ARDUINO ДЛЯ ИЗОБРЕТАТЕЛЕЙ

ОБУЧЕНИЕ ЭЛЕКТРОНИКЕ НА 10 ЗАНИМАТЕЛЬНЫХ ПРОЕКТАХ

БРАЙАН ХУАНГ, ДЕРЕК РАНБЕРГ

sparkfun

bhv®

no starch press

ARDUINO ДЛЯ ИЗОБРЕТАТЕЛЕЙ
Б. ХУАНГ
Д. РАНБЕРГ
bhv®
no starch press

THE ARDUINO INVENTOR'S GUIDE

LEARN ELECTRONICS BY MAKING 10 AWESOME PROJECTS

BY BRIAN HUANG
AND DEREK RUNBERG

no starch
press

SAN FRANCISCO

ARDUINO
ДЛЯ ИЗОБРЕТАТЕЛЕЙ

ОБУЧЕНИЕ ЭЛЕКТРОНИКЕ НА 10 ЗАНИМАТЕЛЬНЫХ ПРОЕКТАХ

БРАЙАН ХУАНГ, ДЕРЕК РАНБЕРГ

Санкт-Петербург
«БХВ-Петербург»
2019

УДК 004
ББК 32.973.26
Х98

Хуанг, Б.

Х98 Arduino для изобретателей. Обучение электронике на 10 занимательных проектах: Пер. с англ. / Б. Хуанг, Д. Ранберг. — СПб.: БХВ-Петербург, 2019. — 288 с.: ил.

ISBN 978-5-9775-3972-2

В книге подробно рассмотрено 10 занимательных проектов с платой Arduino Uno (светофор, светодиодный экран, светочувствительный ночник, мини-теплица, мобильный робот, миниатюрное пианино и др.). Описаны принципы работы и взаимодействие различных электронных компонентов, чтение принципиальных и монтажных схем, создание и тестирование прототипов с помощью беспаечной макетной платы. Показано, как собирать электрические схемы, разрабатывать программный код и создавать готовые конструкции. В каждом проекте приведены советы по его модификации и расширению возможностей. Приведены шаблоны корпусов и деталей, а также пошаговые фотографии их изготовления и сборки. На сайте издательства находятся исходные коды примеров из книги, шаблоны для конструкций проектов, а также коды для дальнейшего экспериментирования с проектами.

Для радиолюбителей

УДК 004
ББК 32.973.26

Группа подготовки издания:

Руководитель проекта	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Компьютерная верстка	*Людмилы Гауль*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.07.18.
Формат 84×108$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 30,24.
Тираж 1500 экз. Заказ № 5226/18.
«БХВ-Петербург», 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано в соответствии с предоставленными материалами в ООО «ИПК Парето-Принт»,
170546, Тверская область, Промышленная зона Боровлево-1, комплекс № 3А, www.pareto-print.ru

ISBN 978-1-59327-652-2 (англ.)
ISBN 978-5-9775-3972-2 (рус.)

ПОСВЯЩАЕТСЯ

ЛИНДСИ ДАЙМОНД — ТЫ НАШ ЗАМЕЧАТЕЛЬНЫЙ РУКОВОДИТЕЛЬ И НАСТАВНИК!

ВСЕЙ КОМАНДЕ ИНТЕРНЕТ-МАГАЗИНА SPARKFUN!

ВСЕМ ЛЮБИТЕЛЯМ САМОДЕЛЬНЫХ КОНСТРУКЦИЙ И ИЗОБРЕТАТЕЛЯМ,
КОТОРЫХ МЫ СМОГЛИ ВДОХНОВИТЬ НА ТВОРЧЕСТВО!

О КОМПАНИИ SPARKFUN ELECTRONICS

Интернет-магазин розничной торговли SparkFun Electronics производит и продает устройства и компоненты, которые используются во многих любительских проектах, промышленных образцах и даже на Международной космической станции.

Мысль создать такую компанию пришла в 2003 году студенту Колорадского университета (г. Боулдер) Натану Зайдлю (Nathan Seidle), когда он сжег печатную плату. В те времена достать печатную плату было весьма трудной задачей. Для этого нужно было передать по факсу номер своей кредитной карточки в компанию, находящуюся в другой стране, после чего ожидать получения заказанной платы в течение шести-восьми недель. Натан решил, что он может сделать этот процесс более удобным, и задался целью создать компанию по поставке электронных деталей, устройств, компонентов и пр. Так и появилась компания SparkFun.com, которая в настоящее время предлагает свыше 3 тыс. различных наименований деталей всевозможных параметров и функционала для использования в цифровых электронных проектах. От базовых плат Arduino до модулей GPS — все это, вместе со всей необходимой документацией, доступно для покупки любому желающему в магазине SparkFun.

Отдел образования компании SparkFun разрабатывает курсы и учебные планы для студентов профильных специальностей, а также предлагает программы повышения профессионального уровня для преподавателей, занятых в сфере электроники и информационных технологий. Этот отдел является зачинателем многих инициатив в области вычислительной техники и любительских проектов, которые с успехом применяются в обучении.

Дополнительная информация о компании SparkFun и ее отделе образования доступна в Интернете на сайтах **https://www.sparkfun.com/** и **http://www.sparkfuneducation.com/**.

ОБ АВТОРАХ

В одно время Брайан Хуанг (Brian Huang) и Дерек Ранберг (Derek Runberg) работали учителями. Брайан преподавал физику в средней школе и одновременно увлекался робототехникой, а Дерек работал учителем технического образования в восьмилетней школе — его любимым коньком было добиться от учеников максимального использования ими своих возможностей.

Они пришли к своему сегодняшнему статусу разными путями, разнятся и их подходы к изучению программирования и электроники, их преподавательская философия и их точки зрения на процесс познания учениками преподаваемых им дисциплин, так что нельзя сказать, что взгляды их всегда совпадают. Тем не менее они надеются, что совместно написанная ими книга сослужит вам хорошую службу и поможет встать на путь поиска приключений в мире изобретательства.

Комментарий от Брайана

Хотя у меня формальное инженерное образование (я изучал электротехнику в институте), мое образование в основном концентрировалось на теории, эмулировании и моделировании, и меня никогда не учили, как пользоваться паяльником, выполнять токарные работы или создавать реальные проекты. По окончании института по будням я работал инженером, а по выходным волонтерствовал в Музее науки штата Миннесота (Science Museum of Minnesota). Именно работая в музее, я и обнаружил в себе любовь к преподаванию. Мне была предоставлена возможность вдохновлять детей на любопытство, задавать вопросы и размышлять о мире вокруг нас. Мой опыт работы в музее побудил меня изменить карьеру — получить степень магистра в области обучения и стать учителем физики в средней школе.

Мы с Дереком дополняем опыт и квалификацию друг друга. Эта книга является квинтэссенцией нашего опыта преподавания и использования Arduino для обучения. Как говорит Дерек, Arduino — просто используемое в наших проектах устройство более высокого уровня, чем раньше. Это было давно известно преподавателям и студентам Программы комплексной телекоммуникации Нью-йоркского университета. Точно так же помещение электронного устройства в корпус или просто ограничение его видимости каким-либо образом немедленно меняет способ взаимодействия с ним. Например, мячик для настольного тенниса, которым накрывают светодиод проекта, рассеивает его свет, что немедленно воздействует на способ нашего восприятия всего проекта. Своеобразное рассеивание света через оболочку мячика и одновременное ее подсвечивание воздействуют на наши эмоции совсем по-другому, чем простой горящий светодиод на макетной плате.

