся окном монитора порта, чтобы через него подавать на Arduino команды, управляющие двига-

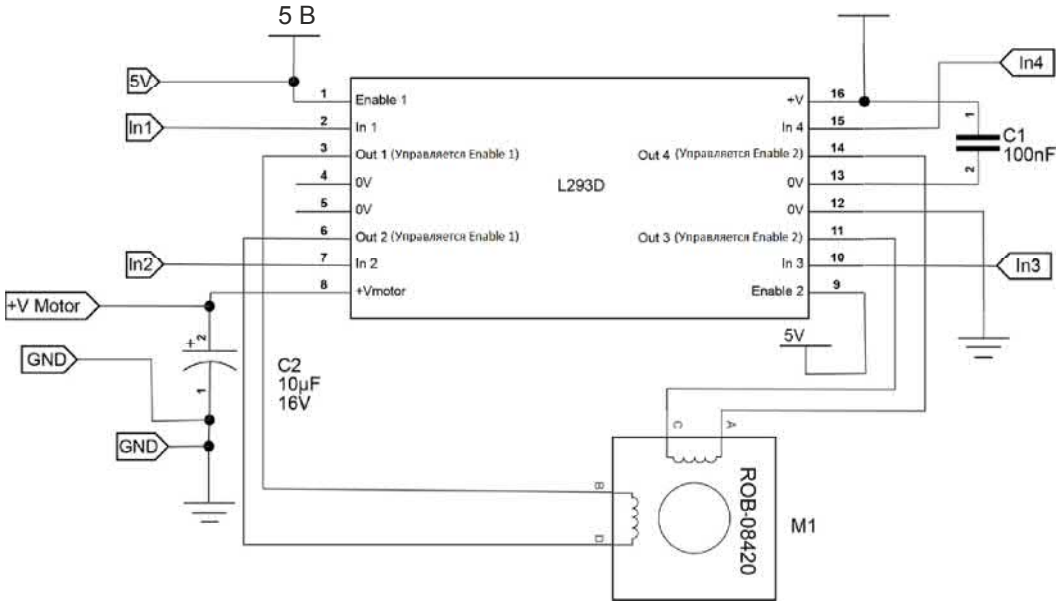


Рис. 10.5. Принципиальная электрическая схема управления биполярным шаговым электродвигателем

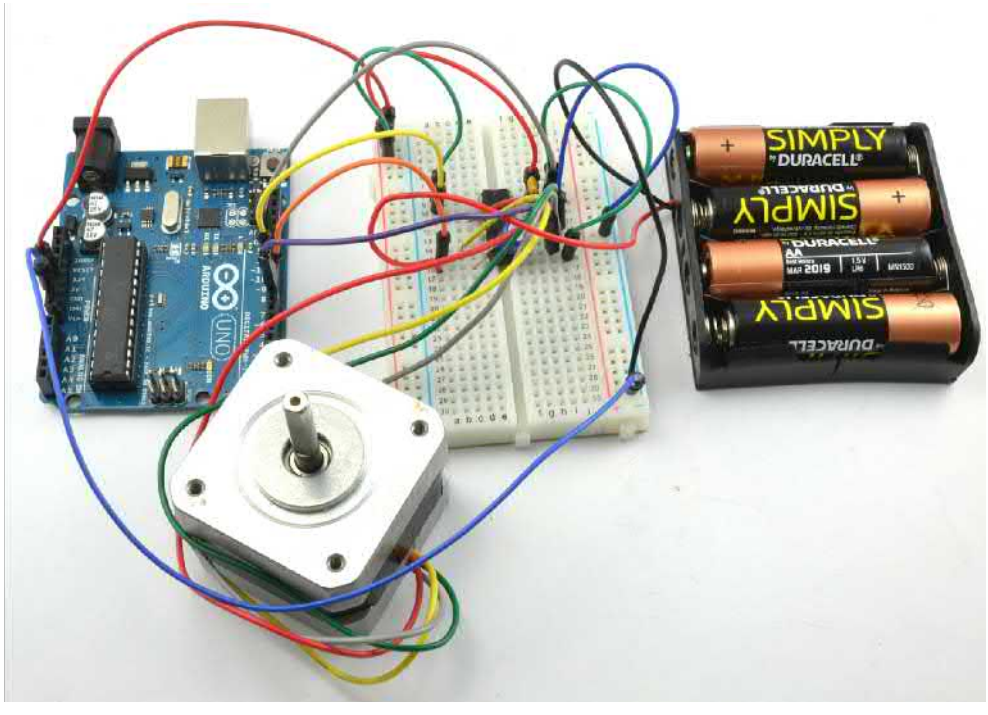


Рис. 10.6. Схема управления шаговым электродвигателем от Arduino в сборе

телем. Таких команд три, и состоят они из одной буквы с последующим числом. Например:

- `f100` — задает движение двигателю на 100 шагов вперед;
- `r100` — задает движение двигателю на 100 шагов в обратном направлении;
- `p10` — задает время задержки между импульсами шагов, равное 10 мс.

Подключение Arduino

При подключении Arduino к микросхеме L293D используются следующие контакты (табл. 10.4).

Таблица 10.4. *Контакты для подключения Arduino к микросхеме L293D*

Наименование контакта L293D	Номер контакта L293D	Контакт Arduino
In1	2	10
In2	7	9
In3	10	11
In4	15	8

Компоновка макетной платы для варианта с Arduino показана на рис. 10.7. Убедитесь, что микросхема установлена правильно (выемка на корпусе должна быть направлена вверх), и что конденсатор C2 емкостью 100 мкФ своим положительным выводом соединен с выводом 8 L293D. Чтобы вам было проще ориентироваться, напомню, что положительный вывод электролитического конденсатора обычно длиннее его отрицательного вывода. Отрицательный вывод конденсатора также может быть помечен на его корпусе символом «–» или ромбиком.

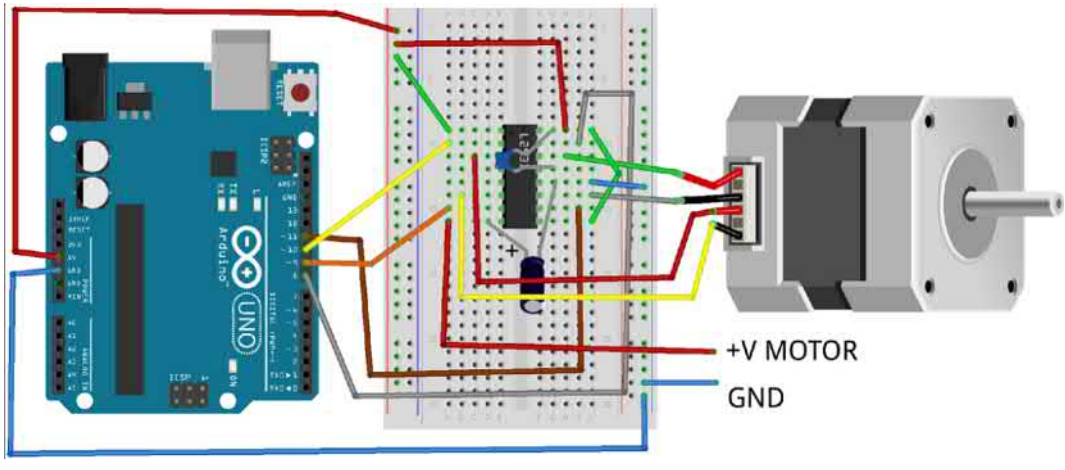


Рис. 10.7. Компоновка макетной платы для управления биполярным шаговым двигателем с Arduino

Программы для Arduino

Для этого эксперимента имеются два варианта программ. Первый — более сложный, он предполагает задание контактов микросхемы L293D для управления катушками шагового двигателя согласно описанию, приведенному ранее, в *разд. «Биполярные шаговые электродвигатели»*. Это полезное упражнение для точного понимания работы шагового двигателя.

Второй вариант использует скетч с обращением к библиотеке `Stepper` для Arduino, которая и делает всю работу. Поэтому второй вариант намного короче.

Сложный вариант программы

Первый вариант скетча для Arduino находится в каталоге arduino/experiments/bi_stepper_no_lib:

```
const int in1Pin = 10; // 1
const int in2Pin = 9;
const int in3Pin = 11;
const int in4Pin = 8;

int period = 20; // 2

void setup() { // 3
   pinMode(in1Pin, OUTPUT);
   pinMode(in2Pin, OUTPUT);
   pinMode(in3Pin, OUTPUT);
   pinMode(in4Pin, OUTPUT);
   Serial.begin(9600);
   Serial.println("Введите букву команды и число");
   Serial.println("p20 - устанавливает скорость 20 мс (управляет скоростью)");
   Serial.println("f100 - вперед 100 шагов");
   Serial.println("r100 - назад 100 шагов");
}

void loop() { // 4
   if (Serial.available()) {
      char command = Serial.read();
      int param = Serial.parseInt();
      if (command == 'p') { // 5
         period = param;

      else if (command == 'f') { // 6
         stepForward(param, period);

      else if (command == 'r') {
         stepReverse(param, period);
```

```
   setCoils(0, 0, 0, 0); // power down
}

void stepForward(int steps, int period) { // 7
   for (int i = 0; i < steps; i++) {
      singleStepForward(period);
   }
}

void singleStepForward(int period) { // 8
   setCoils(1, 0, 0, 1);
   delay(period);
   setCoils(1, 0, 1, 0);
   delay(period);
   setCoils(0, 1, 1, 0);
   delay(period);
   setCoils(0, 1, 0, 1);
   delay(period);
}

void stepReverse(int steps, int period) {
   for (int i = 0; i < steps; i++) {200
      singleStepReverse(period);
   }
}

void singleStepReverse(int period) { // 9
   setCoils(0, 1, 0, 1);
   delay(period);
   setCoils(0, 1, 1, 0);
   delay(period);
   setCoils(1, 0, 1, 0);
   delay(period);
   setCoils(1, 0, 0, 1);
   delay(period);
}

void setCoils(int in1, int in2, int in3, int in4) { // 10
   digitalWrite(in1Pin, in1);
   digitalWrite(in2Pin, in2);
   digitalWrite(in3Pin, in3);
   digitalWrite(in4Pin, in4);
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Программа начинается с определения выводов, которые будут использованы для управления двигателем. Можно, конечно, поменять выводы в соответствии с тем, как выполнен монтаж на макетной плате.

2. Переменная `period` задает величину задержки между включениями катушек на каждом шаге во время вращения двигателя. Изначально задана величина задержки в `20` мс. Эту величину можно изменять с помощью окна монитора порта, чтобы ускорить или замедлить двигатель.
3. Функция `setup()` конфигурирует выводы управления как выходы, запускает последовательный канал связи с окном монитора порта, а затем выводит сообщение, содержащее форматы команд, которые можно использовать.
4. В цикле основной программы `loop()` ожидается приход команд по последовательному каналу, а в случае их получения — обработка этих команд. Если команда получена (это покажет `Serial.available`) в цикле `loop()` сначала будет считана буква команды, а затем следующий за буквой параметр.

   Далее идет ряд выражений, задающих определенные действия, в зависимости от команды.
5. Если указана буква `p`, то переменная `period` принимает значение, равное параметру команды.
6. Если указана команда `f` или `r`, то вызываются функции `stepForward` или `stepReverse` со значением количества шагов и временем задержки между импульсами в качестве параметров.
7. Функции `stepForward` и `stepReverse` очень похожи. Они вызывают функции `singleStepForward` или `singleStepReverse` столько раз, сколько задано шагов.
8. Далее переходим к функции `singleStepForward`, содержащей образцы полярностей четырех фаз переключения катушек двигателя для перемещения на один шаг. Образцы полярностей можно увидеть в качестве параметров для `setCoils`.
9. Функция `singleStepReverse` работает так же, как и `singleStepForward`, только в обратной последовательности. Сравните эти шаги с данными табл. 10.2.
10. И, наконец, функция `setCoils` устанавливает состояние управляющих выводов в соответствии с образцом, заданным в качестве параметров.

Легкий вариант программы

Arduino IDE включает библиотеку `Stepper`, которая отлично работает и может значительно уменьшить размер скетча. Скетч с ее использованием находится в каталоге arduino/experiments/ex_07_bi_stepper_lib/ (см. *разд. «Код к книге» главы 2*):

```
#include <Stepper.h> // 1

const int in1Pin = 10;
const int in2Pin = 9;
const int in3Pin = 8;
const int in4Pin = 11;

Stepper motor(200, in1Pin, in2Pin, in3Pin, in4Pin); // 2

void setup() { // 3
   pinMode(in1Pin, OUTPUT);
```

```
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);
  while (!Serial);
  Serial.begin(9600);
  Serial.println("Command letter followed by number");
  Serial.println("p20 - устанавливает скорость 20");
  Serial.println("f100 - вперед 100 шагов");
  Serial.println("r100 - назад 100 шагов");
    motor.setSpeed(20); // 4
  }

  void loop() { // 5
    if (Serial.available()) {
    char command = Serial.read();
    int param = Serial.parseInt();
    if (command == 'p') {
      motor.setSpeed(param);
    }
    else if (command == 'f') {
      motor.step(param);
    }
    else if (command == 'r') {
      motor.step(-param);
    }
  }
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. В первой строке скетча происходит импорт библиотеки `Stepper`, которая уже включена в состав Arduino IDE, поэтому ее не нужно устанавливать специально.
2. Чтобы использовать библиотеку, в переменной `motor` нужно задать тип шагового двигателя. Первый параметр — количество шагов, которые двигатель делает за один оборот. Для шагового двигателя Adafruit, задействованного в этом проекте, задается значение в `200` шагов. На самом деле это означает 200 переключений фаз катушек на один оборот, что соответствует 50 физическим шагам на один оборот. Остальные четыре параметра — используемые выводы катушек.
3. Функция настройки `setup()` почти такая же, что и для сложного варианта скетча (см. ранее *разд. «Сложный вариант программы»*). Сообщение немного отличается, поскольку в данном случае команда `p` задает скорость вращения двигателя в об/мин (оборотов в минуту), что раньше делалось изменением времени задержки между подачей импульсов. Как бы там ни было, скорость двигателя можно изменить обоими этими способами.
4. По умолчанию скорость вращения двигателя в функции `setup()` устанавливается с помощью функции `motor.setSpeed`, равной `20` об/мин.

5. Цикл основной программы `loop()` также очень похож на вариант сложной версии скетча. Однако в данном случае количество шагов и направление вращения двигателя задается как положительное число при движении вперед и как отрицательное число — при движении назад.

   Число «шагов-четвертушек», задаваемых в параметрах команд `f` и `r`, в данном случае означает количество фаз переключения катушек. Чтобы используемый здесь шаговый двигатель Adafruit совершил полный оборот, нужно ввести число `200`.

Загружаем и выполняем программу

Вы можете попробовать во всех режимах обе программы для шаговых двигателей, но далее мы будем считать, что в Arduino загружен скетч ex_05_bi_stepper_lib (см. ранее *разд. «Легкий вариант программы»*). Откройте монитор порта, и вы увидите сообщение, показанное на рис. 10.8.

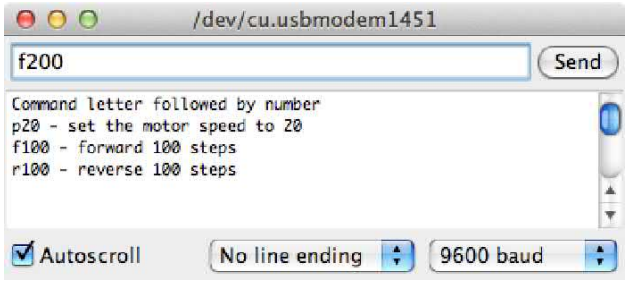


Рис. 10.8. Управление шаговым электродвигателем из монитора порта

Введите команду `f200` и нажмите кнопку **Send** (Отправить) — вал двигателя должен провернуться на один полный оборот. Если он не вращается, а дрожит или гудит, скорее всего, неправильно подключена одна из катушек. Поменяйте местами перемычки, идущие от шагового двигателя к выводам 3 и 6 микросхемы L293D и подайте команду снова.

Выполните команду `r200`, после чего вал двигателя должен сделать один оборот в противоположном направлении.

Далее можно поэкспериментировать со скоростью, используя команду `p` и следующее за ней значение скорости в об/мин, — например `p5` с последующей `f200`. Вал двигателя снова провернется на один оборот, но уже очень медленно. Можно увеличить скорость вращения, задав большее значение, — скажем, `p100`. Но тут обнаружится, что где-то в районе 130 об/мин существует ограничение скорости, выше которой часть шагов уже будет теряться, и вал двигателя не сможет сделать полный оборот.

Экспериментируем с Raspberry Pi

Управление шаговым двигателем с использованием программы на языке Python и Raspberry Pi очень напоминает вариант для Arduino без использования библиотеки.

Для управления микросхемой L293D потребуется четыре управляющих выхода Raspberry Pi.

Программа на Python предложит ввести задержку между фазами переключения, направление и количество шагов.

Подключение Raspberry Pi

Вы вполне можете оставить макетную плату, использованную для подключения Arduino, заменив лишь перемычки, идущие к Raspberry Pi (рис. 10.9). Как уже не раз отмечалось ранее, подключение Raspberry Pi осуществляется перемычками «мама-папа», а не «папа-папа», как в случае Arduino.

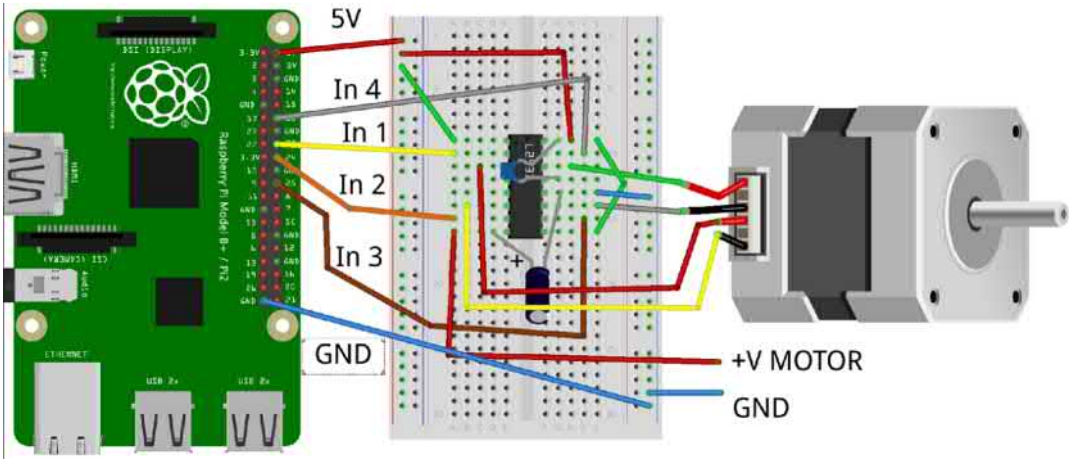


Рис. 10.9. Компоновка макетной платы для управления биполярным шаговым двигателем с Raspberry Pi

Программа для Raspberry Pi

Версия программы управления шагами для Raspberry Pi (см. файл bi_stepper.py из каталога python/experiments) практически полностью повторяет вариант для Arduino без использования библиотек:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

in_1_pin = 23 // 1
in_2_pin = 24
in_3_pin = 25
in_4_pin = 18

GPIO.setup(in_1_pin, GPIO.OUT)
GPIO.setup(in_2_pin, GPIO.OUT)
```

```
GPIO.setup(in_3_pin, GPIO.OUT)
GPIO.setup(in_4_pin, GPIO.OUT)

period = 0.02

def step_forward(steps, period): // 2
   for i in range(0, steps):
      set_coils(1, 0, 0, 1)
      time.sleep(period)
      set_coils(1, 0, 1, 0)
      time.sleep(period)
      set_coils(0, 1, 1, 0)
      time.sleep(period)
      set_coils(0, 1, 0, 1)
      time.sleep(period)

def step_reverse(steps, period):
   for i in range(0, steps):
      set_coils(0, 1, 0, 1)
      time.sleep(period)
      set_coils(0, 1, 1, 0)
      time.sleep(period)
      set_coils(1, 0, 1, 0)
      time.sleep(period)
      set_coils(1, 0, 0, 1)
      time.sleep(period)

def set_coils(in1, in2, in3, in4): // 3
   GPIO.output(in_1_pin, in1)
   GPIO.output(in_2_pin, in2)
   GPIO.output(in_3_pin, in3)
   GPIO.output(in_4_pin, in4)

try:
   print('Command letter followed by number');
   print('p20 - set the inter-step period to 20ms (control speed)');
   print('f100 - forward 100 steps');
   print('r100 - reverse 100 steps');

while True: // 4
   command = raw_input('Enter command: ')
   parameter_str = command[1:] # from char 1 to end
   parameter = int(parameter_str) // 5
   if command[0] == 'p': // 6
      period = parameter / 1000.0
   elif command[0] == 'f':
      step_forward(parameter, period)
```

```
    elif command[0] == 'r':
        step_reverse(parameter, period)

finally:
    print('Сброс')
    GPIO.cleanup()
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Программа определяет константы для четырех управляющих выводов и устанавливает последние в состояние выходов.
2. Функции `step_forward` и `step_reverse` практически такие же, как и их эквиваленты для Arduino. Четыре катушки коммутируются в необходимом порядке при вызове функции `set_coils`. Она многократно повторяет шаги, осуществляя задержку между каждым переключением катушек.
3. Функция `set_coils` устанавливает четыре управляющих вывода в соответствии с ее четырьмя параметрами.
4. В основном цикле программы командная строка считывается с помощью `raw_input`. Параметр, следующий за буквой, отделяется от буквы команды с помощью указания диапазона `[1:]`, что для строк в Python означает строку от позиции `1` (второй символ) до конца строки.
5. Строковая переменная затем преобразуется в число с помощью встроенной функции `int`.
6. Серия из трех выражений, в каждом из которых выполняется определенное действие, в зависимости от команды: меняется значение переменной `period`, происходит запуск двигателя в прямом или обратном направлении.

Загружаем и выполняем программу

Запустите программу командой:

```
sudo python bi_stepper.py
```

Появится сообщение с предложением ввести команды из разрешенного списка. Они точно такие же, как и в случае с Arduino:

```
Command letter followed by number
p20 - set the inter-step period to 20ms (control speed)
f100 - forward 100 steps
r100 - reverse 100 steps
Enter command: p5
Enter command: f50
Enter command: p10
Enter command: r100
Enter command:
```

Униполярные шаговые электродвигатели

Униполярные шаговые электродвигатели работают практически так же, как и биполярные, но не нуждаются для управления в Н-мостах. Этому способствует более сложное устройство его катушек. На рис. 10.10 показана схема работы униполярного шагового электродвигателя (в отличие от рис. 10.3, где показаны обе пары катушек биполярного шагового двигателя, на рис. 10.10 показана всего одна пара катушек, но их у униполярного шагового электродвигателя тоже две).

У униполярного двигателя имеется пять выводов от катушек — в отличие от четырех выводов у биполярного двигателя. Четыре этих вывода аналогичны выводам биполярного двигателя — это просто концы катушек А, Б, В и Г. Так что, при желании можно использовать концы катушек А, Б, В и Г, управляя ими с помощью Н-мостов, как будто это биполярный двигатель, и все будет работать нормально.

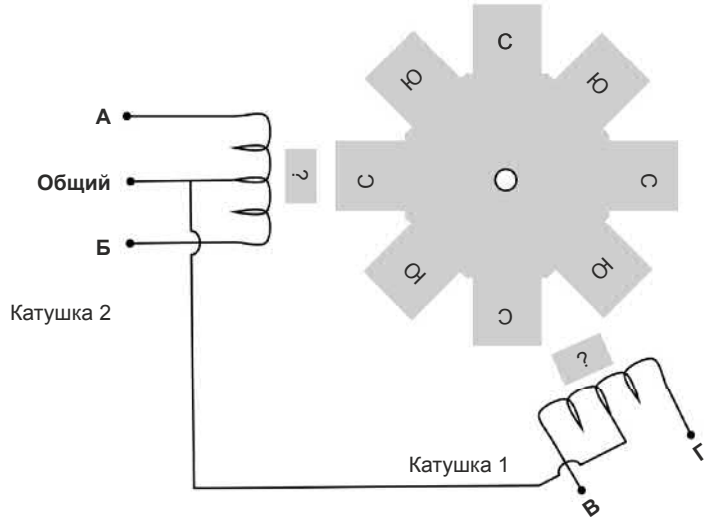


Рис. 10.10. Схема работы униполярного шагового электродвигателя

А вот пятый вывод униполярного шагового двигателя соединяет отводы от половины количества витков каждой из катушек. И если этот вывод соединить с заземляющим контактом (GND), то можно намагнитить катушку северным полюсом, подав питание на вывод А, или южным полюсом, подав питание на вывод Б. При этом необходимость использовать Н-мосты устраняется.

Сборки Дарлингтона

Отсутствие необходимости применять для управления униполярными шаговыми двигателями Н-мосты не отменяет того обстоятельства, что катушки двигателя потребляют слишком большой ток, чтобы их можно было питать непосредственно от выводов Arduino или Raspberry Pi.

Очевидный способ повысить ток — это использовать транзисторы, как это было сделано в *разд. «Эксперимент: управление электродвигателем» главы 4*. Нам по-

надобятся четыре транзистора — по одному на каждый вывод А, Б, В и Г. Понадобятся также токоограничивающие резисторы в цепи базы этих транзисторов и защитные обратные диоды на каждой катушке. На рис. 10.11 показана схема, по которой следует подключить каждую катушку униполярного двигателя, а для в целом управления двигателем вам потребуется четыре такие схемы.

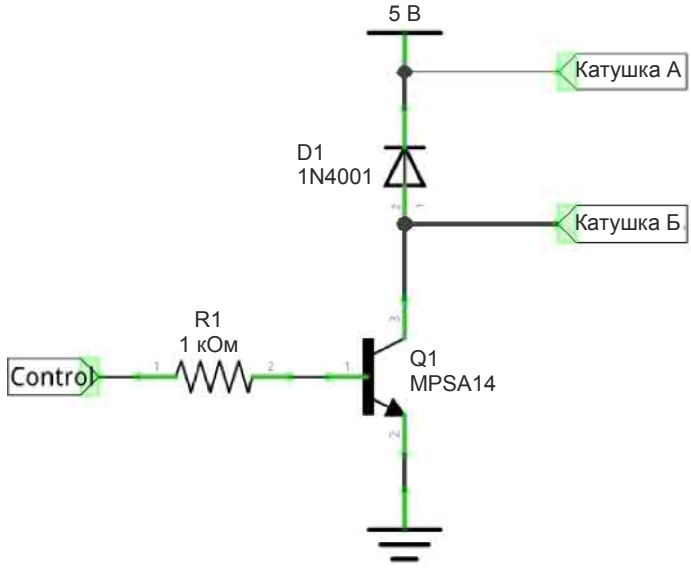


Рис. 10.11. Силовой ключ униполярного шагового двигателя на одном транзисторе

Их можно прекрасно собрать на макетной плате, но для этого потребуется просто куча перемычек.

Более аккуратное решение — использование специализированной микросхемы (транзисторной сборки), например, ULN2803. Эта недорогая микросхема содержит восемь составных транзисторов (пар Дарлингтона) в одном корпусе. Кроме того, там уже есть и резисторы в цепи баз, и защитные диоды. Единственное, что нужно будет добавить, — это конденсатор в цепи питания.

Микросхема ULN2803 (**www.adafruit.com/datasheets/ULN2803A.pdf**) может обеспечить ток до 500 мА в каждом канале при максимальном напряжении 50 В.

В рассмотренном далее *разд. «Эксперимент: управление униполярным шаговым электродвигателем»* мы воспользуемся одним из таких устройств для управления униполярным электродвигателем от Arduino и от Raspberry Pi.

Эксперимент: управление униполярным шаговым электродвигателем

На рис. 10.12 показана плата Raspberry Pi, управляющая униполярным шаговым двигателем одного из широко распространенных его типов, — с понижающим редуктором, имеющим коэффициент редукции 1:16. При этом, хотя сам двигатель

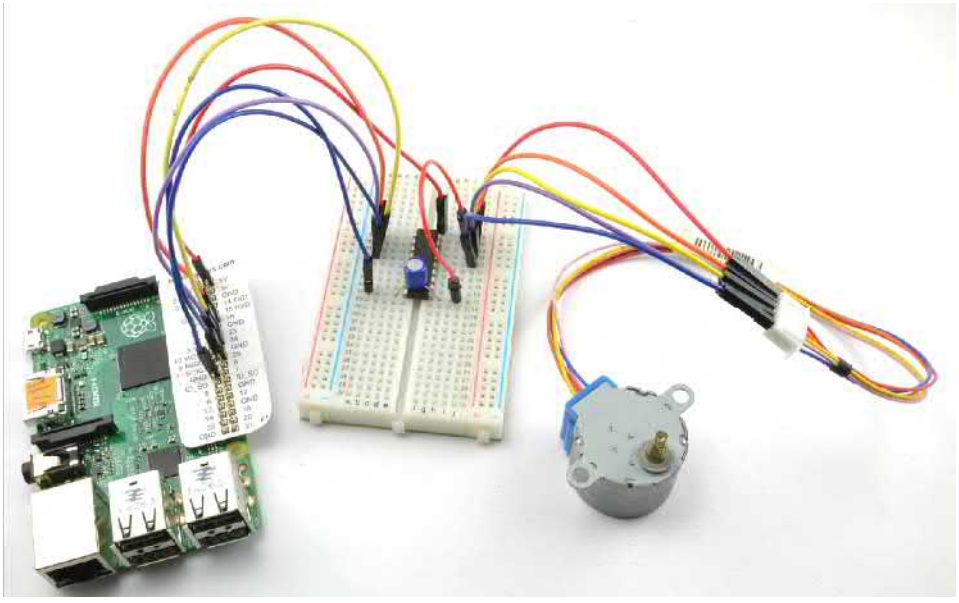


Рис. 10.12. Схема управления униполярным шаговым электродвигателем в сборе (вариант с Raspberry Pi)

совершает 32 шага за оборот, наличие редуктора означает, что для одного оборота нужно выполнить 513 переключений фаз, что эквивалентно 128 шагам на оборот.

Оборудование

На рис. 10.13 показана схема подключения катушек униполярного двигателя (см. рис. 10.11), модернизированная под использование микросхемы ULN2003. Обратите внимание, что незадействованными остаются еще четыре составных транзистора, которые можно использовать для управления вторым шаговым двигателем. Для шаговых двигателей с большим рабочим током транзисторы микросхемы можно сдваивать, соединяя вместе два их входа и два соответствующих им выхода.

Здесь применен маломощный двигатель с напряжением питания 5 В и достаточно малым рабочим током, так что его можно запитать непосредственно от Raspberry Pi или Arduino. При двух включенных катушках потребляемый ток составляет около 150 мА. Если используется более мощный двигатель или двигатель с более высоким напряжением питания, нужно будет подключить отдельный источник питания, как это делалось в случае с биполярным шаговым двигателем.

Найти, какой вывод к чему присоединен внутри двигателя, можно тем же способом, что и рекомендован для биполярного двигателя (см. ранее врезку *«Определение назначения контактов шагового электродвигателя»*). Дополнительные сложности здесь связаны с наличием общего провода. Так что, начать нужно с поиска общего провода, который дает сопротивление вращению, будучи соединенным с любым из других выводов. Затем для нахождения остальных проводов можно использовать тот же прием, что и для биполярного двигателя.

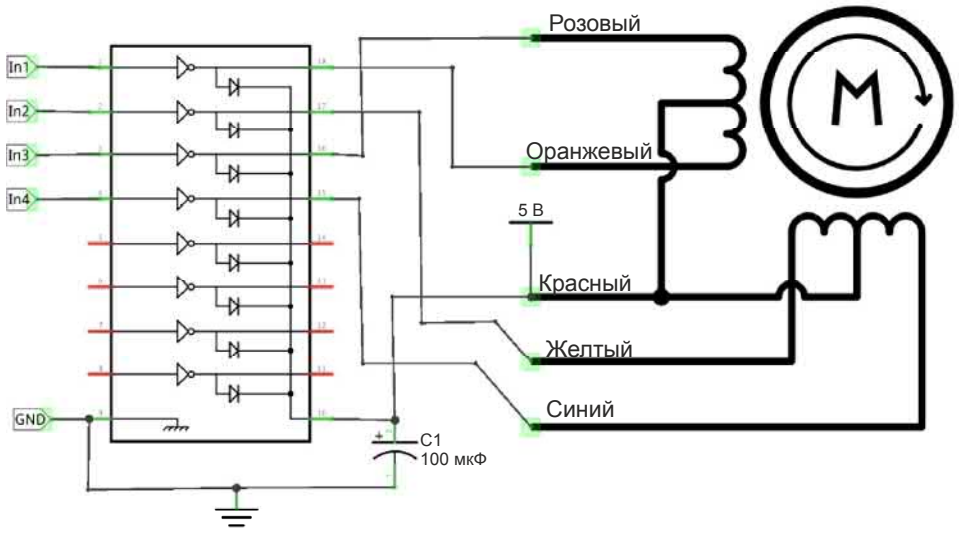


Рис. 10.13. Схема управления униполярным шаговым двигателем (цвета проводов реальной схемы могут отличаться от тех, что здесь указаны)

Комплектующие

В этом эксперименте для работы с Arduino и Raspberry Pi понадобятся следующие комплектующие (табл. 10.5).

Таблица 10.5. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению униполярным шаговым двигателем*

Обозначение	Компонент схемы	Источник
IC1	Микросхема ULN2803	Adafruit: 970 Mouser: 511-ULN2803A
C2	Конденсатор 16 В 100 мкФ	Adafruit: 2193 Sparkfun: COM-00096 Mouser: 647-UST1C101MDD
M1	Униполярный шаговый двигатель 5 В	Adafruit: 858
	Батарейный отсек 4 AA (6 В)	Adafruit: 830
	400-точечная бéспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758
	Перемычки «мама-папа» (только для Pi)	Adafruit: 826

Напомню, что перемычки «мама-папа» понадобятся только для подключения к макетной плате контактов GPIO Raspberry Pi.

Подключение Arduino

На рис. 10.14 показано, как можно подключить Arduino Uno к микросхеме ULN2803.

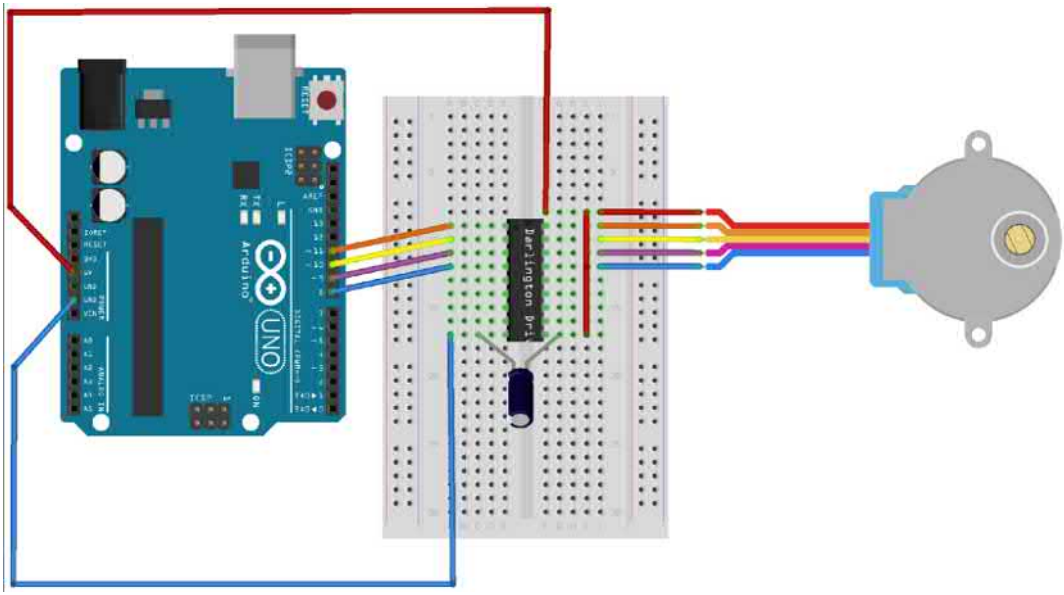


Рис. 10.14. Подключение Arduino к микросхеме ULN2803

Нужно только убедиться, что конденсатор установлен правильно — его более длинный положительный вывод должен быть сориентирован вправо. Микросхема должна быть расположена так, чтобы выемка на корпусе смотрела на верхнюю сторону макетной платы.

Как можно видеть, монтаж получается более аккуратным по сравнению с вариантом для биполярного двигателя.

Подключение Raspberry Pi

На рис. 10.15 показано расположение проводов и подключение Raspberry Pi к микросхеме ULN2803.

Программа

Кроме изменения назначения некоторых выводов, программы для Arduino и Raspberry Pi точно такие же, что приведены ранее, в *разд. «Эксперимент: управление биполярным шаговым одвигателем»*. Программы с модифицированным подключением выводов находятся:

- для Arduino: в файле uni_stepper_lib.ino каталога arduino/experiments/uni_stepper_lib;
- для Raspberry Pi: в файле uni_stepper.py каталога python/experiments.

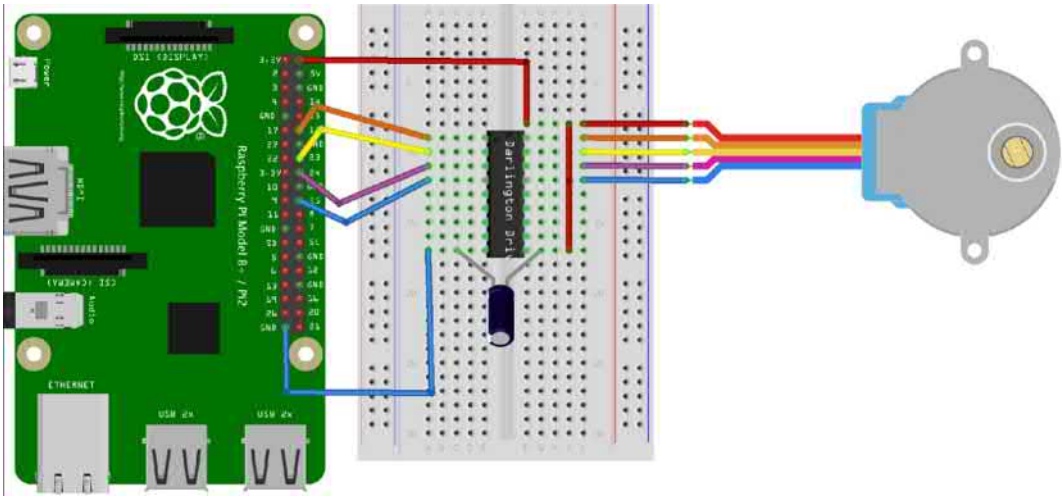


Рис. 10.15. Подключение Raspberry Pi к микросхеме ULN2803

Микрошаги

Можно, наверное, заметить, что движение шагового двигателя происходит не очень гладко, даже когда скорость вращения весьма высока. Для большинства приложений это не составляет проблемы, но иногда было бы лучше, если бы движение было бы более гладким.

Микрошаги — это метод, позволяющий реализовать более гладкое движение двигателя. Вместо того чтобы просто включать и выключать катушки, для их более плавного питания используется ШИМ, что и создает более плавное вращение двигателя.

Микрошаги можно реализовать и программным способом, но в целом гораздо проще использовать специальные аппаратные средства, разработанные для шаговых двигателей, допускающих работу в режиме микрошагов. К таким аппаратным средствам можно отнести базирующуюся на микросхеме A3967 плату EasyDriver, разработанную Брайаном Шмальцем (Brian Schmalz).

Отличный учебник по использованию платы EasyDriver с Arduino можно найти в Интернете на страничке Sparkfun, посвященной этой плате (продукт ROB-12779). Но мы не станем здесь его повторять, и в качестве альтернативы рассмотрим далее *разд. «Эксперимент: микрошаги на Raspberry Pi»*.

Эксперимент: микрошаги на Raspberry Pi

В этом эксперименте используется плата EasyDriver, реализующая режим микрошагов для шагового двигателя 12 В, который был задействован ранее для Raspberry Pi в *разд. «Эксперимент: управление биполярным шаговым двигателем»*.

Можно использовать и униполярный шаговый двигатель, но не подключая общий провод, — при этом униполярный двигатель будет работать как биполярный.

Плата EasyDriver поставляется с контактными штырьками, позволяющими подключить ее к макетной плате напрямую. На рис. 10.16 показан окончательной вид собранной схемы.