Мы основательно продумали, каким образом сделать изучение электроники и программирования доступными для всех. Мы надеемся, что проекты из этой книги помогут вам открыть в себе талант изобретателя.

Комментарий от Дерека

В отличие от Брайана, у меня нет формального образования по части электроники или программирования — все мои знания в этих областях получены путем самообразования. Я был учителем технического образования в восьмилетней школе, перед которым поставили задачу создать программу такого образования для XXI столетия. Составной частью моего видения этой программы была электроника, и на протяжении трех лет модуль Arduino, а затем язык Processing для его программирования заняли центральное место в моем классе. Я изучил Arduino с тем, чтобы предоставлять своим ученикам доступ к технологии, которой они могли бы управлять и с помощью которой могли бы самостоятельно создавать проекты. Мне пришлось также самому изучить электронику и программирование, чтобы иметь возможность обучать своих учеников этим предметам.

Многие проекты, представленные в этой книге, основаны непосредственно на моем опыте обучения Arduino. Мои ученики изучали программирование и электронику в силу необходимости — чтобы иметь возможность реализовать свои идеи, а не потому, что я, их учитель, заставлял их делать это. Я надеюсь, что мой вклад в эту книгу завоюет уважение моих учеников и предоставит электронику и программирование в таком формате, который пробудит также и ваше воображение.

О техническом редакторе английского издания

Даниэль Хайенш (Daniel Hienzsch) является основателем компании Rheingold Heavy, которая предоставляет учебные материалы студентам и любителям-электронщикам. Раньше он в течение 20 лет работал в сфере информационных технологий, включая 10 лет в качестве ИТ-директора инвестиционного банка.

Дан — страстный приверженец образования, и он создал компанию Rheingold Heavy, имея целью обеспечить общество любителей-электронщиков материалами, которых так не хватало ему самому, когда начинал заниматься электроникой и информационными технологиями. Он также сертифицированный инструктор по плаванию с аквалангом.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Приветствуем вас, уважаемые читатели. Эта книга научит вас работать с электронными компонентами, программировать электронные устройства и создавать на этой основе всевозможные интересные и полезные проекты. Любой человек может быть изобретателем, и наша книга пошагово проведет вас по последовательности проектов, в которых обычные детали сочетаются с мощностью модуля Arduino, помогая вам вдохновиться на собственные изобретения.

О чем эта книга?

Эта книга основана на платформе Arduino (**www.arduino.cc**), включающей микроконтроллерную плату, которую можно запрограммировать для управления источниками света, измерения температуры, реагирования на освещение, связи со спутниками глобальной системы позиционирования[1] и, вообще, для того чтобы с ее помощью создать много разных полезных и занимательных устройств. Язык программирования и среда разработки для платы, которые используются в книге, также являются компонентами платформы Arduino.

Эта платформа представляет собой мощный инструмент, с помощью которого любители электронного творчества могут оснащать свои проекты средствами управления. Поиск в Интернете по ключевым словам проекты Arduino возвращает миллионы результатов. Веб-сайты, такие как, например, Instructables, hackster.io или YouTube, содержат тысячи проектов и идей их создания. Все это демонстрирует, насколько много есть мастеров на все руки, использующих Arduino.

Совместно с компанией SparkFun Electronics мы всячески стараемся побудить людей экспериментировать, играть и возиться с обычными бытовыми устройствами, добавляя в них новые электронные компоненты или модифицируя уже имеющиеся. Такая деятельность называется *хакерством*. Чтобы вы могли успешно ею заниматься, наша книга предоставит вам основные знания по электронике и программированию. Мы также надеемся, что она вдохновит вас на создание чего-то нового и уникального из обычных материалов, которые можно найти у каждого в доме.

[1] GPS, от англ. Global Positioning System.

Почему Arduino?

Вы можете задаться вопросом, почему из существующих десятков разных микроконтроллеров и инструментальных платформ мы создаем еще один набор проектов для Arduino?

Ответ на этот вопрос заключается в том, что целью создания платформы Arduino было использование ее не любителями самодельного электронного творчества или инженерами промышленных предприятий, а студентами-проектировщиками из итальянского города Ивреа, — в качестве обучающей платформы, чтобы они могли создавать функционирующие проекты без необходимости изучать в течение нескольких лет теоретическую и практическую электротехнику, электронику и математику. Эта платформа была создана таким образом, чтобы свести к минимуму время от «ничего» к «круто!» — то есть от замысла к работающему проекту, — для людей без технического образования и опыта.

Платформа Arduino проявила себя настолько хорошо, что сообщество любителей-электронщиков также решило взять ее на вооружение. Этому способствовало несколько факторов: низкая цена, качественная документация, открытое аппаратное и программное обеспечение и т. п. Но мы считаем, что основной причиной такой популярности Arduino стала легкость обучения работе на этой платформе. Платформа Arduino — это открытый для любого желающего портал в мир созидания и изобретательства. Проекты из этой книги предназначены для энтузиастов, влекомых желанием познания и мотивируемых исходной сущностью платформы Arduino — быстрым и легким воплощением идеи в работающий проект людьми, не обладающими глубокими познаниями в электронике.

Чем эта книга отличается от других?

Многие книги по программированию похожи на справочные пособия — они сразу же начинаются с написания кода или рассмотрения понятий из области электроники без предоставления каких бы то ни было начальных сведений. В результате большую часть времени они пылятся на полке, пока вам не понадобится информация об определенной команде или забытом понятии.

Эта книга иная. Ее целью является обучение новым знаниям посредством создания занимательных, интересных и практических проектов. Сложность и трудность проектов повышается от предыдущего проекта к последующему. Они дадут вам ответы на старые как мир вопросы: зачем я изучаю это? Почему это важно? Почему это должно меня беспокоить?

Мы предполагаем, что вы читаете эту книгу, потому что горите желанием познания нового или же ищете материалы, которыми хотите поделиться с другими. Будь вы интересующийся новичок-любитель, учитель, библиотекарь или родитель, эта книга станет для любого, кто хочет познавать, практическим руководством, а не справочным пособием, которое будет в основном стоять на полке.

Чтобы начать работать с Arduino, вам не нужен никакой опыт программирования или работы с электроникой. Мы заранее предполагаем, что вначале вы не знаете ничего в этих областях, и опасаетесь броситься в них сломя голову. Не беспокойтесь! Эта книга предоставляет достаточно вводного материала, чтобы реализовывать простые проекты, и постепенно переходит к материалу для реализации более сложных и требовательных проектов.

Мы также знаем, что есть много людей, имеющих некий опыт работы с электроникой, которые хотят испытать что-то неизведанное, — возможно, новый подход к старому проекту. Например, найти не использованный ими ранее способ мигания светодиодом. Многие из наших проектов можно принять в качестве начальных для дальнейшего экспериментирования и модифицирования или как прототипы, на основе которых затем можно собрать проекты из лучших компонентов или с более привлекательным оформлением. В общем, эта книга направлена на активных экспериментаторов, которые без колебаний принимаются за решение задачи, уделяя этому максимальное внимание.

Мы рекомендуем реализовывать проекты при чтении книги, последовательно обучаясь путем приобретения практического опыта. Наши проекты хорошо продуманы с тем, чтобы дать вам необходимые знания как по использованию инструментов, так и по программированию и сборке схем, а также по созданию вспомогательных конструкций из картона, проволоки и прочих обыденных материалов, которые без труда можно найти в любом доме. Удовольствие от обучения заключается в том, чтобы весь процесс был игрой, во что эта книга и старается его превратить.

Необходимые компоненты и материалы

В проектах этой книги в основном используются электронные компоненты из нашего базового продукта — «Набора изобретателя SparkFun» (SparkFun Inventor's Kit, KIT-13969), но все эти компоненты можно свободно приобрести и по отдельности во многих интернет-магазинах. Кроме того, в проектах задействованы также несколько компонентов, не входящих в этот набор, которые также можно приобрести в виде отдельного набора (**www.sparkfun.com/NoStarchArduino**).

Списки необходимых компонентов приведены в табл. В.1 и В.2. В начале каждого проекта также приводится список необходимых для него компонентов и материалов.

Кроме того, для создания корпусов проектируемых устройств используются многие обычные материалы: упаковочный картон, открыточный картон, соломинки для питья, бумажные тарелки и т. п. По мере реализации электронных проектов с использованием таких материалов, вы станете смотреть на все эти бытовые предметы и материалы по-новому.

Таблица В.1. *Компоненты «Набора изобретателя SparkFun», используемые в проектах книги*

К-во	№ компонента	Описание
1	DEV-13975	Плата SparkFun RedBoard (или другая плата, совместимая с Arduino)
1	CAB-11301	Кабель Mini-B USB
1	PRT-12002	Беспаечная макетная плата
30	PRT-11026	Проволочные перемычки
20	COM-12062	Разные светодиоды
1	C0M-09264	Светодиод RGB (общий катод)
20	C0M-11508	Резисторы, 10 кОм
20	COM-11507	Резисторы, 330 Ом
2	COM-10302	Кнопочные переключатели
1	COM-08588	Диод
1	COM-09806	Потенциометр, 10 кОм
1	COM-13689	Транзистор NPN 2N222
1	SEN-09088	Фоторезистор
1	SEN-10988	Датчик температуры TMP36
1	SEN-08680	Потенциометр SoftPot, 50 мм
1	COM-07950	Пьезоэлектрический зуммер
1	LCD-00709	ЖК-дисплей, 16×2 символа
1	R0B-09065	Миниатюрный серводвигатель
1	ROB-11696	Электродвигатель

Таблица В.2. *Дополнительные компоненты, используемые в проектах книги (не входящие в «Набор изобретателя SparkFun»)*

К-во	№ компонента	Описание
1	PRT-12043	Макетная мини-плата
30	PRT-13870	Короткие (10 см) проволочные перемычки со штекерами на обоих концах
10	PRT-09140	Проволочные перемычки со штекером на одном конце и гнездом на другом
20	C0M-12062	Разные светодиоды (дополнительные)
1	SEN-09088	Фоторезистор (дополнительный)
1	PRT-09835	Держатель для 4-х элементов AA
4	PRT-09100	Элементы AA
1	ROB-13845	Драйвер электродвигателя — H-мост TB6612FNG
2	R0B-13302	Электродвигатели для любительских проектов с редуктором
2	ROB-13259	Резиновое колесико для использования с редукторным электродвигателем
1	C0M-00102	Миниатюрный ползунковый переключатель

Набор совместимых компонентов

Для выполнения проектов, описанных в книге, в оригинальном издании рекомендовано использовать набор компонентов SparkFun Inventor's Kit (**https://www.sparkfun.com/products/14265**). Если у вас его нет, предлагаем обратить внимание на наборы издательства «БХВ-Петербург» (**http://www.bhv.ru/books/kits**), специально подготовленные для книг по электронике, выпускаемых издательством. С их помощью вы можете выполнить эксперименты и проекты в том числе и из этой книги.

Необходимые инструменты

Все, что из инструментов вам потребуется для реализации проектов книги — это ножницы, макетный нож и клеевой пистолет. Но это не означает, что можно использовать только эти инструменты. Если у вас есть доступ к лазерному резаку, используйте его. Само собой, если у вас чешутся руки распечатать проект на 3D-принтере, полный вперед! Проекты в книге предназначены для реализации с использованием упаковочного и открыточного картона, но вместо этих материалов можно использовать любой подходящий материал и способ для его обработки.

Вы не должны ничего для реализации проектов покупать, если вы не желаете делать это. Собственно говоря, в качестве материала для нескольких проектов из этой книги можно использовать и саму книгу. Будет просто замечательно, если вы так и сделаете. Как бывшим учителям, нам знакома ситуация, когда приходится работать с очень ограниченным бюджетом, и мы всегда сосредотачиваемся на использовании наиболее экономных материалов — таких как картон, бумага, дерево и отходы пластмассы и металлов.

Большинство проектов этой книги предназначены для реализации в виде прототипов, которые можно легко разобрать и использовать их компоненты в других проектах. Но если вам сильно понравится какой-либо проект, и вы захотите сделать его постоянно действующим, его соединения можно выполнить пайкой (соответствующие инструкции приведены в *разд. «Работа с паяльником» приложения*). Инструменты и расходные материалы для пайки при создании прототипов электронных устройств сравнительно недороги — их можно приобрести как в магазине SparkFun (**www.sparkfun.com**), так и в любом ближайшем магазине хозяйственных товаров или инструментов.

Компьютер

Наконец, для программирования Arduino вам потребуется компьютер и набор специальных программных инструментов. Работа с программным обеспечением для Arduino под силу любому компьютеру средних возможностей. Для компьютеров на ОС Windows подойдут операционные системы Windows XP, Vista, Windows 7, 8/8.1, 10 или более новая. Для компьютеров Mac самая последняя версия программного обеспечения Arduino требует операционную систему OS X 10.7 Lion или более новую. Если у вас компьютер с более или менее стандартной версией Linux, есть хорошие шансы, что для них также найдется подходящая версия программного обеспечения Arduino.

На момент подготовки этой книги поддержка устройств на iOs и Android обеспечивается только посредством бета-версии программного обеспечения Arduino, которое еще находится в процессе разработки и тестирования. Вы можете попробовать использовать это программное обеспечение, но оно может оказаться неработоспособным, а если и будет работать, то ненадежно.

Процесс установки и конфигурирования программного обеспечения для компьютеров Windows, Mac и Linux рассматривается пошагово в первом проекте.