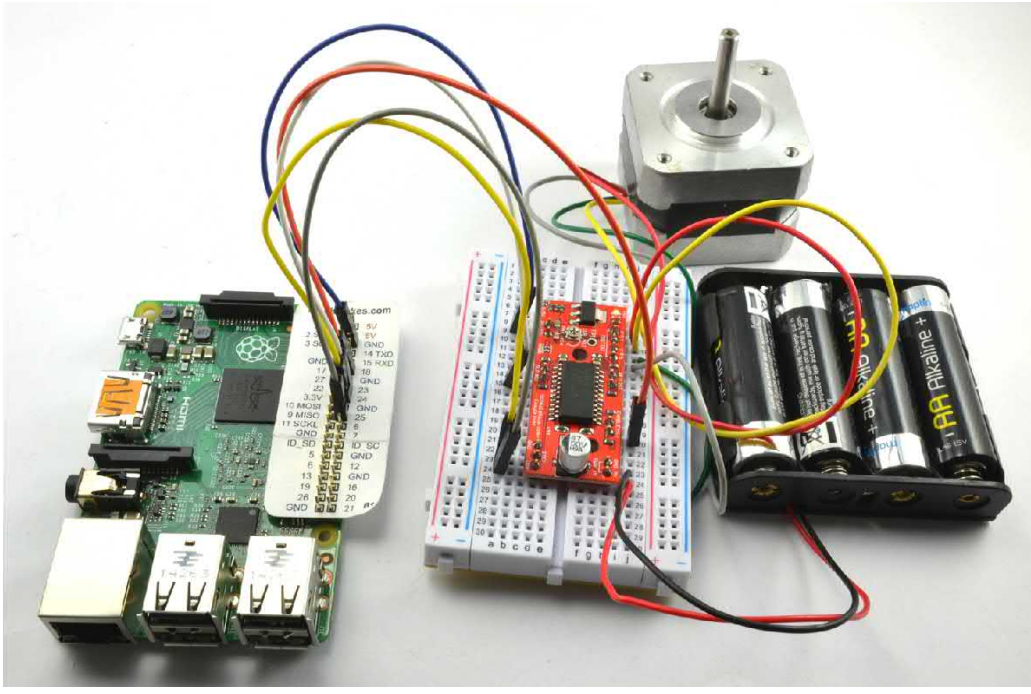


Рис. 10.16. Схема для эксперимента с микрошагами на Raspberry Pi в сборе

Комплектующие

В этом эксперименте для работы с Raspberry Pi понадобятся следующие комплектующие (табл. 10.6).

Таблица 10.6. *Комплектующие для работы с Raspberry Pi в эксперименте с микрошагами*

Компонент схемы	Источники
Плата EasyDriver для регулировки шаговых двигателей	Sparkfun: ROB-12779
Биполярный шаговый электродвигатель 12 В	Adafruit: 324
Батарейный отсек 4 AA (6 В)	Adafruit: 830
400-точечная беспаечная макетная плата	Adafruit: 64
Перемычки «мама-папа»	Adafruit: 826

Примечание

Большинство шаговых двигателей, рассчитанных на 12 В, будут нормально работать и от 6 В, но лучше все же использовать блок питания на 12 В.

Подключение Raspberry Pi

На рис. 10.17 показана компоновка макетной платы с платой EasyDriver и подключение ее к плате Raspberry Pi.

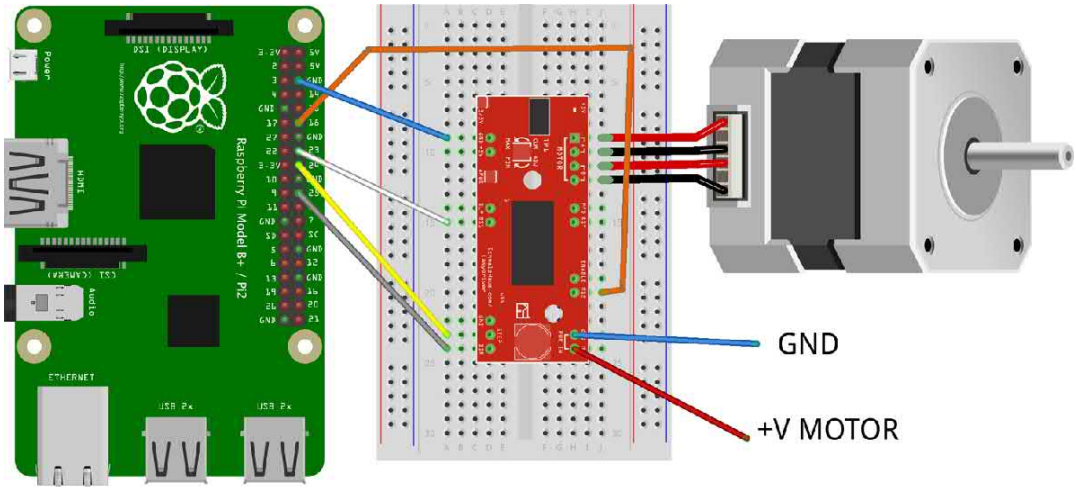


Рис. 10.17. Компоновка макетной платы для реализации микрошагов на Raspberry Pi

На плате EasyDriver уже установлены конденсаторы, которые в прочих случаях обычно приходится устанавливать самостоятельно. На ней также имеется и стабилизатор напряжения, обеспечивающий питание микросхемы A3967. Соответственно, ее даже не надо подключать к питанию Raspberry Pi. Все, что нужно сделать, — это подключить общий провод (GND) и четыре сигнала управления (Control).

Программа

Как следует из самого названия платы EasyDriver («Простой Драйвер»), ею очень удобно управлять программно. Основные команды на плату передаются с помощью только двух выводов. Когда вывод `step` получает импульс `HIGH`, двигатель делает один шаг в направлении, заданном на направляющем выводе.

Два других вывода: `ms1` и `ms2` — нужны для того, чтобы задать размер микрошага от 0 до $^1/_8$.

Код программы содержится в файле **microstepping.py** каталога **python/experiments** (см. *разд. «Код к книге» главы 3*):

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
```

```
step_pin = 24
dir_pin = 25
ms1_pin = 23
ms2_pin = 18

GPIO.setup(step_pin, GPIO.OUT)
GPIO.setup(dir_pin, GPIO.OUT)
GPIO.setup(ms1_pin, GPIO.OUT)
GPIO.setup(ms2_pin, GPIO.OUT)

period = 0.02

def step(steps, direction, period): // 1
   GPIO.output(dir_pin, direction)
   for i in range(0, steps):
      GPIO.output(step_pin, True)
      time.sleep(0.000002)
      GPIO.output(step_pin, False)
      time.sleep(period)

def step_mode(mode): // 2
   GPIO.output(ms1_pin, mode & 1) // 3
   GPIO.output(ms2_pin, mode & 2)

try:
   print('Command letter followed by number');
   print('p20 - set the inter-step period to 20ms (control speed)');
   print('m - set stepping mode (0-none 1-half, 2-quater, 3-eighth)');
   print('f100 - forward 100 steps');
   print('r100 - reverse 100 steps');

while True: // 4
   command = raw_input('Enter command: ')
   parameter_str = command[1:] # from char 1 to end
   parameter = int(parameter_str)
   if command[0] == 'p':
      period = parameter / 1000.0
   elif command[0] == 'm':
      step_mode(parameter)
   elif command[0] == 'f':
      step(parameter, True, period)
   elif command[0] == 'r':
      step(parameter, False, period)

finally:
   print('Сброс')
   GPIO.cleanup()
```

Примечание

Вы уже должны знать цоколевку GPIO и код инициализации, запускающий большинство Python-программ из этой книги. Если не знаете — вернитесь к соответствующим разделам предыдущих глав.

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Функция `step` содержит код для перемещения двигателя на ряд шагов (микрошагов). В качестве параметров функция берет количество шагов, направление (0 или 1) и величину задержки между каждым шагом. Внутри тела функции прежде всего устанавливается вывод `dir_pin`, отвечающий за направление на плате EasyDriver, а затем генерируется необходимое количество импульсов на выводе `step_pin`. Длительность каждого импульса на выводе `step_pin` составляет всего 2 микросекунды (в техническом описании микросхемы A3967 сказано, что длительность импульсов должна быть не менее 1 микросекунды).
2. Функция `step_mode` устанавливает состояние выводов тактового режима в соответствии с выбранным значением режима, которое должно находиться между 0 и 3.
3. Код внутри функции `step_mode` отделяет два бита от номера режима и устанавливает `ms1_pin` и `ms2_pin`, используя логический оператор И (`&`). В табл. 10.7 показана зависимость состояния выводов режима микрошагов и поведения двигателя.

Таблица 10.7. *Выводы управления микрошагами*

Режим	ms1_pin	ms2_pin	Микрошаги
0	LOW	LOW	Нет
1	HIGH	LOW	Половина шага
2	LOW	HIGH	Четверть шага
3	HIGH	HIGH	Восьмая шага

4. Основной цикл программы очень похож на Python-программу из *разд. «Эксперимент: управление биполярным шаговым электродвигателем»*, за исключением добавленной команды ***m***, позволяющей произвести установку режима микрошагов.

Загружаем и выполняем программу

Запустите программу microstepping.py и в ответ на приглашение введите команды `m0`, `p8`, а затем `f800`:

```
$ sudo python microstepping.py
Command letter followed by number
p20 - set the inter-step period to 20ms (control speed)
m - set stepping mode (0-none 1-half, 2-quater, 3-eighth)
```

```
f100 - forward 100 steps
r100 - reverse 100 steps
Enter command: m0
Enter command: p8
Enter command: f800
```

Это должно вызвать поворот вала двигателя на четыре оборота (для 200-шагового двигателя). Постарайтесь запомнить плавность движения вала при вращении.

Затем введите следующие команды: `m3`, `p1`, `f6400` — вал двигателя сделает четыре оборота с той же скоростью, но намного более плавно.

Обратите внимание, что для сохранения той же скорости период следования шагов был уменьшен в 8 раз, а число шагов (в данном случае — микрошагов) увеличено в 8 раз.

Бесколлекторные двигатели постоянного тока

Бесколлекторные двигатели постоянного тока (БДПТ, BLDC) (рис. 10.18) — это маленькие и очень мощные двигатели, которые можно отыскать в квадрокоптерах и радиоуправляемых моделях самолетов. При том же весе они могут создать гораздо больший крутящий момент, чем обычные двигатели постоянного тока. Они включены в эту главу, поскольку имеют много общего с шаговыми двигателями, хотя формально и относятся к двигателям постоянного тока.

Рис. 10.18. Бесколлекторный двигатель постоянного тока

В отличие от коллекторного двигателя постоянного тока (см. *главу 7*) бесколлекторные двигатели не имеют механического коммутатора-коллектора, меняющего направление тока при вращении двигателя. По конструкции они больше похожи на

шаговые двигатели, при том отличии, что вместо двух пар катушек они содержат несколько блоков из трех катушек. По этому признаку их можно скорее отнести к трехфазным электродвигателям, и приводы бесколлекторных двигателей существенно сложнее, чем обычные Н-мосты. Действительно, каждая катушка может пропускать ток в прямом и обратном направлении, а также находиться в «высокоимпедансном», т. е. отключенном состоянии. Для эффективного управления двигателем его управляющая микросхема (драйвер) должна замерять напряжение на «отключенной» катушке, чтобы настроить для каждой конкретной катушки время переключения на следующую фазу работы.

К сожалению, практически невозможно найти микросхемы управления бесколлекторными двигателями, которые можно было бы легко установить на макетную плату. Так что, лучше поискать готовые платы драйверов. Много разных моделей таких плат можно найти на аукционе eBay.

Чтобы получить мощность и размеры бесколлекторного двигателя без заморочек с платами управления, можно купить двигатель с уже встроенным контроллером и только двумя проводами, выходящими из корпуса. Его тогда можно подключать как обычный двигатель постоянного тока. Кстати, динамический насос на рис. 7.13, *справа*, оборудован как раз таким двигателем.

Заключение

В этой главе были рассмотрены шаговые электродвигатели и способы управления ими с помощью Arduino и Raspberry Pi. Следующая глава посвящена вопросам управления нагревом и охлаждением.

11 Нагрев и охлаждение

В этой главе рассматриваются устройства для генерации тепла и холода — такие как резистивные нагревательные элементы и элементы Пельтье. При этом основное внимание уделяется именно устройству нагревателей и охладителей. А о том, как эти нагреватели и охладители могут использоваться вместе с Arduino и Raspberry Pi для точного регулирования температуры, будет рассказано в *главе 12*.

Резистивные нагреватели

В *главе 5* мы выяснили, что, сопротивляясь протеканию тока, резистор генерирует тепло. Как вы, наверное, помните, количество сгенерированного тепла в ваттах равно произведению тока, протекающего через резистор, в амперах на величину падения напряжения на резисторе в вольтах.

Эксперимент: нагрев резистора

С достаточно простой процедурой включения и отключения электродвигателя вы уже хорошо знакомы. Точно так же питание подается на резистор и снимается с него. Любой из предыдущих экспериментов по включению и отключению двигателя вы легко сможете повторить, заменив двигатель резистором. С резистивным нагревателем будет также хорошо работать даже ШИМ-модуляция мощности. Таким образом, при проведении этого эксперимента вполне можно обойтись без Arduino и Raspberry Pi, используя просто батарейный блок питания и резистор.

Обратите внимание, что в этом эксперименте резистор задействован в качестве нагревательного элемента и батарейный блок — в качестве источника питания. Ни то, ни другое нельзя назвать практичным решением для нагрева какого-либо даже совсем небольшого устройства. Тем не менее, такой выбор весьма удобен для проведения экспериментов.

Комплектующие

В этом эксперименте нам понадобятся следующие комплектующие (табл. 11.1).

Таблица 11.1. *Комплектующие для работы в эксперименте по нагреву резистора*

Компонент схемы	Источники
Резистор 100 Ом 0,25 Вт	Mouser· 291-100-RC
Термометр	Хозяйственный магазин
400-точечная беспаечная макетная плата	Adafruit: 64
Батарейный отсек 4 AA (6 В)	Adafruit: 830

Схема эксперимента

Для проведения эксперимента резистор просто подключается к клеммам батарейного источника питания, а макетная плата нужна лишь для фиксации положения резистора, как показано на рис. 11.1.

Но не подключайте пока батарею — если это сделать, резистор сразу начнет нагреваться.

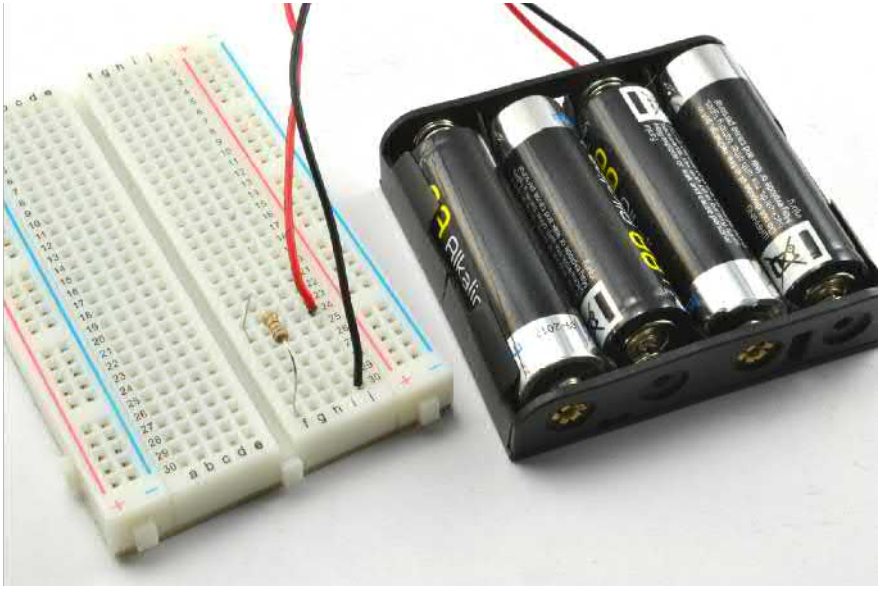


Рис. 11.1. Схема эксперимента с резистивным нагревателем в сборе

Проведение эксперимента

Подключите батарею и с помощью термометра убедитесь, как быстро резистор разогреется. Его температура резистора может достичь значения в 75 °С. Будьте внимательны — этого достаточно, чтобы обжечь палец!

Сопротивление резистора 100 Ом, напряжение в цепи 6 В — отсюда можно вычислить силу тока по формуле:

$$I = V / R = 6 / 100 = 0{,}06 \text{ А}$$

Мощность, рассеиваемая на резисторе, равна току, умноженному на напряжение:

$$6 \times 0{,}06 = 0{,}36 \text{ Вт или } 360 \text{ мВт (милливатт)}$$

Это несколько больше той мощности, которую резистор может рассеивать длительное время (250 мВт). Поэтому при длительной работе в таком режиме он может быстро выйти из строя.

Проект: лопнем шарик с помощью Arduino

Не подумайте, что в основе этого проекта лежит шарикобоязнь. Мы просто воспользуемся платой Arduino для управления прикрепленным к воздушному шарику резистором, играющим роль нагревательного элемента. По истечении случайного промежутка времени резистор включится и, разогревшись, заставит шарик лопнуть (рис. 11.2).

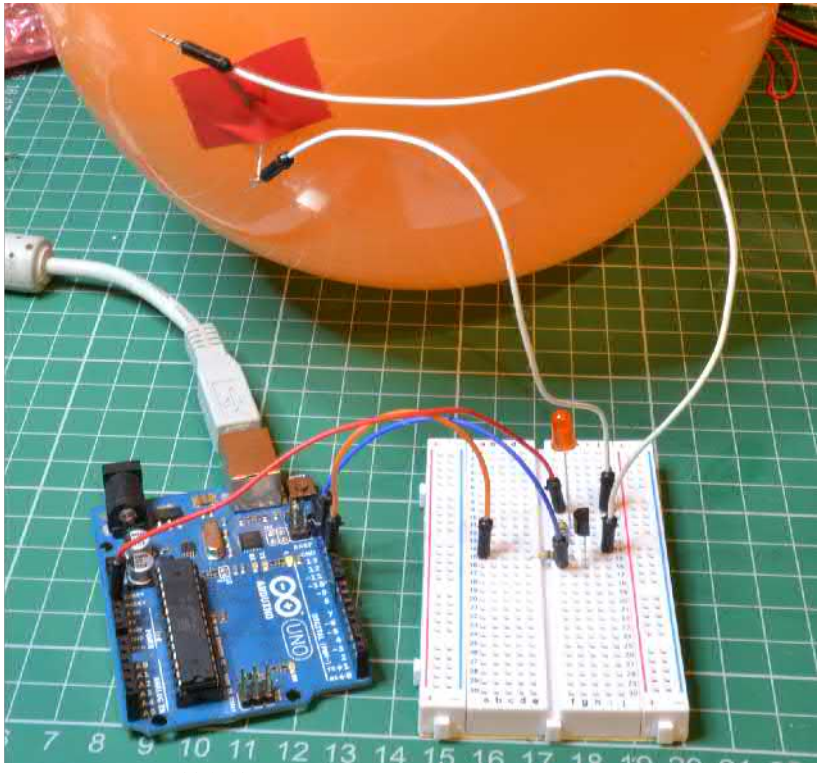


Рис. 11.2. Установка, заставляющая шарик лопнуть, в сборе

Учтите, что этот проект весьма опасный — нагрева резистора, достаточного для того, чтобы шарик лопнул, вполне хватит и для нанесения вреда экспериментатору. В ходе эксперимента вполне может даже пойти дымок.

Опасность ожога

В этом проекте нагрев резистора ведет к тому, чтобы шарик лопнул, а значит, он достаточно горяч и для того, чтобы подпалить пальцы экспериментатора. Пока на резистор подается питание, его нельзя касаться! Прикоснуться к резистору можно не ранее, чем через 30 секунд после отключения питания.

Существует также некоторый риск устроить пожар, поэтому, на всякий случай, под рукой лучше иметь огнетушитель. Кроме того, при схлопывании шарика резистор может с него слететь. И чтобы избежать попадания горячего резистора в глаз, во время эксперимента следует пользоваться защитными очками.

Комплектующие

В этом эксперименте для работы с Arduino понадобятся следующие комплектующие (табл. 11.2).

Таблица 11.2. *Комплектующие для работы с Arduino в эксперименте с шариком*

Обозначение	Компонент схемы	Источники
R1, R3	Резистор 470 Ом 0,25 Вт	Mouser: 291-470-RC
R2	Резистор 10 Ом 0,25 Вт	Mouser: 291-10-RC
Q1	Составной транзистор MPSA14	Mouser: 833-MPSA14-AP
LED1	Красный светодиод	Adafruit: 297 Sparkfun: COM-09590
	400-точечная беспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758

Следует приобрести несколько резисторов по 10 Ом, поскольку они, скорее всего, в ходе эксперимента буквально превратятся в дым.

Схема проекта

Когда сработает составной транзистор, напряжение на резисторе достигнет порядка 3,5 В. При этом через резистор пойдет ток:

$$3{,}5\ \text{В} / 10\ \text{Ом} = 350\ \text{мА}$$

Такой ток выдержит любое USB-устройство, обеспечивающее питание для этого проекта.

Тепловая мощность, выделившаяся на резисторе, составит:

$$I \times V = 350\ \text{мА} \times 3{,}5\ \text{В} = 1{,}225\ \text{Вт}$$

Это намного больше номинальной мощности в 0,25 Вт, на которую рассчитан резистор, тем не менее, даже при такой перегрузке он сможет продержаться достаточно долго, чтобы шарик успел лопнуть.

На рис. 11.3 показана компоновка макетной платы для этого проекта.

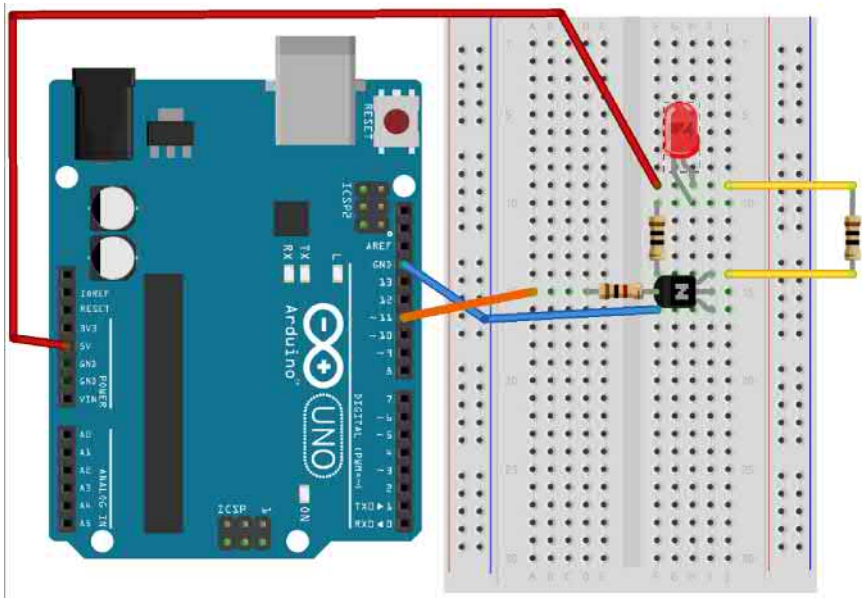


Рис. 11.3. Компоновка макетной платы для проекта с шариком

Обратите внимание: резистор должен быть надежно присоединен к разъемам питающих его проводов (рис. 11.4), — чтобы он не болтался в них в момент, когда шарик лопнет.

Рис. 11.4. Подключение выводов резистора

Программа

В качестве инициатора пуска в этом проекте используется кнопка сброса Arduino. Программа Arduino запускается сразу после загрузки, так что пока вы не готовы лопнуть шарик, один из проводов, идущих к резистору, следует держать отключенным.

Перейдем к скетчу из каталога /arduino/projects/pr_balloon_popper (см. *разд. «Код к книге» главы 2*):

```
const int popPin = 11;

const int minDelay = 3;      // Секунды 1
const int maxDelay = 5;      // Секунды
const int onTime = 3; // Секунды 2

void setup() {
   pinMode(popPin, OUTPUT);
   randomSeed(analogRead(0)); // 3
   long pause = random(minDelay, maxDelay+1); // 4
   delay(pause * 1000); // 5
   digitalWrite(popPin, HIGH); // 6
   detay(onTime * 1000); digitalWrite(popPin, LOW);
}

void loop() {
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Константы `minDelay` и `maxDelay` задают границы допустимого диапазона времени, в который на резистор может быть подано питание, что приведет к схлопыванию шарика.
2. Константа `onTime` определяет, как долго резистор должен быть включен. Чтобы правильно подобрать ее значение, потребуется провести ряд экспериментов. Резистор не следует держать включенным длительное время, иначе он быстро выйдет из строя.
3. Случайные числа в Arduino C на самом деле не совсем случайные — они представляют собой часть длинной псевдослучайной последовательности. Чтобы получить гарантированно разную задержку при каждом старте программы, в этой строке устанавливается позиция в псевдослучайной последовательности, основанная на чтении значения аналогового входа A0. Поскольку этот вход ни к чему не подключен, и на него действуют помехи, то значение на нем можно считать более или менее случайным.
4. Пауза перед включением резистора определяется функцией `random`, возвращающей случайное число, находящееся между двумя значениями, задаваемыми как параметры функции. Ко второму параметру добавляется `1`, чтобы диапазон был всегда реализуемым. Чтобы диапазон задержек мог выходить за пределы 32 секунд, используется переменная типа `long`, поскольку тип `int` может содержать только числа до 32 767.
5. Задержка включения в секундах.
6. Включение транзистора и подача питания на резистор, задержка во включенном состоянии на время `onTime()` и отключение резистора.

Загружаем и выполняем программу

На этапе подготовки проекта к выполнению лучше сначала установить значения переменных `minDelay` и `maxDelay` равными `2` и `3` соответственно — это добавит вам уверенности, что время, требующееся на то, чтобы шарик под нагревом лопнул, выбрано правильно.

Приклейте резистор к шарику липкой лентой и прикрутите выводы резистора к питающим проводам, а провода подключите к макетной плате. Нажмите кнопку сброса и ждите подрыва шарика.

Нагревательные элементы

В практическом плане резисторы — не лучший выбор в качестве нагревателей. Если вам нужно что-либо нагреть, следует использовать специальные *нагревательные элементы* (нагреватели). Нагревательные элементы по своей сути — это большие резисторы, сделанные из материалов, специально разработанных для нагрева и передачи тепла к объекту, который вы собираетесь нагреть.

Нагрев, как правило, требует больше энергии, чем генерирование света, звука или даже движение. Таким образом, нагревательные элементы электрических чайников, бытовых стиральных машин и других приборов, нагревающих воду, питаются переменным током высокого напряжения непосредственно из электрической сети просто потому, что должны получать достаточную мощность для своей работы. Поэтому, перед подключением реальных нагревателей, вам, возможно, следует ознакомиться с материалом *главы 13*.

Мощность нагревательных элементов измеряется в ваттах (Вт) или киловаттах (кВт) (тысячах ватт) так же, как и номинальная мощность резисторов. Это действительно одно и то же. Номинальная мощность резистора — это количество тепловой энергии, которую он может вырабатывать в секунду без того, чтобы слишком разогреться и выйти из строя.

Ранее, в *разд. «Эксперимент: нагрев резистора»*, чтобы рассчитать мощность, мы воспользовались законом Ома и формулой мощности ($P = I \times V$). Если эти выражения объединить, то мы получим простую удобную формулу, с помощью которой, зная сопротивление нагревательного элемента и напряжение на его контактах, можно вычислить его мощность:

$$P = \frac{V^2}{R}$$

Другими словами, тепловая мощность, выделяемая нагревательным элементом, равна квадрату напряжения в вольтах, деленному на сопротивление в омах. Формула справедлива как для постоянного, так и для переменного тока.

Например, резистор (или нагревательный элемент) сопротивлением 10 Ом при напряжении 12 В будет генерировать $(12 \times 12) / 10 = 14{,}4$ Вт тепловой мощности. Если же взять переменное напряжение 120 В, то мощность составит $(120 \times 120) / 10 = 1440$ Вт или 1,44 кВт.

Итак, для любого нагревательного элемента можно определить пару параметров:

- его рабочее напряжение (12 В, 120 В, 220 В);
- ео мощность (50 Вт, 1 кВт, 5 кВт).

Если нагревательный элемент предназначен для нагрева воды, то он и должен находиться в воде, которая будет уносить от него тепло. Без воды он быстро перегреется и перегорит.

Мощность и энергия

На бытовом уровне слова *мощность* и *энергия* часто используются как синонимы. С научной же точки зрения, понятия мощности и энергии столь же различны, как понятия скорости и расстояния.

На самом деле, мощность — это скорость передачи или преобразования энергии. Мощность нагревательного элемента — это количество энергии, выделившейся в виде тепла за секунду.

Единица измерения энергии — джоуль, а ватт, единица измерения мощности, — это изменение энергии на один джоуль в секунду.

От мощности к повышению температуры

Если вы знаете, какова мощность нагревательного элемента и какую субстанцию он нагревает, то можете рассчитать, за какое время температура в этой субстанции поднимется на заданную величину.

В научном мире принято, что для таких расчетов обычно используются международные единицы измерения: ватты, градусы Цельсия, граммы и секунды. Результат вычислений всегда можно впоследствии преобразовать в более привычные вам единицы.

Вещества обладают свойством, называемым *удельной теплоемкостью* (С). Это значение энергии, необходимой, чтобы нагреть один грамм вещества на один градус Цельсия. Например, вода имеет теплоемкость, равную 4,2 Дж/(г·°С). Другими словами, необходимо 4,2 Дж энергии, чтобы нагреть 1 грамм воды на 1 градус Цельсия. Если вы хотите нагреть 100 граммов воды на 1°С, понадобится уже 420 Дж энергии, а чтобы нагреть те же 100 граммов на 10 °С, потребуется энергия в 4200 Дж.

Теплоемкость воздуха составляет около 1 Дж/(г·°С), а теплоемкость стекла — 0,8 Дж/(г·°С). Все вещества разные...

Кипящая вода

Для примера рассчитаем, как наш небольшой резистор вскипятит некоторое количество воды.

Предположим, что мы хотим вскипятить стакан воды (примерно 250 г). Предположим также, что изначально вода находится при комнатной температуре (20 °С)

и закипит при температуре 100 °С — т. е., температура воды должна возрасти на 80 °С.

Общее количество энергии, необходимой, чтобы нагреть 250 г воды на 80 °С, составит:

$$4{,}2 \times 250 \times 80 = 84\ 000 \text{ джоулей}$$

В *разд. «Эксперимент: нагрев резистора»* наш резистор вырабатывал 0,36 Вт тепловой мощности, т. е. 0,36 Дж в секунду. Отсюда получается, что он мог бы нагреть стакан воды за 84 000 / 0,36 = 233 333 секунды, что составляет примерно 64 часа!

Однако в силу практических причин, изложенных в следующей врезке, наш резистор вообще никогда не вскипятит стакан воды.

ТЕПЛОВЫЕ ПОТЕРИ И ТЕПЛОВОЕ РАВНОВЕСИЕ

Очень важно убедиться, что ваш нагревательный (охладительный) элемент имеет достаточную для выполнения заданной работы мощность.

Причина, по которой вы никогда не сможете вскипятить стакан воды резистором мощностью 0,4 Вт, — *потери тепла*. Если бы вода находилась в идеально термически изолированном контейнере, исключающем утечки тепла, тогда — да, вода становилась бы все горячее и горячее, пока бы не вскипела.

Потеря тепла пропорциональна разнице температур нагреваемого объекта и его окружающей среды. При этом, чем вода горячее, тем быстрее уходит тепло. Таким образом, баланс достигается, и температура стабилизируется тогда, когда емкость с водой теряет такое же количество тепла (в ваттах), какое добавляет нагревательный элемент. Вот почему температура резистора в *разд. «Эксперимент: нагрев резистора»* достигает отметки примерно в 75 °С и так на ней и остается.

Элементы Пельтье

Элементы Пельтье (рис. 11.5) обладают очень полезным свойством — когда через них пропускается ток, одна сторона элемента становится более горячей, а другая — более холодной.

Чтобы это произошло, пропускаемый ток должен иметь солидную величину (обычно от 2 до 6 А при 12 В), т. е. для использования элемента Пельтье вам понадобится достаточно мощный блок питания. Такие элементы можно часто обнаружить в переносных холодильниках и в охладителях напитков. Элементы Пельтье в ряде случаев предпочтительнее обычных холодильников, поскольку у них нет движущихся частей, которые могут сломаться.

Как работают элементы Пельтье?

Когда электрический ток протекает через переход между двумя различными проводящими материалами, то материал с одной стороны перехода становится несколько горячее, а с другой — несколько холоднее. Это явление назвали *эффектом Пельтье* в честь французского физика Жана Пельтье, открывшего его в 1894 году. Явление это известно также как *термоэлектрический эффект*.

Рис. 11.5. Элемент Пельтье

Термоэлектрический эффект относительно мал, и чтобы сделать его нам полезным, — например, для охлаждения напитков, его нужно многократно усилить. Это достигается за счет размещения групп переходов друг за другом — тогда при прохождении тока каждый переход вносит свой вклад в общий эффект. Обычные недорогие элементы Пельтье имеют, как правило, около 12 переходов (рис. 11.6). По обеим сторонам элемента Пельтье расположен базовый материал, образующий основу для блока групп переходов.

Материалы, образующие стыки (переходы) элемента Пельтье, — это полупроводники двух типов, аналогичные тем, что используются при производстве транзисто-

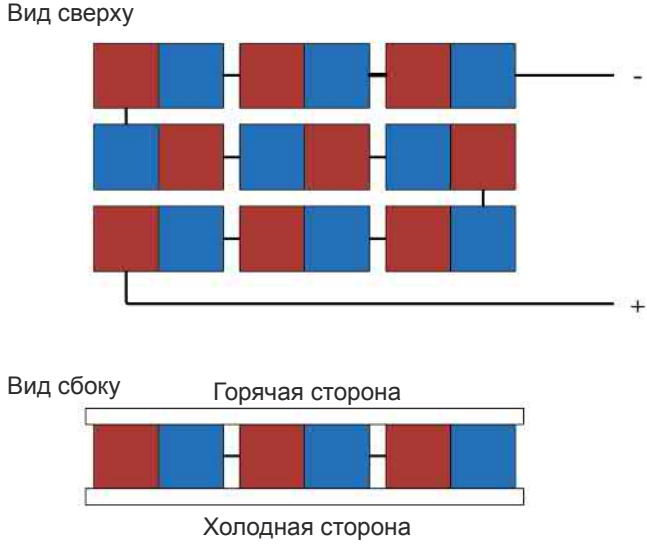


Рис. 11.6. Устройство элемента Пельтье

ров и микросхем, но оптимизированные для термоэлектрического эффекта. Они известны как полупроводники N-типа и P-типа — от слов Negative (отрицательный) и Positive (положительный).

Элементы Пельтье имеют еще одну очень интересную особенность — их можно использовать не только для охлаждения. Если нагреть одну сторону элемента Пельтье больше, чем другую, то элемент будет генерировать небольшое количество электроэнергии.

Особенности практического применения

Основная проблема элементов Пельтье состоит в том, что горячая и холодная его стороны находятся в тесном контакте, вследствие чего горячая сторона очень быстро нагревает холодную. Чтобы уменьшить этот неприятный эффект, нужно снимать тепло с холодной стороны как можно скорее.

Решением такой проблемы может служить применение теплоотводящего радиатора (рис. 11.7). Такой радиатор представляет собой алюминиевую пластину с ребрами, увеличивающими площадь охлаждаемой поверхности, что способствует лучшему отводу тепла. Холодная сторона элемента сконструирована в виде блока, вдающегося внутрь теплоотводящего радиатора.

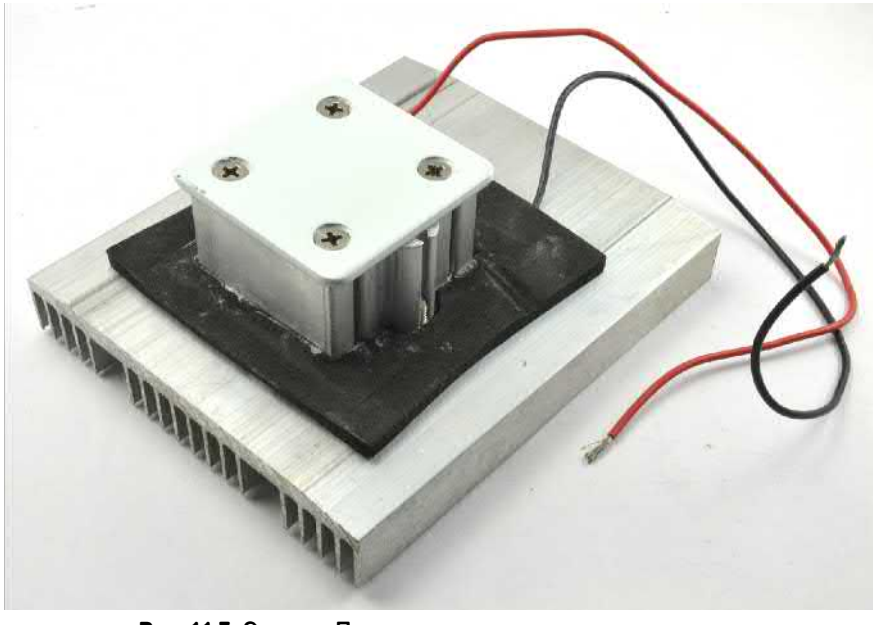


Рис. 11.7. Элемент Пельтье на теплоотводящем радиаторе

Простой радиатор сам по себе не столь эффективен, как тот же радиатор, но с вентилятором. Вентилятор уносит от пластин радиатора нагретый за счет конвекции воздух и заменяет его новой порцией холодного воздуха. На практике некоторые устройства имеют вентиляторы с обеих сторон элемента Пельтье, что позволяет повысить эффективность его работы (рис. 11.8).

Рис. 11.8. Охлаждение элемента Пельтье двумя вентиляторами

Проект: охладитель напитков

В этом проекте не используется ни Raspberry Pi, ни Arduino. Проект просто демонстрирует, как нужно подключать элемент Пельтье, и как можно самому собрать простой охладитель напитков (рис. 11.9). Этот базовый проект мы дополним в *главе 12*, добавив в него контроль за температурой и термостабилизацию. Затем, в *главе 14*, в него будет добавлен OLED-дисплей для отображения температуры.

Комплектующие

В этом проекте нам понадобятся следующие комплектующие (табл. 11.3).

Таблица 11.3. *Комплектующие для работы с проектом охладителя напитков*

Компонент схемы	Источники
Элемент Пельтье с двумя охлаждающими вентиляторами (ток 4 А или меньше)	eBay
Переходник с круглым гнездом и винтовыми зажимами	Adafruit: 368
Источник питания (12 В 5 А)	Adafruit: 352
Большая емкость из-под молока или сока	Б/у
Освежающий напиток	

Рис. 11.9. Охладитель напитков в сборе

Если потребуется задействовать более мощный элемент Пельтье, чем тот, что указан в табл. 11.3, следует воспользоваться более мощным блоком питания, чтобы его максимально допустимый ток наверняка превышал ток, потребляемый элементом. Предусмотрите, как минимум, превышение в половину ампера для вентиляторов и еще в половину ампера на всякий случай.

Конструкция

Рассмотрев провода, идущие к блоку охлаждения, можно выделить три пары проводов: одна пара идет к элементу Пельтье, и по паре проводов — к каждому из вентиляторов. Им всем требуется 12 В от источника питания, и самый простой способ подключения состоит в использовании винтового зажима на разъеме, идущем к блоку питания. Проще говоря, все три красных провода от охлаждающего устройства нужно зафиксировать в винтовой клемме с маркировкой (+), а все три черных провода — в винтовой клемме с маркировкой (–), как показано на рис. 11.10.

Крышку пластикового контейнера, предназначенного для охлаждения емкости с напитком, нужно отрезать, а в его боковой стенке вырезать квадратное отверстие с таким расчетом, чтобы охлаждающая часть элемента Пельтье в этом отверстии

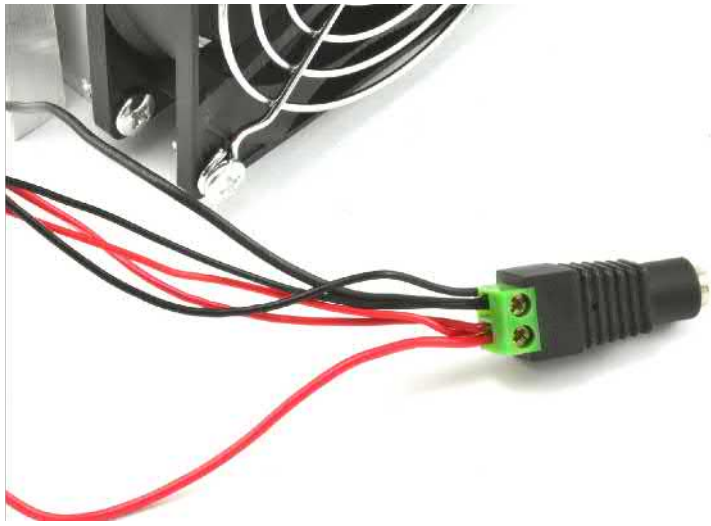


Рис. 11.10. Подключение проводов в проекте охладителя напитков

Рис. 11.11. Подготовка пластикового контейнера

плотно держалась (рис. 11.11). Контейнер должен быть достаточно большим, чтобы в нем можно было разместить для охлаждения любимый стакан или бутылочку.

Если на охлаждающей стороне элемента Пельтье имеются отверстия под винты для крепления, то под них нужно проделать отверстия также и в контейнере, — с помощью винтов элемент можно будет закрепить на контейнере надежнее. При правильно вырезанном отверстии дно контейнера и основание элемента Пельтье

окажутся на одном уровне, и в собранном виде у контейнера не будет причин опрокидываться.