Состав книги

Книга содержит 10 практических проектов, а также краткое изложение основ электроники и приложение, в котором рассматриваются основы работы с паяльником, а также даются другие полезные советы. Проекты начинаются с простого мигающего светодиода и с каждым следующим проектом постепенно обрастают разными электронными компонентами, концепциями программирования и слоями сложности конструирования. Каждый проект содержит отдельные разделы по монтажу электрических соединений, программированию и сборке, что позволяет досконально разобраться с каждым из этих аспектов проекта. Проекты завершаются разделом *Идем дальше*, в котором излагаются идеи по экспериментированию с проектом и его модификации. Не забывайте, что эти проекты следует использовать в качестве отправной точки для своих разработок, а не как конечную цель.

- **Основы электроники.** Прежде чем приступить к работе с проектами, вам рекомендуется познакомиться с основами электричества и электроники и понятиями, которые используются в этой книге.
- **Проект 1. Начало работы с Arduino.** В проекте описывается установка программного обеспечения и даются основы приемов сборки и программирования схем на основе пошаговых инструкций при реализации проекта мигающего светодиода.
- **Проект 2. Домашний светофор.** Здесь исследуется работа с макетной платой и рассматривается управление несколькими компонентами одновременно на примере модели светофора из трех светодиодов.
- **Проект 3. Девятипиксельный анимационный дисплей.** Здесь производится расширение проекта светофора из *проекта 2* до девяти светодиодов в матрице размером 3×3, а также изучается создание пользовательских функций Arduino.
- **Проект 4. Измеритель скорости реакции.** В проекте описывается использование кнопки и светодиода при создании игры для измерения скорости реакции.
- **Проект 5. Разноцветный ночник.** Здесь делитель напряжения и датчик освещенности используются для определения уровня освещенности и активирования в зависимости от этого уровня многоцветного светодиода.
- **Проект 6. Балансирная балка.** В проекте используется серводвигатель под управлением внешнего устройства для создания настольной игры с балансирной балкой.
- **Проект 7. Миниатюрная настольная теплица.** В проекте создается модель теплицы, которая определяет температуру, включает вентилятор и открывает вентиляционное отверстие, если температура в ней станет слишком высокой. Отрабатывается здесь также управление электродвигателем с помощью транзистора.
- **Проект 8. Робот-рисовальщик.** Проект представляет собой введение в робототехнику с использованием Н-моста для регулирования частоты оборотов электродвигателя. Здесь осуществляется сборка простого робота, которого можно запрограммировать на рисование различных фигур.
- **Проект 9. Хронометрист автогонок.** В проекте рассматривается создание гоночной трассы для игрушечных машин, которая записывает их скорость. В конструкции используются сервопривод, светочувствительный датчик и жидкокристаллический дисплей. Проект настолько занимательный, что его можно подарить на Рождество!
- **Проект 10. Миниатюрное пианино.** Проект помогает создавать с помощью Arduino музыку, используя мембранный потенциометр в качестве клавиатуры. Здесь также исследуется пьезоэлектрический зуммер и рассматривается использование функции tone(). Хорошая возможность открыть в себе пианиста!
- **Приложение. Дополнительные практические сведения по электронике.** Здесь приводятся полезные советы по использованию мультиметра, паяльника и определению номинального значения резисторов по цветным полосам.

Интернет-ресурсы

Все ресурсы, необходимые для проектов этой книги, доступны для загрузки, использования и модифицирования. Они содержат коды всех примеров, которые приводятся и обсуждаются в книге, шаблоны для конструкций проектов, а также коды для дальнейшего экспериментирования с проектами и их модифицирования.

Ресурсы организованы в виде ZIP-файла, который можно загрузить по адресу: **www.nostarch.com/arduinoinventor**.

Если вы застопоритесь на каком-либо аспекте того или иного проекта или у вас возникнут проблемы с его работой, можно всегда обратиться к этим файлам за справкой для решения проблемы.

Электронный архив

Все необходимые для работы с проектами книги ресурсы вы также найдете в сопровождающем книгу электронном архиве, который можно загрузить с FTP-сервера издательства «БХВ-Петербург» по ссылке: **ftp://ftp.bhv.ru/9785977539722.zip** или со страницы книги на сайте **www.bhv.ru**.

Распространяем информацию: делимся результатами своей работы

Компания SparkFun, будучи поставщиком аппаратных и электронных компонентов, прилагает также большие усилия, чтобы быть открытым источником информации, — это один из основных принципов, на котором она основана. Разрабатывая проекты, мы любим делиться своими идеями и файлами проектов с нашим сообществом, чтобы все желающие могли воспользоваться нашей базой знаний в своем следующем проекте. Мы бы хотели, что бы вы также исповедовали такой подход и делились с другими информацией о своих проектах. Покажите их своим друзьям или опубликуйте в Интернете. В последнем случае это можно сделать во многих местах.

Например, поделиться своей работой можно в социальных сетях Twitter, Instagram или Facebook. Пометьте опубликованные проекты тегами @sparkfunedu и @nostarch. Мы также предлагаем для общего доступа через Интернет галерею проектов InventorSpace, доступную по адресу: **https://invent.sparkfun.com**. Если у вас есть идея или проект, которыми вы хотите поделиться, вы можете опубликовать его в этой галерее. Мы надеемся, что эта книга вдохновит вас начать делать что-то поразительное.

Наконец, вы можете также отправить нам свои проекты, их фотографии или общие замечания и вопросы электронной почтой по адресу: **ArduinoInventorsGuide@sparkfun.com**. Время от времени мы выбираем некоторые замечательные проекты и фотографии для главной публикации в нашем блоге. Кто знает, возможно мы попросим у вас разрешения использовать один из ваших проектов в нашей следующей книге.

ОСНОВЫ ЭЛЕКТРОНИКИ

Эта глава дает общие сведения об электричестве, электронике и схемотехнике для тех, у кого нет никакого опыта работы в этих сферах знания или этот опыт очень небольшой. Если вы почувствуете, что некоторые представленные в этом проекте темы вам более или менее знакомы, можете пропускать их и переходить к тем темам, о которых вы хотите знать более подробно, или вообще перейти сразу к **проекту 1**.

Тем не менее, даже если вы и не абсолютный новичок в электронике, мы рекомендуем вам прочитать эту главу, просто чтобы освежить свои знания. И пусть она и не представляет собой полное руководство по электронике (этой области посвящены целые тома монографий и учебные курсы), можете считать ее удобным справочным пособием, предназначенным дать вам представление об основных ее понятиях и терминологии. Для тех же, кто хочет получить более глубокие знания по электричеству, электронике и схемотехнике, в конце этой главы приведен список рекомендуемой литературы.

Электричество: ток, проводимость и основная терминология

Электричество — это очень странный зверь. Во многих отношениях оно предсказуемо, но временами может быть весьма коварным. Если просто посмотреть определение электричества в учебнике, то, скорее всего, оно не много скажет вам о том, что такое электричество, как оно работает и, самое главное, как его можно использовать. В данной главе предоставляются основы всего этого.

Что такое электричество?

Чтобы ответить на этот вопрос, прежде всего нужно разобраться в строении атома. Атомы являются основными составляющими единицами всего, что находится вокруг нас, и нас самих. Состоят они из протонов, нейтронов и электронов. Протоны и нейтроны образуют ядро атома, а электроны вращаются вокруг ядра по более или менее круговым орбитам. Протоны и, соответственно, ядро атома обладают положительным зарядом, а электроны — отрицательным. Нейтроны заряда не несут. Количество электронов типичного атома равно количеству его протонов, поэтому атом имеет нейтральный заряд.