Огромное преимущество использования одного из серии бывших в употреблении пластиковых контейнеров заключается в том, что если сделать вырез в неправильном месте, то для исправления ошибки всегда можно взять другой контейнер и начать все сначала.

Использование охладителя

Когда вся конструкция будет собрана, подключите блок питания, — оба вентилятора должны начать вращаться, и если опустить в контейнер руку, то сразу же можно будет почувствовать холодный воздух, идущий от меньшего вентилятора.

Как показали расчеты, проведенные при попытке вскипятить воду, изменение температуры даже относительно небольшого количества воды занимает много времени. И хотя наш охладитель способен охладить и теплый напиток, гораздо лучше до использования сохранять напиток прохладным, чем дожидаться, пока до нужной температуры охладится теплый напиток.

Этот проект также несколько неэкономичен, на сохранение прохладным небольшого количества напитка расходует 50 Вт электрической энергии. В *главе 12* мы сделаем наш проект несколько эффективней, добавив в него режим термостатирования.

Заключение

В этой главе мы узнали, что быстрый нагрев и охлаждение требуют довольно высокой мощности, и что схемы с использованием транзисторов и реле могут успешно использоваться для управления как нагревателями, так и элементами Пельтье.

В следующей главе мы рассмотрим способы точного регулирования температуры и узнаем, как можно улучшить охладитель напитков, используя терморегулятор.

Контуры управления 12

Хотя эта книга посвящена в основном исполнительным устройствам (двигателям, нагревателям, источникам света и прочим «выходным приспособлениям»), во многих схемах, использующих исполнительные устройства, также находят применение и датчики для отслеживания поведения исполнительных устройств. В качестве характерного примера можно привести термостатический управляемый нагреватель, где управление мощностью нагревательного элемента осуществляется на основе считывания показателей температурного датчика. Такое управление может осуществляться простым включением/выключением элемента, или же быть основано на более совершенной технологии, — например, на так называемом *пропорционально-интегрально-дифференциальном управлении* (от англ. proportional-integral-derivative, PID).

В этой главе рассматриваются вопросы объединения датчиков и исполнительных устройств с целью создания системы управления — эта технология будет применена для доработки проекта охладителя напитков, реализованного в *главе 11*.

Простой термостат

Наверное, для понимания принципов объединения в единую систему управления датчика и исполнительного устройства, лучше всего начать с попытки поддержания постоянного температурного режима за счет использования температурного датчика и нагревательного элемента. Принципы, усвоенные при рассмотрении этого примера, могут с таким же успехом применяться для управления позицией двигателя, или уровнем воды в резервуаре, или в любой другой сфере, допускающей как измерение существующего состояния, так и управление его изменением с применением электронных устройств.

На рис. 12.1 показано, почему такие системы называются *контурами управления*.

В случае с термостатом в доме устанавливается желаемая температура — назовем ее *контрольной*, или *заданной*, *температурой*. Температурный датчик измеряет *фактическую температуру*. Вычитанием *фактической температуры* из *заданной*

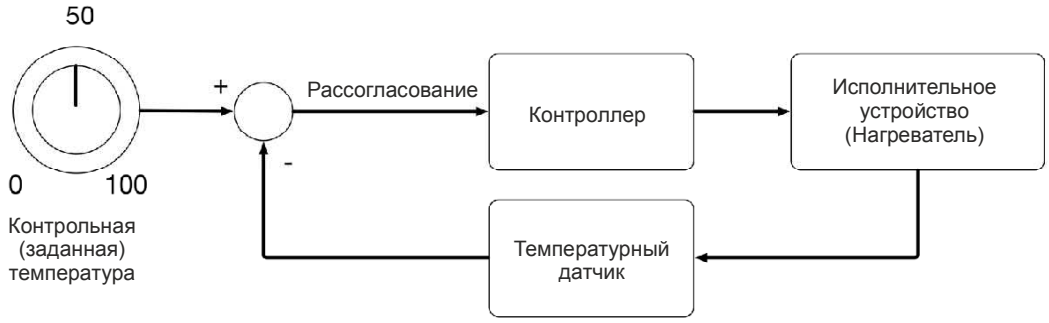


Рис. 12.1. Контур управления (на примере термостата)

определяется величина *рассогласования*. Затем контроллер получает эту величину для вычисления мощности, подаваемой на нагреватель.

В самом простом варианте нагреватель только включается и выключается. Если рассогласование температур имеет положительную величину (т. е. заданная температура выше фактической), значит, в доме достаточно холодно, чтобы включить нагреватель. Или же, если фактическая температура стала выше заданной, значит, в доме достаточно жарко, чтобы снова отключить нагреватель.

В *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»* для создания простого термостата мы воспользуемся Arduino в сочетании с резистором в качестве нагревателя и цифровым температурным датчиком. Это же оборудование будет использовано при изучении более точных способов управления исполнительным устройством.

Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?

В этом эксперименте для реализации простой системы температурного регулирования путем включения/выключения нагревателя используется плата Arduino и интегральная микросхема цифрового температурного датчика DS18B20. Роль нагревательного элемента исполнит резистор сопротивлением 100 Ом, физически прикрепленный к микросхеме DS18B20, измеряющей его температуру.

Для установки заданной температуры через окно монитора порта будут подаваться соответствующие команды, считываемая температура будет также отображаться на этом мониторе. Считанные температурные показатели можно копировать и вставлять в электронную таблицу, создавая затем с ее помощью графики, показывающие качество регулирования температуры.

Небольшие габариты и близкое расположение компонентов позволяют проводить эксперименты с температурным регулятором, не прибегая к использованию крупных (и в силу этого медленно поддающихся изменениям) объектов.

Схема этого эксперимента показана на рис. 12.2 — обратите внимание на прищепку, крепко прижимающую резистор к микросхеме температурного датчика.

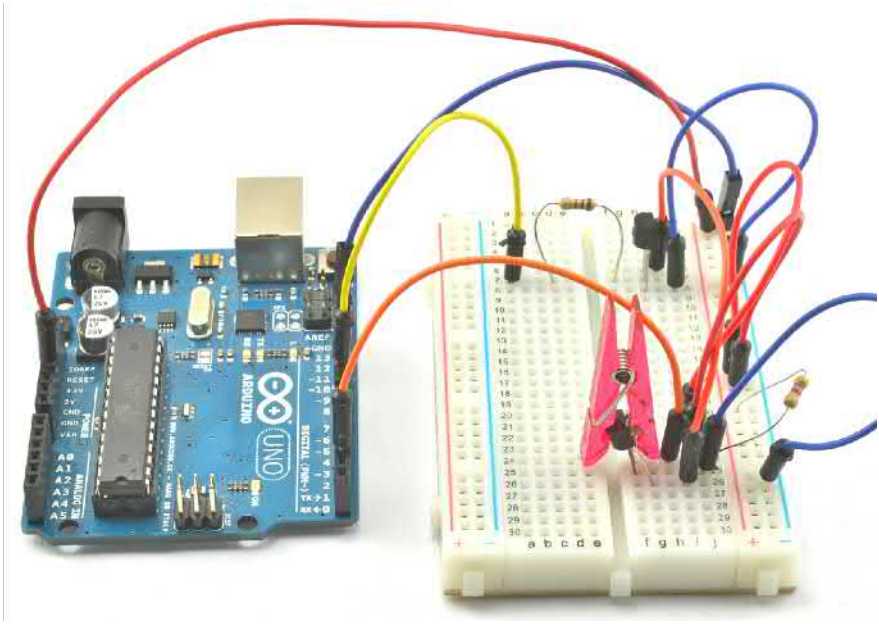


Рис. 12.2. Схема эксперимента по созданию термостатической системы регулирования на основе включения/выключения нагревателя с использованием Arduino в сборе

Комплектующие

В этом эксперименте для работы с Arduino или Raspberry Pi (независимо от того, какая именно плата используется) понадобятся следующие комплектующие (табл. 12.1).

Таблица 12.1. *Комплектующие для работы с Arduino и Raspberry Pi в эксперименте по управлению светодиодом*

Обозначение	Компонент схемы	Источник
IC1	Микросхема цифрового термометра DS18B20	Adafruit: 374 (включает резистор на 4,7 кОм)
R1	Резистор 4,7 кОм	Mouser: 291-4.7k-RC
R2	Резистор 1 кОм	Mouser: 291-1k-RC
R3	100 Ом 0,25 Вт	Mouser: 291-100-RC
Q1	Составной транзистор MPSA14	Mouser: 833-MPSA14-AP
	400-точечная беспаечная макетная плата	Adafruit: 64
	Перемычки «папа-папа»	Adafruit: 758
	Небольшая прищепка	

Если нет небольшой прищепки, можно взять обычную, но обязательно пластмассовую, — чтобы не закоротить ненароком выводы резистора или микросхемы.

Иногда микросхема DS18B20 предлагается совместно с резистором на 4,7 кОм. Кроме того, ее часто продают в водонепроницаемом корпусе. Для эксперимента желательно использовать «голую» микросхему (выглядящую как обычный транзистор), но если есть стремление продолжить работу и создать настоящий термостат, то для проекта из рассматриваемого в этой главе далее *разд. «Проект: термостатический охладитель напитков»* идеально подойдет как раз ее вариант в виде водонепроницаемого датчика.

Принципиальная схема эксперимента

Принципиальная схема для нашего эксперимента показана на рис. 12.3.

Она фактически состоит из двух частей: температурного датчика DS18B20, которому требуется гасящий резистор R1 номиналом 4,7 кОм, и нагревательной части, в центре которой установлен транзистор Q1. Резистор R2 ограничивает ток,

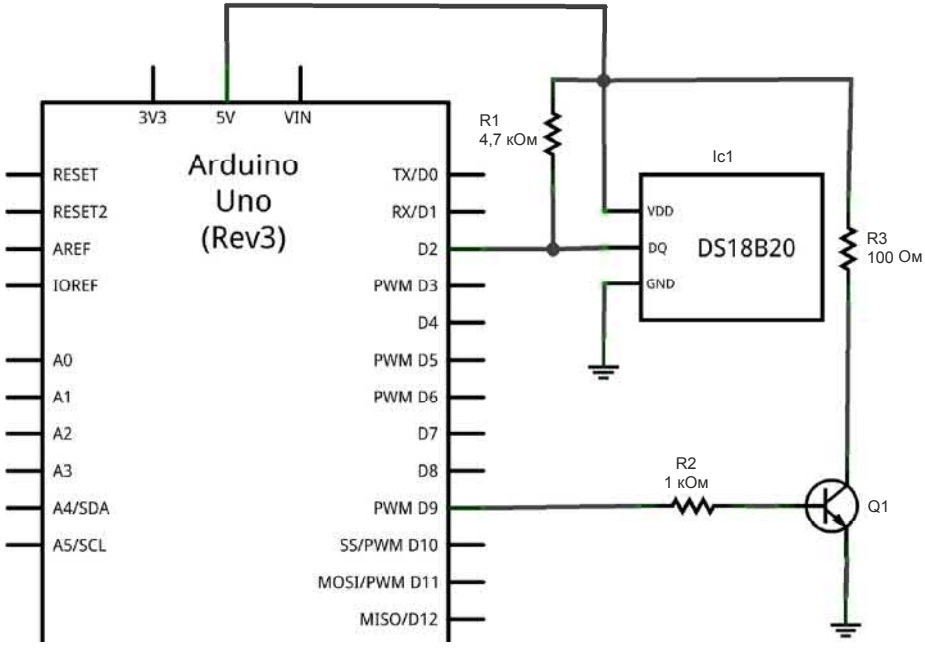


Рис. 12.3. Принципиальная схема эксперимента с термостатом на основе включения/выключения

поступающий на базу транзистора, а резистор R3 служит в качестве нагревательного элемента.

Микросхема DS18B20

Микросхема DS18B20 — весьма полезное малогабаритное устройство. И дело не только в ее точности (±0,5 градуса по Цельсию), но и в том, что она допускает создание цепочки из подобных устройств с использованием только одного контакта Arduino или Raspberry Pi.

Инструкцию по подключению нескольких датчиков DS18B20 и отдельному обращению к ним можно найти в программе для Arduino от Майлза Бартона (Miles Burton) по адресу: **https://milesburton.com/Dallas_Temperature_Control_Library**.

При извлечении данных из нескольких устройств на Raspberry Pi при открытии файлов устройств нужно использовать для этих устройств разные идентификаторы — в соответствии с описанием из *разд. «Программа для Raspberry Pi»* этой главы.

Макетная схема эксперимента

Схема, собранная для проведения эксперимента на макетной плате, показана на рис. 12.4.

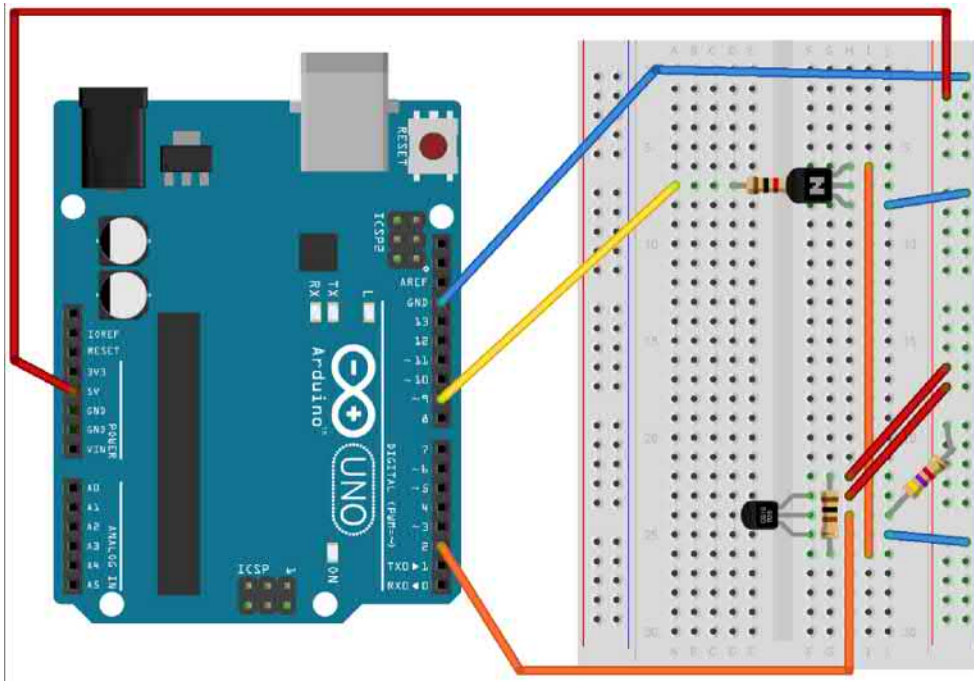


Рис. 12.4. Собранная на макетной плате схема для термостата на основе включения/выключения

Резистор нагревателя R3 размещается сразу же за плоской стороной датчика DS18B20 — чтобы их можно было крепко прижать друг к другу небольшой прищепкой (рис. 12.5).

В схеме задействованы два контакта Arduino: контакт **D9** служит для управления транзистором, включая и выключая с его помощью питание резистора, выступаю-

щего в роли нагревателя, а контакт **D2** является входом сигнала от цифрового датчика.

Размещайте транзистор и микросхему DS18B20 правильно — плоскими сторонами к правому краю макетной платы.

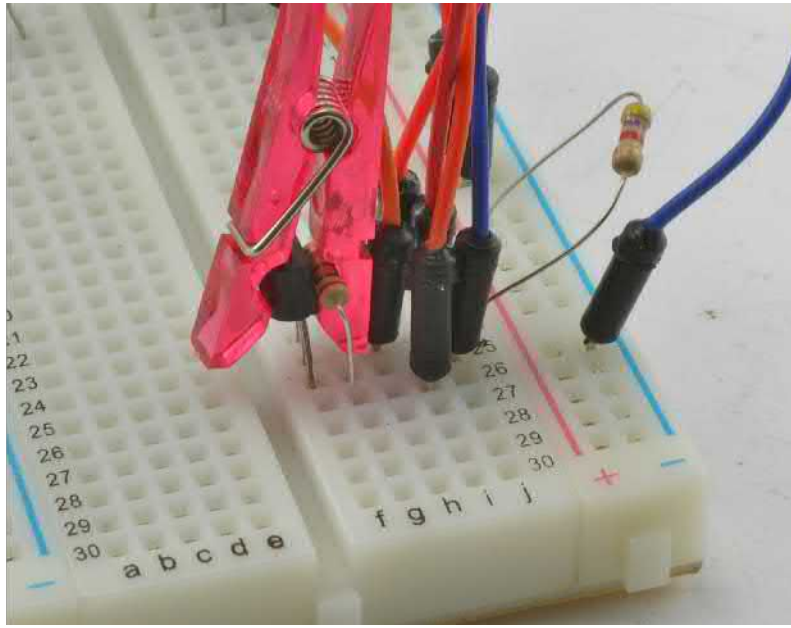


Рис. 12.5. Резистор и датчик DS18B20 прижаты друг к другу небольшой прищепкой

Программа

Микросхема DB18B20 работает на шине под названием *1-Wire* («один провод»). Чтобы использовать эту микросхему в качестве датчика для Arduino, нужно загрузить две библиотеки Arduino и установить их в среду Arduino IDE.

Сначала загрузите библиотеку `OneWire` (**http://www.pjrc.com/teensy/arduino_libraries/OneWire.zip**). Распакуйте загруженный архив и каталог с именем OneWire и установите в вашу среду Arduino (см. далее врезку *«Установка библиотек Arduino»*).

Для загрузки второй библиотеки — непосредственно для самой микросхемы DS18B20 — зайдите на страницу **https://github.com/milesburton/Arduino-Temperature-Control-Library** и щелкните там по кнопке **Download ZIP** (Загрузить ZIP-файл). После распаковки архива у вас образуется папка с именем dallas-temperature-control, которую, перед тем как переместить ее в ваш каталог libraries, нужно переименовать в DallasTemperature.

Установка библиотек Arduino

Среда Arduino IDE включает множество предустановленных библиотек, но иногда возникает необходимость загрузить библиотеку, не включенную в IDE-среду из Интернета. Делается это весьма просто: обычно библиотека загружается в виде ZIP-архива,

который нужно распаковать в отдельный каталог. Имя этого каталога должно совпадать с именем библиотеки, но случается, что загруженный каталог не будет иметь точно такое же имя, особенно когда загрузка ведется из GitHub.

После переименования каталога (при необходимости) его нужно переместить в ваш, предназначенный для Arduino, каталог *libraries*, который находится в вашем же каталоге *arduino*. Среда Arduino IDE должна была автоматически создать каталог *arduino* внутри вашего каталога *Documents* — именно в нем Arduino хранит свои небольшие программы.

Создавая самую первую библиотеку, прежде чем перемещать загруженную библиотеку в каталог *libraries* (его пока еще нет), в каталоге *arduino/* нужно создать каталог с этим именем: *libraries*.

Чтобы после установки новой библиотеки она была распознана, понадобится провести перезапуск среды Arduino IDE.

Скетч Arduino для этого проекта находится в каталоге arduino/experiments/simple_thermostat/ (см. *разд. «Код к книге» главы 2*):

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int tempPin = 2; // 1
const int heatPin = 9;
const long period = 1000; // 2

OneWire oneWire(tempPin); // 3
DallasTemperature sensors(&oneWire);

float setTemp = 0.0; // 4
long lastSampleTime = 0;

void setup() {
   pinMode(heatPin, OUTPUT);
   Serial.begin(9600);
   Serial.println("t30 - устанавливает температуру на 30");
   sensors.begin(); // 5
}

void loop() {
   if (Serial.available()) { // 6
      char command = Serial.read();
      if (command == 't') {
         setTemp = Serial.parseInt();
         Serial.print("Установлена температура ");
         Serial.println(setTemp);


   long now = millis(); // 7
   if (now > lastSampleTime + period) {
      lastSampleTime = now;
```

```
    float measuredTemp = readTemp(); // 8
    float error = setTemp - measuredTemp;
    Serial.print(measuredTemp);
    Serial.print(", ");
    Serial.print(setTemp);
    if (error > 0) { // 9
       digitalWrite(heatPin, HIGH);
       Serial.println(", 1");
    }
    else {
       digitalWrite(heatPin, LOW);
       Serial.println(", 0");
```

```
float readTemp() { // 10
   sensors.requestTemperatures();
   return sensors.getTempCByIndex(0);
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Код начинается с определения констант для контакта считывания температуры (`tempPin`) и контакта управления нагревателем (`heatPin`).
2. Константа `period` используется для установки интервала времени между последовательными считываниями температуры. Микросхема DS18B20 не обладает особо высоким быстродействием, и этот интервал может составлять до 750 мс, поэтому для `period` лучше выбрать значение не меньше, чем `750`.
3. Обеим библиотекам, `DallasTemperature` и `OneWire`, нужны переменные, определяемые для получения доступа к ним.
4. Переменная `setTemp` содержит текущую заданную температуру, а переменная `lastSampleTime` используется для показателя времени съема последнего значения температуры.
5. Инициализация библиотеки `DallasTemperature`.
6. Первая часть функции `loop()` проверяет последовательные команды, поступающие из окна монитора порта. Фактически, имеется только одна возможная команда (`'t'`), за которой следует желаемая устанавливаемая температура в градусах Цельсия. После установки нового значения `setTemp`, оно возвращается эхом в окно монитора порта для подтверждения.
7. Вторая половина `loop()` сначала проверяет, не настало ли время для нового считывания, т. е. не истек ли период, указанный в миллисекундах.
8. Температура измеряется, и вычисляется рассогласование, после чего значения `measuredTemp` и `setTemp` записываются в окне монитора порта.

9. Если рассогласование больше нуля, нагреватель включается, и в конце строки на мониторе появляется цифра `1`, означающая включение питания нагревателя. В противном случае нагреватель выключается, и на экран отправляется цифра `0`.
10. Всю цепочку, состоящую из датчиков DS18B20, можно подключать к одному и тому же входному контакту, поэтому команда `requestTemperatures` просит все подключенные микросхемы DS18B20 (в нашем случае, только одну микросхему) провести измерение температуры и отправить результат. Затем для возвращения считанного результата с первой и единственной подключенной микросхемы DS18B20 используется вызов функции `getTempCByIndex`.

Загружаем и выполняем программу

Загрузите программу и откройте окно монитора порта — в нем должен появиться постоянный поток считанных температурных данных (учтите, что микросхема DS18B20 разработана для работы по температурной шкале Цельсия, и, поскольку этой главе присуща некая научная составляющая, я на всем ее протяжении придерживаюсь температурных единиц, выраженных в градусах Цельсия):

```
t30 - устанавливает температуру на 30
21.75, 0.00, 0
21.69, 0.00, 0
21.75, 0.00, 0
21.69, 0.00, 0
21.75, 0.00, 0
```

В трех выводимых столбцах показываются: измеренная температура, заданная температура (в данном случае `0`) и индикатор включения электропитания нагревателя (`0` или `1`).

Введите команду `t30`, чтобы установить желаемую температуру равной 30 °С. Температура тут же начнет подниматься, поскольку нагреватель сразу включится, о чем будет свидетельствовать цифра `1` в третьем столбце:

```
21.75, 0.00, 0
21.75, 0.00, 0
Установленная температура=30.00
21.75, 30.00, 1
21.75, 30.00, 1
21.81, 30.00, 1
21.94, 30.00, 1
22.06, 30.00, 1
```

Когда температура достигнет 30 °С, нагреватель должен будет отключиться, и вы должны увидеть цикл включения/выключения нагревателя при колебании температуры возле значения 30 °С:

```
29.87, 30.00, 1
29.94, 30.00, 1
```

```
29.94, 30.00, 1
30.00, 30.00, 0
30.06, 30.00, 0
30.12, 30.00, 0
30.06, 30.00, 0
29.94, 30.00, 1
29.81, 30.00, 1
29.75, 30.00, 1
```

Получилось! Вы создали термостат, и он вполне может пригодиться для решения многих задач. Но не трудно заметить, что температура «рыскает» вокруг заданного значения в 30 градусов. Размах этого рысканья можно увидеть, скопировав фрагмент данных, вставив его в файл, используя текстовый редактор, и задав для имени файла расширение csv. Затем этот файл можно импортировать в удобную для вас программу электронной таблицы и создать из данных файла график, получив в результате изображение, один из вариантов которого показан на рис. 12.6.

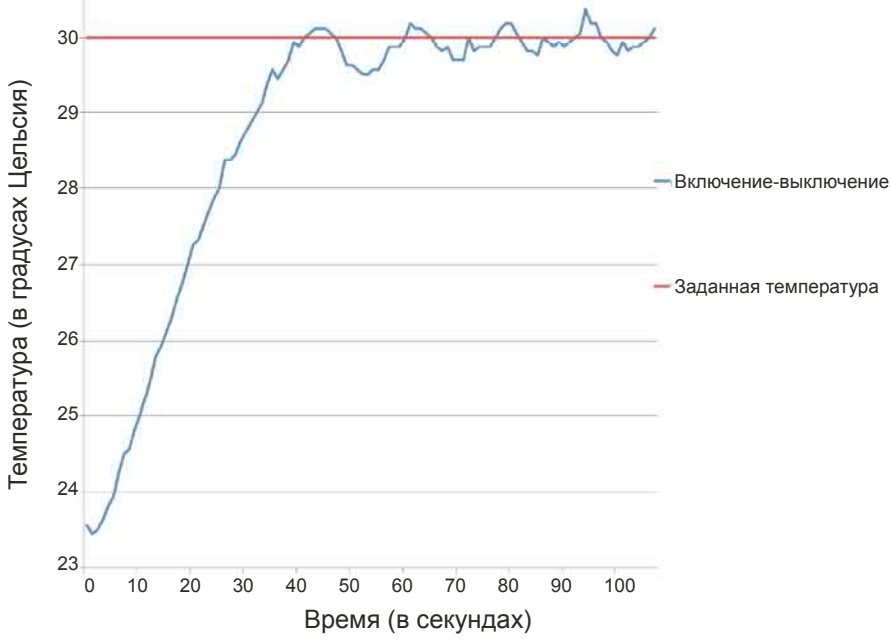


Рис. 12.6. Построение графика температуры простого термостата, работающего на основе включения/выключения

Как видите, температура колеблется выше-ниже заданной почти на полградуса. Существенно улучшить наши результаты можно за счет использования уже упомянутой ранее технологии управления, называемой *пропорционально-интегрально-дифференциальным регулированием* (ПИД). И хотя само название — ПИД — наводит на мысль о применении сложных математических выкладок, всю работу можно упростить, следуя нескольким основным правилам.

Гистерезис

Использование транзистора для включения/выключения питания позволяет добиться высокой скорости этого процесса и воспользоваться технологией ШИМ. Но для некоторых систем очень частое включение/выключение электропитания устройства крайне нежелательно. Это может относиться и к бытовому нагревателю, которому нужно время для начала выработки тепла после его включения в сеть. К тому же, управление им с использованием электромеханических выключателей и других подобных устройств в режиме постоянного включения/выключения сократит срок службы нагревателя.

Чтобы предотвратить слишком быстрое переключение, нужно установить ему минимальное время, не позволяющее устройству отключаться, пока оно не истечет, или же ввести *гистерезис* (рис. 12.7).

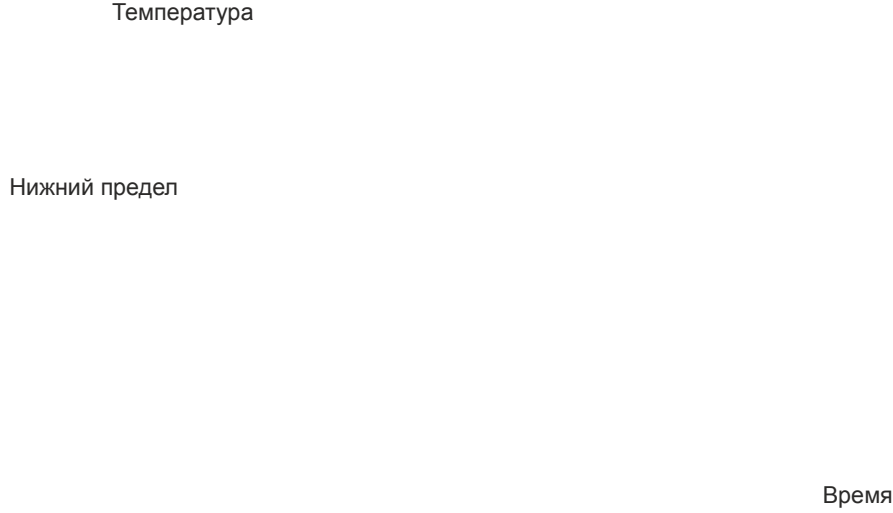


Рис. 12.7. Гистерезис в термостате

Применительно к термостату, гистерезис означает наличие не одного, а двух температурных порогов, один из которых выше другого на определенную фиксированную величину. Если температура падает ниже нижнего порога, нагреватель включается, но выключается он только при превышении температурой верхнего порога.

Таким образом, для введения задержки в процесс переключения используется естественная инерционность системы.

ПИД-управление

Выключение нагревателя в тот момент, когда температурный датчик показывает, что нагреватель горячее заданной температуры, и его включение, когда он холоднее ее, приводит к созданию системы, где температура «рыскает» вокруг заданного значения (см. рис. 12.6).

Для более точного поддержания температуры следует воспользоваться пропорционально-интегрально-дифференциальным (ПИД) регулированием.

Вместо простого включения/выключения нагревателя, ПИД-контроллер регулирует выходную мощность нагревателя (или другого исполнительного устройства), принимая в расчет три фактора: пропорциональность, интегральность и дифференциальность.

> ***ПРЕДУПРЕЖДЕНИЕ***
>
> В этом разделе нам предстоит более глубокое погружение в теорию. Вопросы по ПИД-управлению мне задавались весьма часто, и в качестве одной из основных целей этой главы я выбрал рассмотрение принципов работы и применения ПИД-регулирования.

Пропорциональность (П)

Используя одну лишь П-часть ПИД-контроллера и игнорируя две остальные части — И и Д, можно на многих системах добиться весьма неплохих результатов. Поэтому на каждой новой системе в любом случае следует начинать с одной только части П, и смотреть, насколько приемлемым окажется результат.

Пропорциональное управление означает, что подаваемая на нагреватель мощность пропорциональна рассогласованию. Чем больше рассогласование, т. е. чем ниже фактическая температура от заданной, тем больше мощности подается на нагреватель. Как только фактическая температура близко подходит к заданной, мощность снижается, и получается, что фактическая температура не настолько сильно превышает заданную, как показано на рис. 12.6. В зависимости от системы, она, скорее всего, все же превысит заданную, но совсем не настолько, как это происходит при простом управлении на основе включения/выключения. Этот процесс отчасти напоминает приближение на машине к знаку СТОП — подъезжая к нему, вы начинаете тормозить заранее, а не непосредственно у самого знака.

Если фактическая температура выше заданной, рассогласование будет отрицательным, и это должно выразиться в отъеме мощности от нагревателя. Если нагреватель был бы элементом Пельтье (см. *главу 11*), то можно было бы, используя Н-мост, пустить ток через него в обратном направлении и превратить его из нагревателя в охладитель. Впрочем на практике, если только окружающая температура не сильно отличается от заданной, так делать не приходится. Вместо этого можно просто дать системе остыть (или нагреться) естественным образом.

Рассогласование, на котором основывается вычисление подаваемой выходной мощности в случае с термостатом, — это разница температур. Выражая его в градусах Цельсия, получим, что если заданная температура равна 30 °С, а измеренная — 25 °С, то рассогласование равно 5 °С (30 °С – 25 °С). Если для установки выходной мощности используется ШИМ-схема Arduino, то выход будет иметь значение в диапазоне от 0° до 255 °С. Поэтому непосредственная установка на величину рассогласования (5) приведет к подаче на нагреватель слишком малой мощности, которой, скорее всего, не хватит, чтобы он нагрелся до 30 °С. Следовательно, чтобы получить значение подаваемой выходной мощности, рассогласование умножается на число, называемое `kp` (которое иногда называют также *увеличением*, от англ.

gain). Изменение kp определит, насколько быстро температура станет подниматься до заданной. При низком значении kp температура может никогда не дойти до заданной, но при слишком высоком значении kp система будет вести себя точно так же, как контроллер температуры на основе включения/выключения, и фактическая температура станет колебаться вокруг заданной температуры. На рис. 12.8 показано, как различные значения kp меняют поведение идеализированной системы.

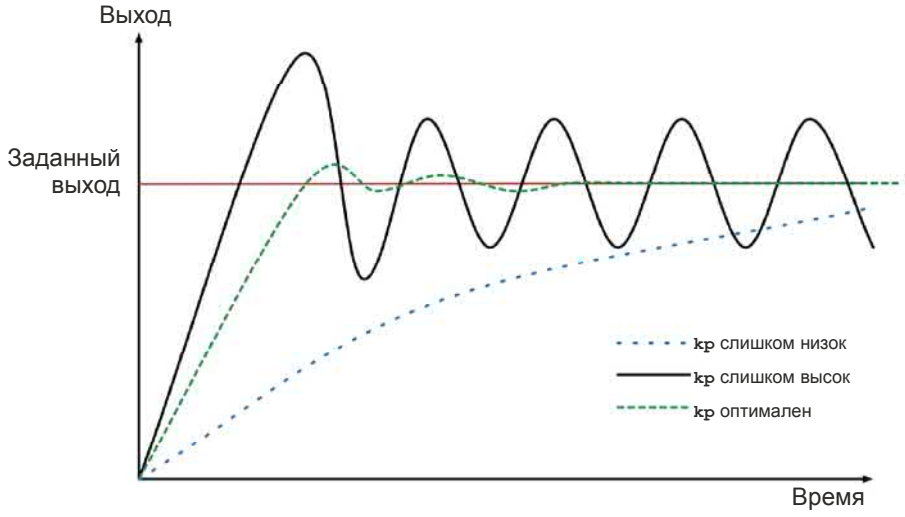


Рис. 12.8. Влияние величины kp (gain) на выход в пропорциональном контроллере

Здесь видно, что если значение kp недостаточно высоко, температура может никогда не подняться до заданной, или же на это может уйти слишком много времени. С другой стороны, если значение kp слишком велико, на выходе начнутся колебания, и их величина не станет уменьшаться. Подходящее значение kp будет быстро подгонять выход до заданного уровня, возможно, с небольшим превышением, а затем колебания быстро сойдут (успокоятся) на приемлемо низкий уровень.

Поиск подходящего значения kp называется *настройкой системы*. Настройка ПИД-регулирования будет рассмотрена далее, в *разд. «Настройка ПИД-контроллера»*.

Вернемся к примеру с заданной температурой, равной 30 °С, но при фактической температуре 25 °С. В этом случае по-прежнему желательно, чтобы на нагревателе рассеивалась максимальная мощность, т. е. для ШИМ был бы установлен коэффициент заполнения 255 (100%). Если в качестве значения kp выбрать 50, то рассогласование в 5 единиц приведет к подаче следующей выходной мощности:

$$\text{Выход} = \text{рассогласование} \times kp = 5 \times 50 = 250$$

Как только система окажется всего в 1 °С от заданной температуры, выходная мощность упадет до следующего значения:

$$\text{Выход} = \text{рассогласование} \times kp = 1 \times 50 = 50$$

Этой мощности может хватить, а может и не хватить для достижения системой заданной температуры. Системы отличаются друг от друга, поэтому контроллер нуждается в настройке.

Интегральность (И)

Если предположить, что от пропорционального управления мощностью нужной точности добиться не удается, в вычисления может понадобиться добавить некое значение I, чтобы выходная мощность вычислялась по следующей формуле:

$$\text{Выход} = \text{рассогласование} \times \texttt{kp} + \text{I} \times \texttt{ki}$$

Здесь присутствует новая константа `ki`, масштабирующая наше новое загадочное свойство по имени I (откроем секрет — это как раз и есть *Интегральность*), и выходная мощность будет вычисляться путем сложения этой *интегральной составляющей* с уже рассмотренной нами ранее пропорциональной составляющей. Такой тип контроллера называется ПИ-контроллером (PI, или иногда P + I, не путайте с Pi в названии Raspberry Pi). Как применение чисто пропорционального контроллера, так и применение ПИ-контроллера, иногда приводит к получению вполне приемлемых результатов, исключая необходимость добавления третьей составляющей ПИД-регулятора, Д.

Интегральная составляющая вычисляется путем сохранения текущей суммы рассогласований при каждом измерении температуры. Когда рассогласование имеет положительное значение (разогрев), I-составляющая будет становиться все больше и больше, превышая значение первоначального ответа. Уменьшение начнется только тогда, когда рассогласование станет отрицательным после превышения заданной температуры.

Как только фактическая температура достигнет заданной, I-составляющая окажет успокаивающее воздействие, сглаживая температурные изменения, чтобы температура лучше закреплялась на желаемом значении.

Единственная проблема, связанная с I-составляющей, заключается в том, что она дает большой прирост по мере подъема температуры, это приводит к возможному большему превышению температуры, после чего требуется время на то, чтобы она снизилась до заданной и стабилизировалась.

Дифференциальность (Д)

На практике D-составляющая зачастую в реальных системах управления не используется, потому что преимущества от ее использования для снижения промахов перевешиваются усложнением настройки.

При добавлении к программе управления D-составляющей, противодействующей эффекту промахов, выходная мощность вычисляется по следующей формуле:

$$\text{Выход} = \text{рассогласование} \times \texttt{kp} + \text{I} \times \texttt{ki} + \text{D} \times \texttt{kd}$$

D-составляющая является мерой того, насколько быстро меняется рассогласование между каждыми замерами температуры, и поэтому она в некотором смысле предсказывает, куда пойдет температура.

Настройка ПИД-регулятора

Настройка ПИД-контроллера означает поиск значений `kp`, `ki` и `kd`, заставляющих систему вести себя подобающим образом. Я рекомендую не усложнять себе жизнь, и использовать только ПИ-регулировку, устанавливая для `kd` нулевое значение, — тогда останется настроить только два параметра. Фактически, это упрощение облегчает настройку большинства систем, однако несколько увеличивает затрачиваемое на нее время.

В рассматриваемом в этой главе далее *разд. «Эксперимент: термостатический ПИД-регулятор»*, я проведу вас через метод проб и ошибок, который неплохо подходит для тех температур, с которыми работает контроллер. Остальная часть раздела будет посвящена наиболее популярному способу ПИД-настройки, носящему название *метод Циглера-Никольса* (Ziegler-Nichols). Этот метод существенно сокращает процесс получения значений `kp`, `ki` и `kd`, сводя его к серии экспериментов с последующими простыми вычислениями.

Настройка Циглера-Никольса начинается с установки нулевых значений для `ki` и `kd`, чтобы контроллер стал работать в пропорциональном режиме. Значение `kp` затем постепенно увеличивается, пока выходная мощность не начнет колебаться. Значение `kp`, при котором это происходит, называется `ku`.

После определения значения `ku` нужно измерить количество колебаний в секунду (период колебаний, называемый здесь `pu`) (рис. 12.9).

Чтобы его получить, нужно нанести на график показания мощности на нагрузке, как это делалось в эксперименте из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»*.

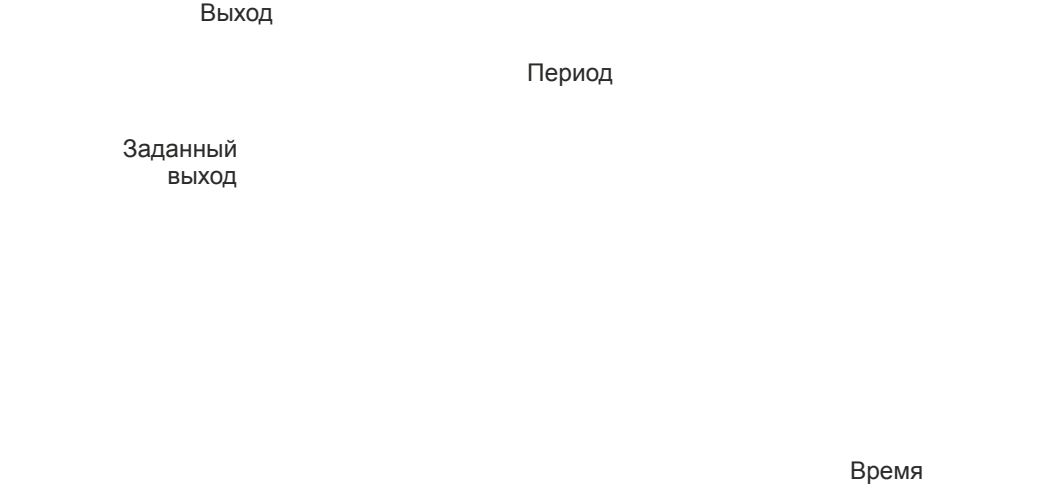


Рис. 12.9. Определение количества колебаний в секунду (периода колебаний)

Затем можно вычислить значения для `kp`, `ki` и `kd`:

```
kp = 0.6 × ku
ki = (2 × kp) / pu
kd = (kp × pu) / 8
```

Если у вас используется ПИ-контроллер, то лучше воспользоваться следующими вычислениями:

```
kp = 0.45 × ku
ki = (1.2 × kp) / pu
kd = 0
```

Статью, посвященную методу Циглера-Никольса, можно найти в Википедии (**https://ru.wikipedia.org/wiki/ПИД-регулятор**).

Эксперимент: термостатический ПИД-регулятор

В этом эксперименте мы исследуем работу ПИД-регулятора с использованием как Arduino, так и Raspberry Pi. Программа для ПИД-контроллера с использованием Arduino берется из библиотеки, а программу для ПИД-контроллера с использованием Raspberry Pi придется набирать вручную.

Оборудование

В этом эксперименте используется точно то же самое оборудование, что и в эксперименте из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»*. Но при подключении его к Raspberry Pi, нужно проявить особое внимание, поскольку микросхему DS18B20 и нагреватель из резистора следует запитать на Pi отдельно: микросхему DS18B20 — от напряжения 3,3 В, а резисторный нагреватель — от напряжения 5 В. Дело в том, что цифровой выход из DS18B20 должен повышаться до 3,3 В, а не до 5 В, чтобы не повредить входные цепи Raspberry Pi.

Экспериментируем с Arduino

Программа для Arduino

В этом эксперименте используется готовая к использованию ПИД-библиотека (**https://github.com/br3ttb/Arduino-PID-Library/**), которую можно загрузить и установить согласно рекомендациям, представленным ранее во врезке *«Установка библиотек Arduino»*. Полную документацию по этой библиотеке можно найти на веб-сайте Arduino (**http://playground.arduino.cc/Code/PIDLibrary**).

Скетч, используемый в эксперименте с Arduino, можно найти в каталоге /arduino/experiments/pid_thermostat:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <PID_v1.h>

const int [redacted] = 2;
const int [redacted] = 9;
const long [redacted] = 1000; // >750

double kp = [redacted] 1
double kd = [redacted]
double ki = [redacted]

OneWire oneWire(tempPin);
DallasTemperature sensors(&oneWire);

double setTemp = 0.0;
double measuredTemp = 0.0;
double outputPower = 0.0; // 2
long lastSampleTime = 0;

PID myPID(&measuredTemp, &outputPower, &setTemp, kp, ki, kd, DIRECT); // 3

void setup() {
   pinMode(heatPin, OUTPUT);
   Serial.begin(9600);
   Serial.println("t30 - устанавливает температуру на 30");
   Serial.println("k50 20 10 - устанавливает, соответственно, kp, ki и kd");
   sensors.begin();
   myPID.SetSampleTime(1000); // 4
   myPID.SetMode(AUTOMATIC);
}

void loop() {
   checkForSerialCommands(); // 5
   long now = millis();
   if (now > lastSampleTime + period) { // 6
      lastSampleTime = now;
      measuredTemp = readTemp();
      myPID.Compute();
      analogWrite(heatPin, outputPower);

      Serial.print(measuredTemp); // 7
      Serial.print(", ");
      Serial.print(setTemp);
      Serial.print(", ");
      Serial.println(outputPower);
```

```
void checkForSerialCommands() { // 8
      if (Serial.available()) {
      char command = Serial.read();
      if (command == 't') {
         setTemp = Serial.parseFloat();
         Serial.print("Установлена температура ");
         Serial.println(setTemp);
      }
      if (command == 'k') {
         kp = Serial.parseFloat();
         ki = Serial.parseFloat();
         kd = Serial.parseFloat();
         myPID.SetTunings(kp, ki, kd);
         Serial.print("Установка констант kp=");
         Serial.print(kp);
         Serial.print(" ki=");
         Serial.print(ki);
         Serial.print(" kd=");
         Serial.println(kd);
      }
   }
}

double readTemp() {
   sensors.requestTemperatures();
   return sensors.getTempCByIndex(0);
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях.

> ***Примечание***
>
> Код, относящийся к считыванию температурных данных с DS18B20, ничем не отличается от кода для эксперимента из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»*, поэтому здесь мы сосредоточим внимание на коде, относящемся к ПИД-регулированию.

1. В качестве коэффициентов масштабирования для P-, I- и D-компонентов выхода определяются три переменные: `kp`, `ki` и `kd`. Использование переменных обусловлено тем, что они могут изменяться с помощью команд в окне монитора порта в ходе выполнения программы. Тип `double` использован вместо типа `float` отчасти потому, что он позволяет работать с более точными числами, чем `floats`, а также потому, что именно этот тип ожидается библиотекой.
2. Переменная `outputPower` будет содержать коэффициент заполнения ШИМ в пределах от 0 до 255.
3. Для обращения к коду ПИД-библиотеки определена переменная `myPID`. Заметьте, что первые три параметра при создании переменной для обращения к ПИД-

библиотеке являются именами переменных `measuredTemp`, `outputPower` и `setTemp`, перед которыми стоит префикс `&`. Это прием языка C, позволяющий ПИД-библиотеке изменять значения этих переменных, даже если они не являются частью библиотеки. Если вас интересуют дополнительные сведения об этой технологии (указателях языка C), обратитесь к материалу Tutorials Point (**http://www.tutorialspoint.com/cprogramming/c_pointers.htm**). Последний параметр (`DIRECT`) устанавливает для ПИД-операции прямой режим, который в случае применения этой библиотеки означает, что выход будет пропорционален рассогласованию, и не будет инвертирован. Для удобства по умолчанию диапазон выходных значений для ШИМ устанавливается от 0 до 255.

4. Время периодичности снятия показаний нужно установить равным 1 секунде (1000 мс). ПИД-вычисления запускаются установкой режима на `AUTOMATIC`.
5. Теперь проверка поступления последовательных команд находится в своей собственной функции, останавливающей цикл `loop()`, что облегчает чтение кода, который ранее был слишком многословен. См. также сноску номер 8.
6. Как только наступает время очередного замера температуры, происходит ее считывание в переменную `measuredTemp`, а затем ПИД-библиотека получает предписание на обновление своих вычислений (`myPID.Compute`). При этом автоматически производится обновление значения `outputPower`, которое затем используется для установки коэффициента заполнения ШИМ на контакте, используемом для управления резисторным нагревателем.
7. Все значения выводятся в окне монитора порта, поскольку они нужны нам для построения графиков и отслеживания работы контроллера.
8. Функция `checkForSerialCommands` проверяет выдачу команды `'t'` для задания температуры, как и в эксперименте из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»*, но также проверяет и наличие команды `k`, за которой следуют три числа: `kp`, `ki` и `kd`, устанавливая параметры настройки в случае получения команды.

Загружаем и выполняем программу

Оборудование и программа, предназначенные для этого эксперимента, дают нам все необходимое, чтобы провести эксперимент с ПИД-контроллером. Мы можем изменить заданную температуру и три значения — `kp`, `ki` и `kd` — и записать влияние, оказываемое ими на выходе. В нашем случае вполне приемлемые результаты будут получены с ПИ-регулятором, поэтому для `kd` может быть просто установлено значение `0`.

Итак, загрузите программу в Arduino и откройте окно монитора порта (рис. 12.10).

Настройка контроллера требует времени. Нужно будет записать данные и построить на их основе график, чтобы увидеть, насколько приемлемо поведение системы. Сначала следует подобрать подходящее значение для `kp`. Давайте просто попробуем значения 50, 500 и 5000, чтобы получить представление о системе.

Рис. 12.10. Использование окна монитора порта для тестирования ПИД-регулирования

Прежде всего установим параметры настройки, введя в окне монитора порта следующие значения:

```
k50 0 0
```

В результате для `kp` будет установлено значение 50, а для `ki` и `kd` — значения, равные нулю. Если хотите, можете ввести значения с указанием десятичной части (например, `k30.0 0.0 0.0`). В обоих случаях, числа будут преобразованы в формат чисел с плавающей точкой.

Теперь давайте установим заданную температуру на 30 °С (это значение я выбрал как наиболее подходящее, превышающее значение температуры окружающей среды примерно на 7 или 8 градусов):

```
t30
```

Температура должна начать подниматься, что станет отображаться на экране, где появятся три столбца, показывающие фактическую температуру, заданную температуру и значение на выходе ШИМ (в диапазоне от 0 до 255):

```
25.06, 30.00, 246.88
25.19, 30.00, 240.63
25.31, 30.00, 234.38
25.44, 30.00, 228.13
```

Заметьте, что значение на ШИМ-выходе сохранено в виде числа с плавающей точкой, поэтому у него имеются цифры после десятичной точки. При вызове функции `analogWrite` это значение должно быть усечено до целого числа в диапазоне от 0 до 255.

Как можно видеть, значение коэффициента заполнения ШИМ в последнем столбце при заданном для `kp` значении 50 начинает почти сразу же снижаться. Это означает,

что значение kp выбрано слишком низким, но вы продолжайте собирать данные еще несколько минут, а затем перенесите их в текстовый файл и импортируйте этот файл в программу электронной таблицы. Я начал копировать данные, когда температура достигла 25 °С. Постройте по показателям температуры график и получите примерно такую же картину, что изображена на рис. 12.11.

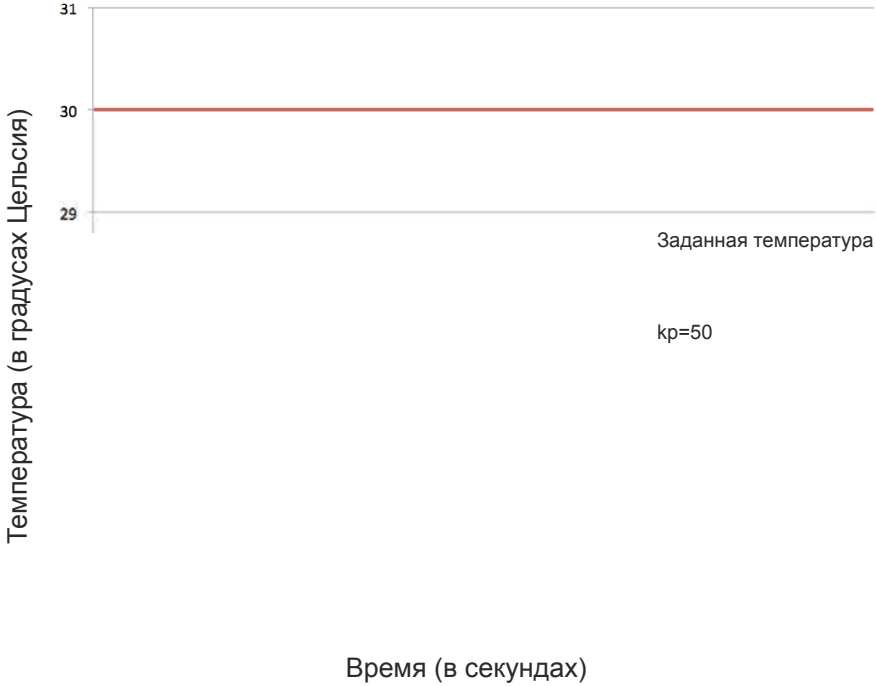


Рис. 12.11. График изменения температуры с течением времени для kp = 50

Импорт данных в электронную таблицу

Помещение полученных из окна монитора порта данных в электронную таблицу с последующим построением графика с помощью имеющихся там средств позволяет увидеть, что происходит с данными, принося тем самым немало пользы.

Если вы пользуетесь программным пакетом OpenOffice, выделите данные из окна монитора порта и, чтобы скопировать данные в буфер обмена, нажмите в Windows/Linux комбинацию клавиш <Ctrl>+<C> (или <Command>+<C> в Mac OS). Перейдите в документ электронной таблицы OpenOffice, выберите ячейку, начиная с которой хотите поместить данные, и нажмите в Windows/Linux комбинацию клавиш <Ctrl>+<V> (или <Command>+<V> в Mac OS).

Поскольку данные представлены в нескольких столбцах, в OpenOffice откроется диалоговое окно (рис. 12.12), позволяющее отделить столбцы друг от друга.

Выберите вариант **Comma** (Запятая) в разделе **Separated by** (Разделитель) и щелкните на кнопке **OK**, после чего данные будут вставлены в отдельные столбцы электронной таблицы.

При использовании Microsoft Excel вам предварительно придется воспользоваться текстовым редактором — например, Notepad++ или Textmate. Сначала перенесите данные в новый текстовый документ, а затем сохраните их в файле с расширением

csv (comma separated values — значения, разделенные запятыми). После этого можно будет открыть этот файл непосредственно в Excel. Для импортирования файла можно также воспользоваться имеющейся в Excel командой импорта данных.

Рис. 12.12. Импорт данных в OpenOffice

Похоже, что при значении `kp`, равном 50, температура никогда не поднимется до 30 °С. Задайте температуру 0 °С (`t0` — в окне монитора порта) и подождите, пока система не остынет, после чего повторите процедуру сначала для `kp` со значением `500`, а потом со значением `5000`. Результаты, полученные при этих трех значениях `kp`, показаны на рис. 12.13.

Как мы и предполагали, для `kp` значение 50 слишком мало, 500 уже больше подходит, но при нем значение заданной температуры все еще не достигается, а при значении 5000 система ведет себя как термостат, работающий по принципу включение/выключение. Можно выдвинуть предположение, что для `kp` вполне подойдет значение 700, особенно, если система получит ускорение в подъеме температуры за счет добавления значения для I-составляющей.

В методе настройки ПИ-контроллера от Циглера-Никольса предлагается вычислять значение `ki` по следующей формуле:

$$ki = (1{,}2 \times kp) / pu$$

Мы прикинули, что для `kp` вполне подойдет значение 700, а из рис. 12.13 следует, что значение `pu` составляет приблизительно 15 секунд, из чего для `ki` предлагается значение 56.

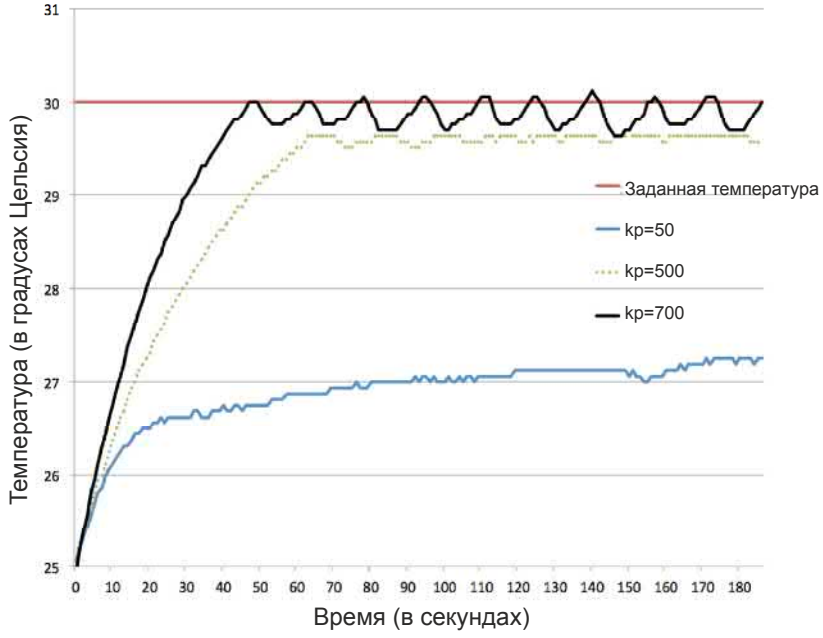


Рис. 12.13. График изменения температуры с течением времени для трех значений `kp`

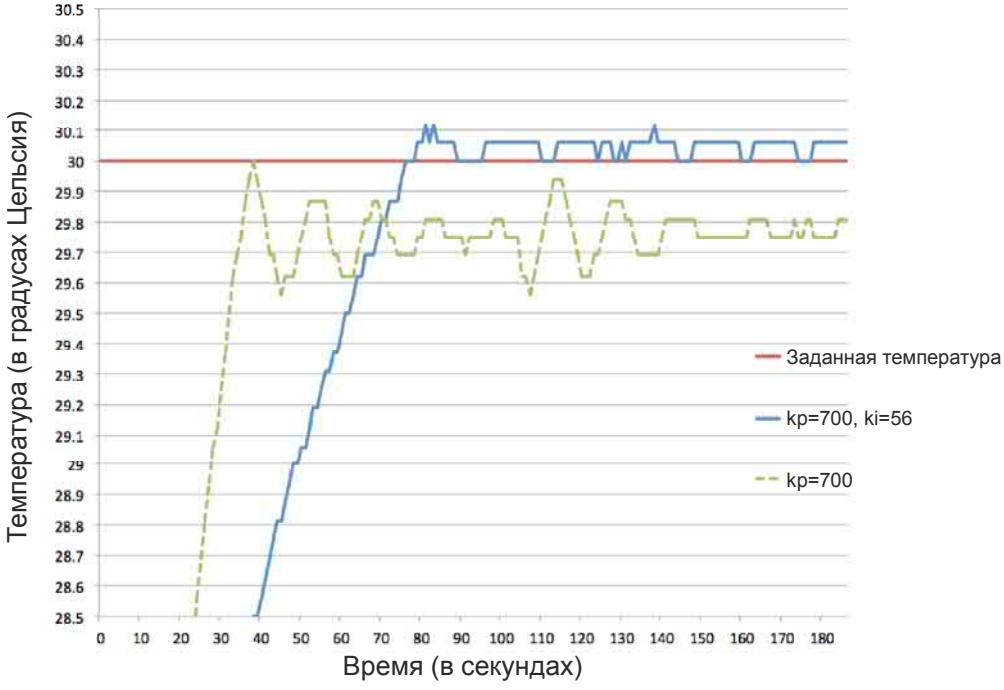


Рис. 12.14. График изменения температуры с течением времени для П- и ПИ-регулирования

Задайте Arduino еще один набор данных: с `kp = 700` и `ki = 56`. Результаты, полученные на его основе, показаны на рис. 12.14, где они даны в сравнении с результатами для чисто пропорционального регулирования при значении `kp` равном `700`. Ось Y здесь несколько растянута, чтобы можно было увидеть, насколько хороши результаты для ПИ-регулирования.

Как только кривая ПИ-графика достигнет заданной температуры, она будет изменяться на величину, слегка превышающую 0,1 градуса. Температура изменяется ступенчато, а не плавно, поскольку микросхема DS18B20 является цифровым устройством с фиксированным шагом.

Потратив дополнительные усилия на проведение экспериментов, можно, наверное, еще улучшить результат, но вряд ли этим стоит заниматься.

Экспериментируем с Raspberry Pi

Подключение Raspberry Pi

Схема, собранная на макетной плате для Raspberry Pi (рис. 12.15), немного отличается от того, что было собрано для Arduino. Разница в том, что нам нужно, чтобы резисторный нагреватель по-прежнему работал от 5 В, а датчику температуры DS18B20, чтобы быть совместимым с GPIO-контактами Raspberry Pi, нужно работать от 3,3 В.

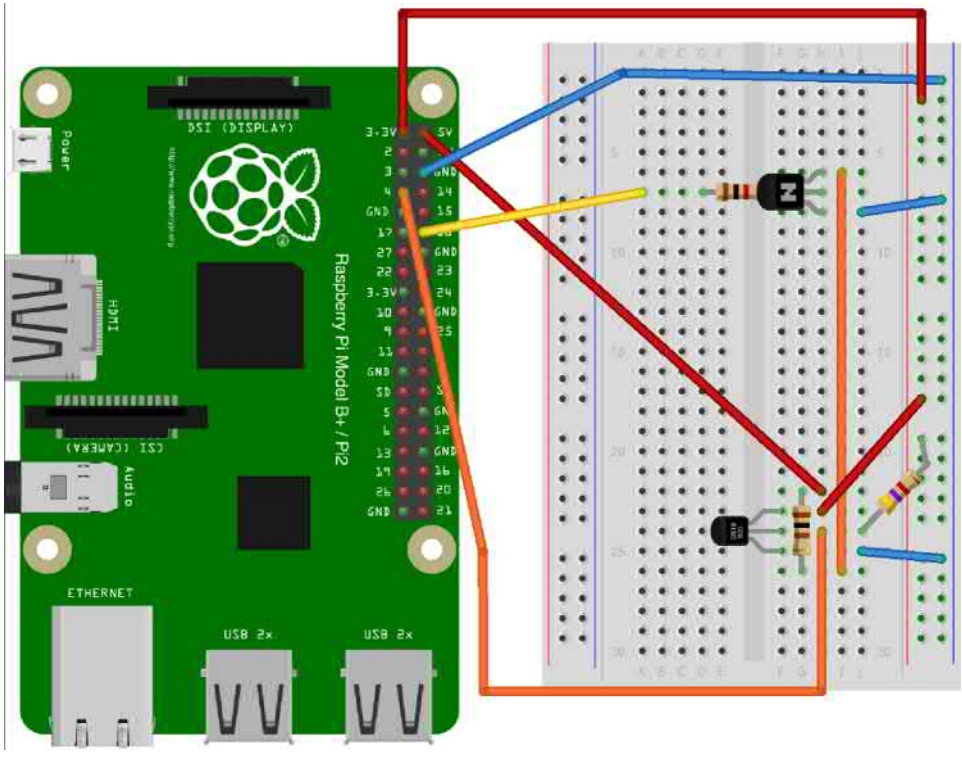


Рис. 12.15. Компоновка макетной платы для подключения к Raspberry Pi

Программа для Raspberry Pi

Чтобы микросхема DS18B20 работала с Raspberry Pi, требуется небольшая подготовка. Сначала нужно включить шину 1-Wire. Для этого отредактируйте файл /boot/config.txt, воспользовавшись следующей командой:

```
$ sudo nano /boot/config.txt
```

Добавьте к концу файла следующую строку:

```
dtoverlay=w1-gpio
```

Теперь нужно перезапустить Raspberry Pi, чтобы изменения вступили в силу. В микросхеме DS18B20 используется интерфейс в стиле текстового файла, значит, программе на языке Python придется считывать файл, а затем извлекать из него температурные измерения. Можно попытаться сделать это до запуска всей программы и посмотреть, как выглядит формат сообщения, изменив с помощью следующей команды текущий каталог на /sys/bus/w1/devices:

```
$ cd /sys/bus/w1/devices
```

Затем нужно получить список каталогов, находящихся в этой папке, воспользовавшись следующей командой:

```
$ ls
28-000002ecba60 w1 bus master1
pi@raspberrypi /sys/bus/w1/devices $
```

Сделайте текущим каталог, чье имя начинается с `28`. В нашем случае это каталог 28-000002ecba60 (учтите, что у вас, скорее всего, он будет иметь другое имя):

```
$ cd 28-000002ecba60
```

И, наконец, запустите следующую команду для извлечения последних считанных значений температуры:

```
$ cat w1_slave
```

Ответ приходит в двух строках:

```
53 01 4b 46 7f ff 0d 10 e9 : crc=e9 YES
53 01 4b 46 7f ff 0d 10 e9 t=21187
pi@raspberrypi /sys/bus/w1/devices/28-000002ecba60 $
```

Первая часть каждой строки представляет собой уникальный идентификатор для датчика температуры, а первая строка заканчивается словом `YES`, показывающим, что чтение прошло успешно. Вторая же строка заканчивается показанием температуры, выраженным в тысячных долях градуса Цельсия. В данном случае это 21 187 (или 21,187 °C).

Хотя для ПИД-регулирования имеются доступные библиотеки на языке Python, использовать их сложнее, чем их двойников для Arduino, и поэтому для версии Raspberry Pi ПИД-алгоритм будет реализован с чистого листа (хотя и не совсем — код был написан с оглядкой на библиотеку Arduino и с попыткой сделать две версии как можно более похожими друг на друга).

Код этот можно найти в файле pid_thermostat.py каталога python/experiments/:

```
import os
import glob
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

heat_pin = 18
base_dir = '/sys/bus/w1/devices/' // 1
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

GPIO.setup(heat_pin, GPIO.OUT)
heat_pwm = GPIO.PWM(heat_pin, 500)
heat_pwm.start(0)

old_error = 0 // 2
old time = 0
measured_temp = 0
p_term = 0
i term   0
d term = 0

def read_temp_raw(): // 3
   f = open(device_file, 'r')
   lines = f.readlines()
   f.close()
   return lines

def read_temp(): // 4
   lines = read_temp_raw()
   while lines[0].strip()[-3:] != 'YES':
      time.sleep(0.2)
      lines = read_temp_raw()
   equals_pos = lines[1].find('t=')
   if equals_pos != -1:
      temp_string = lines[1][equals_pos+2:]
      temp_c = float(temp_string) / 1000.0
      return temp_c

def constrain(value, min, max): // 5
   if value < min
      return min
   if value > max
      return max
```

```
   else:
      return value

def update_pid(): // 6
   global old_time, old_error, measured_temp, set_temp
   global p_term, i_term, d_term
   now = time.time()
   dt = now - old_time // 7
   error = set_temp - measured_temp # // 8
   de = error - old_error # // 9

   p_term = kp * error // 10
   i_term += ki * error // 11
   i_term = constrain(i_term, 0, 100) // 12
   d_term = (de / dt) * kd // 13

   old_error = error
   # print((measured_temp, p_term, i_term, d_term))
   output = p_term + i_term + d_term // 14
   output = constrain(output, 0, 100)
   return output

set_temp = input('Введите заданную температуру в градусах Цельсия ') # // 15
kp   input('kp: ')
ki = input('ki: ')
kd   input('kd: ')

old_time = time.time() // 16
try:
   while True:
      now = time.time()
      if now > old_time + 1 : // 17
         old_time = now
         measured_temp = read_temp()
         duty = update_pid()
         heat_pwm.ChangeDutyCycle(duty)

         print(str(measured_temp) + ', ' + str(set_temp) + ', ' + str(duty))

finally:
   GPIO.cleanup()
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Этот код определяет каталог, в котором находится файл для DS18B20. Это делается практически так же, как и в ранее рассмотренном методе с использованием команды `glob` для нахождения первого каталога, начинающегося с `28`.

2. Эти глобальные переменные используются ПИД-алгоритмом. Переменная `old_error` служит для вычисления изменения рассогласования для D-составляющей.
3. Функция `read_temp_raw` считывает в виде двух строк текста показания микросхемы DS18B20.
4. Функция `read_temp` отвечает за фактическое извлечение показателя температуры с конца второй строки — после проверки, что в первой строке получен ответ `YES`.
5. Эта вспомогательная функция ограничивает значение первого параметра, чтобы оно всегда находилось внутри диапазона, указанного вторым и третьим параметрами.
6. Функция `update_pid` содержит код ПИД-вычислений.
7. Вычисление `dt` (сколько времени прошло с последнего вызова функции `update_pid`).
8. Вычисление рассогласования.
9. Вычисление изменения в рассогласовании `de`.
10. Вычисление пропорциональной составляющей.
11. Добавление к `i_term` текущего значения `error * ki`.
12. Ограничение интервала значений `i_term` тем же диапазоном, что и на выходе (от 0 до 100).
13. Вычисление `d_term`.
14. Сложение всех составляющих и ограничение интервала значений в диапазоне выходных значений от 0 до 100.
15. В отличие от версии Arduino, позволяющей корректировать настроечные переменные при работе контроллера, программа на языке Python делает однократный запрос на ввод температуры, `kp`, `ki` и `kd`.
16. Переменная `old_time` инициализируется текущим временем непосредственно перед началом основного управляющего цикла.
17. Если со времени предыдущего замера прошла 1 секунда, производится измерение температуры, а затем получение нового значения на выходе (`duty`) и соответствующее изменение коэффициента заполнения ШИМ-канала.

Загружаем и выполняем программу

Одно из отличий версии программы для Arduino от версии программы для Raspberry Pi заключается в том, что у Raspberry Pi выход в диапазоне от 0 до 100, а у Arduino — от 0 до 255. Поэтому параметры `kp` и `ki`, найденные при настройке Arduino, нуждаются в корректировке под Raspberry Pi. По сути, чтобы подогнать выход под интервал от 0 до 100, можно просто разделить значения `kp` и `ki` на 2,5. Это приведет к тому, что у `kp` будет значение 280, а у `ki` — 22.

Запустите программу, задайте температуру 30, подключите эти числа, и в результате должны быть получены данные, сходные с теми, что были при использовании версии под Arduino:

```
$ sudo python ex_11_pid_thermostat.py
Введите заданную температуру в градусах Цельсия 30
kp: 280
ki: 22
kd: 0
23.437, 30, 100
23.437, 30, 100
23.5, 30, 100
23.562, 30, 100
23.687, 30, 100
```

Построив по этим показателям с помощью электронной таблицы график, я получил результаты, показанные на рис. 12.16. Здесь также используется растянутое представление температуры, которая регулируется довольно точно.

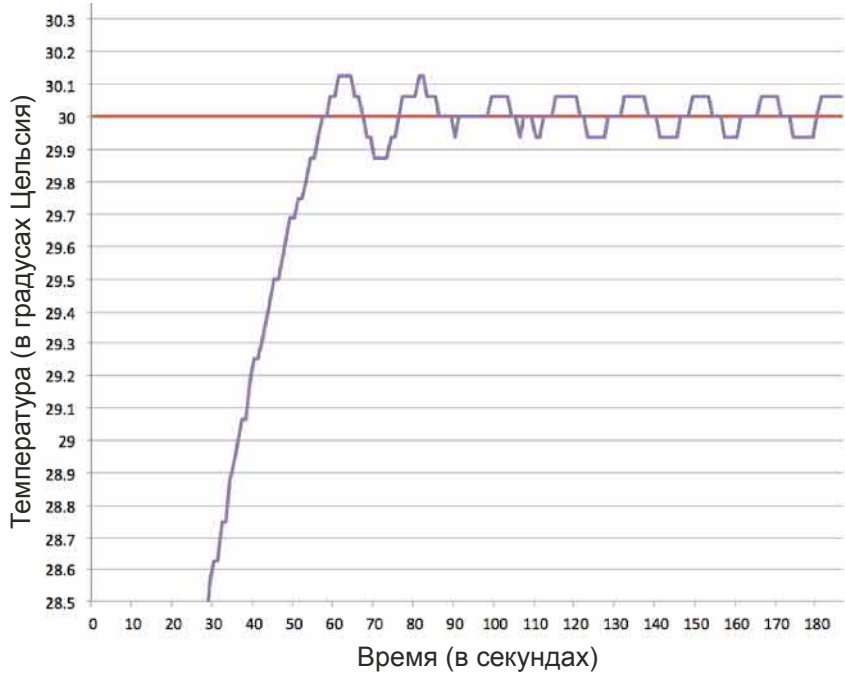


Рис. 12.16. Результаты ПИД-регулирования с использованием Raspberry Pi

Проект: термостатический охладитель напитков

В этом проекте мы добавим термостатическое регулирование к проекту из *разд. «Проект: охладитель напитков» главы 11*, чтобы напитки могли более точно охлаждаться до нужной температуры (рис. 12.17). В *главе 14* проект получит свое

дальнейшее развитие — к нему будет добавлен дисплей, показывающий заданную и фактическую температуры.

Как уже было сказано, этот проект реализуется с использованием Arduino, но учитывая, что вы уже научились использовать Raspberry Pi совместно с микросхемой DS18B20, у вас не должно возникнуть проблем и по изменению проекта под работу с Raspberry Pi.

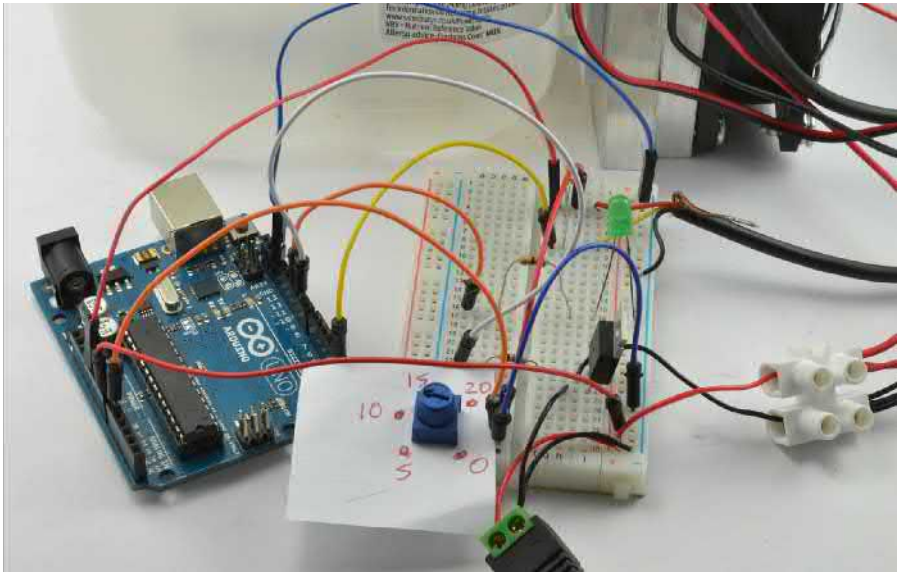


Рис. 12.17. Термостатический охладитель напитков в сборе

Оборудование

Этот проект основывается на проекте из *разд. «Проект: охладитель напитков» главы 11*, но с добавлением в рабочий цикл платы Arduino и датчика температуры DS18B20, поэтому, если вы еще не проработали тот проект, следуйте инструкциям по его реализации, имеющимся в *главе 11*.

Комплектующие

В этом проекте для работы с Arduino понадобятся следующие комплектующие (табл. 12.2).

Таблица 12.2. *Комплектующие для работы с Arduino в проекте по управлению термостатическим охладителем напитков*

Обозначение	Компонент схемы	Источники
R1	Резистор 4,7 кОм	Mouser: 291-4.7k-RC
R2	Резистор 1 кОм	Mouser. 291-1k-RC
R3	Резистор 270 Ом	Mouser 291-270-RC

Таблица 12.2 *(окончание)*

Обозначение	Компонент схемы	Источники
	[redacted]	
	Большая емкость из-под молока или сока	Б/у

Герметичный температурный датчик DS18B20 содержит точно такую же микросхему, что использовалась в экспериментах из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»* и из *разд. «Эксперимент: термостатический ПИД-регулятор»*, за исключением того, что он поставляется в удобной водонепроницаемой капсуле с длинными проводами, которые могут быть подключены непосредственно к макетной плате.

Если потребуется задействовать более мощный элемент Пельтье, чем тот, что указан в табл. 12.2, следует воспользоваться более мощным блоком питания, чтобы его максимально допустимый ток наверняка превышал ток, потребляемый элементом. Предусмотрите, как минимум, превышение в половину ампера для вентиляторов и еще в половину ампера на всякий случай.

Схема проекта

Принципиальная схема этого проекта изображена на рис. 12.18. В левой части схемы показан подстроечный резистор R4, который также называют *потенциометром* (см. далее врезку *«Потенциометры»*). Подвижный контакт потенциометра подключен к контакту A0, представляющему собой аналоговый вход Arduino (см. *разд. «Аналоговые входы» главы 2*). Положением ручки потенциометра на контакте A0 устанавливается напряжение, которое замеряется Arduino, а затем используется для установки нужной температуры охладителя.

Потенциометры

Компонент под названием *потенциометр* должен быть вам известен по регуляторам громкости радиоприемника или усилителя. У него есть ручка, вращающаяся почти на полный оборот.

Область на рис. 12.18 вокруг R4 показывает, как потенциометр используется в качестве устройства ввода данных в Arduino: верхний контакт потенциометра подключен к линии 5 В, а нижний — к заземлению, при этом на среднем контакте потенциометра напряжение в зависимости от положения ручки будет изменяться в диапазоне от 0 до 5 В.

Правая часть схемы на рис. 12.18 очень похожа на схему эксперимента из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»*, за исключением того, что вместо маломощного транзистора MPSA14 используется мощный МОП-транзистор FQP30N06L. Этот транзистор способен коммутировать подводимый к охладителю ток силой 4 А и более, при этом степень его нагрева позволяет обойтись без радиатора.

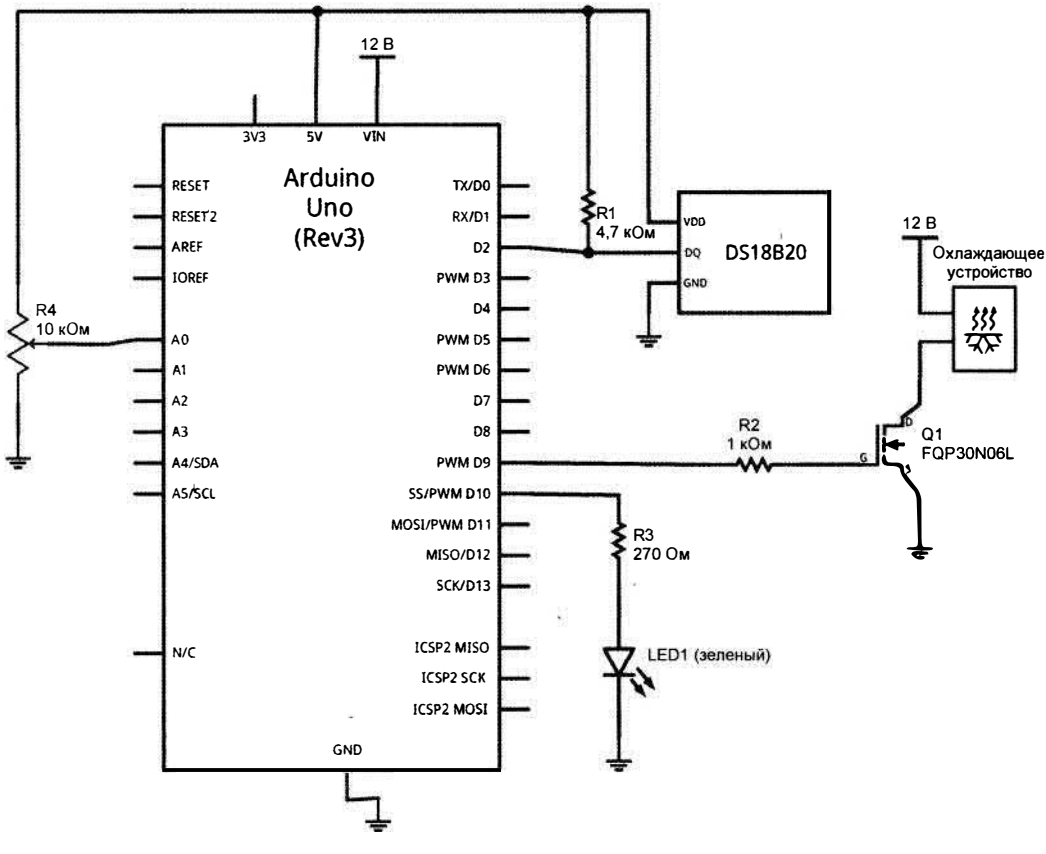


Рис. 12.18. Принципиальная схема термостатического охладителя

Сборка проекта

Предполагая, что вы уже реализовали проект из *разд. «Проект: охладитель напитков» главы 11*, для реализации этого проекта нужно выполнить лишь следующие дополнительные действия.

Шаг 1. Добавление температурного датчика

Физическая конструкция охладителя остается точно такой же, что и в проекте из *разд. «Проект: охладитель напитков» главы 11*, но теперь к ней добавляется температурный датчик, который следует разместить на дне контейнера — под ставящиеся на него охлаждаемые стаканы или бутылки (рис. 12.19). В данном случае я просто прикрепил датчик ко дну контейнера клейкой лентой, но лучше его все же основательно приклеить.

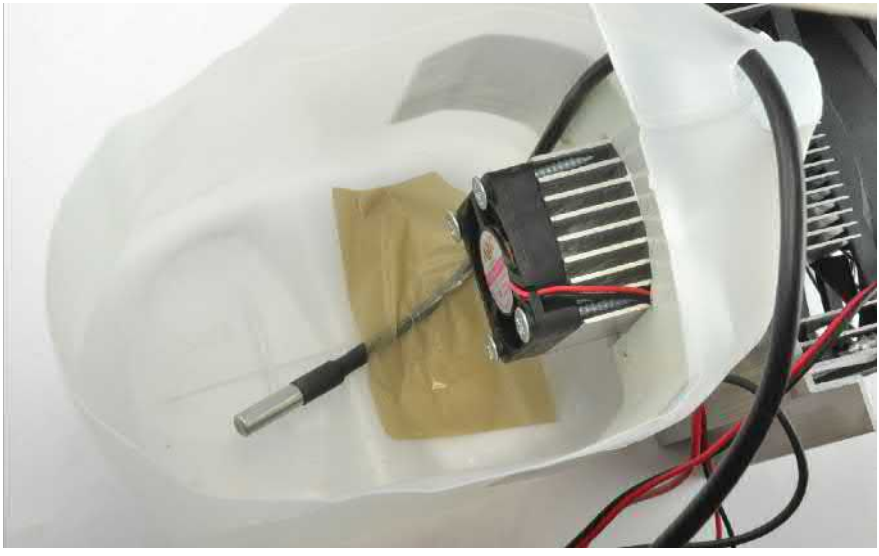


Рис. 12.19. Добавление температурного датчика

Шаг 2. Сборка схемы на макетной плате

На рис. 12.20 изображена собранная на макетной плате схема, используемая для проекта, а также показаны соединения различных деталей проекта.

Расположите компоненты на макетной плате, убедившись в правильном подключении МОП-транзистора и светодиода. Температурный датчик имеет в своем кабеле четыре провода. Провода с красной и черной изоляцией подключаются соответственно к общей линии питания (VCC) и к заземлению (GND), а провод с желтой изоляцией является цифровым выходом зонда. Четвертый провод никуда подключать не нужно.

На ручку потенциометра я наколол небольшой клочок бумаги, соорудив из него примитивную шкалу с рисками, позволяющими увидеть задаваемую температуру охладителя.

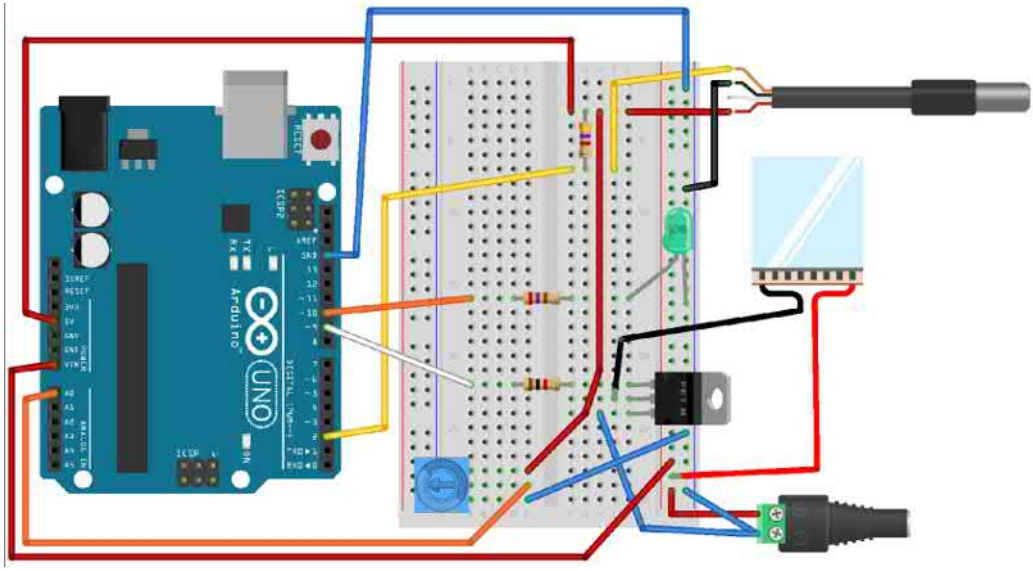


Рис. 12.20. Схема проекта, собранная на макетной плате

Шаг 3. Подключение охладителя

У охладителя имеются три пары проводов: две — для вентиляторов и одна — для самого элемента Пельтье. Чтобы упростить подключение охладителя, используется двунаправленная клеммная колодка, позволяющая подключить охладитель к макетной плате всего двумя проводами (рис. 12.21).

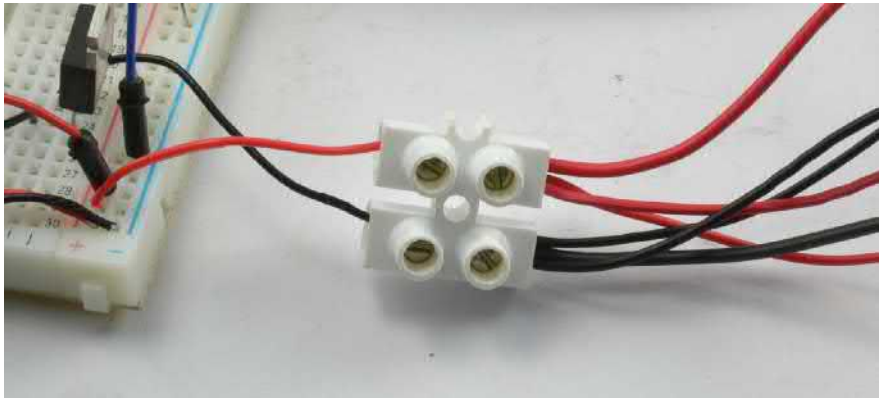


Рис. 12.21. Подключение охладителя

Шаг 4. Подключение блока питания

Переходник с круглым гнездом и винтовыми зажимами, конечно, может быть подключен к макетной плате перемычками «папа-папа». С этим вариантом можно согласиться в случае использования высококачественных проводов-перемычек относительно большого поперечного сечения, однако многие соединительные провода

содержат слишком тонкие проводники, которые могут сильно нагреваться от проходящего по ним тока силой в несколько ампер. Само по себе это не создает проблем до тех пор, пока эти провода не станут сильно греться, вместо того, чтобы быть просто теплыми. Но это означает, что из 12 В на элемент Пельтье станет попадать не все напряжение, и на то, чтобы войти в рабочий режим, охладителю потребуется больше времени.

Для подключения макетной платы к переходнику питания можно вместо простых соединительных проводов-перемычек воспользоваться каким-нибудь одножильным изолированным проводом большого сечения. То же самое касается и соединительных проводов, подводимых к охладителю.

Программа для Arduino

Использование ПИД-регулятора для охладителя напитков можно посчитать излишеством. Но вопрос в данном случае упирается только в программу, поэтому на использование «крутого» алгоритма поддержки напитков в охлажденном состоянии дополнительных затрат не предвидится.

Программа этого проекта во многом похожа на ту, что использовалась в экспериментах из *разд. «Эксперимент: насколько хорош терморегулятор, основанный на включении и выключении?»* и из *разд. «Эксперимент: термостатический ПИД-регулятор»*, включая весь код для создания интерфейса с температурным датчиком DS18B20, поэтому, чтобы разобраться в этом коде, следует вернуться к описанию упомянутых экспериментов:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <PID_v1.h>

const double minTemp = 0.0; // 1
const double maxTemp = 20.0;
const float tempOKMargin = 0.5;

double kp = 1500;
double ki = 50.0;
double kd = 0.0;

const int tempPin = 2;
const int coolPin = 9;
const int ledPin = 10; // 3
const int potPin = A0;
const long period = 1000; // >750

OneWire oneWire(tempPin);
DallasTemperature sensors(&oneWire);

double setTemp = 0.0;
double measuredTemp = 0.0;
```

```
double outputPower = 0.0;
long lastSampleTime = 0;

PID myPID(&measuredTemp, &outputPower,
      &setTemp, kp, ki, kd, REVERSE); // 4

void setup() {
   pinMode(coolPin, OUTPUT);
   pinMode(ledPin, OUTPUT);
   Serial.begin(9600);
   sensors.begin();
   myPID.SetSampleTime(1000);
   myPID.SetMode(AUTOMATIC);
}

void loop() {` // 5
   long now = millis();
   if (now > lastSampleTime + period) {
      checkTemperature();
      lastSampleTime = now;
   }
   setTemp = readSetTempFromPot(); // 6
}

void checkTemperature() { // 7
   measuredTemp = readTemp();
   Serial.print(measuredTemp);
   Serial.print(", ");
   Serial.print(setTemp);
   Serial.print(", ");
   Serial.println(outputPower);
   myPID.Compute();
   analogWrite(coolPin, outputPower);
   float error = setTemp - measuredTemp; // 8
   if (abs(error) < tempOKMargin) {
      digitalWrite(ledPin, HIGH);
   }
   else {
      digitalWrite(ledPin, LOW);
   }
}

double readSetTempFromPot() { // 9
   int raw = analogRead(potPin);
   double temp = map(raw, 0, 1023, minTemp, maxTemp);
   return temp;
}
```

```
double readTemp() {
   sensors.requestTemperatures();
   return sensors.getTempCByIndex(0);
}
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Диапазон температур, задаваемых с помощью потенциометра, устанавливается двумя константами: `minTemp` и `maxTemp`. Переменная `tempOKMargin` определяет значение выше или ниже заданной температуры, которое может иметь фактическая температура до того, как погаснет зеленый светодиод.
2. Для `kp` установлено довольно высокое значение, чтобы включение и выключение охладителя происходило более четко. Сделано это в основном с тем, чтобы избавиться от унылого звучания моторов вентиляторов, когда они питаются в режиме низкого уровня выходной мощности. Вместо этого вентиляторы можно запитать отдельно, чтобы они постоянно работали, и заниматься только регулировкой мощности на элементе Пельтье.
3. Определение дополнительных контактов для светодиода и потенциометра.
4. ПИД-регулятор инициализируется в режиме `REVERSE`, а не `DIRECT` (как ранее), поскольку добавление выходной мощности будет снижать, а не повышать температуру.
5. В основном цикле проверяется истечение секундного интервала, после чего для включения и выключения охладителя по мере надобности вызывается функция `checkTemperature`.
6. При каждом прохождении цикла (которое должно осуществляться несколько раз в секунду), для установки значения переменной `setTemp` на основании позиции ручки потенциометра вызывается функция `readSetTempFromPot`.
7. Функция `checkTemperature` производит измерение температуры, считывает полученные данные, а затем производит обновление ПИД-контроллера. Эта функция также записывает прочитанные данные в окно монитора порта, позволяя настроить охладитель или отследить его работу. Arduino не нуждается в подключении через порт USB, поскольку получает электропитание через свой контакт **Vin**, но если его подключить через порт USB, выводимые данные можно будет увидеть на экране в окне монитора порта.
8. Остальная часть этой функции включает светодиод, если измеренная температура находится в пределах допустимого отклонения от заданной температуры, определяемого с помощью константы `tempOKMargin`. Функция `abs` (абсолютное значение) удаляет знак минуса перед числом.
9. Код превращения позиции потенциометра в значение между `minTemp` и `maxTemp`. Необработанное аналоговое считывание (значения в диапазоне от 0 до 1023) производится в переменную `raw`. Затем для преобразования считанного значения в желаемый диапазон температур вызывается функция `map` (см. далее врезку *«Функция `map`, используемая в Arduino»*).

Функция map, используемая в Arduino

При управлении какими-либо устройствами с помощью Arduino или Raspberry Pi часто возникает проблема преобразования числа, имеющего один диапазон значений, в число в каком-нибудь другом диапазоне значений.

Например, на аналоговом входе Arduino установлен диапазон значений от 0 до 1023, и если нужно отобразить этот диапазон на температуру, например, между 0 и 20, можно просто разделить число на 51,15 (т. е. на 1023 / 20). Тогда 1023 превратится в 1023 / 51,15 = 20.

Задача усложняется, если оба диапазона начинаются не с нуля. И тут может пригодиться имеющаяся в Arduino функция `map`. Как далее показано, она получает пять параметров, которые преобразуют число в диапазоне от 0 до 1023 в число в диапазоне от 20 до 40:

```
map(value, 0, 1023, 20, 40);
```

Первый параметр представлен здесь значением, которое нужно преобразовать, второй и третий параметры задают диапазон имеющегося значения, а четвертый и пятый — диапазон, в котором нужно получить соответствующее значение (в данном случае, это диапазон от 20 до 40).

В языке Python отсутствует встроенная функция диапазона, но ее довольно просто создать, а затем использовать в своей программе. Она должна выглядеть примерно так:

```
def map(value, from_low, from_high,
to_low, to_high):
       from_range = from_high -

from_low
   to_range = to_high - to_low
   scale_factor = from_range / to_range
   return to_low + (value /
scale_factor)
```

Затем эту функцию на языке Python можно будет вызвать с такими же параметрами, что и у ее двойника в Arduino. Например:

```
map(510, 0, 1023, 20, 40)
```

В результате будет возвращено значение 30, которое является средним значением для диапазона от 20 до 40, точно так же, как и значение 510, которое расположено примерно посредине между значениями в диапазоне от 0 до 1023.

Заключение

Хотя для иллюстрации принципов точного управления чем-либо при наличии возможности проведения измерений и внесения корректив была использована температура, те же самые принципы вполне применимы и к управлению другими параметрами — например, положением исполнительного механизма. Именно так и работает серводвигатель, рассмотренный в *главе 9*.

В следующей главе мы обсудим способы безопасного управления высоковольтными устройствами с использованием Raspberry Pi или Arduino.

Управление устройствами переменного тока

13

Коммутация устройств переменного тока требует четкого понимания возникающих при этом опасностей и принятия специальных мер, направленных на обеспечение безопасности при работе с высоким напряжением. В этой главе мы рассмотрим способы безопасного управления устройствами переменного тока с использованием электромеханических и твердотельные реле, а также таких технических приемов, как коммутация при пересечении синусоидой переменного тока нулевого значения (zero-crossing switching).

> ***ВЫСОКОЕ НАПРЯЖЕНИЕ ОПАСНО ДЛЯ ЖИЗНИ!***
>
> Каждый год от бытовых электроприборов только в Соединенных Штатах гибнут сотни людей. Неисправности электропроводки приводят к многочисленным возгораниям и пожарам в жилых домах, при которых люди часто страдают от ожогов. В бытовой электросети используется высокое напряжение переменного тока, и она рассчитана на выдачу больших мощностей.
>
> При проработке практических разделов этой главы **никогда** ничего не делайте под напряжением. Лично я, когда работаю над схемой, предпочитаю видеть выдернутую из стенной розетки вилку перед собой, чтобы быть уверенным, что она в любом случае ни в какую розетку не включена.
>
> При работе над проектом следует пользоваться розеткой переменного тока, подключенной к автомату защиты от коротких замыканий, и никогда не оставлять оголенные провода или печатные платы любого проекта, управляющие цепями питания переменным током, вне изолированного корпуса и не закрепленными в нем.
>
> И не пытайтесь использовать в конструкциях переменного тока макетные платы — они не рассчитаны на такое напряжение и силу тока.
>
> Короче говоря, если вы не прошли обучение по работе с бытовой электросетью переменного тока, используйте готовые к использованию модули типа PowerSwitch Tail (см. далее).

Теоретические основы коммутации цепей переменного тока

В этом разделе мы рассмотрим некоторые теоретические основы коммутации цепей переменного тока и различные конструкции электрических схем и компонентов, используемых в этой области электроники. Практической работе по коммутации цепей переменного тока в этой главе также будет посвящен отдельный раздел.

Что такое переменный ток?

В отличие от *постоянного тока* (DC), который всегда течет в одном направлении, *переменный ток* (AC) не сохраняет такое постоянство. Направление его течения в некоторых странах (включая Соединенные Штаты) меняется на противоположное 120 раз в секунду и 100 раз в секунду во всем остальном мире. Такое изменение направления тока сопровождается соответствующим изменением напряжения на нагрузке (рис. 13.1). Каждый полный цикл состоит из двух перемен направления тока, поэтому частота переменного тока равна либо 60, либо 50 Гц (циклов в секунду). Герц — это единица измерения частоты, и 1 Гц равен одному циклу в секунду.

В Соединенных Штатах и еще в некоторых странах напряжение в сети переменного тока составляет 120 В, а в большей части остального мира в розетках переменного тока присутствует куда более опасное для жизни напряжение в 220–230 В.

Рис. 13.1. Напряжение переменного тока

На рис. 13.1 видно, что пик положительной и отрицательной фаз напряжения на самом деле существенно выше 120 В. Дело в том, что значение 120 В переменного тока фактически является некой усредненной величиной, представляющей собой эквивалент напряжению постоянного тока, способному поставлять нагрузке такой же объем мощности. Следовательно, постоянный ток напряжением 120 В заставит старомодную лампу накаливания светиться с той же яркостью, что и при ее питании от сети 120 В переменного тока.

Реле

Вы уже работали с *реле* при изучении *разд. «Управление двигателем постоянного тока при помощи реле» главы 7* — там реле использовалось для включения двига-

теля постоянного тока. Обмотка реле изолирована от той части, где находятся коммутирующие контакты, что играет весьма важную роль для безопасности коммутирования цепей переменного тока. Коммутационные возможности большинства релейных устройств указаны на их корпусах. Так, обычное реле типа «кубик сахара» может иметь маркировку с указанием возможности коммутирования переменного тока силой 10 А при напряжении 250 В и постоянного тока силой 10 А при напряжении 24 В.

На рис. 13.2 показано, как реле может использоваться для коммутации нагрузки переменного тока. Следует помнить, что для управления с цифрового выхода Arduino или Raspberry Pi обмотке реле требуется ток, существенно превышающий тот, который обеспечивается на этом выходе, поэтому в цепь управления реле включается небольшой транзистор Q1.

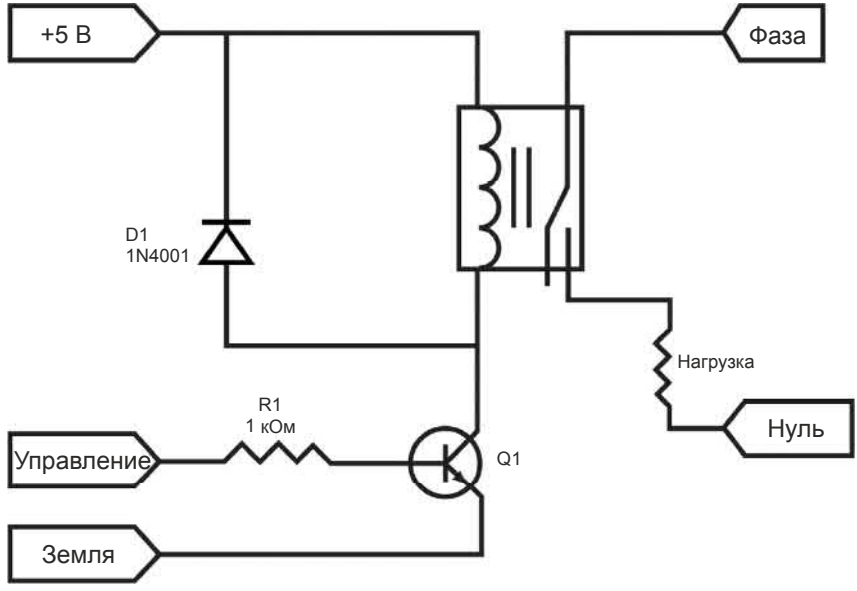


Рис. 13.2. Коммутация переменного тока с помощью реле

Линии управляющего сигнала, 5 В и заземления подключаются к платам Arduino или Raspberry Pi. Когда в линии управления выставляется высокий уровень напряжения, транзистор Q1 открывается, запитывая обмотку реле, которая замыкает его контакты, соединяющие провод фазы переменного тока с одним из концов нагрузки (коммутируемого устройства). Другой же конец нагрузки подключен к нулевому проводу переменного тока.

Оптрон

На смену стремительно устаревающим реле во многих конструкциях коммутационных устройств переменного тока приходят *твердотельные реле* (solid-state relays, SSR), рассматриваемые далее в *разд. «Твердотельные реле (SSR)»*. Одним из основных компонентов твердотельных реле является *оптрон* — построенный на

основе транзисторной технологии компонент, главная цель которого — отделение низковольтной цепи управления, используемой в проекте, от опасной высоковольтной цепи переменного тока нагрузки (рис. 13.3).

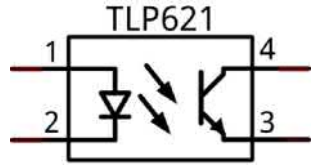


Рис. 13.3. Оптрон

Оптрон объединяет в едином пластмассовом корпусе светодиод и светочувствительный элемент (обычно это фототранзистор).

Ключевым моментом в конструкции оптрона является отсутствие электрического соединения между светодиодом и фототранзистором — связь между ними исключительно оптическая: проводящая цепь фототранзистора открывается, когда светодиод начинает излучать свет. Фототранзистор представляет собой устройство маломощное, и прежде чем управлять каким-либо агрегатом переменного тока, ему нужна дополнительная электронная обвязка.

Оптроны весьма чувствительные устройства, и их светодиодной цепью можно управлять непосредственно с вывода Arduino и даже с универсального контакта ввода/вывода (GPIO) Raspberry Pi, — при использовании развязывающего резистора в 1 кОм, ограничивающего ток до нескольких миллиампер.

Оптроны и симисторы с переключением при переходе нулевого значения

Оптроны, разработанные для коммутации цепей переменного тока, обладают рядом особенностей. Во-первых, на светочувствительной стороне у них стоит не обычный биполярный фототранзистор, а устройство, называемое *фотосимистором* (photo-TRIAC — от англ. TRIode for AC, триод для работы в цепях переменного тока). Внутреннее устройство фотосимистора (к примеру, типа MOC3031) показано на рис. 13.4, а паспорт этого изделия в PDF-формате можно загрузить по адресу **http://www.farnell.com/datasheets/1639837.pdf**.

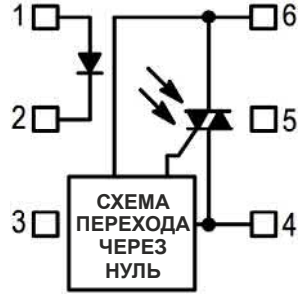


Рис. 13.4. Оптрон со схемой перехода через нуль

Симистор представляет собой специализированный тип транзистора, разработанный для коммутации тока, идущего в обоих направлениях, что, собственно, и нужно для управления цепями переменного тока.

Еще одна из особенностей симистора заключается в том, что, открыв цепь, он так и остается в открытом состоянии. И будет находиться в нем, пока проходящий через него ток не окажется сведенным практически к нулю. Это делает его совершенно бесполезным для управления устройствами постоянного тока, однако учитывая, что переменный ток меняет свою полярность 120 раз в секунду, у симистора появляется возможность те же 120 раз в секунду выключаться.

Преимущество фиксации симистора в открытом состоянии заключается в том, что он закрывается только тогда, когда проходящий через него ток (а следовательно, и напряжение на нем) падает, — тем самым снижается ток коммутации, который в иных случаях имел бы весьма большие значения. Такая коммутация тока — на его минимальных значениях — снижает еще и электрические помехи. «Мягкость» коммутации усиливается за счет использования *схемы перехода через нуль*, включаемой в некоторые оптроны. Эта схема откладывает включение симистора до момента перехода напряжения через нуль, гарантируя тем самым мягкость не только его выключения, но и включения.

Симистор, встроенный в оптрон типа MOC3031 — устройство слаботочное, и предназначен исключительно для использования в схемах управления более мощным симистором, который и занимается фактической коммутацией цепей переменного тока.

Типовая электрическая схема использования оптрона со схемой переключения при переходе через нуль для управления симистором высокой мощности показана на рис. 13.5.

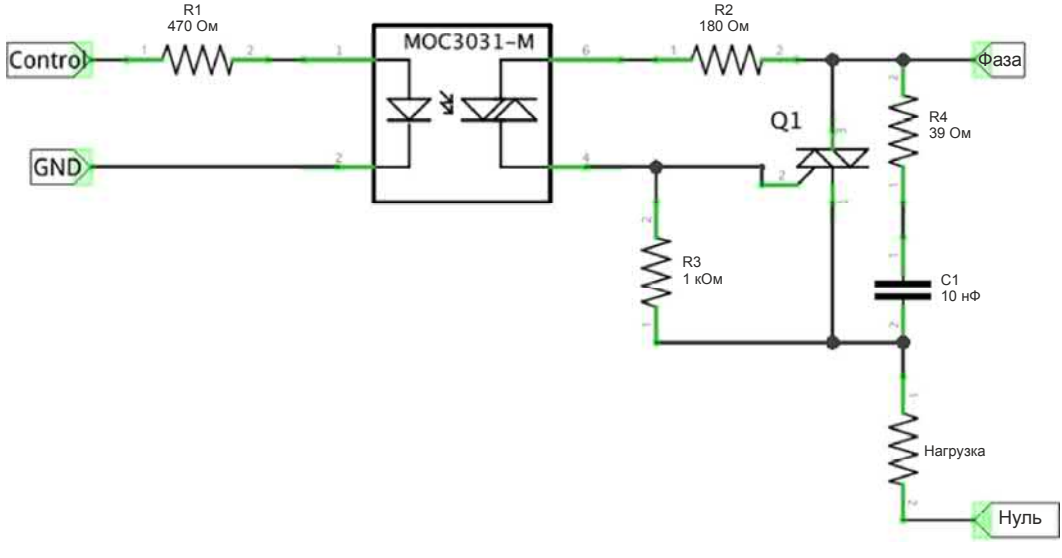


Рис. 13.5. Коммутация цепей переменного тока с помощью оптрона, имеющего схему переключения при переходе через нуль

Управление высокой нагрузкой с Arduino или Raspberry Pi в таких цепях превращается в выдачу всего лишь одного-двух миллиампер на светодиод, находящийся внутри оптрона.

Резисторы R2 и R3 ограничивают силу тока, протекающего через маломощный фотосимистор, находящийся в оптроне, а резистор R4 и конденсатор C1 предназначены для «сглаживания» любых скачков напряжения, иногда возникающих в цепи, несмотря на мягкое переключение.

Практическая коммутация цепей переменного тока

Не пытайтесь собрать рассматриваемые в этой главе схемы на макетной печатной плате — это опасно. И если вы хотите коммутировать нагрузку переменного тока с помощью Arduino или Raspberry Pi, следуйте инструкциям этого раздела.

Релейные модули

В *разд. «Релейные модули» главы 7* мы уже рассматривали готовые к использованию релейные модули. При работе с цепями переменного тока они обладают большим преимуществом перед простыми реле типа «сахарный кубик», поскольку коммутируемые устройства переменного тока можно подключить к ним, используя имеющиеся на модулях зажимные винтами силовые клеммы. Тем самым релейный модуль позволит изолировать низковольтную сторону создаваемой вами схемы от ее опасной высоковольтной части.

Не следует забывать, что металлические детали конструкции реле могут повредиться при соприкосновении с обмоткой реле контактов, подключенных к фазе, — при сильных ударах или случайных повреждениях. Поэтому в реле часто имеется дополнительный уровень безопасности, обеспечиваемый применением оптрона.

Безопасное использование релейного модуля

Надо постоянно иметь в виду, что у релейного модуля, подобного тому, что изображен на рис. 13.6, на верхней и нижней сторонах платы имеются оголенные металлические контакты и паяные соединения, подключенные к линии фазы. Прикосновение к ним может иметь для вас роковые последствия, поэтому релейный модуль, а также любые другие части создаваемой конструкции **всегда** должны размещаться в пластмассовом корпусе, чтобы ни вы, ни кто-либо другой не могли ненароком к чему-нибудь прикоснуться. Части схемы внутри корпуса должны быть также надежно закреплены, чтобы ничего не болталось.

Уделите внимание и кабельным зажимам, закрепляющим провода в корпусе и не дающим им выдергиваться из зажимных клемм релейного модуля, создавая возможность коротких замыканий.

Не работайте с релейным модулем под током, не подсоединяйте подключенные к цепи переменного тока провода к его зажимным клеммам, и, как только провода будут смонтированы, тут же закройте корпус крышкой.

Остерегайтесь применения в цепях переменного тока дешевых релейных модулей, продаваемых невесть какими поставщиками, поскольку многие из таких модулей небезопасны.

Рис. 13.6. Релейный модуль в пластмассовом корпусе

Не думайте, что если на самом реле написано, что оно выдерживает ток 10 А при 25 В, то релейный модуль в целом предназначен для работы в таком режиме. Мелкие зажимные клеммы на многих дешевых модулях выдерживают ток силой не более 2 А. На некоторых дешевых платах контактные ламели высоковольтной части реле находятся слишком близко (иногда в одном-двух миллиметрах) от низковольтных цепей. При высоком переменном напряжении это опасно — такие реле должны использоваться только в низковольтных цепях постоянного тока небольшой силы. В самых лучших релейных модулях стоят оптроны, а с печатной платы вокруг клемм контактной группы реле для обеспечения максимальной изоляции удален проводящий слой.

Самым безопасным способом коммутации цепей переменного тока является использование готового закрытого модуля типа PowerSwitch Tail (см. далее *разд. «Модуль PowerSwitch Tail»*).

Нужно также проверить, на какой уровень сигнала срабатывает ваш релейный модуль — на низкий или на высокий. Если вам достался релейный модуль, срабатывающий на низкий уровень сигнала, то он активизируется, когда на цифровом выходе будет низкий уровень (low). Это означает, что, назначив вывод схемы в качестве цифрового выхода, вы также должны в следующей строке кода установить на этом выводе высокий уровень сигнала (high), иначе реле станет на короткий период времени включаться при каждом перезапуске Arduino:

```
pinMode(relayPin, OUTPUT);
digitalWrite(relayPin, HIGH);
```

При использовании с Raspberry Pi релейного модуля, срабатывающего на низкий уровень сигнала, для установки на выводе схемы высокого уровня (high) следует воспользоваться дополнительным параметром `initial`:

```
GPIO.setup(relay_pin, GPIO.OUT, initial=True)
```

Для релейных модулей с оптронами характерно наличие дополнительной удаляемой перемычки, позволяющей предоставить обмотке реле положительное питание, независимое от светодиодной цепи оптрона. Удаление этой перемычки обеспечивает получение дополнительного уровня изоляции, но и означает необходимость питания обмотки реле от отдельного источника.

Твердотельные реле (SSR)

Модуль *твердотельного реле* показан на рис. 13.7. Он представляет собой герметично закрытый блок, содержащий электронную схему, которая может быть очень похожа на ту, что показана на рис. 13.5.

Рис. 13.7. Твердотельное реле, предназначенное для работы в цепях переменного тока

Твердотельные реле — это общедоступные устройства, позволяющие легко и просто коммутировать цепи переменного тока. Но проблема с открытыми металлическими частями, находящимися под напряжением, все еще остается, поэтому это устройство тоже нужно помещать в изолирующий корпус.

Низковольтная сторона устройства может подключаться непосредственно к Raspberry Pi или Arduino, поскольку в ней имеется подходящий дополнительный резистор для светодиода.

Модуль PowerSwitch Tail

Устройство под названием PowerSwitch Tail (рис. 13.8) представляет собой заключенное в изолирующий корпус твердотельное реле, имеющее с одной стороны вилку, а с другой — розетку переменного тока.

В устройстве имеются зажимные клеммы, подключенные к светодиодной стороне оптрона (со встроенным дополнительным резистором), и небольшой светодиод

красного свечения, который также излучает свет, когда твердотельное реле активировано. Одно из этих весьма удобных устройств будет задействовано при реализации проекта из *разд. «Проект: реле времени на основе Raspberry Pi»*.

Рис. 13.8. Твердотельное реле PowerSwitch Tail

Проект: реле времени на основе Raspberry Pi

В этом проекте для управления питанием небольшого электроприбора используются Raspberry Pi и PowerSwitch Tail. В *главе 16* этот весьма простой проект будет усовершенствован путем добавления к нему веб-интерфейса, позволяющего включать и выключать электроприбор с помощью браузера (см. *разд. «Проект: веб-выключатель на основе Raspberry Pi» главы 16*).

Собрать проект совсем нетрудно — нам понадобится лишь отвертка для выкручивания и закручивания винтов зажимных клемм.

Комплектующие

В этом проекте для работы с Raspberry Pi понадобятся следующие комплектующие (табл. 13.1).

Таблица 13.1. *Комплектующие для работы с Raspberry Pi в проекте по управлению реле времени*

Компонент схемы	Источники
[redacted]ail	Adafruit: 268
Перемычки «мама-папа»	Adafruit: 826
Настольная лампа или другой небольшой электроприбор	

Схема проекта

Монтажная схема проекта показана на рис. 13.9.

Присмотритесь к маркировке PowerSwitch Tail — там написано, что на входе должен быть сигнал постоянного тока напряжением 3–12 В при силе тока 3–30 мА. Необходимый для входа ток будет варьироваться в зависимости от напряжения, поэтому наименьшая возможная в допустимом диапазоне сила тока соответствует напряжению на входе, равному 3 В. Фактически, при напряжении 3,3 В модуль PowerSwitch Tail потребляет около 6 мА, что вполне допустимо для одного универсального GPIO-контакта Raspberry Pi.

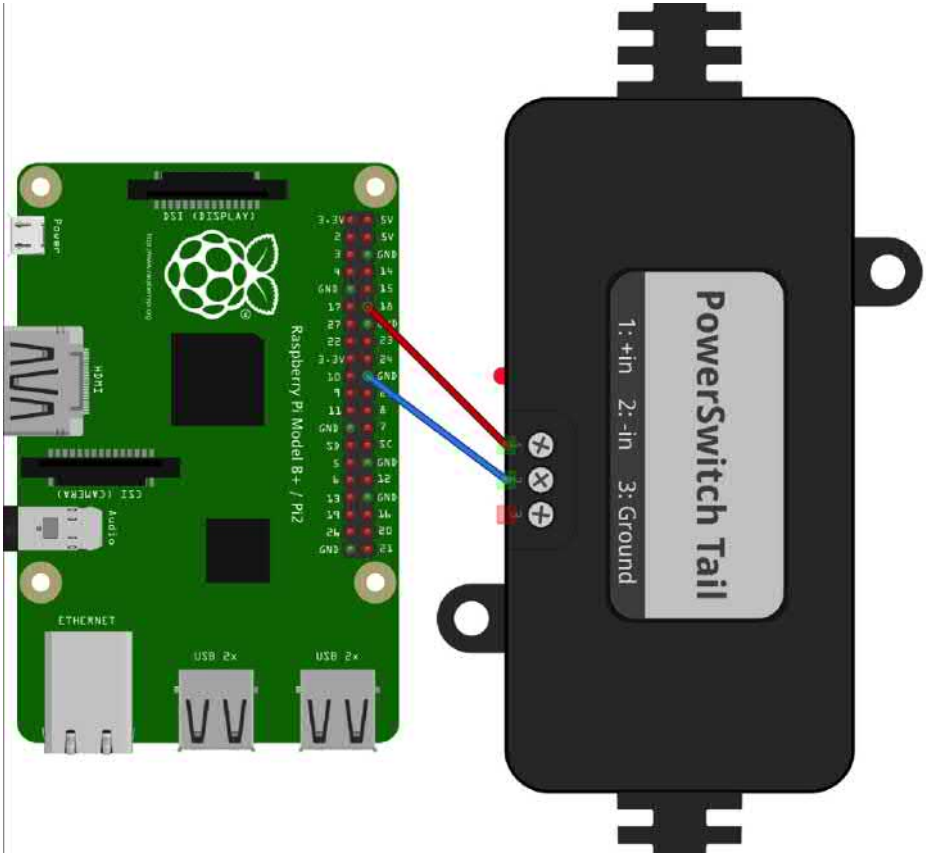


Рис. 13.9. Монтажная схема́ проекта реле времени

В ходе тестирования подключать PowerSwitch Tail к сети переменного тока нет необходимости, поскольку при включении твердотельного реле будет загораться имеющийся у него светодиод состояния.

Программа

Программа для этого проекта находится в файле /python/projects/ac_timer_switch.py (см. *разд. «Код к книге» главы 3*):

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

control_pin = 18

GPIO.setup(control_pin, GPIO.OUT)

try:
   while True: // 1
      duration_str = input("Точное время в минутах: ") # // 2
      duration = int(duration_str) * 60 // 3

      GPIO.output(control_pin, True) // 4
      time.sleep(duration)
      GPIO.output(control_pin, False) // 5

finally:
   print("Сброс")
   GPIO.cleanup()
```

Программа также предельно проста — она начинается с обычных инструкций импорта и определения констант. Уточним некоторые ее моменты по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Условие `True` для `While` является способом задания бесконечного цикла, поскольку условие `True` никогда не превратится в значение `False`. По этой причине цикл никогда не заканчивается (пока не будет нажата комбинация клавиш <Ctrl>+<C>).
2. В основном цикле вам предлагается ввести количество минут, в течение которых свет должен был включен.
3. Преобразование строкового значения продолжительности в целое число минут путем использования `int` с последующим умножением на 60 для преобразования этого значения в секунды.
4. Установка на GPIO-контакте `18` высокого уровня (`True`) для включения устройства PowerSwitch Tail, которое, в свою очередь, включает электроприбор, подключенный к его розетке переменного тока.
5. После соответствующей задержки на GPIO-контакте устанавливается низкий уровень сигнала (`LOW`), чтобы выключить твердотельное реле, после чего цикл запускается снова, запрашивая у вас точное время.

Загружаем и выполняем программу

Модуль PowerSwitch Tail способен коммутировать цепи переменного тока силой до 15 А, поэтому к нему можно подключать большинство электроприборов, за исключением очень мощных устройств вроде электрочайников или фенов для сушки

волос. Неплохим устройством для начала может стать небольшая настольная лампа.

Запустите программу, и когда она запросит ввод точного времени, введите `1` для одной минуты и нажмите клавишу <Enter>. Электроприбор, подключенный к PowerSwitch Tail, должен включиться, кроме того, должен загореться небольшой светодиод состояния на самом устройстве PowerSwitch Tail. По истечении минуты устройство Power-Switch Tail должно отключиться.

Заключение

В этой главе большое значение придавалось предостережениям об опасностях использования высоковольтных цепей переменного тока, но наряду с этим также было показано, насколько просто, имея подходящее оборудование, можно включать и выключать электроприборы.

В следующей главе мы рассмотрим вопросы применения дисплеев при работе с Arduino и Raspberry Pi.

14 Дисплеи

Мало того, что к Raspberry Pi можно подключить монитор, — имеется еще множество различных устройств вывода, способных отображать текст, цифры или графику, можно даже управлять целыми светодиодными панелями. Чтобы рассказать о каждом типе существующих дисплеев, пришлось бы уделить этому слишком много времени, так что в этой главе нам придется остановиться лишь на наиболее практичных и интересных дисплеях и некоторых согласующих устройствах.

Светодиодные ленты

RGB-светодиоды, рассмотренные в *главе 6*, состоят из трех светоизлучающих диодов, совмещенных в одном корпусе. Другой тип светодиодов — так называемый *адресуемый светодиод* — имеет в своем корпусе еще и управляющую микросхему. Эта микросхема обеспечивает ШИМ-управление тремя цветами светодиода — так же, как это делалось нами с помощью Arduino или Raspberry Pi в *главе 6*, с учетом того обстоятельства, что управляющая микросхема здесь встроена непосредственно в сам светодиод.

Такие адресуемые светодиоды разработаны для управления большим их количеством с использованием одного микроконтроллера или компьютера типа Arduino или Raspberry Pi. В Adafruit можно приобрести адресуемые светодиоды, которые у них называются NeoPixels (обратите внимание, что название NeoPixels используется довольно часто, особенно на eBay, в описаниях и тех адресуемых светодиодов, которые не имеют отношение к Adafruit). Наиболее востребованным из них является адресуемый светодиод типа WS2812. Он поддерживает стандарт последовательной передачи данных, позволяющий составлять длинные цепочки из этих светодиодов для создания больших дисплеев. Кроме применения в качестве RGB-адресуемых светодиодов, светодиоды WS2812 можно использовать и как одноцветные светоизлучатели.

Иногда адресуемые светодиоды скомпонованы в формате матрицы, но их также можно приобрести и в виде свернутой в рулон ленты, от которой можно отрезать светодиодную полоску той длины, которая нужна для реализации проекта (рис. 14.1).

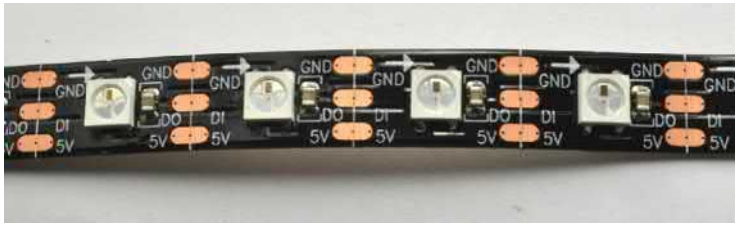


Рис. 14.1. Лента светодиодов NeoPixel

Если присмотреться к изображению на рис. 14.1, можно заметить, что лента сегментирована наподобие ленточного червя. По трем ламелям с каждой стороны светодиода подводится заземление (**GND**) и напряжение в 5 В (**5V**) для электропитания, а также последовательные данные **D0**, которые переходят по гирлянде от одного светодиода к другому. Иными словами, выход одного светодиода соединен со входом другого. Посредине ламелей проходит вертикальная черта, обозначающая линию отреза светодиодной ленты на тот случай, если ее нужно укоротить.

Обратите также внимание на имеющиеся на ленте стрелки, указывающие направление потока последовательных данных. Согласно этому направлению для управления светодиодами всегда нужно подключаться к левой стороне ленты.

Эксперимент: управление дисплеем из ленты RGB-светодиодов

Приобретя один-два метра светодиодной ленты либо на eBay, либо в магазине Adafruit, вам, вероятно, захочется испробовать ее на деле. К Arduino подключить ее весьма просто (рис. 14.2), а вот для подключения ленты к Raspberry Pi придется немного постараться.

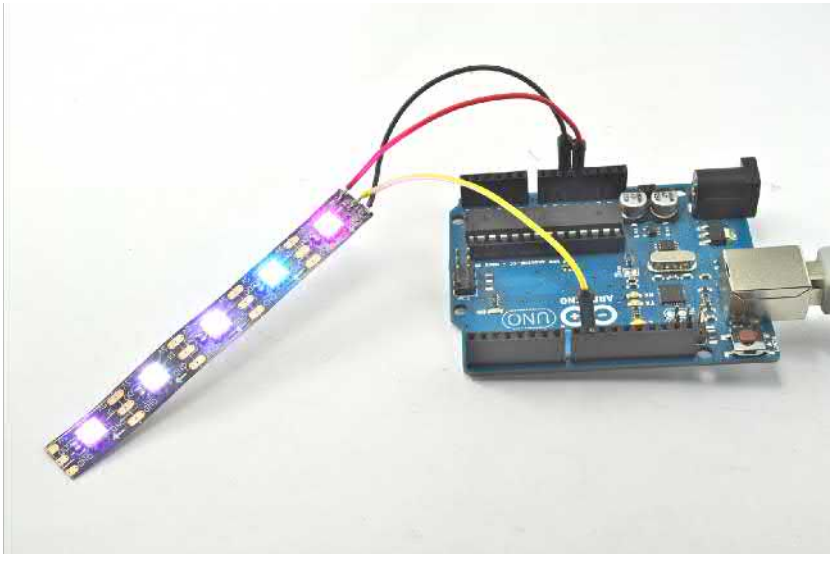


Рис. 14.2. Светодиодная лента, подключенная к Arduino

Комплектующие

В этом эксперименте для работы с Arduino понадобятся следующие комплектующие (табл. 14.1).