Под воздействием тех или иных сил несущие заряды электроны могут срываться со своих орбит и перемещаться внутри вещества. Такое перемещение зарядов и представляет собой вид энергии, называемый электричеством. Молния, которую наверняка когда-либо видел каждый из нас, является наглядной демонстрацией перемещения зарядов между облаками или облаками и землей.

Перемещение зарядов в веществе называется *током*. Величина тока измеряется в единицах, называемых *амперами* (А). Для удобства на практике могут использоваться и более мелкие единицы — например, *миллиамперы* (мА).

> **Примечание**
>
> Условно электрическим током называется перемещение положительных зарядов. И хотя в действительности ток создается перемещением электронов, которые несут отрицательный заряд, принято считать, что ток протекает от положительного источника к отрицательному.

За исключением нескольких явлений, таких как молния, сварочная дуга или разряд статического электричества, электрический ток обычно нельзя наблюдать непосредственно. Даже яркий свет молнии является лишь следствием изменения состояния молекул воздуха, вызываемого прохождением электричества через них.

Заряды перемещаются в веществе под воздействием внешней электрической силы и при наличии пути, по которому они могут перемещаться. Сила электрического воздействия создается *разностью потенциалов*, которая обычно называется *напряжением*. Напряжение и является силой, которая, в конечном счете, заставляет заряды

двигаться. Величина напряжения измеряется в *вольтах* (В). Для общей информации: типичные электрические батареи имеют напряжение в диапазоне от около 1,5 до 12 В. Батарея с напряжением в 12 В заставляет перемещаться большее количество зарядов, чем батарея с напряжением в 1,5 В.

Типы электрического тока

Существуют два основных типа электрического тока: *постоянный* и *переменный*. Самый распространенный пример переменного тока — электричество в линиях электропередачи, которое заходит к нам в дом, в частности, через розетки. Переменный ток отлично подходит для выработки электричества (например, электростанциями), передачи его на дальние расстояния (например, от электростанции к нам в дом) и для приведения в действие мощных электрических устройств (например, электродвигателей и обогревателей). Но для работы большинства электронных приборов (телевизоры, компьютеры и т. п.) переменный ток не нужен — им требуется постоянный ток, который они получают с помощью специального устройства (обычно встроенного в вилку, вставляющуюся в розетку), преобразовывающего переменный ток в постоянный. Более глубокое рассмотрение переменного и постоянного тока выходит за рамки этой книги. В завершении скажем только, что для питания проектов в этой книге используется электричество постоянного тока.

Что такое цепь?

Даже когда на заряды воздействует электрическая сила, понуждающая их перемещаться от более высокого потенциала к более низкому, между этими потенциалами необходим путь какого-либо рода. Путь, по которому заряды перемещаются от более высокого (положительного: «+») потенциала батареи к более низкому (отрицательному: «–») ее потенциалу, называется *электрической цепью*. Таким образом, электрическая цепь представляет собой замкнутый путь, проходящий от положительного вывода источника питания через потребитель тока (например: светодиод, резистор, лампочку или электродвигатель) к отрицательному его выводу. На рис. 1 показана простая электрическая цепь, состоящая из батарейки, светодиода и резистора.

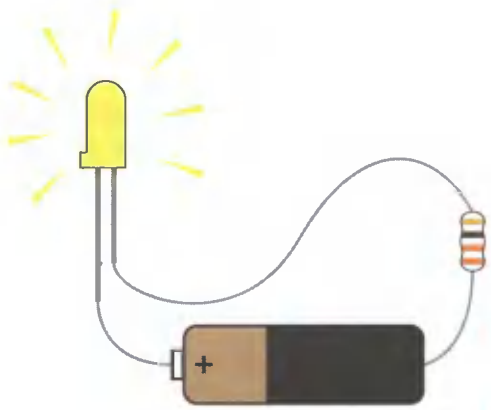


Рис. 1. Простая цепь постоянного тока

Но чтобы заряды (электроны) могли перемещаться по цепи, она должна быть из материала, проводящего электричество, или, иными словами, обладать *электрической проводимостью*. Проводимость того или иного материала зависит от вещества, из которого он состоит. Притом, что некоторые материалы обычно считаются абсолютными проводниками или непроводниками (изоляторами), проводимость большинства материалов варьируется в широком диапазоне. Иными словами, заряды перемещаются через некоторые материалы более свободно, чем через другие.

Это можно сравнить с ездой на машине по дорогам с разным покрытием. На гладкой асфальтированной дороге можно ехать намного быстрее, чем по проселочной или вообще по пересеченной местности. Таким образом, как дороги могут иметь разное покрытие, позволяющее развивать разную скорость перемещения по ним (разную автомобильную проводимость, так сказать), так же и разные материалы имеют различную электрическую проводимость, позволяющую более или менее свободное перемещение электрических зарядов через них. Величина, обратная проводимости, называется *сопротивлением* и выражает степень затруднения материалом перемещения по нему электрических зарядов.

Закон Ома

Как вы уже, наверное, догадались, ток, напряжение и сопротивление взаимосвязаны. Эта взаимосвязь называется законом Ома и выражается следующей математической формулой:

$$U = I \times R$$

В этом уравнении буква U обозначает напряжение, I — ток, а R — сопротивление. (Не стоит пугаться, что мы сейчас начнем углубляться в математические дебри — это всего лишь одно из трех уравнений, которые вы увидите в этой книге.)

Модель электрического тока: вода в трубе

Чтобы лучше понять прохождение тока по электрической цепи, представим себе трубу, по которой протекает вода. Открывая кран, мы позволяем воде из резервуара протекать по трубе и вытекать из другого ее конца (рис. 2).

Рис. 2. Модель электрического тока: вода в трубе

Молекулы воды, перемещающиеся по трубе, представляют в нашей модели поток зарядов (ток). Закрывая или открывая кран, мы можем менять давление воды в трубе, — повышение давления воды ускоряет ее течение по трубе. Давление воды в трубе подобно напряжению в электрической цепи — повышение напряжения увеличивает ток. Труба представляет собой последнюю часть аналогии — пережав ее (уменьшив ее диаметр), мы создадим сопротивление протеканию воды. Такое повышение сопротивления замедлит протекание воды в трубе — возвращаясь к нашей модели, понизит величину тока в электрической цепи.

Эта модель хорошо описывает электрический ток, но мы не хотим собрать всю эту систему резервуаров, труб и кранов, чтобы вода только проливалась на землю (разве что собираемся поливать газон). Нам нужно, чтобы она делала какую-то полезную работу. В случае электрических цепей мы используем устройства, которые преобразуют электричество в полезную работу, — например, в свечение лампочки, вращение электродвигателя или звучание зуммера. Устройство, которое преобразовывает электрическую энергию в другой вид энергии, называется *нагрузкой*. То, что электрическую энергию можно преобразовывать в световую энергию с помощью лампочки, открыл Томас Эдисон. В этой книге мы будем занимать такими и многими другими преобразованиями.