Таблица 14.1. *Комплектующие для работы с Arduino в эксперименте со светодиодной лентой*

Компонент схемы	Источники
Адресуемая светодиодная лента WS2812	Adafruit: 1376 eBay
Три перемычки «папа-папа»	Adafruit: 758

Поскольку логический вход светодиодов NeoPixels не совместим с логическими сигналами с уровнем 3 В, для эксперимента с участием Raspberry Pi нам могут понадобиться дополнительные компоненты. Надо ли говорить, что прежде чем предпринимать какие-либо дополнительные усилия, можно попробовать обойтись и без преобразователя логического уровня, т. к. вполне возможно, что ваша светодиодная лента сможет нормально работать и с логическими сигналами, имеющими уровень 3 В.

Тем не менее, надо все же иметь в виду, что для проведения этого эксперимента с использованием Raspberry Pi могут также понадобиться и следующие комплектующие (табл. 14.2).

Таблица 14.2. *Дополнительные комплектующие для работы с Raspberry Pi в эксперименте со светодиодной лентой*

Обозначение	Компонент схемы	Источники
	400-точечная беспаечная макетная плата	Adafruit: 64
R1, R2	Два резистора 470 Ом 0,25 Вт	Mouser: 291-470-RC
Q1	МОП-транзистор 2N7000	Mouser: 512-2N7000
	Перемычки «мама-папа»	Adafruit: 826

В этом наборе допущено отклонение от нашего стандартного перечня транзисторов — для преобразования логических уровней использован маломощный МОП-транзистор 2N7000, поскольку он намного лучше реагирует на высокочастотные последовательные данные, чем, к примеру, 2N3904. Если хотите, можете воспользоваться транзистором типа FQP30N06L, но его мощность для этого эксперимента будет избыточна.

Экспериментируем с Arduino

Подключение Arduino

С одной из сторон светодиодного рулона обычно имеется кабель с трехконтактным разъемом, подсоединенный на ленте к контактам GND, 5V и D0. Если такой разъем

сохранился, то для подключения его к макетной плате или к Arduino можно воспользоваться перемычками «мама-папа». Но если вы отрезали фрагмент ленты так, что контактного кабеля на нем уже нет, для проведения эксперимента нужно будет аккуратно подпаять провода к контактам GND, 5V и D0 на левом краю ленты.

Я пожертвовал тремя перемычками «папа-папа», срезав у них контакт с одной стороны и припаяв перемычки этой стороной к ленте из пяти светодиодов NeoPixel (рис. 14.3).

Полученные три контакта затем могут быть подключены непосредственно к Arduino, причем, как показано на рис. 14.2, контакт D0 светодиодной ленты должен быть подключен на Arduino к контакту D9.

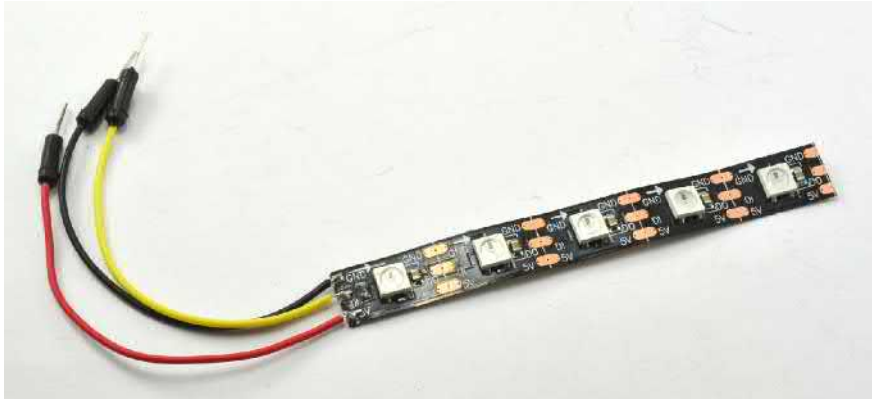


Рис. 14.3. Удобный NeoPixel-дисплей

Энергопотребление адресуемых светодиодов

При наращивании мощности адресуемых светодиодов до максимальной яркости свечения белым цветом, они потребляют довольно большой ток (около 60 мА на каждый светодиод). И если пять адресуемых светодиодов могут потреблять 300 мА, что вполне допустимо при непосредственном подключении к Arduino или Raspberry Pi, то при большем количестве светодиодов вам придется, наверное, подумать о подаче на светодиодную ленту 5-вольтового питания от внешнего источника.

Программа для Arduino

Услужливые сотрудники Adafruit создали библиотеку Arduino, упрощающую управление длинными вереницами адресуемых светодиодов. Эту библиотеку можно загрузить по адресу **https://github.com/adafruit/Adafruit_NeoPixel** и установить в вашу среду Arduino (см. в *главе 12* врезку *«Установка библиотек Arduino»*).

Скетч Arduino для этого эксперимента находится в каталоге /arduino/experiments/neopixel (см. *разд. «Код к книге» главы 2*):

```
#include <Adafruit_NeoPixel.h> // 1

const int pixelPin = 9; // 2
const int numPixels = 5; // 3
```

```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(numPixels, pixelPin,
                                                NEO_GRB + NEO_KHZ800); // 4

void setup() {
  pixels.begin(); // 5
}

void loop() {
  for (int i = 0; i < numPixels; i++) { // 6
    int red = random(255);
    int green = random(255);
    int blue = random(255);
    pixels.setPixelColor(i, pixels.Color(red, green, blue)); // 7
    pixels.show();
  }
  delay(100L);
}
```

Уточним некоторые моменты скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Импортируйте библиотеку `Adafruit NeoPixel`.
2. Если для управления светодиодами NeoPixel потребуется воспользоваться другим контактом, сюда следует внести изменения.
3. Если в вашей светодиодной ленте больше светодиодов, внесите сюда изменение, но перед тем как добавлять слишком большое количество светодиодов, ознакомьтесь с приведенной ранее врезкой *«Энергопотребление адресуемых светодиодов»*.
4. Инициализация библиотеки `NeoPixel` под ваши настройки.
5. Начало отправки данных на дисплей.
6. Назначение для каждого пиксела произвольного цвета, обновление дисплея, а затем задержка на $^1/_{10}$ секунды перед новым изменением цвета всех светодиодов. Настало время танцев в стиле диско!
7. Функция `setPixelColor` принимает два параметра: индексную позицию устанавливаемого пиксела и цвет, который составляется из трех значений в диапазоне от 0 до 255 для каждого из трех цветовых каналов.

Экспериментируем с Raspberry Pi

Подключение Raspberry Pi

Возможно, ваш Raspberry Pi будет работать и без преобразования уровня. Так что, прежде чем браться за макетную плату и использовать преобразователь уровня, попробуйте обойтись без него.

Итак, возьмите дисплей NeoPixel из пяти светодиодов, сделанный для эксперимента с Arduino, и воспользуйтесь перемычками «мама-мама» для создания следующих подключений:

- GND NeoPixel — к GND Raspberry Pi;
- 5V NeoPixel — к 5V Raspberry Pi;
- D0 NeoPixel — к GPIO 18.

На рис. 14.4 показана светодиодная лента, подключенная непосредственно к Raspberry Pi.

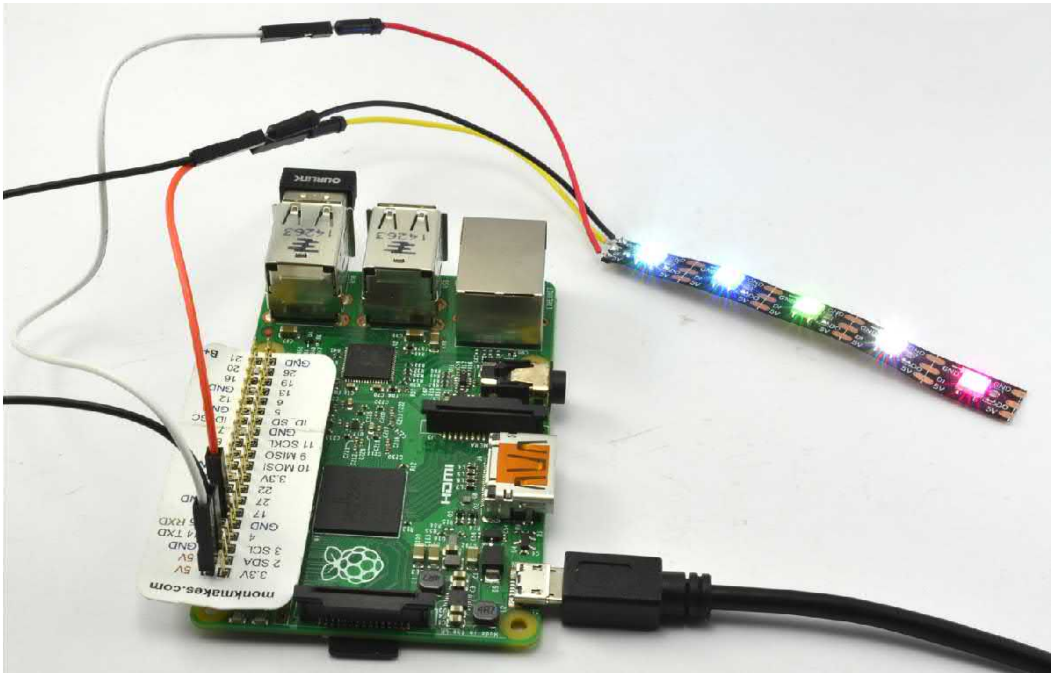


Рис. 14.4. Лента светодиодов NeoPixel, подключенная непосредственно к Raspberry Pi

Перейдите к следующему далее *разд. «Программа для Raspberry Pi»* и попробуйте запустить программу python/experiments/neo-pixel_no_level_conv.py.

Если ваша светодиодная лента не станет светиться, и у вас не получится красивый многоцветный дисплей, то придется, наверное, приложить дополнительные усилия и воспользоваться преобразователем уровня, подняв напряжение сигнала с 3 до 5 В.

Поднимаем напряжение сигнала с 3 до 5 В

Вам, конечно, может повезти, но сигнал управления напряжением в 3 В из Raspberry Pi находится ниже минимально ожидаемого адресуемым светодиодом WS2812 высокого уровня (high) напряжением в 4 В. Схема, приведенная на рис. 14.5, показывает, как можно воспользоваться МОП-транзистором, чтобы поднять уровень сигнала до 5 В.

Побочным эффектом от такого смещения уровня является инвертирование выходного сигнала. То есть, когда на GPIO-контакте Raspberry Pi выставлен логически низкий уровень (low, 0 В), то на выходе для светодиодной ленты появится сигнал напряжением 5 В, а когда на GPIO-контакте Raspberry Pi выставлен высокий уровень (high, 3,3 В), то на выходе для светодиодной ленты появится сигнал напряжением 0 В.

К счастью, эта особенность может быть легко скорректирована в управляющей программе.

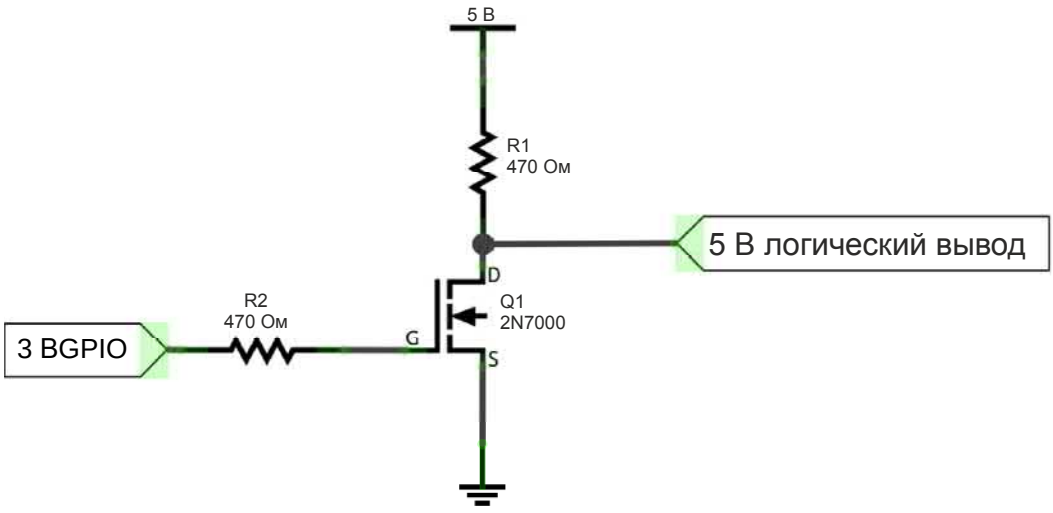


Рис. 14.5. Преобразование уровня с 3 до 5 В

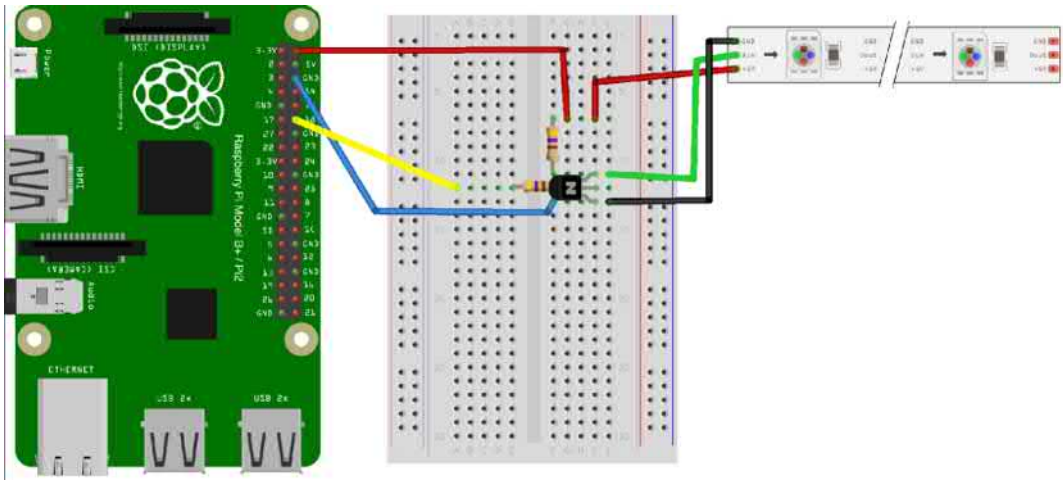


Рис. 14.6. Подключение светодиодной ленты к Raspberry Pi с использованием преобразователя уровня

На рис. 14.6 показана макетная плата со схемой для эксперимента с преобразованием уровня.

Программа для Raspberry Pi

Программа этого эксперимента для Raspberry Pi основана на предоставленном компанией Adafruit руководстве (см. **https://learn.adafruit.com/neopixels-on-raspberry-pi**). Однако библиотека языка C, на которую опирается руководство Adafruit, на момент подготовки книги была несовместима с Raspberry Pi 2.

К счастью для владельцев Raspberry Pi 2, Ричард Херст (Richard Hurst) создал версию программы, работающую с Raspberry Pi 2. Эта версия также совместима и

с более ранними версиями Raspberry Pi. Для установки указанной библиотеки и необходимых программных пакетов воспользуйтесь командой:

```
$ sudo apt-get install build-essential python-dev git scons swig
```

Следующий этап нашей работы расходится с направлением, заданным в руководстве Adafruit, потому что вам нужно получить версию программы, исправленную для работы с Raspberry Pi 2.

Извлеките измененный код NeoPixel из GitHub, используя следующую команду:

```
$ git clone https://github.com/richardghirst/rpi_ws281x.git
```

Измените в извлеченной из GitHub программе каталог, а затем произведите сборку кода на языке C, воспользовавшись следующей командой:

```
$ scons
```

По завершении компиляции кода на языке C вам нужно установить библиотеку Python, взаимодействующую с быстрым кодом на языке C, воспользовавшись следующей командой:

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

Загружаем и выполняем программу

Существуют две практически одинаковые версии программы на языке Python: одна настроена на работу с инвертирующим преобразователем уровня, а вторая — на работу со светодиодной лентой, подключенной непосредственно к Raspberry Pi. Соответственно, запустите программу либо из файла neopixel_no_level_conv.py (для работы со светодиодной лентой, подключенной непосредственно к Raspberry Pi), либо из файла neopixel.py (если вы пользуетесь схемой, собранной на макетной плате). Обе эти программы находятся в каталоге python/experiments/.

Далее показана версия программы, предполагающая использование схемы инвертирующего преобразователя уровня:

```
import time, random
from neopixel import * // 1

# Лента светодиодов // 2
LED_COUNT = 30 # Количество светодиодных пикселов
LED_PIN = 18 # GPIO-контакт, подключенный к пикселам (должен поддерживать ШИМ!)
LED_FREQ_HZ = 800000 # Частота сигнала, подаваемого на светодиоды в герцах
(обычно 800 кГц)
LED_DMA = 5 # DMA-канал, используемый для генерирования сигнала (пробуем 5)
LED_BRIGHTNESS = 255 # Установка 0 для самого темного и 255 для самого светлого
уровня
LED_INVERT = True # True для инвертирования сигнала (при использовании смещения
уровня с помощью транзистора со структурой n-p-n)
```

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

while True: // 3
   for i in range(strip.numPixels()):
      red = random.randint(0, 255)
      green = random.randint(0, 255)
      blue = random.randint(0, 255)
      strip.setPixelColor(i, Color(red, green, blue))
      strip.show()
   time.sleep(0.1)
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Импорт библиотеки `NeoPixel`.
2. Эту установку параметров не нужно изменять, если только не понадобится использовать другой контакт для `LED_PIN`.
3. Основной цикл программы работает практически так же, как и его двойник для Arduino, генерируя произвольный цвет, а затем назначая его пикселу.

Дисплеи I²C на органических светодиодах

Raspberry Pi может быть подключен к любому монитору (большому или маленькому), лишь бы у него были разъемы HDMI или AV, но иногда нужно, чтобы на экран выводилась всего лишь пара строк. У Arduino Uno нет видеовыхода вовсе, и в таком случае опять может пригодиться небольшой дисплей, способный выводить на экран какую-нибудь графическую картинку или несколько строк текста.

Небольшие *дисплеи на основе органических светодиодов* (OLED) имеют невысокую цену, не потребляют много тока и выдают ясное и хорошо читаемое изображение (рис. 14.7). Во многих потребительских товарах они идут на смену жидкокристаллическим дисплеям. OLED-дисплеи доступны как в монохромном, так и в цветном исполнении.

Наиболее широко распространен тип OLED-дисплея в виде модуля, включающего сам дисплей и управляющую микросхему на печатной плате. В качестве управляющей обычно используется микросхема SSD1306, имеющая интерфейс I²C (произносится «ай ту си»), требующий всего два контакта для данных и два контакта для питания.

Экраны таких дисплеев имеют небольшой размер и высокое разрешение, поэтому, если используется Arduino, временами, если не предпринять меры предосторожности, может испытываться дефицит памяти.

Рис. 14.7. Дисплей на основе технологии OLED

Эксперимент: использование модуля I²C-дисплея с Raspberry Pi

OLED-дисплеи I²C могут использоваться и с Arduino, и Raspberry Pi. В рассматриваемом в этой главе далее *разд. «Проект: добавление дисплея к проекту охладителя напитков»* светодиод индикатора температуры мы заменим OLED-дисплеем, показывающим как текущую, так и заданную температуру.

Этот же эксперимент носит чисто демонстрационный характер — просто показать, как дисплей используется с Raspberry Pi. Программа эксперимента выводит на экран время и небольшую анимацию (рис. 14.8).

Комплектующие

OLED-дисплей имеет четыре контакта для подключения — они и должны быть подключены напрямую к Raspberry Pi с помощью четырех перемычек «мама-мама». Соответственно, в этом эксперименте для работы с Raspberry Pi понадобятся следующие комплектующие (табл. 14.3).

Вам следует найти дисплей с разрешением 128×64 пиксела, использующий микросхему управления SSD1306. В некоторых таких дисплеях используется еще один контакт из SSD1306, в котором нет никакой необходимости, поэтому, чтобы не усложнять задачу, ищите дисплей с четырьмя контактами: GND (заземление), VCC (+V), SDA (данные) и CLK (тактовые сигналы). Дисплей, которым я воспользовался, был монохромным, но в этом эксперименте будет работать и цветной дисплей.

Таблица 14.3. *Комплектующие для работы с Raspberry Pi в эксперименте по управлению модулем OLED-дисплея*

Компонент схемы	Источник
OLED-дисплей I²C с разрешением 128×64 пиксела	eBay
Четыре перемычки «мама-мама»	Adafruit: 266

Рис. 14.8. Схема использования OLED-дисплея с Raspberry Pi в сборе

Если дисплей вы приобретели в Adafruit, то для подключения его дополнительных контактов следуйте инструкциям, опубликованным на сайте Adafruit (**https://learn.adafruit.com/096-mini-color-oled/**).

Подключение Raspberry Pi

Микросхема SSD1306 будет работать от 3 или 5 В. Поскольку более ранние модели Raspberry Pi могли при напряжении 3 В выдавать только лишь небольшие токи, есть смысл запитать эту микросхему от 5 В.

Выполните перемычками следующие четыре соединения:

- GND на дисплее соедините с GND на Raspberry Pi;
- VCC на дисплее соедините с 5V на Raspberry Pi;
- CLK на дисплее соедините с GPIO 3 на Raspberry Pi;
- SDA на дисплее соедините с GPIO 2 на Raspberry Pi.

Программа для Raspberry Pi

Если ваша плата Raspberry Pi еще не настроена на поддержку I^2C, вернитесь к *главе 9* и воспользуйтесь рекомендациями, приведенными там во врезке *«Настройка I^2C на Raspberry Pi»*.

В Adafruit разработана большая библиотека Python, предназначенная для поддержки OLED-дисплеев, которая весьма неплохо работает с любым дисплеем, управляемым микросхемой SSD1306, независимо от того, у кого он был приобретен, у Adafruit или у другого поставщика. Чтобы загрузить эту библиотеку, нужно клонировать каталог библиотеки непосредственно в свой Raspberry Pi, используя следующую команду:

```
$ git clone https://github.com/adafruit/Adafruit-SSD1331-OLED-Driver-Library-
for-Arduino.git
```

Для установки библиотеки перейдите в каталог, созданный командой `clone`, а затем запустите сценарий установки:

```
$ cd Adafruit_Python_SSD1306
$ sudo python setup.py install
```

Система координат дисплея

OLED-дисплеи относятся к категории графических — они позволяют рисовать на них различные фигуры, а также выводить текст. Но вам нужно указывать позиции элементов, выводимых на экран. В библиотеке Adafruit используется система координат, показанная на рис. 14.9.

Все пиксельные позиции указываются по отношению к верхнему левому углу экрана, а пиксел в нижнем правом углу имеет координаты x = 127, y = 63.

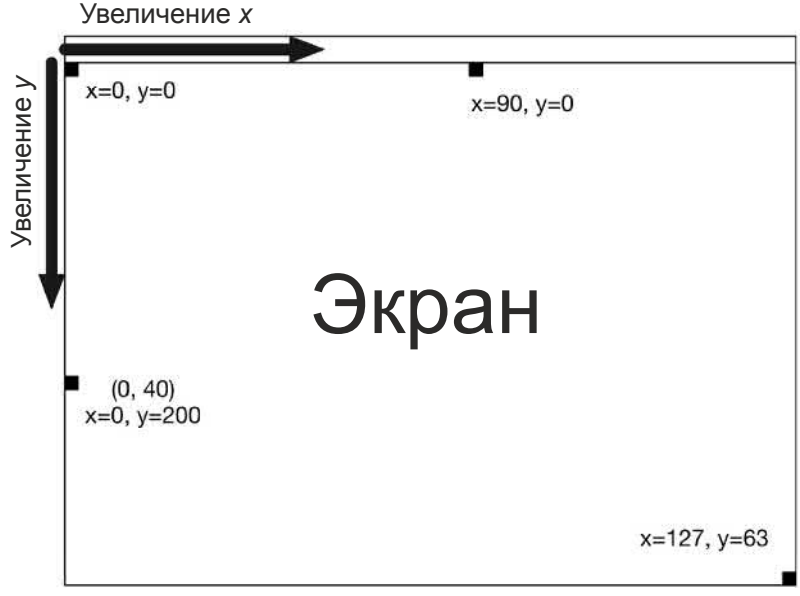


Рис. 14.9. Система координат дисплея

Программа для этого эксперимента находится в файле oled.py каталога python/experiments. Рассмотрим ее код:

```
from oled.device import ssd1306 // 1
from oled.render import canvas
from PIL import ImageFont // 2
import time

device = ssd1306(port=1, address=0x3C) // 3
large_font = ImageFont.truetype('FreeMono.ttf', 24) # // 4

x = 0
while True:
   with canvas(device) as draw: // 5
      draw.pieslice((x, 30, x+30, 60), 45, -45, fill=255) # // 6
      x += 10 // 7
   if x > 128:
      x = 0
   now = time.localtime() // 8
   draw.text((0, 0), time.strftime('%H:%M:%S', now), font=large_font, fill=255)
time.sleep(0.1)
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Из OLED-библиотеки импортируются два модуля: один — `ssd1306`, предназначен для самого устройства, и еще один предназначен для холста, являющегося объектом языка Python, на котором рисуются фигуры и на который выводится текст.
2. Импортируется также библиотека `PIL` (Python Image Library).
3. Создание переменной `device` для получения доступа к дисплею. Указанное здесь значение `0x3C` является I^2C-адресом дисплея. Он часто используется в дешевых дисплеях, приобретаемых на eBay, но оказывается, что для других дисплеев, включая те, что приобретаются у Adafruit, используются иные адреса. Чтобы установить правильный адрес вашего дисплея, нужно изучить его документацию.
4. Указание используемого шрифта высотой `24` пиксела. Он будет применяться для записи показаний времени на дисплей.
5. Конструкция `with ... as` используется для объединения всего кода вывода текста на экран.
6. Рисование фигурки из игры Pac-Man размером 30×30 пикселов с углом окружности от 45° до –45°, чтобы у круга получился своеобразный «рот». Цвет заполнения `fill`, равный `255`, означает «белый».
7. Прибавление `10` к `x`. Переменная `x` устанавливает стартовую позицию рисунка `pieslice`, поэтому при увеличении этой позиции на 10 единиц при каждом прохождении цикла можно получить примитивную анимацию.

8. Получение текущего времени, придание ему формата строки, а затем запись его на дисплей.

Загружаем и выполняем программу

Запустите программу следующей командой:

```
$ sudo python oled.py
```

Если экран остается пустым, то это, скорее всего, из-за неправильного указания адреса I²C.

В этом примере использована функция `piesclice`. Она рисует круг с вырезанным из него куском. Вы можете воспользоваться многими другими функциями рисования графических образов, поэтому стоит обратиться по адресу **http://effbot.org/imagingbook/imagedraw.htm** и попробовать для оживления дисплея некоторые из них.

Проект: добавление дисплея к проекту охладителя напитков

Чтобы разобраться, как работает OLED-дисплей в связке с Arduino, возьмем за основу материал *разд. «Проект: термостатический охладитель напитков» главы 12* и заменим зеленый светодиод, фиксирующий выход охладителя на нужный температурный режим, OLED-дисплеем, отображающим как фактическую, так и заданную температуру (рис. 14.10).

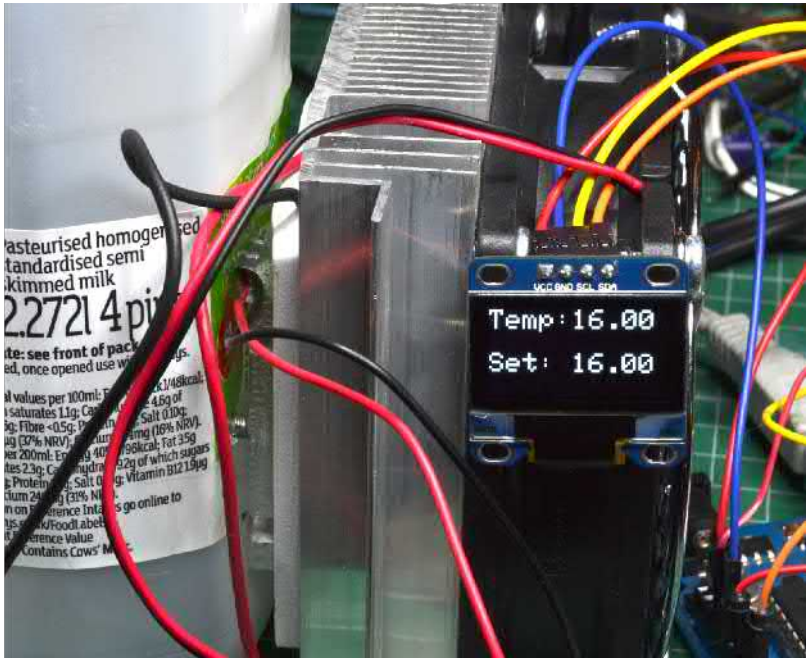


Рис. 14.10. Добавление OLED-дисплея к проекту охладителя напитков

Комплектующие

Кроме компонентов, задействованных в *разд. «Проект: термостатический охладитель напитков» главы 12* (за исключением светодиода и резистора R3), вам понадобятся OLED-дисплей, подобный тому, что использовался ранее в *разд. «Эксперимент: использование модуля I^2C-дисплея с Raspberry Pi»* этой главы, и еще четыре перемычки «мама-папа».

Подключение Arduino

Если вы этого в свое время еще не сделали, соберите схему, рассмотренную в *разд. «Проект: термостатический охладитель напитков» главы 12*. При этом добавлять светодиод и резистор R3 не нужно.

OLED-дисплей можно подключить непосредственно к Arduino, уменьшив помехи, связанные с применением макетной платы. На рис. 14.11 показана только новая часть реализации проекта с OLED-дисплеем, подключенным к Arduino.

Будьте внимательны при подключении проводов от Arduino к выходным контактам OLED-модуля, не перепутайте контакты 5V и GND.

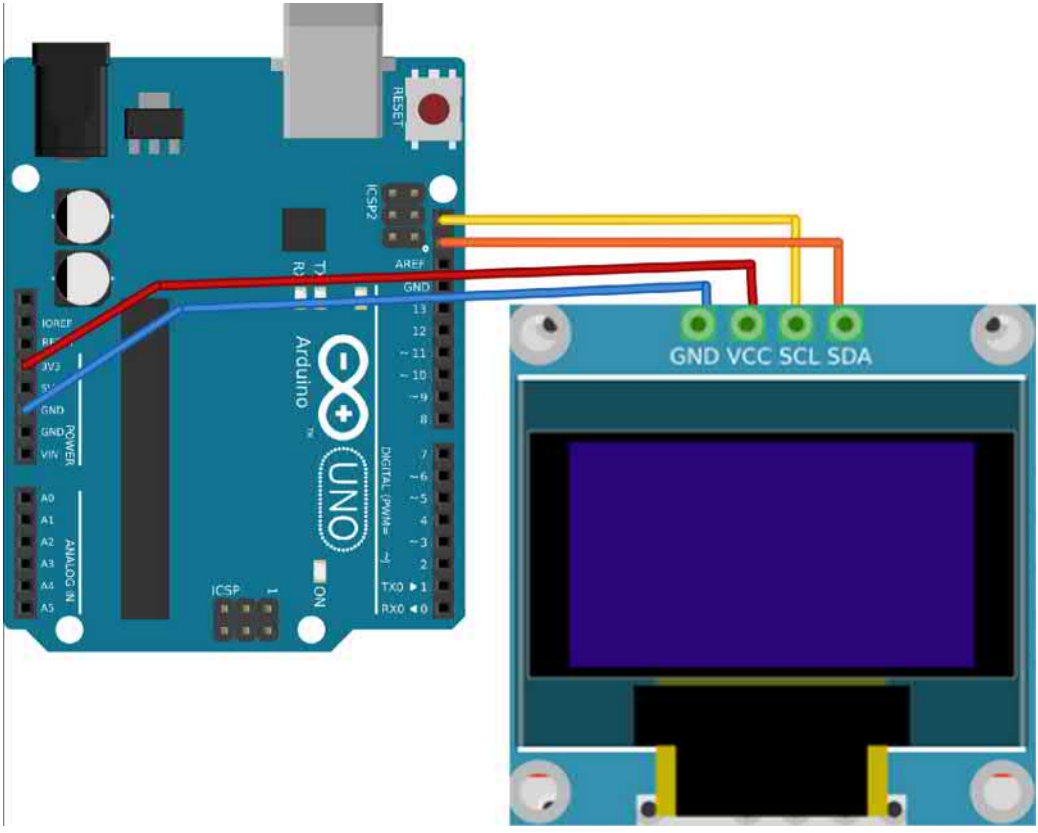


Рис. 14.11. Подключение Arduino к OLED-дисплею

Программа для Arduino

Как вы, наверное, и предполагали, для OLED-дисплея также существует библиотека Arduino (**https://github.com/adafruit/Adafruit_SSD1306**), поэтому ее нужно загрузить и установить в IDE вашего Arduino (см. в *главе 12* врезку *«Установка библиотек Arduino»*).

Для этой версии охладителя используется почти такая же программа, что и для варианта без использования дисплея. Она находится в каталоге arduino/projects/thermostatic_cooler_display (см. *разд. «Код к книге» главы 2*). Изменения касаются включения в нее библиотек `Adafruit GFX` и `SSD1306`: библиотека `SSD1306` предназначена для самого устройства, а библиотека `GFX` предоставляет полезные функции для отображения текста, рисования фигур и т. д.:

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Переменная `display` определена для предоставления ссылки на библиотеку. Параметр `4` здесь является фактическим номером контакта для подключения к контакту Enable дисплея, если таковой у него имеется. У наших дисплеев контакта Enable нет, поэтому число `4`, выбранное в качестве номера контакта, ни для чего другого в Arduino не используется:

```
Adafruit_SSD1306 display(4);
```

Функция `setup()` включает следующую строку, инициализирующую дисплей:

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
```

Функция `checkTemperature` теперь включает вызов новой функции `updateDisplay` для обновления содержимого дисплея текущим показателем температуры каждую секунду:

```
void updateDisplay() {
   display.clearDisplay();
   display.setTextSize(2);
   display.setTextColor(WHITE);
   display.setCursor(0,0);
   display.print("Температура:");
   display.println(measuredTemp);
   display.print("Установка: ");
   display.println(setTemp);
   display.display();
}
```

Функция `readSetTempFromPot` была немного изменена с тем, чтобы если заданная температура изменяется, сразу же вызывалась функция `updateDisplay`, т. е., чтобы при повороте подстроечного потенциометра установленная температура обновлялась сразу же, без необходимости ожидания наступления следующей секунды:

```
double readSetTempFromPot() {
   static double oldTemp = 0;
   int raw = analogRead(potPin);
```

```
    double temp = map(raw, 0, 1023, minTemp, maxTemp);
    if (oldTemp != temp) {
      updateDisplay();
      oldTemp = temp;
    }
    return temp;
}
```

Ключевое слово `static` перед определением `oldTemp` означает, что `oldTemp` будет сохранять свое значение между успешными вызовами функции `readSetTempFromPot`. Обе переменные, `temp` и `oldTemp`, имеют тип `double` (число с плавающей точкой двойной точности), поскольку именно этот тип используется в библиотеке DS18B20.

Заключение

Существует множество различных типов дисплеев, но в этой главе были рассмотрены только некоторые, наиболее полезные из них.

В следующей главе мы рассмотрим использование Arduino и Raspberry Pi для создания звука.

Звук 15

Разобравшись с управлением движением, светом, температурой и дисплеями, настало время обратить внимание и на создание звуков.

На Raspberry Pi получать высококачественный звук проще, поскольку активные колонки можно подключить к его аудиоразъему, а на Arduino эта задача решается несколько сложнее.

Эксперимент: громкоговоритель без усилителя на Arduino

Для воспроизведения более или менее сложного звука понадобится какой-нибудь громкоговоритель. Громкоговорители, появившиеся почти сто лет назад, в большинстве своем работают на основе соленоида (см. *разд. «Соленоиды» главы 7*) — содержащаяся в них электромагнитная катушка с достаточно высокой частотой колеблет жесткий конус, создающий звуковые волны.

На громкоговорители обычно наносится маркировка в омах. И хотя этой единицей измеряется резистивное сопротивление, но применительно к громкоговорителям говорят, что это их *импеданс*. По своему смыслу этот термин тоже означает сопротивление, но он применяется к приборам, не являющимся резисторами в полном смысле этого слова, и имеющаяся в громкоговорителе проводная катушка (как и любая такая катушка) не работает как простой резистор. Если вы заинтересовались этой проблематикой, можете почитать статьи, посвященные *индуктивности*.

Обычно на громкоговорителях встречаются значения 4 или 8 Ом. И если вы собираетесь подключить 8-омный громкоговоритель к выходу Arduino с напряжением 5 В, то с полным основанием можете ожидать возникновения тока, равного по силе:

$$I = V / R = 5 / 8 = 625 \text{ мА}$$

То есть, тока намного большего, чем те 40 мА, которые рекомендуются для выходного контакта Arduino. Похоже, нам и здесь понадобится согласующий резистор!

Итак, в этом эксперименте мы подключим громкоговоритель к Arduino через резистор, а затем воспользуемся окном монитора порта, чтобы научить Arduino воспроизводить звук конкретной частоты.

Звуковые частоты

Частота звуковой волны в музыкальных терминах называется *тоном* и выражается в количестве звуковых волн, достигающих ваших ушей за одну секунду. Мне нравится представлять звуковые волны в виде ряби на пруду. Звук высокой частоты — скажем, 10 кГц (килогерц), создает 10 000 звуковых волн в секунду, а звук низкой частоты (скажем, 100 Гц) — всего 100 звуковых волн в секунду. Человек обычно слышит звуки в диапазоне от 20 Гц до 20 кГц, но верхняя граница с возрастом снижается. Звуки выше 20 кГц обычно называют *ультразвуком*.

У животных другие диапазоны слышимых звуков. Например, кошки могут слышать частоты вплоть до 55–79 кГц, а летучие мыши известны тем, что вообще пользуются ультразвуковой эхолокацией.

В музыке самая низкая нота «До» на обычном пианино соответствует частоте 32,7 Гц, а самая высокая — частоте 4186 Гц. Известное всем, имеющим хоть какое-то касательство к музыке, понятие *октава* означает удваивание частоты. Поэтому если заставить на клавиатуре пианино зазвучать две ноты «До» смежных октав, то вторая нота будет звучать с удвоенной частотой первой ноты.

Комплектующие

Для этого эксперимента много деталей не потребуется — только громкоговоритель и резистор, хотя, если воспользоваться макетной платой и перемычками, подключения можно упростить (табл. 15.1).

Таблица 15.1. *Комплектующие для работы с Arduino в эксперименте с громкоговорителем*

Компонент схемы	Источники
Небольшой громкоговоритель с импедансом 8 Ом	Adafruit: 1891
Резистор 270 Ом 0,25 Вт	Mouser: 291-270-RC
400-точечная беспаечная макетная плата	Adafruit: 64
Перемычки «папа-папа»	Adafruit: 758

Громкоговоритель я извлек из старого радиоприемника, и у него оказался разъем, в который можно было вставить штекерные наконечники перемычек. Вам может достаться громкоговоритель с припаянными к нему проводами, которые удастся воткнуть в разъемы Arduino, или же вам придется подпаять к этим проводам выводы, толщина которых позволит воткнуть их в гнезда монтажной платы.

Макетная схема эксперимента

Схема, собранная для проведения эксперимента на макетной плате, показана на рис. 15.1.

Один провод от громкоговорителя подключается к контакту **GND** (заземление) Arduino, а второй — к контакту **D11** через резистор.

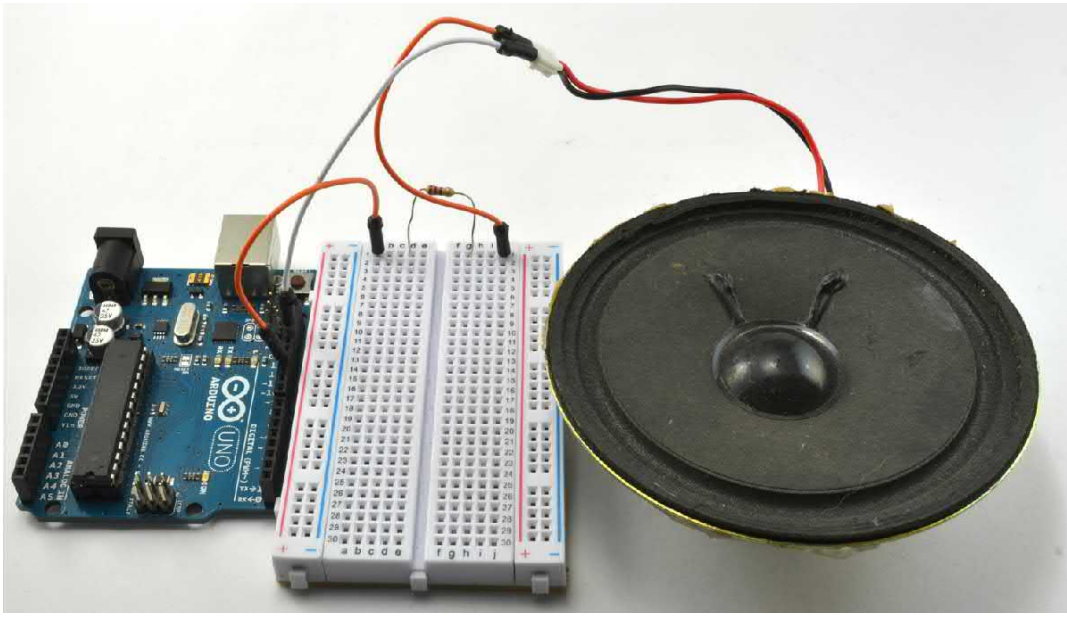


Рис. 15.1. Схема эксперимента с громкоговорителем на Arduino в сборе

Программа для Arduino

Скетч Arduino для этого эксперимента находится в каталоге /arduino/experiments/ex_speaker (см. *разд. «Код к книге» главы 2*):

```
const int soundPin = 11;

void setup() {
   pinMode(soundPin, OUTPUT);
   Serial.begin(9600);
   Serial.println("Введите частоту ");
}

void loop() {
   if (Serial.available()) {
      int f = Serial.parseInt();
      tone(soundPin, f); // 1
      delay(2000);
      noTone(soundPin); // 2
   }
}
```

Незнакомая вам часть кода может находиться внутри цикла `loop()`:

1. `tone` настраивает один из выходных контактов Arduino на воспроизведение звука с указанной частотой (в данном случае, с частотой, набранной в окне монитора порта).

2. После двухсекундной задержки команда `noTone` отменяет воспроизведение звука, возвращая умиротворяющую тишину.

Загружаем и выполняем программу

Загрузите в устройство программу, а затем откройте окно монитора порта. Попробуйте ввести значение 1000 — должен услышаться не самый приятный звук умеренной громкости, но не настолько громкий, чтобы быть услышанным в шумном помещении.

Продолжите эксперимент, вводя различные частоты и наблюдая изменение звука.

В случае, если вы попытаетесь получить в этом эксперименте частоты на границах слышимости, вас, к сожалению, будет ожидать разочарование, поскольку у громкоговорителя, скорее всего, окажется свой, еще более узкий диапазон звучания, и на частотах, превышающих примерно 10 кГц, громкость звука резко пойдет на спад. Небольшие громкоговорители также обычно не способны нормально воспроизводить звуки с частотой, ниже 100 Гц.

Синусоида и меандр

Звук, излучаемый громкоговорителем при подключении его непосредственно к выходному контакту Arduino, режет слух и кажется грубым. Дело в том, что цифровой выход может находиться только во включенном и выключенном состоянии, и звуковая волна, получаемая при этом, по форме напоминает *меандр*.

По сравнению с более мягкой и приближенной к синусоиде звуковой волной музыкального инструмента, звуковая волна в форме меандра считается весьма ненатуральным звуком (рис. 15.2).

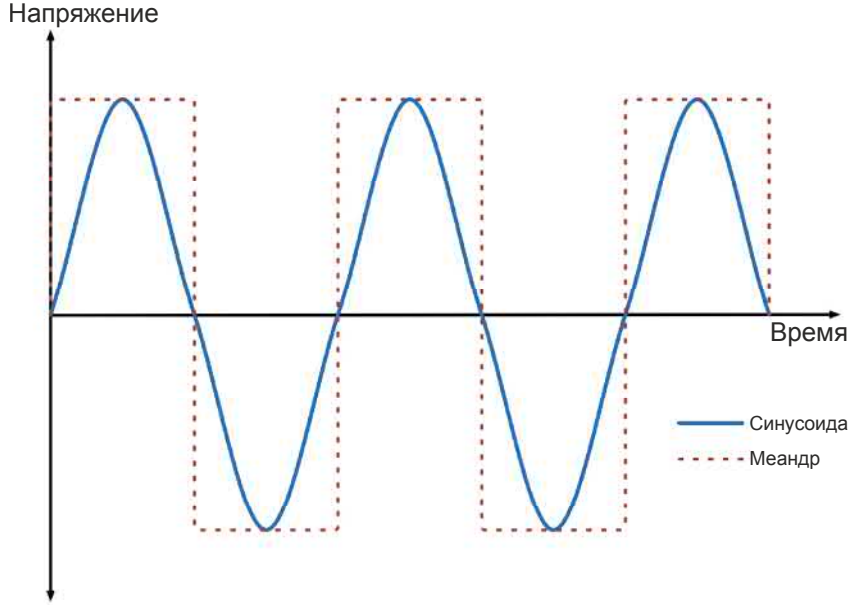


Рис. 15.2. Звуковые волны в виде меандра и синусоиды

Усилители

Чтобы сделать генерируемый звук громче, требуется выдать на громкоговоритель больше мощности. Иными словами, сигнал нужно *усилить*.

В конструкции, собранной для эксперимента из *разд. «Эксперимент: громкоговоритель без усилителя на Arduino»*, мощность звука, излучаемого громкоговорителем, можно существенно усилить, используя транзистор, как это делалось для включения и выключения реле или электродвигателя, — звук останется столь же неприятным, но станет значительно громче.

Если же требуется сгенерировать более гладкую звуковую волну — скажем, для музыки или речи, принцип включения/выключения звукового сигнала, ведущий к выдаче ужасного звучания, уже не подойдет, и придется воспользоваться настоящим *усилителем* звука.

Усилитель можно собрать самостоятельно, но намного проще воспользоваться уже готовым изделием или же парой колонок, которыми оснащен ваш персональный компьютер. Задействовать такие колонки, в которые усилитель уже встроен, окажется особенно заманчиво, когда дело дойдет до использования Raspberry Pi, поскольку их входной штекер можно будет просто подключить к аудиоразъему Raspberry Pi.

Такой подход мы рассмотрим позже, когда кукла Пепе (из *разд. «Проект: танцующая кукла Пепе на Raspberry Pi» главы 9*) станет обретать голос (см. *разд. «Проект: кукла Пепе обретает голос»* далее в этой главе).

Эксперимент: воспроизведение звуковых файлов на Arduino

Звуковые файлы формата WAV можно проигрывать на Arduino с помощью оборудования из *разд. «Эксперимент: громкоговоритель без усилителя на Arduino»* и библиотеки Arduino под названием PCM (от англ. pulse code modulation, *импульсно-кодовая модуляция*). Для генерирования колебаний, способных вызвать звук, в этой библиотеке используется технология, слегка похожая на технологию широтно-импульсной модуляции (ШИМ). Флэш-памяти Arduino хватает приблизительно на 4 секунды записи, а если требуется проигрывать более длинные звуковые клипы, надо добавить к Arduino кардридер SD и следовать рекомендациям, приведенным на веб-сайте Arduino по адресу:
https://www.arduino.cc/en/Tutorial/SimpleAudioPlayer.

Звук мы запишем на компьютере, воспользовавшись пакетом программ Audacity, а затем запустим утилиту, предназначенную для преобразования звукового файла в набор чисел, соответствующих этому звуку, который можно будет вставить в программу для Arduino и воспроизвести.

Исходная статья с описанием такого подхода к воспроизведению звука опубликована на веб-сайте High-Low Tech (**http://highlowtech.org/?p=1963**). Наш экспери-

мент немного отличается от этого описания тем, что в нем для записи аудиоклипа используется свободно распространяемое приложение Audacity.

Оборудование и софт

В этом эксперименте используется то же самое оборудование, что и в *разд. «Эксперимент: громкоговоритель без усилителя на Arduino»*. Но вам придется установить на компьютер следующие программы, позволяющие записывать и обрабатывать звуковые клипы:

- приложение Audacity: **audacityteam.org**;
- утилиту Audio Encoder — ссылку на версию под вашу операционную систему можно найти на веб-сайте **highlowtech.org/?p=1963**.

Создание звукового файла

> ***Примечание***
>
> Если записывать свой собственный звуковой клип желания у вас нет, вы можете сразу перейти к *разд. «Загружаем и выполняем программу»*, входящему в состав этого эксперимента, и запустить программу *wav_arduino*, содержащую в закодированном виде небольшой звуковой блок.

Чтобы создать звуковой файл, запустите программу Audacity, переключите режим записи в моно (**Mono**) и выберите частоту проекта (**Project Rate**) равной 8000 Гц. Эти варианты настройки выделены на рис. 15.3.

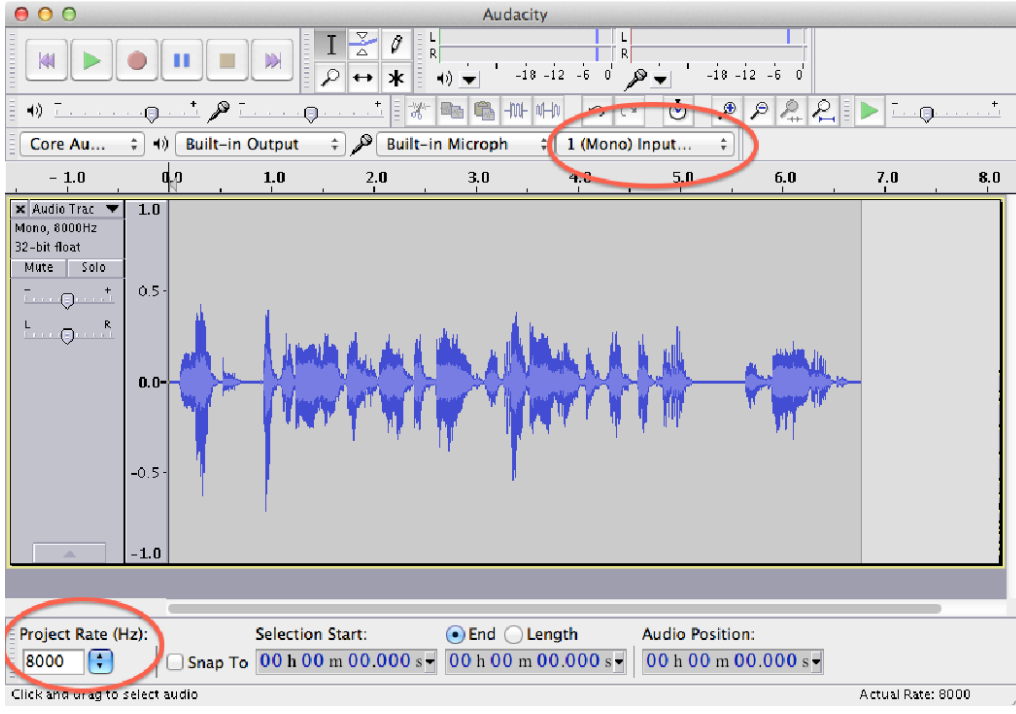


Рис. 15.3. Программа Audacity: запись звукового клипа

Чтобы начать запись, щелкните на красной кнопке **Record** (Записать) и запишите свое сообщение. Учтите, что запись не должна быть длиннее 4 секунд. Как только запись будет сделана, в окне Audacity появится изображение звуковой волны. Можно выбрать любую «немую» область в начале или в конце записи и удалить ее, оставив только нужную часть звукового блока.

Записанный звук надо экспортировать в нужный формат. Для этого опять понадобится выполнить специальные настройки. Выберите в меню **File** (Файл) пункт **Export** (Экспорт аудио), затем в раскрывающемся меню выберите вариант **Other uncompressed files** (Прочие несжатые файлы), перейдите в область настроек **Format options** (Настройки формата) и укажите характеристики **WAV (Microsoft)** и **Unsigned 8 bit PCM** (рис. 15.4). Присвойте файлу имя и пропустите страницу, приглашающую ввести данные об исполнителе.

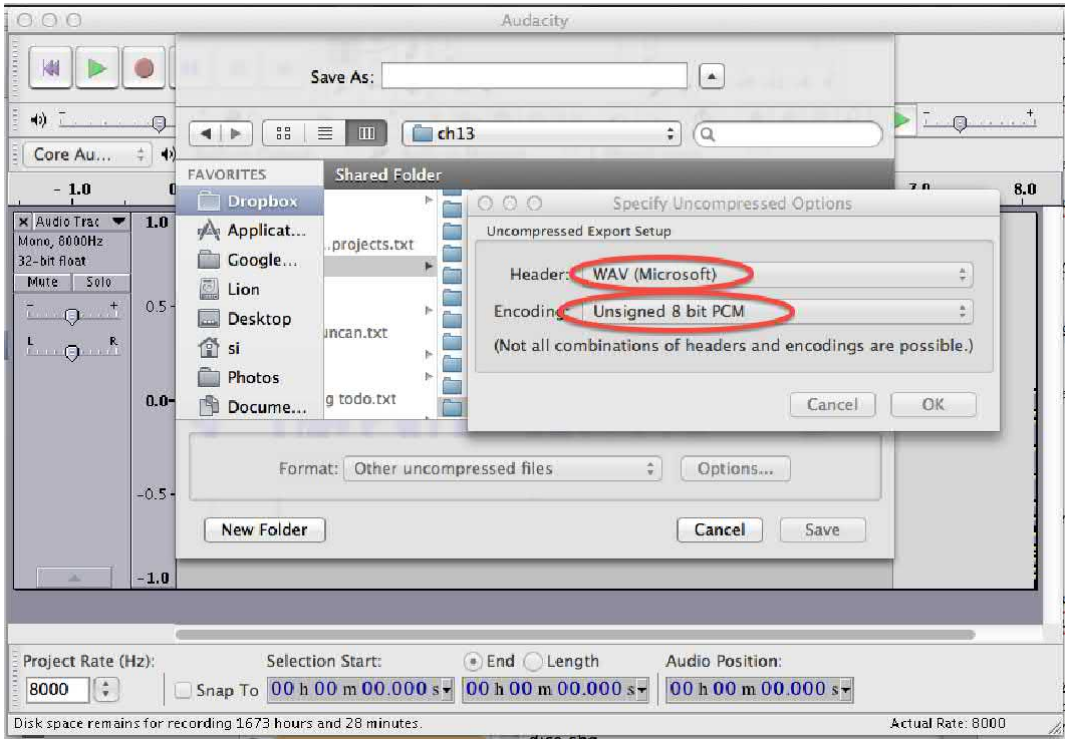


Рис. 15.4. Программа Audacity: выбор параметров экспорта

Только что сгенерированный файл представляет собой двоичные данные. Его нужно преобразовать в список чисел в текстовом виде, отделенных друг от друга запятыми, чтобы его можно было вставить в нашу программу. Запустите для этого утилиту Audio Encoder, загруженную с веб-сайта **highlowtech.org**. Она выведет на экран приглашение на выбор конвертируемого файла — выберите файл, который только что был экспортирован из Audacity.

Через несколько секунд появится диалоговое окно с утверждением, что все данные находятся в вашем буфере обмена.

Откройте программу /arduino/experiments/wav_arduino и замените всю строку, начинающуюся с последовательности 125, 119, 115, данными из буфера обмена. Это очень длинная строка, поэтому лучше всего выбрать ее, поместив курсор в начало строки, а затем, удерживая нажатой клавишу <Shift>, опустить курсор вниз и переместить его на последний символ строки. Для замены выбранного текста данными из буфера обмена воспользуйтесь пунктом контекстного меню **Insert** (Вставить).

Если нанести все вставленные числа на график, его форма будет напоминать ту, что вы уже видели в Audacity при записи звукового клипа.

Программа для Arduino

Перед тем как приступить к компиляции и запуску программы, нужно установить библиотеку PCM. Загрузите ZIP-архив из GitHub (**https://github.com/damellis/PCM/zipball/master**), распакуйте его, переименуйте папку в PCM и переместите ее в каталог своих библиотек Arduino, воспользовавшись рекомендациями, представленными во врезке *«Установка библиотек Arduino» главы 12*.

Программа для Arduino (если проигнорировать звуковые данные) имеет весьма скромный размер, но массив данных представлен очень длинной строкой!

```
#include <PCM.h>

const unsigned char sample[] PROGMEM = { // 1
125, 119, 115, 115, 112, 116, 114, 113, 124, 126, 136, 145, 139,
};

void setup() {
   startPlayback(sample, sizeof(sample)); // 2
}

void loop() { // 3
}
```

Уточним некоторые моменты этого скетча по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Данные хранятся в массиве типа char, состоящем из восьмиразрядных чисел без знака. Команда PROGMEM обеспечивает хранение данных во флэш-памяти Arduino (в ней должно быть доступно около 32 Кбайт).
2. Проигрывание звукового сэмпла осуществляет библиотека PCM. Функции startPlayback передается массив проигрываемых данных и размер данных в байтах.
3. Звуковой клип проигрывается однократно при каждом перезапуске Arduino, поэтому функция loop() остается пустой.

Загружаем и выполняем программу

Загрузите программу в свой Arduino, и как только загрузка завершится, звуковой клип будет воспроизведен!

Сразу же после загрузки программы в нижней части Arduino IDE появится сообщение о том, сколько флэш-памяти Arduino было задействовано, примерно следующего содержания: **Binary sketch size: 11,596 bytes (of a 32,256 byte maximum)** (Размер двоичной программы: 11 596 байтов (из максимально возможных 32 256 байтов)). Если звуковой файл окажется слишком большим, вы получите сообщение об ошибке.

Подключение Arduino к усилителю

Как ни удивительно, но только что проведенный эксперимент оказался успешным, хотя его удалось проделать с применением только лишь Arduino.

Ток звукового сигнала, поступающего от Arduino, ограничивается резистором, но напряжения 5 В, подводимого к выводному контакту Arduino, слишком много для подключения ко входу обычного усилителя звука. Поэтому, когда мы хотим подключить Arduino к звуковым колонкам с целью сделать звук громче, сначала, как ни странно, нужно уменьшить напряжение на выходе.

Для этого вполне подойдут два резистора, включенные по схеме *делителя напряжения*.

Делитель напряжения

Делителем напряжения называется устройство, в котором для уменьшения напряжения используются два резистора. По сравнению с напряжением, изменяющимся, скажем, пропорционально изменению звукового сигнала, делитель напряжения изменяет его в фиксированной пропорции.

Например, на рис. 15.5 показан делитель напряжения, используемый для снижения сигнала напряжением 5 В от Arduino до более подходящего в качестве входа для усилителя звука значения в 0,5 В (или около того).

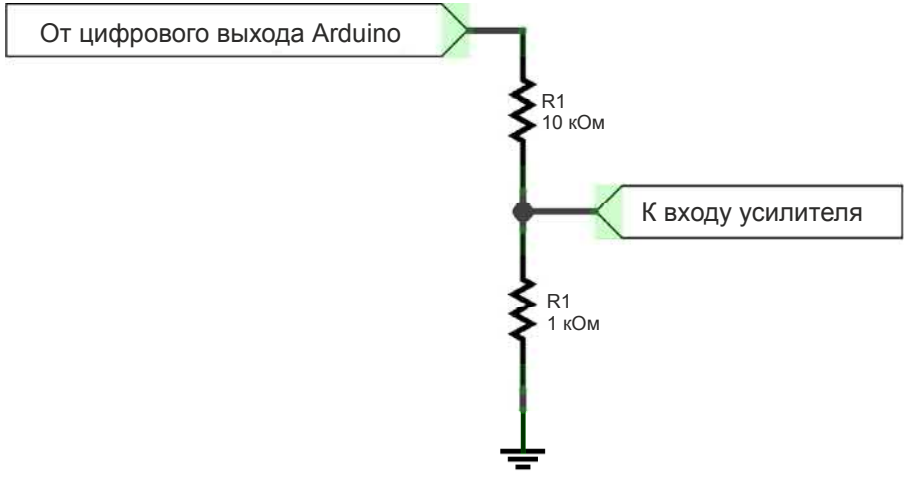


Рис. 15.5. Делитель напряжения

Напряжение в точке соединения двух резисторов (V_{out}) вычисляется по формуле:

$$v_{out} = \frac{R_2}{R_1 + R_2} \times V_{in}$$

В данном случае, когда на цифровом выходе выставлен высокий уровень сигнала (high) и Vin = 5 В, то:

$$V_{out} = 5 \times 1/(1+10)=0{,}45 \text{ В}.$$

Макетную плату из *разд. «Эксперимент: громкоговоритель без усилителя на Arduino»* можно приспособить под подключение к усилителю, поместив один резистор над другим, как показано на рис. 15.6, и подключив верхний резистор номиналом 10 кОм к выводу **D11**, а нижний вывод нижнего резистора — к выводу **GND** (заземление). После этого останется только выбрать способ подключения контактной линии **GND** и линии, на которой встретились резисторы, к усилителю.

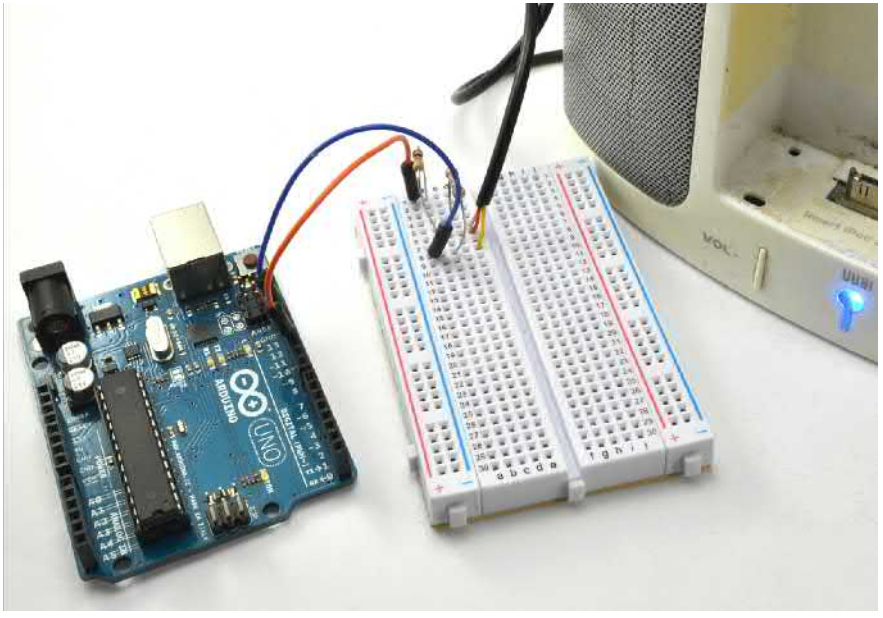


Рис. 15.6. Схема подключения Arduino ко входу усилителя звукового сигнала в сборе

Для этого можно пожертвовать входным кабелем усилителя, укоротив его наполовину и зачистив провода на конце, чтобы их можно было вставить в гнезда монтажной платы. Обычно такой кабель имеет три жилы, поскольку большинство звуковых систем рассчитано на работу со стереосигналом. То есть, в нем имеется один общий (GND) провод и два отдельных провода: для левого и правого звуковых каналов (они обычно имеют изоляцию красного и белого цвета).

Вам нужно найти только общий (GND) провод, поскольку провода для левого и правого каналов лучше соединить вместе, чтобы монофонический сигнал от Arduino был слышен в обеих колонках. Найти общий (GND) провод можно с помощью мультиметра, установленного в режим проверки наличия контакта (рис. 15.7).

Рис. 15.7. Тестирование входного кабеля звукового сигнала

Прикрепите один из проводов мультиметра к самой дальней от конца штекера звукового кабеля контактной площадке. Затем по очереди прикасайтесь другим проводом мультиметра к каждому из трех проводов кабеля, пока не услышите звуковой сигнал мультиметра или не обнаружите наличие контакта по каким-либо другим признакам. Это и будет общий провод, который нужно вставить в гнездо контактной линии GND макетной платы. Два других провода нужно скрутить вместе и вставить в гнездо контактной линии материнской платы, служащей выходом, т. е. той линии, на которой соединяются два резистора.

Повторите теперь один из ранее проводившихся в этой главе экспериментов с Arduino, и получите намного более громкий и чистый звук.

Проигрывание звуковых файлов на Raspberry Pi

Raspberry Pi представляет собой полноценный компьютер с выходным аудиоразъемом, поэтому проигрывание звукового файла на Raspberry Pi сводится к поиску подходящего для этого пакета программ.

Существует несколько подходов к решению этой задачи, и вопросы, рассматриваемые на форуме StackExchange (**http://raspberrypi.stackexchange.com/questions/7088/playing-audio-files-with-python**), охватывают большинство из них.

Здесь мы воспользуемся методом, связанным с применением Python-библиотеки `pygame`, которая уже установлена в Raspberry Pi.

Файлы WAV-формата, несмотря на то, что они намного длиннее файлов формата MP3, имеют большое преимущество перед ними в том, что Raspberry Pi их очень просто декодировать, и поэтому они не приводят к сколько-нибудь заметному замедлению работы Pi-устройства. Ранее, в *разд. «Эксперимент: воспроизведение звуковых файлов на Arduino»*, отмечалась привередливость Arduino в вопросах проигрывания WAV-файлов, а вот Raspberry Pi не имеет ограничений по памяти или скорости процессора при проигрывании звуковых файлов, поэтому практически любой звуковой файл будет на нем воспроизводиться без проблем.

Убедиться в этом можно, воспользовавшись командной строкой Python и звуковым файлом, который будет далее использован в *разд. «Проект: кукла Пепе обретает голос»*. Измените в командной строке Raspberry Pi каталог на тот, в котором загружены файлы для этой книги, а затем, находясь в нем, на каталог python/projects/puppet_voice. В нем вы найдете pepe_1.wav. Для проигрывания этого файла подключите колонки или наушники к аудиовыходу вашего Raspberry Pi, а затем запустите консоль Python, используя команду `python`:

```
>>> from pygame import mixer
>>> mixer.init()
>>> mixer.music.load("pepe_1.wav")
>>> mixer.music.play()
```

Вы сможете услышать короткое сообщение от Пепе.

Проект: кукла Пепе обретает голос

Теперь, когда вы научили Raspberry Pi проигрывать звуковые файлы, эту его возможность можно объединить с конструкцией из *разд. «Проект: танцующая кукла Пепе на Raspberry Pi» главы 9*, добавив к ней ИК-датчик, чтобы заставить Пепе танцевать и разговаривать, когда к нему кто-нибудь приближается (рис. 15.8).

Комплектующие

Для осуществления этого проекта понадобятся все комплектующие из *разд. «Проект: танцующая кукла Пепе на Raspberry Pi» главы 9*, а также следующие дополнительные компоненты (табл. 15.2).

Таблица 15.2. *Комплектующие для работы с Raspberry Pi в проекте, где кукла Пепе обретает голос*

Компонент схемы	Источники
Пассивный инфракрасный (ИК) датчик (Passive infrared, PIR)	Adafruit: 189 eBay
Перемычки «мама-папа»	Adafruit: 826
Активные громкоговорители (колонки)	
400-точечная беспаечная макетная плата	Adafruit: 64

Рис. 15.8. Схема проекта с танцующей и говорящей куклой в сборе

ИК-датчики обычно используются для обнаружения движения в охранной сигнализации. Это недорогое устройство идеально подходит для оживления Пепе в случае, когда к нему кто-нибудь приближается. С учетом включения в конструкцию ИК-датчика, для его подключения и для подключения платы управления серводвигателями есть смысл воспользоваться макетной платой.

Макетная схема проекта

В этом проекте макетная плата позволяет получить надежное крепление проводов от ИК-датчика и от платы управления серводвигателями, которые будут подключены непосредственно к ней. У многих макетных плат снизу имеется липкая поверхность, с помощью которой плату можно закрепить на монтажной панели с серводвигателями, придав всей конструкции более прочную основу. На рис. 15.9 приведена схема подключений на макетной плате проекта, а на рис. 15.10 показано, как все соединения выглядят на самом деле.

У ИК-датчика имеется выход с напряжением 3 В, а для питания ему требуется напряжение 5 В. Поэтому он идеально подходит для использования вместе с Raspberry Pi.

Пассивные инфракрасные датчики

ИК-датчики обнаруживают движение любого объекта, излучающего тепло (например, человека). Как только датчик регистрирует изменение уровня тепла в окружающей его среде или — на некоторых устройствах — изменение картины падающего на датчик инфракрасного излучения, на цифровом выходе датчика на одну-две секунды появляется сигнал высокого уровня.

Его подключение к Arduino или Raspberry Pi сводится к подсоединению к цепям электропитания и к подсоединению его цифрового выхода к цифровому входу Raspberry Pi или Arduino.

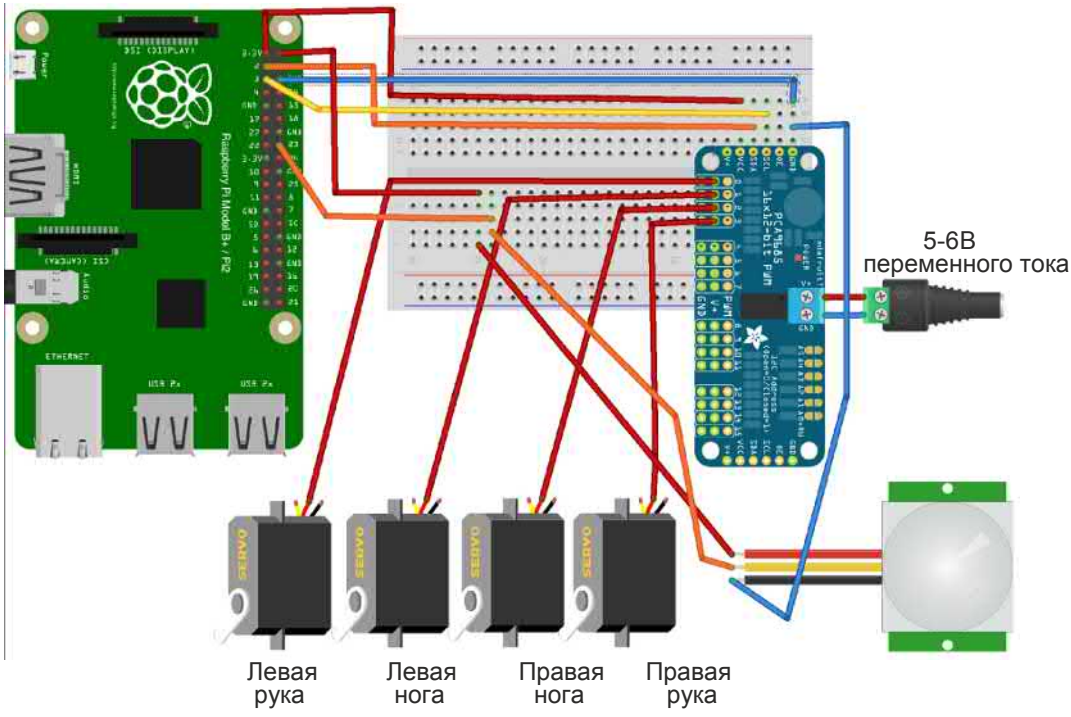


Рис. 15.9. Схема подключений на макетной плате для создания говорящей и танцующей куклы

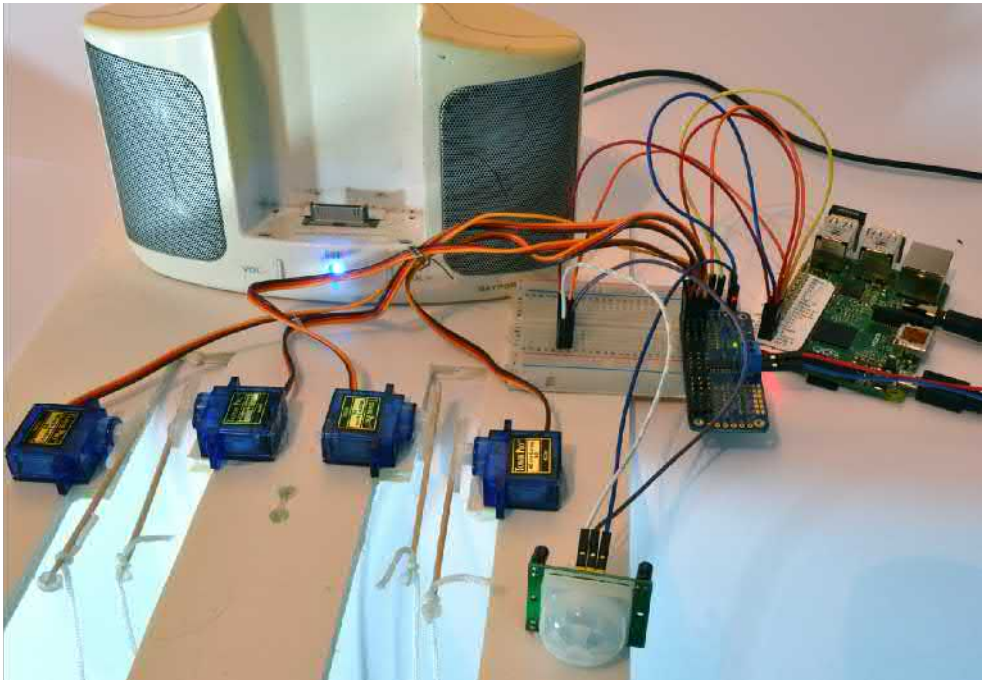


Рис. 15.10. Электронная часть проекта в сборе

Программа

Программа для этого проекта основана на программе из *разд. «Проект: танцующая кукла Пепе на Raspberry Pi» главы 9*, поэтому, возможно, вам придется вернуться к ее описанию в данной главе.

Все файлы для этого проекта можно найти в каталоге python/projects/puppet_voice. Наряду с кодом для серводвигателей компании Adafruit и самой программой (puppet_voice.py), там присутствует также звуковой файл pepe_1.wav. Именно он и будет воспроизводиться при активизации Пепе в случае обнаружения движения рядом с ним. Рассмотрим управляющую говорящей куклой программу:

```
from Adafruit_PWM_Servo_Driver import PWM
import RPi.GPIO as GPIO
from pygame import mixer
import time

PIR_PIN = 23 // 1
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_PIN, GPIO.IN)

pwm = PWM(0x40)
mixer.init() // 2
mixer.music.load("pepe_1.wav")
```

```
servoMin = 150 # Минимальная продолжительность импульса до 4096
servoMax = 600 # Максимальная продолжительность импульса до 4096

dance = [
   #lh lf rf rh
   [130, 20, 20, 130],
   [30, 160, 160, 30],
   [90, 90, 90, 90]
]

delay = 0.2

def map(value, from_low, from_high, to_low, to_high):
   from_range = from_high - from_low
   to_range = to_high - to_low
   scale_factor = float(from_range) / float(to_range)
   return to_low + (value / scale_factor)

def set_angle(channel, angle):
   pulse = int(map(angle, 0, 180, servoMin, servoMax))
   pwm.setPWM(channel, 0, pulse)

def dance_step(step):
   set_angle(0, step[0])
   set_angle(1, step[1])
   set_angle(2, step[2])
   set_angle(3, step[3])

def dance_pupet(): // 3
   for i in range(1, 10):
      for step in dance:
         dance_step(step)
         time.sleep(delay)

pwm.setPWMFreq(60)

while True:
   if GPIO.input(PIR_PIN) == True: // 4
      mixer.music.play()
      dance_pupet()
      time.sleep(2)
```

Уточним некоторые моменты программы по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Первым нововведением является код назначения контакта 23 цифровым вводом.
2. Также добавился код инициализации микшера с целью его подготовки к воспроизведению звукового клипа.

3. Новая функция `dance_pupet` используется для десятикратного повторения куклой танцевальных движений.
4. Если ИК-датчик активирован (на контакте 23 сигнал `True`), проигрывается музыкальный трек и начинается танец. Функция `music.play` выполняется в фоновом режиме.

Что еще можно сделать с говорящей куклой?

С помощью программы Audacity, которой мы с вами уже воспользовались ранее (см. *разд. «Эксперимент: воспроизведение звуковых файлов на Arduino»*), вы можете записать для произнесения куклой новый звуковой клип, — просто замените им файл pepe_1.wav. В этом проекте можно пойти еще дальше и записать несколько различных звуковых клипов, чтобы проигрывать их в произвольном порядке или в зависимости от наступившего времени суток.

Заключение

В этой главе вы узнали, как с помощью Arduino и Raspberry Pi воспроизводить различные звуки.

В заключительной главе книги мы рассмотрим возможности использования Arduino и особенно Raspberry Pi в Интернете вещей.

Интернет вещей 16

Основной способ взаимодействия людей с Интернетом — это использование браузера для просмотра веб-страниц. *Интернет вещей* (Internet of Things, или IoT), основан на концепции, что в Интернете могут присутствовать и другие «вещи». Например, подключенные к Интернету смарт-устройства домашней автоматизации, способные предоставлять полезную информацию домовладельцам и обслуживающим компаниям. Частью Интернета вещей могут считать себя и пользователи так называемых *носимых устройств*, — например, смарт-часов или фитнес-трекеров, поскольку персональная информация об их местоположении и частоте пульса уходит прямиком в облачные сервисы Интернета.

Управление электронными устройствами, когда оно производится через Интернет, обретает абсолютно новое измерение. В этой главе мы рассмотрим вопросы управления представленными в книге исполнительными устройствами с помощью веб-интерфейса.

С практической точки зрения для использования Raspberry Pi в качестве компонента Интернета вещей существуют два основных способа.

Один из них заключается в организации сети, где Raspberry Pi сможет выступить в роли веб-сервера. Тогда к предоставленному им веб-интерфейсу можно будет подключиться и наладить взаимодействие из любого браузера. К примеру, вы щелкаете по кнопке на веб-странице, которую демонстрирует вам Raspberry Pi, и включаете выход GPIO. Этот подход мы реализуем далее в *разд. «Проект: веб-выключатель на основе Raspberry Pi»*.

Второй способ предусматривает обмен данными между Raspberry Pi и облачным сервисом, предоставляющим услуги посредника для сообщений, которыми устройства и люди обмениваются через Интернет. Например, в *разд. «Проект: твиттер-партнер куклы»* (см. далее в этой главе) облачный сервис IFTTT (If This Then That, «Если это, тогда то») будет отслеживать вашу страничку в сервисе Twitter и заставлять куклу Пепе танцевать и воспроизводить звуковой клип, как только кто-нибудь пришлет вам сообщение с хештегом `#dàncepepe`.

Если вы не захотите привлечь Twitter, вместо него можно будет воспользоваться электронной почтой, записью в Facebook — да практически любой разновидностью

сетевой активности, способной взаимодействовать с сервисом IFTTT (**http://www.ifttt.com**).

Raspberry Pi и среда Bottle

Bottle представляет собой облегченную и весьма несложную в использовании веб-серверную среду, целиком написанную на языке Python. Она прекрасно подходит для создания простых веб-серверных приложений для Raspberry Pi.

Чтобы установить Bottle, введите следующие команды:

```
$ sudo apt-get update
$ sudo apt-get install python-bottle
```

Затем нужно создать Python-программу, использующую Bottle для реализации простого веб-сервера, к которому можно будет подключиться с браузера из любого места вашей сети. Введите для создания нового файла Python команду:

```
$ nano test_bottle.py
```

И добавьте к файлу следующий текст:

```
from bottle import route, run

@route('/')
def index():
   return '<h1>Hello World</h1>'

run(host='0.0.0.0', port=80)
```

Маркер `@route` перед функцией `index` показывает, что `index` отвечает за создание кода HTML, отправляемого любому браузеру, посещающему корневой каталог веб-сервера (сам веб-сайт, а не его конкретную страницу). В данном случае браузеру будет возвращен только текст `Hello World`, отформатированный в виде заголовка первого уровня. Чтобы проверить программу в работе, запустите следующую команду:

```
$ sudo python test_bottle.py
Bottle server starting up (using WSGIRefServer())...
Listening on http://0.0.0.0:80/
Hit Ctrl-C to quit.
```

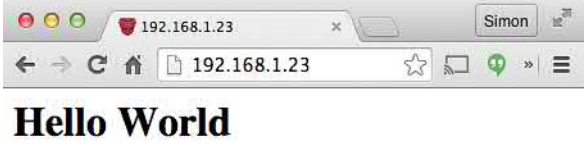


Рис. 16.1. Проверка работы среды Bottle

Затем откройте браузер либо непосредственно в Raspberry Pi, либо на любом компьютере вашей сети, и воспользуйтесь в качестве интернет-адреса (URL) IP-адресом своего Raspberry Pi (рис. 16.1). Чтобы определить IP-адрес Raspberry Pi, обратитесь к врезке *«Поиск IP-адреса Raspberry Pi»* в *главе 3*.

Проект: веб-выключатель на основе Raspberry Pi

Поскольку веб-сервер Bottle является всего лишь программой на языке Python, он может не только поставлять код HTML для браузера, но и использовать библиотеку `RPi.GPIO` для управления сигналами на контактах ввода/вывода GPIO в ответ на щелчки по гиперссылкам и кнопкам веб-интерфейса обслуживаемого им браузера (рис. 16.2).

Вы можете, например, воспользоваться этими возможностями, чтобы управлять описанным в *разд. «Проект: реле времени на основе Raspberry Pi» главы 13* устройством PowerSwitch Tail для включения/выключения тех или иных электроприборов домашней электросети.

Рис. 16.2. Веб-интерфейс для проекта интернет-выключателя

Оборудование

В этом проекте нас интересует веб-интерфейс, который будет устанавливать из браузера высокий или низкий уровень на контакте 18 GPIO. Выбор устройства, подключаемого к контакту 18, отдается целиком на ваше усмотрение. Вы можете, ничего не усложняя, просто воспользоваться светодиодом. В таком случае соберите конструкцию из *разд. «Эксперимент: управление светодиодом» главы 4*. Можно также предложить контакту управлять устройством PowerSwitch Tail с подключенным к нему каким-либо бытовым прибором, как это было сделано в *разд. «Проект: реле времени на основе Raspberry Pi» главы 13*.