Принципиальные схемы

Представление электрической цепи рисунками ее компонентов выглядит красиво, но это не очень эффективно, особенно когда цепь состоит из большого количества элементов. Поэтому на практике цепь представляют упрощенными символами ее компонентов (рис. 3), а само такое представление называют *принципиальной схемой*.

Принципиальная схема показывает, какие компоненты содержит электрическая цепь и каким образом они соединяются между собой[1]. Приведенная на рис. 3 принципиальная схема в действитель-

[1] Принципиальные схемы в этой книге отображаются согласно американскому стандарту IEEE (Institute of Electrical and Electronics Engineers, Институт инженеров электротехники и радиоэлектроники).

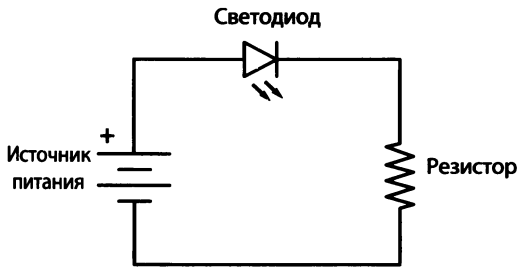


Рис. 3. Принципиальная схема электрической цепи, состоящей из батарейки, светодиода и резистора

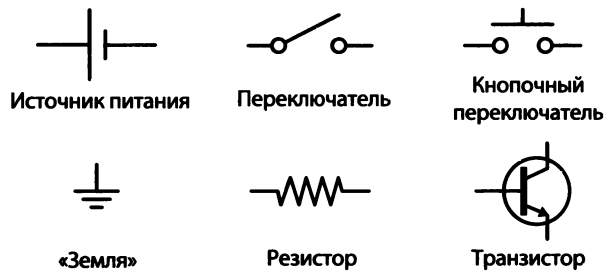


Рис. 4. Некоторые схемные символы стандарта IEEE

ности представляет ту же самую электрическую цепь, что и на рис. 1. Прямые линии здесь — это провода, а компоненты обозначены соответствующими символами. На рис. 4 показано несколько распространенных схемных символов, которые используются в этой книге.

Формат схемных символов стандарта IEEE предназначен для быстрого представления компонентов схем с использованием очень простых линий и символов. Он признается и применяется для представления и распространения принципиальных схем во всем мире.

Создание прототипов схем

По мере изучения проектов этой книги, вы будете создавать и проверять множество вариантов их разработки. При сборке схемы может понадобиться перегруппировать компоненты, убрать некоторые из них или добавить новые. Этот процесс называется *разработкой прототипа*. Для разработки прототипов электронных устройств обычно используется *беспаечная макетная плата* (рис. 5).

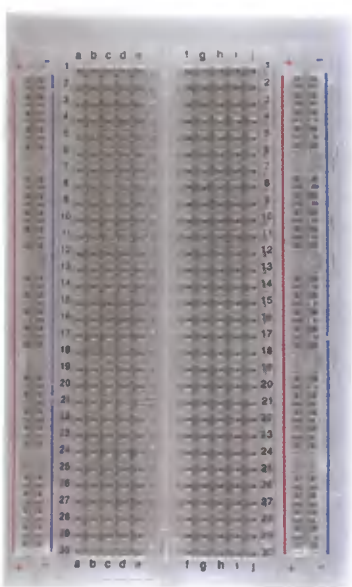


Рис. 5. Прозрачная беспаечная макетная плата с горизонтальными контактными рядами и вертикальными шинами питания

Беспаечная макетная плата представляет собой пластмассовый прямоугольник с множеством отверстий. Эти отверстия расположены решеткой с расстоянием между ними в 0,1 дюйма (2,54 мм). Диаметр отверстий таков, чтобы выводы большинства электронных компонентов плотно входили в них.

Ориентация макетной платы

Расположите макетную плату вертикально (книжная ориентация), чтобы буквы на ее концах были размещены правильным образом. При такой ориентации макетной платы горизонтальные группы из пяти гнезд называются *рядами*, а вертикальные секции по сторонам платы — *столбцами*.

Под отверстиями расположены небольшие зажимы, сделанные из мягкого металла (рис. 6). Эти зажимы электрически соединяют компоненты, вставленные в отверстия в том же самом ряду. Таким образом, отпадает необходимость соединять компоненты, скручивая вместе их выводы. Обратите внимание на то, что зажимы перекрывают только пять соседних отверстий в ряду. Макетная плата разделяется на две половины вертикальной выемкой, и зажимы рядов правой стороны не соединены с зажимами рядов левой.

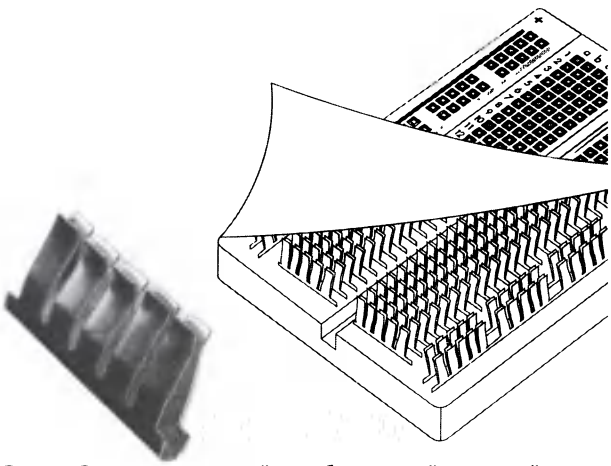


Рис. 6. Внутреннее устройство беспаечной макетной платы (*справа*) и крупный план металлического зажима (*слева*)

Форм и размеров беспаечных макетных плат доступно несколько, но большинство их будут иметь вертикальные столбцы гнезд на внешних сторонах. Эти столбцы гнезд называются *шинами питания*, и каждые пять гнезд в них соединены зажимом, так же, как и гнезда рядов (рис. 7). Столбцы шин питания часто обозначаются символами «+» и «–», указывающими полярность подключаемого питания, а также маркируются красным и синим цветом соответственно.

Проекты этой книги собираются на беспаечных макетных платах, чтобы в случае ошибки можно было бы быстро исправить ее, а также чтобы можно было усовершенствовать готовый проект, добавив к нему дополнительные необходимые компоненты (на рис. 8 показан вариант использования макетной платы для создания прототипа схемы, содержащей восемь светодиодов).

Для сборки крупных и сложных проектов рекомендуется иметь в наличии несколько беспаечных макетных плат, чтобы схему можно было собрать по частям, соединив их затем вместе. Это позволяет собрать и протестировать каждую часть проекта по отдельности, что намного легче, чем тестирование всего проекта.