Программа

Среда Bottle предоставляет интересный способ отделения кода HTML, обслуживающего браузер, от программной логики языка Python, управляющей какими-либо

устройствами. Этот механизм основан на *использовании шаблонов*. Для нашего проекта потребуются два файла, которые находятся в каталоге projects/pr_web_switch/.

В файле-шаблоне home.tpl содержится код HTML, предназначенный для создания веб-интерфейса:

```
<html>
<body>

<h1>Web Switch</h1>

<a href="/on">ON</a>

<a href="/off">OFF</a>

</body>
</html>
```

При просмотре в браузере этот код формирует страницу, показанную на рис. 16.2. Ключевыми строками в нем являются два тега с гиперссылками. В атрибуте `href` этих тегов указывается веб-адрес перехода при щелчке на надписи **ON** или **OFF**. Если вы знакомы с языком HTML и CSS-стилями, можете сделать эти ссылки более удобными и придать им вид, похожий на настоящие кнопки.

При щелчке на гиперссылке **ON** или **OFF** на IP-адрес вашего Raspberry Pi отправляется веб-запрос, содержащий в конце строку символов: `/on` или `/off` соответственно. Этот запрос обрабатывается следующим кодом на языке Python (из файла web_switch.py):

```
from bottle import route, run, template, request
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM) // 1
CONTROL_PIN = 18
GPIO.setup(CONTROL_PIN, GPIO.OUT)

@route('/') // 2
def index():
   return template('home.tpl')

@route('/on') // 3
def index():
   GPIO.output(CONTROL_PIN, 1)
   return template('home.tpl')

@route('/off') // 4
def index():
   GPIO.output(CONTROL_PIN, 0)
   return template('home.tpl')
```

```
try:
   run(host='0.0.0.0', port=80) // 5

finally:
   print('Сброс GPIO')
   GPIO.cleanup()
```

Уточним некоторые моменты кода по пунктам, воспользовавшись разметкой строк, сделанной в комментариях:

1. Назначение контакта `18` в качестве выходного контакта сигнала управления.
2. Если браузер посещает корневой каталог этого веб-сервера, то возвращается просто содержимое шаблона `home.tpl`.
3. Обработчик URL-адреса с окончанием `on` (в этом случае перед возвращением шаблона главной страницы на контакте управления устанавливается сигнал высокого уровня).
4. Обработчик гиперссылки `OFF`.
5. Запуск веб-сервера, работающего с использованием порта `80` (этот порт используется для веб-страниц по умолчанию).

Загружаем и выполняем программу

Для запуска веб-сервера воспользуйтесь следующей командой:

```
$ sudo python web_switch.py
```

Затем откройте на вкладке браузера страницу, указав IP-адрес вашего Raspberry Pi, и убедитесь, что в браузере отобразилась страница, показанная на рис. 16.2.

Попробуйте пощелкать по гиперссылкам **ON** и **OFF** — при этом устройство, подключенное к контакту GPIO 18, должно включаться и выключаться.

Arduino и сети

За счет встроенного сетевого интерфейса и с учетом доступности дешевых USB модулей Wi-Fi, Raspberry Pi является гораздо более подходящим устройством для IoT-проектов, чем Arduino Uno.

Вы, конечно, можете приобрести к своему Arduino шилд Wi-Fi, но это не слишком дешевое удовольствие. В некоторые модели Arduino — например, в Arduino Yun, встроен модуль Wi-Fi, но эти модели также весьма недешевы да и не так просты в использовании.

Но если вам в IoT-проектах все же нужно задействовать какое-либо Arduino-подобное устройство с возможностями Wi-Fi, я рекомендую воспользоваться чем-нибудь вроде платы Photon компании Particle.io (рис. 16.3).

Плата Photon стилизована под версию Arduino Nano, но имеет встроенный модуль Wi-Fi. Относится она к устройствам, полностью предназначенным для работы

в облачной среде, — связь с ней и установка на нее программ может осуществляться через Интернет. Это означает, что плату Photon можно встроить в проект и программировать ее работу, не имея к ней физического доступа, — нужно лишь подвести к плате питание и подключить ее к вашей сети Wi-Fi.

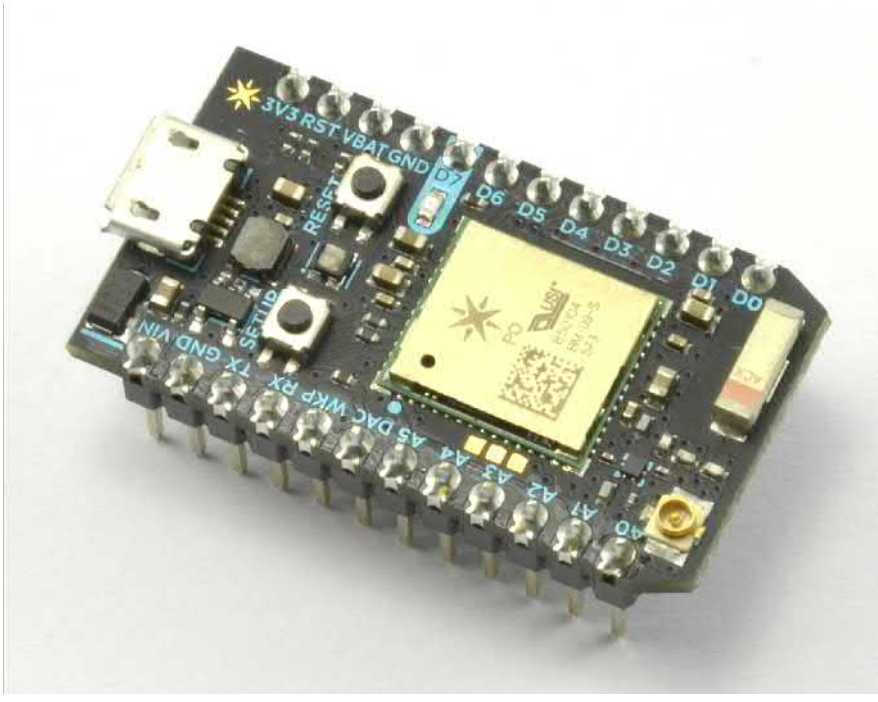


Рис. 16.3. Плата Photon

Язык программирования для Photon также основан на Arduino C, но вместо стандартного Arduino IDE, программирование Photon ведется из IDE-среды, созданной на основе веб-технологий. Дополнительные сведения по работе с платой Photon можно найти на сайте **particle.io** или в моей книге «Make: Getting Started with the Photon».

Еще одним полезным Arduino-подобным устройством, которое также используется в IoT-проектах, является модуль ESP8266. Этот модуль (рис. 16.4) продается по фантастически низкой цене и может быть настроен на программирование из среды Arduino IDE, как будто он и есть Arduino. Впрочем, его можно просто подсоединить к плате Arduino, чтобы обеспечить ей подключение Wi-Fi.

С использованием этого устройства все сильно упрощается, но на момент подготовки книги оно все еще требовало множества различных настроек на правильную работу. Если есть желание ознакомиться с ним поглубже, поищите информацию во Всемирной паутине по запросу `ESP8266` и изучите одно из найденных таким образом руководств для начинающих, — например, **http://makezine.com/2015/04/01/esp8266-5-microcontroller-wi-fi-now-arduino-compatible/**.

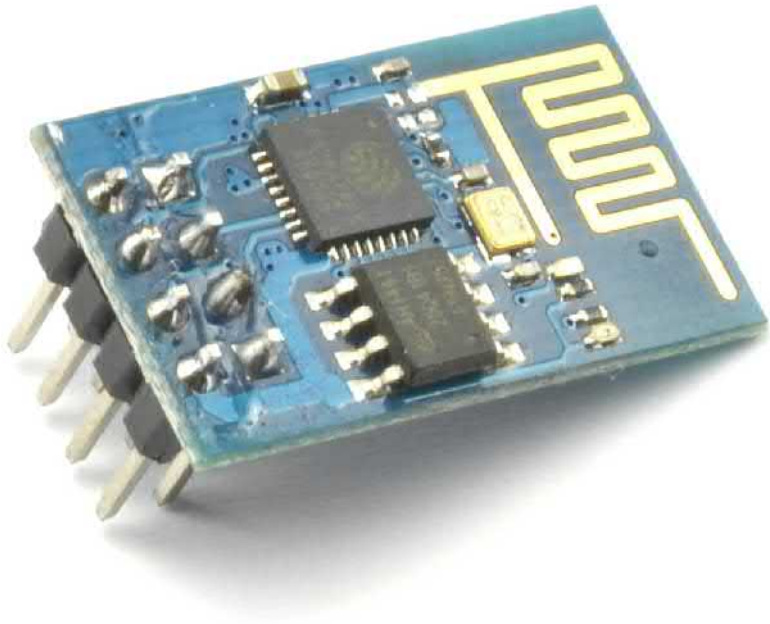


Рис. 16.4. Модуль ESP8266

Проект: твиттер-партнер куклы

Давайте на основе конструкции из *разд. «Проект: кукла Пепе обретает голос» главы 15* заставим Пепе реагировать на твиты с хештегом `#dancepepe`, исполняя небольшой танец и проигрывая звуковой клип. В нашем новом проекте может использоваться то же самое оборудование, что и в *разд. «Проект: кукла Пепе обретает голос» главы 15*, только без ИК-датчика (рис. 16.5). Изменится лишь программное обеспечение.

Подключение Пепе к Интернету

Процесс превращения Пепе в куклу, реагирующую на твиты, осуществляется в два этапа. На первом этапе Пепе будет позволено отвечать на веб-запросы, и вы сможете включать и выключать его танец из веб-браузера. На втором этапе мы привлечем веб-сервис IFTTT для отслеживания твитов на наличие хештега `#dancepepe`, при обнаружении которого Пепе, подготовленному на первом этапе, будет отправляться веб-запрос, на который он не сможет не отреагировать.

Чтобы Пепе отвечал на веб-события, мы воспользуемся веб-сервисом **dweet.io**. Это бесплатный сервер (при условии разумного ограничения на количество отправляемых в течение одного месяца сообщений), который позиционирует себя в качестве обработчика твитов для IoT. Он не требует регистрации и весьма прост в использовании. У него также имеется библиотека Python, позволяющая легко и просто объединить его с программой для Пепе.

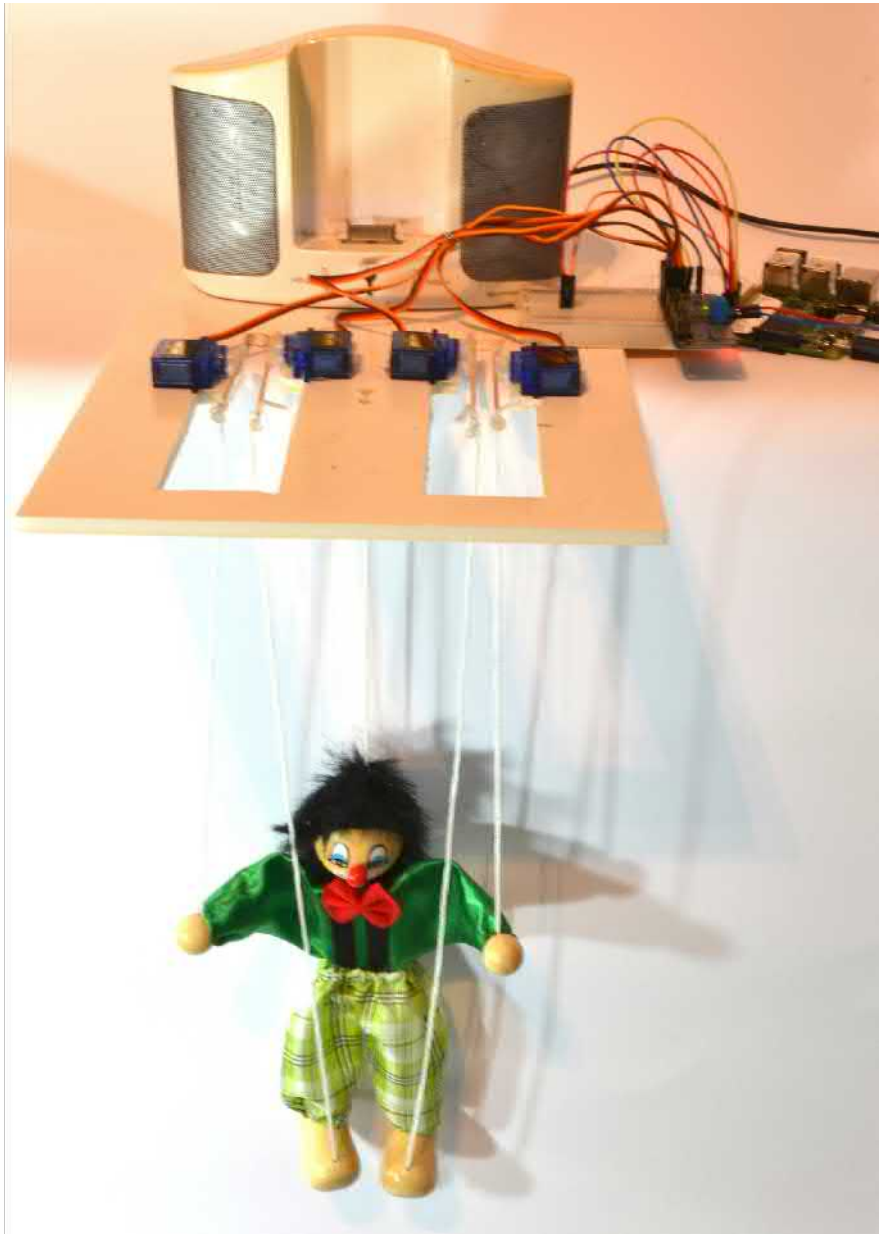


Рис. 16.5. Схема проекта куклы, реагирующей на твиты, в сборе

Для установки dweepy, библиотеки языка Python для **dweet.io**, выполните следующие команды:

```
$ git clone git://github.com/paddycarey/dweepy.git
$ cd dweepy
$ sudo python setup.py install
```

Небольшая проблема, связанная с применением этой библиотеки для Python 2 и SSL, легко решается выполнением следующих команд, предназначенных для изменения версии HTTP-запросов, используемых в Python 2:

```
$ sudo apt-get install python-pip
$ sudo pip install requests==2.5.3
```

Теперь программу puppet_web.py для этого проекта, находящуюся в каталоге python/projects/puppet_web, можно запустить с помощью следующей команды:

```
$ sudo python puppet_web.py
```

Отследить ход выполнения проекта можно с помощью веб-браузера. Для этого надо перейти по адресу: **https://dweet.io/dweet/for/pepe_the_puppet** (рис. 16.6).

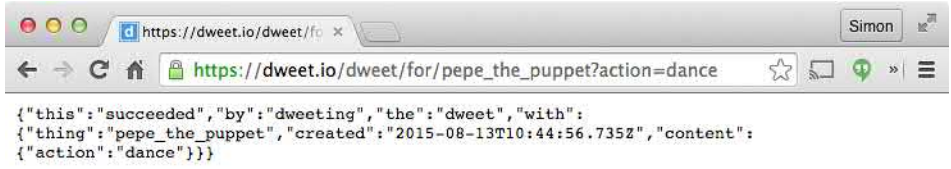


Рис. 16.6. Управление куклой из веб-браузера

Как только будет осуществлен переход по этому адресу, Пепе должен начать свое выступление.

Во многом программа из файла puppet_web.py похожа на ту, что использовалась в *разд. «Проект: кукла Пепе обретает голос» главы 15*, так что за разъяснениями по основной части кода вы можете обратиться к этому проекту. А здесь мы поясним лишь ту часть кода, которая имеет касательство к веб-сервису **dweet.io**:

```
from Adafruit_PWM_Servo_Driver import PWM
from pygame import mixer
import time
import dweepy // 1

pwm = PWM(0x40)
mixer.init()
mixer.music.load("pepe_1.wav")

dweet_key = 'pepe_the_puppet' // 2

servoMin = 150 # Минимальная продолжительность импульса до 4096
servoMax = 600 # Максимальная продолжительность импульса до 4096

dance = [
   #lh lf rf rh
   [130, 20, 20, 130],
```

```
    [30, 160, 160, 30],
    [90, 90, 90, 90]
]

delay = 0.2

def map(value, from_low, from_high, to_low, to_high):
    from_range = from_high - from_low
    to_range = to_high - to_low
    scale_factor = float(from_range) / float(to_range)
    return to_low + (value / scale_factor)

def set_angle(channel, angle):
    pulse = int(map(angle, 0, 180, servoMin, servoMax))
    pwm.setPWM(channel, 0, pulse)

def dance_step(step):
    set_angle(0, step[0])
    set_angle(1, step[1])
    set_angle(2, step[2])
    set_angle(3, step[3])

def dance_pupet():
    for i in range(1, 10):
        for step in dance:
            dance_step(step)
            time.sleep(delay)

pwm.setPWMFreq(60)

while True:
    try: // 3
        for dweet in dweepy.listen_for_dweets_from(dweet_key): # // 4
            print("Танцуй, Пепе! Танцуй!")
            mixer.music.play()
            dance_pupet()
except Exception:
    pass
```

Уточним элементы кода:

1. Импортируем библиотеку `dweepy`.
2. Сервис `dweet` использует это значение в качестве ключевого для сообщений в `dweet`, которыми вы интересуетесь. Если этот ключ оставить неизменным, то другие читатели этой книги, также работающие над этим проектом, смогут оживить вашу куклу (и наоборот, вы сможете оживить их куклу), что сделает проект еще интереснее. Если вам не нужна такая открытость, выберите для этого ключа другое значение.

3. Весь код внутри основного цикла содержится в обработчике ошибок `try...except`, потому что веб-подключение, к которому программа прислушивается время от времени, выдает исключение. Код `try...except` маскирует эту выдачу исключения, позволяя программе совершать новую попытку выполнения после таких ошибок.
4. Сервис **dweet.io** позволяет находиться в режиме ожидания и отслеживать новые dweet-сообщения на предмет наличия вашего ключа, а также выполнять действия при поступлении таких сообщений.

Веб-сервис IFTTT

IFTTT — веб-сервис, позволяющий настраивать *инициаторы* (trigger), вызывающие *действие* (action). Можно, например, создать IFTTT-рецепт, отправляющий ваше сообщение электронной почты (действие) при каждом упоминании о вас в Твиттере (инициатор).

В нашем проекте сервис IFTTT будет отслеживать вашу страничку в сервисе Twitter и заставлять куклу Пепе танцевать и воспроизводить звуковой клип, как только кто-нибудь пришлет вам сообщение с хештегом `#dancepepe`.

Для настройки IFTTT на эту работу следует выполнить следующие действия.

Шаг 1. Создайте новый рецепт

Подпишитесь на использование IFTTT-сервиса (подписка бесплатная). Затем щелкните на кнопке **Create Recipe** (Создать рецепт) — вам будет показана страница, изображенная на рис. 16.7.

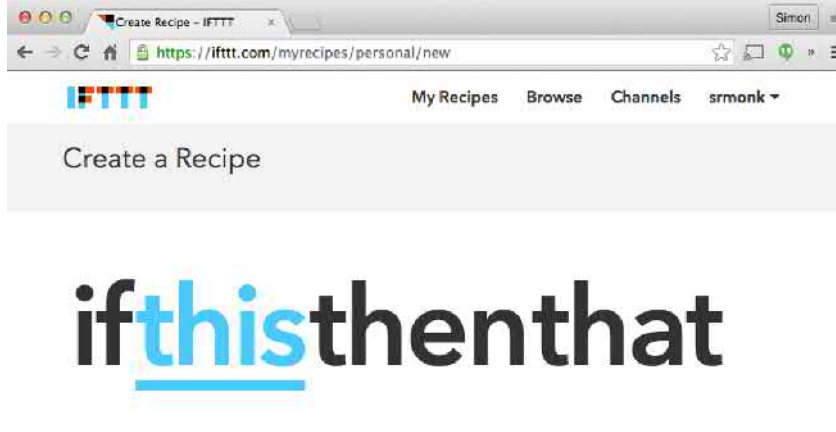


Рис. 16.7. Создание в IFTTT нового рецепта

Шаг 2. Определите инициатор

Щелкните по большой гиперссылке **this**, а затем в списке каналов найдите канал **Twitter**. Найдите внутри этого канала триггер **New tweet from search**, а затем введите `#dancepepe` в поле **Search for** (Искать) (рис. 16.8).

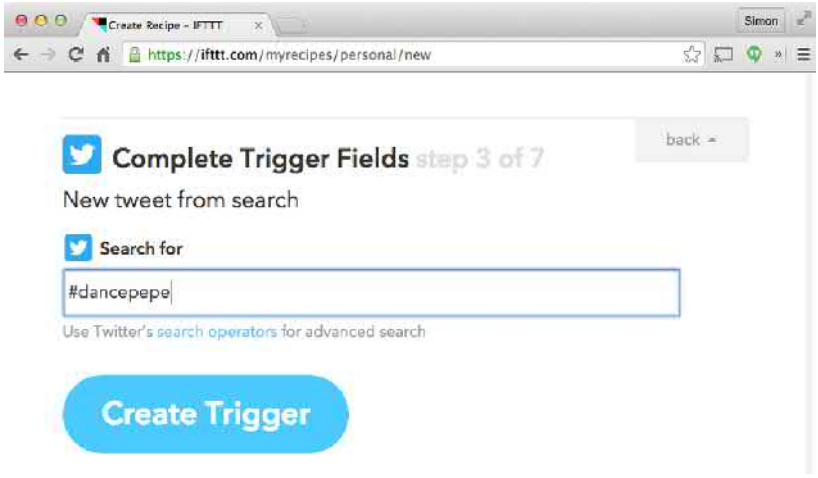


Рис. 16.8. Настройка инициатора Twitter

Шаг 3. Добавьте действие в виде веб-запроса

После определения инициатора снова должно открыться окно **ifthisthenthat**, в котором теперь нужно будет выбрать действие, щелкнув по гиперссылке **that**. Далее найдите канал действия **Maker**, выберите единственное доступное действие **Make a web request** (Создать веб-запрос) и заполните форму, показанную на рис. 16.9.

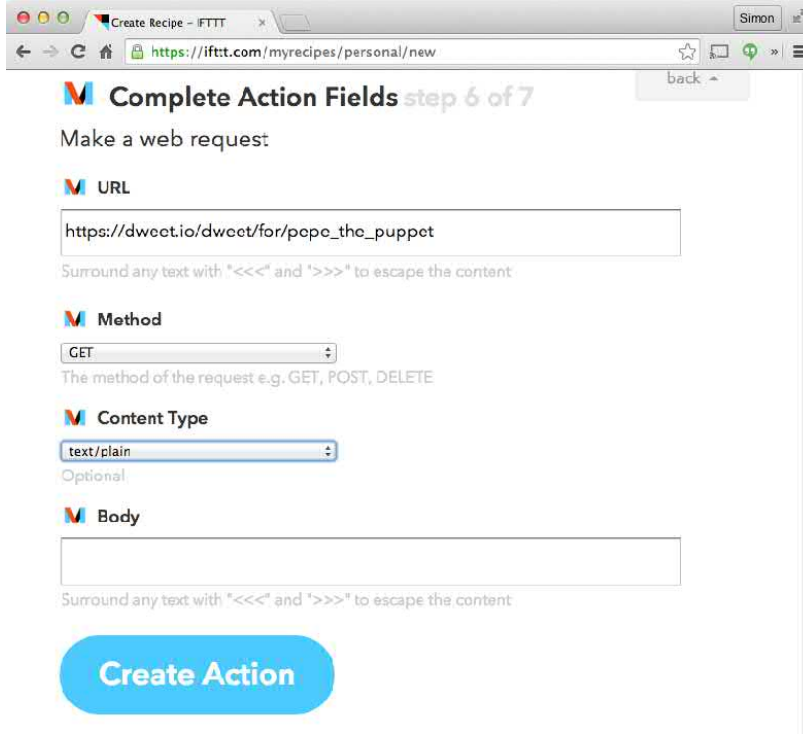


Рис. 16.9. Заполнение формы действия

Шаг 4. Завершите создание рецепта

Щелкните на кнопке **Create Action** (Создать действие), после чего подтвердите завершение рецепта, щелкнув на кнопке **Create Recipe** (Создать рецепт) (рис. 16.10).

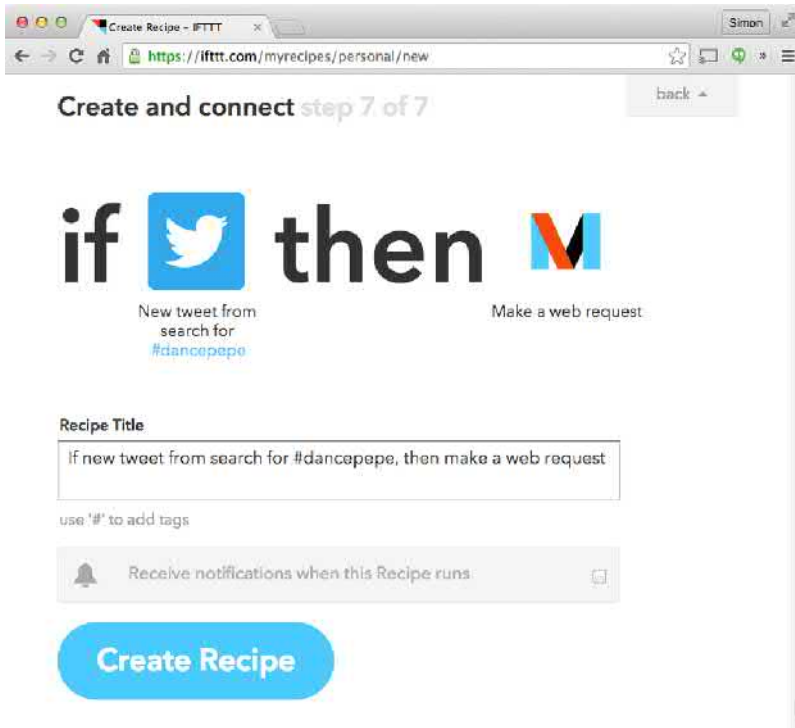


Рис. 16.10. Завершение создания рецепта

Сразу же после завершения создания рецепта, он автоматически будет активирован. В ходе этого процесса IFTTT может попросить вас авторизоваться на сервисе Twitter.

Работа с проектом

Итак, можно запускать проект в работу. Протестируйте его, создав твит с хештегом `#dancepepe`. Чтобы IFTTT нашел твит, нужно некоторое время, поэтому не волнуйтесь, если ничего не будет происходить в течение одной-двух минут.

Если по истечении этого времени так ничего и не случится, надо проверить IFTTT-журнал вашего рецепта, который вызывается щелчком на значке на странице вашего рецепта, похожем на маркированный список. Запись покажет, был ли инициирован рецепт, а также любые возникшие при этой попытке ошибки.

Если вы захотите записать для Пепе свой собственный музыкальный фрагмент, обратитесь к *разд. «Создание звукового файла» главы 15*.

Можно также воспользоваться и другими инициаторами, доступными в IFTTT, поскольку интересных возможностей по запуску Пепе там бесконечно много!

Заключение

В этой главе мы рассмотрели способы запуска веб-сервера на Raspberry Pi, а также использование для реализации IoT-проектов таких веб-сервисов, как IFTTT и **dweet.io**.

Надеюсь, что эта книга послужила для вас источником полезной информации и подсказала ряд идей для серьезных проектов, к реализации которых вы теперь сможете приступить.

Приложение 1. Комплектующие

1

Отыскать комплектующие для проектов порой бывает весьма сложно. В этом приложении указаны места, где можно приобрести различные детали для проектов из книги, а также дополнительная полезная информация, — например, схемы расположения контактов некоторых компонентов.

Поставщики

Сегодня существует множество поставщиков электроники, чья целевая аудитория — такие мастера, как мы с вами. Некоторые наиболее популярные из них представлены в табл. П1.1.

Как можно видеть, большинство поставщиков продают детали для Arduino Uno и Raspberry Pi 3, — именно с такими устройствами рекомендует работать эта книга.

Набор для экспериментов с Arduino компании Adafruit (код товара 170) и набор для ардуинщика-любителя компании Sparkfun (KIT-11227) — хорошие варианты для начала, поскольку в них имеются все базовые компоненты Arduino. А комплект MonkMakes Basic Components включает большинство резисторов, конденсаторов, транзисторов и светодиодов, упомянутых в этой книге.

Таблица П1.1. *Поставщики*

Поставщик	Сайт	Примечания
Adafruit	**www.adafruit.com**	Хороший вариант для работы с модулями
Digikey	**www.digikey.com**	Широкий ассортимент комплектующих
MakerShed	**www.makershed.com**	Удобные варианты модулей, наборов и инструментов
MCM	**www.mcmelectronics.com**	Широкий ассортимент комплектующих
Mouser	**www.mouser.com**	Широкий ассортимент комплектующих
SeeedStudio	**www.seeedstudio.com**	Интересные недорогие модули
SparkFun	**www.sparkfun.com**	Хороший вариант для работы с модулями

Таблица П1.1 (окончание)

Поставщик	Сайт	Примечания
CPC	**cpc.farnell.com**	Британская компания, широкий ассортимент комплектующих
Farnell	**www.farnell.com**	Международная компания, широкий ассортимент комплектующих
Maplins	**www.maplin.co.uk**	Британская компания, небольшие магазины
Proto-pic	**proto-pic.co.uk**	Британская компания, стоковые модули SparkFun и Adafruit
Pimoroni	**shop.pimoroni.com**	Специализируется на Raspberry Pi
MonkMakes	**www.monkmakes.com**	Наборы комплектующих и специальные подборки деталей для проектов, описанных в книгах Саймона Монка
Амперка	**amperka.ru**	Российский магазин для гиков, ассортимент невелик, только основные комплектующие
Чип и Дип	**www.chipdip.ru**	Российский магазин радиоэлектронных компонентов, измерительной техники, паяльного оборудования, инструментов и товаров для хобби в области электроники. Очень широкий ассортимент

Резисторы и конденсаторы

Упомянутые в этой книге резисторы и конденсаторы, а также источники, где их можно приобрести, представлены в табл. П1.2.

Таблица П1.2. Резисторы и конденсаторы

Компонент схемы	Источники
Резистор 10 Ом 0,25 Вт	Mouser: 291-10-RC
Резистор 100 Ом 0,25 Вт	Mouser: 291-100-RC
Резистор 150 Ом 0,25 Вт	Mouser: 291-150-RC
Резистор 270 Ом 0,25 Вт	Mouser: 291-270-RC
Резистор 470 Ом 0,25 Вт	Mouser: 291-470-RC
Резистор 1 кОм 0,25 Вт	Mouser: 291-1k-RC
Резистор 4,7 кОм 0,25 Вт	Mouser: 291-4.7k-RC
Подстроечный резистор на 10 кОм	Adafruit: 356 Sparkfun: COM-09806
Фоторезистор 1 кОм	Adafruit: 161 Sparkfun: SEN-09088

Таблица П1.2 (окончание)

Компонент схемы	Источники
Конденсатор 100 нФ	Adafruit: 753 Mouser: 810-FK16X7R2A224K
Конденсатор 16 В 100 мкФ	Adafruit: 2193 Sparkfun: COM-00096 Mouser: 647-UST1C101MDD

Полупроводниковые компоненты и светодиоды

Детали с кодами вроде 2N3904 искать удобно, но когда возникает необходимость в самых обычных компонентах, например в светодиодах, оказывается, что их найти уже сложнее. В таких случаях лучше искать специализированный набор светодиодов на eBay или Amazon.com, либо посмотреть, что есть в стартовом наборе.

Упомянутые в этой книге полупроводниковые компоненты и светодиоды, а также источники, где их можно приобрести, представлены в табл. П1.3.

Таблица П1.3. Полупроводниковые компоненты и светодиоды

Компонент схемы	Источники
Транзистор 2N3904	Adafruit: 756 Sparkfun: COM-00521 Mouser: 610-2N3904
Составной транзистор MPSA14	Mouser: 833-MPSA14-AP
МОП-транзистор 2N7000	Mouser: 512-2N7000
Составной транзистор TIP120	Adafruit: 976 Mouser: 512-TIP120
МОП-транзистор FQP30N06L	Mouser: 512-FQP30N06L
Диод 1N4001	Adafruit: 755 Sparkfun: COM-08589 Mouser: 512-1N4001
Красный светодиод	Adafruit: 297 Sparkfun: COM-09590
Зеленый светодиод	Adafruit: 298 Sparkfun: COM-09650

Таблица П1.3 *(окончание)*

Компонент схемы	Источники
Желтый светодиод	Sparkfun. COM-09594
RGB-светодиод	Sparkfun. COM-11120
Микросхема с Н-мостом L293D	Adafruit. 807 Mouser 511-L293D
Модуль L298N с Н-мостом	Mouser 511-L298
Микросхема ULN2803	Adafruit. 970 Mouser: 511-ULN2803A
Микросхема цифрового термометра DS18B20	Adafruit 374 (включает резистор на 4,7 кОм)
Герметичный температурный датчик DS18B20	Adafruit 381 eBay

Оборудование

Макетную плату и подборку проводов-перемычек, пожалуй, лучше всего взять в одном из наборов для начинающих от поставщиков, упомянутых в *разд. «Поставщики»*.

Упомянутые в этой книге макетная плата и провода-перемычки, а также источники, где их можно приобрести, представлены в табл. П1.4.

Таблица П1.4. *Оборудование*

Компонент схемы	Источники
400-точечная беспаечная макетная плата	Adafruit. 64
Перемычки «папа-папа»	Adafruit 758
Перемычки «мама-мама»	Adafruit. 266
Монтажные провода	Adafruit 1311
Перемычки «мама-папа»	Adafruit 826
Батарейный отсек 2 AA (3 В)	Adafruit 770
Батарейный отсек 4 AA (6 В)	Adafruit 830
Переходник с круглым гнездом и винтовыми зажимами	Adafruit 368
Двунаправленная клеммная колодка	Магазин строительных или электротоваров

Прочее

Упомянутые в этой книге прочие компоненты, а также источники, где их можно приобрести, представлены в табл. П1.5. Многие такие компоненты легко найти на eBay или Amazon.

Таблица П1.5 *Прочее*

Компонент схемы	Источники
Небольшой двигатель постоянного тока с напряжением питания 6 В	Adafruit: 711
Биполярный шаговый двигатель 12 В	Adafruit: 324
Униполярный шаговый двигатель 5 В	Adafruit 858
Серводвигатель 9g	eBay Adafruit. 169
Твердотельное реле PowerSwitch Tail	Adafruit: 268
Источник питания 5 В 2 А	Adafruit: 276
Источник питания 12 В 1 А	Adafruit: 798
Источник питания 12 В 5 А	Adafruit: 352
Плата EasyDriver для регулировки шаговых двигателей	Sparkfun: ROB-12779
16-канальный 12-битный модуль PWM/Servo	Adafruit: 815
Небольшой громкоговоритель с импедансом 8 Ом	Adafruit: 1891
Адресуемая светодиодная лента WS2812	Adafruit: 1376 eBay
OLED-дисплей I^2C с разрешением 128×64 пикселов	eBay
Пассивный инфракрасный (ИК) датчик (Passive infrared, PIR)	Adafruit: 189 eBay
Релейный модуль	eBay

Схемы расположения выводов

На рис. П1.1 показаны схемы расположения выводов для некоторых компонентов, используемых в книге.

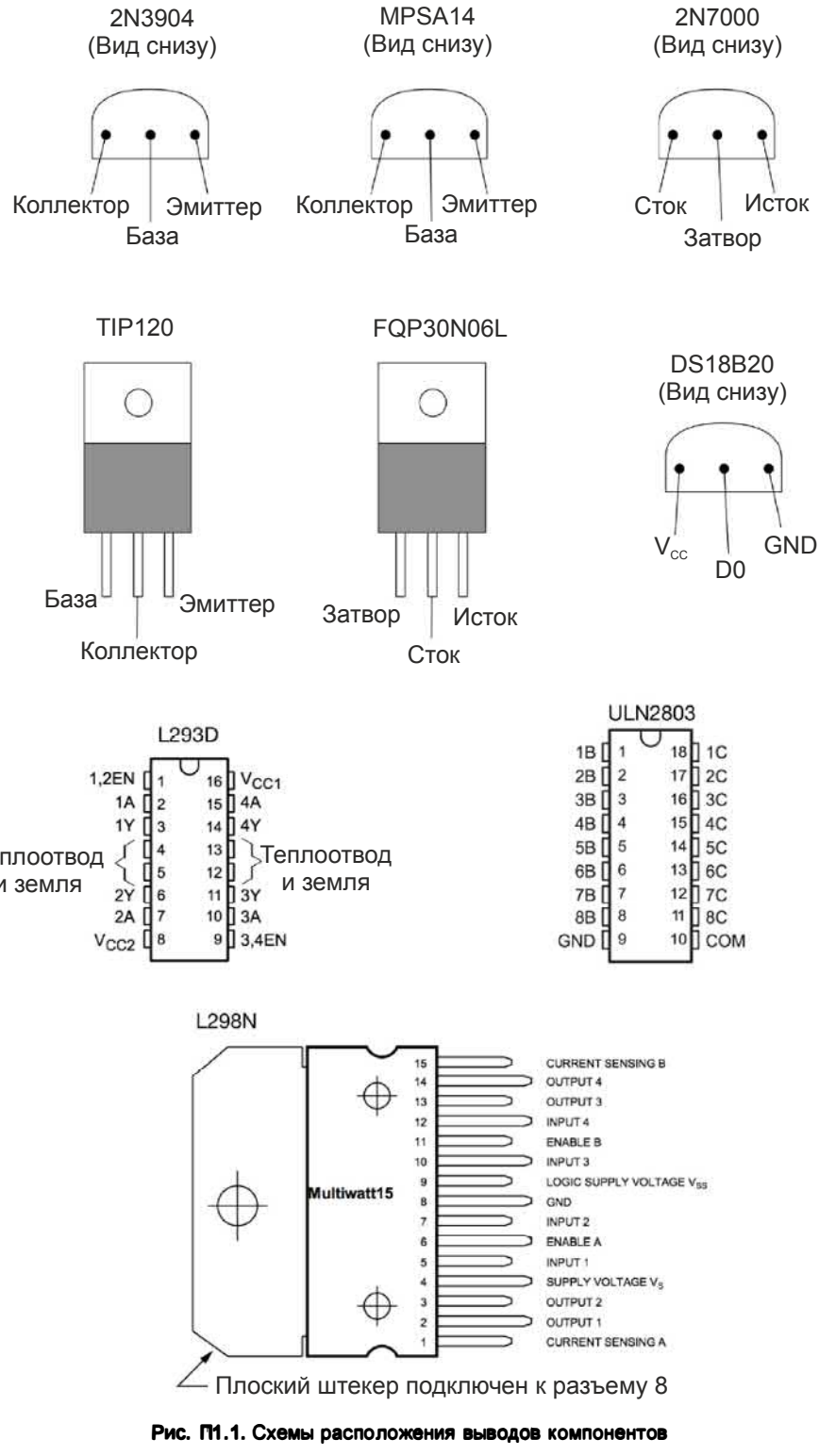


Рис. П1.1. Схемы расположения выводов компонентов

Приложение 2. Схема контактов GPIO Raspberry Pi

2

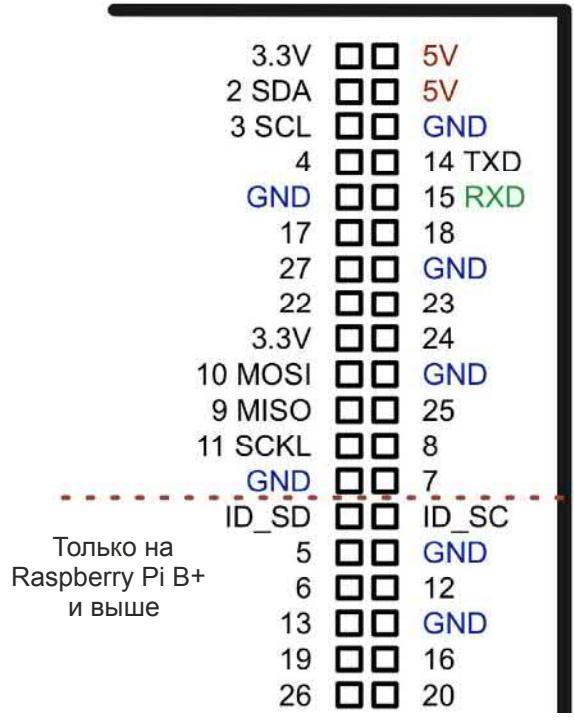


Рис. П2.1. Схема контактов Raspberry Pi

Примечания

- Интегральные схемы I^2C на контактах 2 и 3.
- Последовательный периферийный интерфейс на контактах 9, 10 и 11.
- Последовательный интерфейс TTL на контактах 14 и 15.
- Контакты ID_SD и ID_SC зарезервированы для использования с последовательным интерфейсом EEPROM для идентификации подключаемых плат (HAT).

Предметный указатель

L

N

O

P

R

S

T

U

Ω

К

Л

М

Н

О

П

Р

С

Т

У

Ф

Ц

Ш

Э

Я