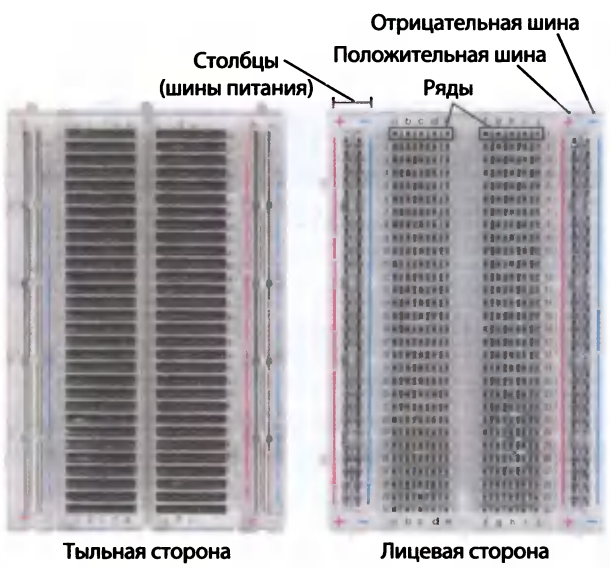


Рис. 7. На *тыльной стороне* беспаечной макетной платы видны горизонтальные контактные ряды и вертикальные шины питания

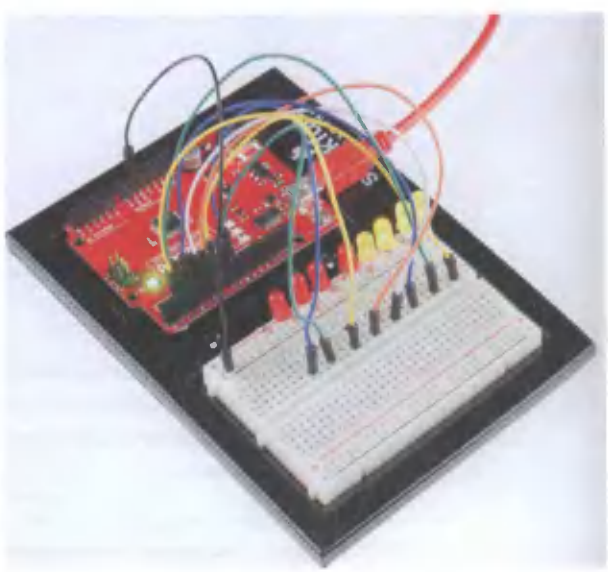


Рис. 8. Сборка схемы на макетной плате

Дискретные компоненты и адаптерные платы

Существуют буквально сотни, если не тысячи разнообразных электронных компонентов. Под *компонентами* в нашем случае мы подразумеваем дискретные электронные детали, т. е. самые элементарные составляющие электронных схем. В качестве примеров дискретных компонентов можно привести резистор, конденсатор и светодиод (рис. 9).

Рис. 9. Примеры дискретных компонентов: резисторы (*слева*), конденсатор (*посередине*) и светодиод (*справа*)

С другой стороны, *адаптерная плата* представляет собой электронную схему из нескольких компонентов, собранную на отдельной плате, — это облегчает установку ее на макетной плате проекта. Таким образом, адаптерные платы помогают значительно ускорить процесс создания прототипов.

На рис. 10 приведен пример сложного дискретного компонента — интегральной схемы датчика акселерометра (P/N ADXL345 компании Analog Devices) и его адаптерной платы, предлагаемой компанией SparkFun. Размеры самой интегральной схемы всего лишь 5×3 мм! Она оснащена миниатюрными металлическими выводами, выполняющими ту же функцию, что и длинные металлические выводы более традиционных дискретных компонентов, — таких как, например, резисторы. Но выводы этой интегральной схемы настолько малого размера, что подсоединить монтажные провода к ним практически невозможно. Поэтому она поставляется в составе адаптерной платы, где эти выводы соединены дорожками с пролуженными насквозь отверстиями на краю платы. Расстояние между этими отверстиями составляет точно 0,1 дюйма (2,54 мм), в результате чего они точно совмещаются с гнездами на беспаечной макетной плате. Поскольку отверстия пролужены, в них можно впаять монтажные провода. Кстати, вместо проводов можно впаять в эти отверстия контактные штырьки (рис. 11), и тогда эту плату будет удобно вставлять в беспаечную макетную плату. (Не озабочивайтесь, если вы никогда раньше не работали с паяльником. В *разд. «Работа с паяльником» приложения* вы найдете всю информацию, необходимую для выполнения монтажа пайкой.)

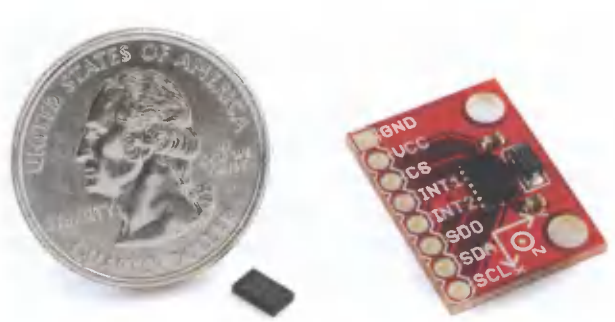


Рис. 10. Дискретный миниатюрный акселерометр (*слева*) и адаптерная плата, на которой он установлен (*справа*). Обратите внимание на пролуженные насквозь отверстия с левой стороны адаптерной платы

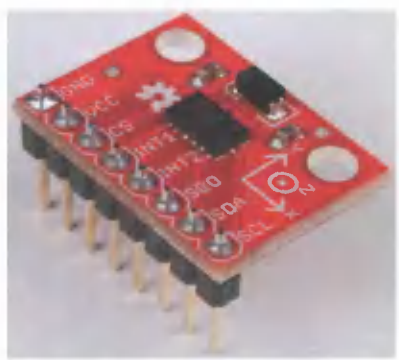


Рис. 11. Адаптерная плата ADXL345 с контактными штырьками

Обратите внимание, что каждое контактное отверстие адаптерной платы обозначено названием подключенного к нему вывода интегральной схемы акселерометра. Таким образом, адаптерная плата позволяет сразу же использовать ее в беспаечной макетной плате, без необходимости тратить время на решение проблемы, каким образом подключить дискретный компонент.

Аналоговая и цифровая электроника

Рассмотрев понятия тока, напряжения, сопротивления, электронных компонентов и схем, можно перейти к обсуждению двух типов электроники: аналоговой и цифровой. Эти два типа электроники взаимосвязаны, и для понимания работы электронных схем в целом необходимо разобраться с основными принципами работы каждого из этих двух типов.

Аналоговая электроника работает со значениями токов и напряжений, которые варьируются плавно в пределах определенного диапазона. Например, контроллер плавного изменения яркости освещения является аналоговым устройством. Аналоговое значение может быть нулевым, максимальным или каким угодно между этим двумя крайностями. С другой стороны, *цифровые* величины имеют только два значения: нулевое и максимальное или включенное и выключенное.

Устройства цифровой электроники, такие как микроконтроллер и микропроцессор, включают и выключают другие устройства в зависимости от запрограммированных в них условий. Устройства аналоговой электроники обычно плавно изменяют ток, напряжение или сопротивление, чтобы добиться того же результата.

Оба типа устройств имеют свои преимущества и недостатки, но в современной электронике нельзя обойтись только одним из них. Например, для работы цифрового термометра на микроконтроллере необходим целый ряд аналоговых компонентов.

Что такое микроконтроллер?

Рис. 12. Микроконтроллерная плата RedBoard компании SparkFun (*вверху слева*) и плата Arduino Uno (*внизу справа*)

Микроконтроллер — это небольшой компьютер, который можно программировать, загружая в него набор инструкций, называемый *программой*. Микроконтроллеры применяются для автоматизации простых операций — например, управления температурой воздуха в помещениях или поливкой газона.

В проектах этой книги используется микроконтроллерная плата RedBoard компании SparkFun, которая на 100% совместима с платой Arduino Uno. Обе эти платы показаны на рис. 12.

В любой день вы в среднем можете использовать 15–20 микроконтроллеров, даже не подозревая об этом. Микроконтроллеры управляют вашей кофеваркой, будильником, микроволновкой и другими бытовыми устройствами (более-менее современными, конечно, а не доставшимися вам по наследству от бабушки). Ваш автомобиль может использовать от 5 до 10 микроконтроллеров для управления системой зажигания, тормозами,

Примечание

Вы узнаете больше о плате Arduino, включая ее возможности и как ее программировать, в процессе реализации проектов. А сейчас вам будет достаточно знать, что микроконтроллер — это программируемый кремниевый «мозг», который значительно облегчает превращение идеи об автоматизировании какого-либо аспекта нашей жизненной деятельности в прототип электронного устройства, реализующего эту автоматизацию.

аудиосистемой и т. п. Современный мир практически стоит на микроконтроллерах. Эта книга поможет вам научиться использовать это обстоятельство для своей выгоды.

Мы надеемся, что представленное вам здесь краткое введение в основы электроники дало вам достаточно знаний для работы с дальнейшим материалом книги. Мы также надеемся, что вы найдете этот материал интересным, увлекательным и полезным. Итак, вперед, к первому проекту!

ДОПОЛНИТЕЛЬНЫЕ ИСТОЧНИКИ ПО ОСНОВАМ ЭЛЕКТРИЧЕСТВА И ЭЛЕКТРОНИКИ

Если вы желаете узнать больше об электричестве и электронике, мы рекомендуем вам ознакомиться со следующими книгами:

- Basic Electricity by the Bureau of Naval Personnel (Основы электричества. Учебник Бюро флотского персонала), Dover Publications, 1970.
- Arduino Workshop by John Boxall (Джон Баксол. Мастерская для Arduino), No Starch Press, 2013.
- Getting Started in Electronics by Forrest M. Mims III (Форрест М. Мимс III. Первые шаги в электронике), Master Publishing, 2003.
- Practical Electronics for Inventors, 4th edition by Paul Scherz and Simon Monk (Поль Шерц, Саймон Монк. Практическая электроника для изобретателей), McGraw-Hill Education, 2016.

Читателям в России могут быть полезны и такие издания на русском языке:

- Платт Ч. Электроника для начинающих, 2-е изд. — СПб.: БХВ-Петербург, 2017. — 416 с.
- Блум Дж. Изучаем Arduino: инструменты и методы технического мастерства. — СПб.: БХВ-Петербург, 2015. — 336 с.
- Ревич Ю. В. Занимательная электроника, 5-е изд. — СПб.: БХВ-Петербург, 2018.
- Монк С., Шерц П. Электроника. Теория и практика. — СПб.: БХВ-Петербург, 2018. — 1168 с.

НАЧАЛО РАБОТЫ С ARDUINO

В этом проекте содержится вся необходимая информация, чтобы начать работать с Arduino. В нем мы познакомим вас с самой платой Arduino, расскажем, как установить среду программирования, а также поможем вам проверить работоспособность собранной схемы, загрузив и выполнив простую программу. Если вы все сделаете правильно, на плате должен будет мигать встроенный светодиод, — это станет подтверждением того, что вы уже готовы двигаться дальше, к новым увлекательным приключениям с Arduino.

Необходимые компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 1.1):

- плата Arduino Uno (DEV-11021), или плата RedBoard компании SparkFun (DEV-13975), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель USB (A – B) (CAB-00512) для платы Arduino Uno или кабель Mini-B USB (CAB-1101) для платы RedBoard, 1 шт.;
- светодиод (COM-09590 или COM-12062 для пакета, содержащего 20 шт.), 1 шт.

Рис. 1.1. Компоненты для первого проекта

О плате Arduino

Плата Arduino представляет собой небольшое программируемое устройство, с помощью которого обычным электронным схемам можно придать определенный уровень интеллектуальности. Ее можно использовать для управления роботами, создания графики на светодиодах или даже как ручную игровую консоль. В этом разделе мы рассмотрим более подробно, что такое Arduino и как она может изменить ваше мировоззрение.

Доступная аппаратная платформа

Плата Arduino подобна небольшому компьютеру. Ее можно программировать, используя очень простые инструкции, а для ее питания достаточно лишь нескольких батареек типоразмера AA. Но Arduino отличается от обычного компьютера тем, что для обработки информации и выполнения действий в ней используется не процессор, а микроконтроллер — небольшая интегральная схема, выполняющая в проектах функцию своеобразного мозга. Она может получать входные данные от датчиков (например, оптических датчиков, датчиков температуры или просто переключателей) и выдавать выходные сигналы для управления светодиодами, электродвигателями, зуммерами и т. п. Плата Arduino (рис. 1.2) содержит все вспомогательные компоненты и схемы, необходимые для работы ее микроконтроллера.

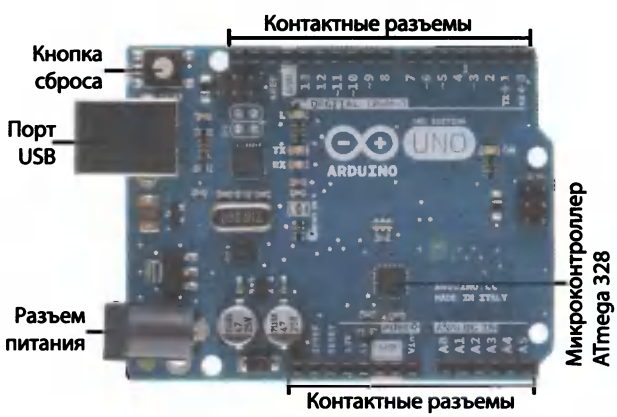


Рис. 1.2. Arduino Uno — открытая программируемая платформа для любителей электроники

Программируется Arduino на языке, который фактически является версией языка C/C++. В качестве среды программирования используется Arduino IDE (интегрированная среда разработки). Разработчики среды оснастили ее многими готовыми функциями и библиотеками функций, чтобы упростить создание кода для интерфейса платы Arduino с другими аппаратными устройствами. Например, одной из таких функций является функция включения светодиода, которая сводит множество строк кода всего лишь в одну инструкцию.

Плата RedBoard компании SparkFun

Существует много плат, официально использующих марку Arduino, но поскольку это открытая платформа (то есть исходный дизайн ее аппаратной и программной части доступен для просмотра и модифицирования любому желающему), существует также множество плат, производных от Arduino, ее клонов и совместимых с ней. Все эти платы разработаны по лицензии Creative Commons «Attribution-ShareAlike» (Атрибуция — С сохранением условий), и в часто задаваемых вопросах по Arduino (**https://www.arduino.cc/en/Main/FAQ**) указывается, что любой желающий «может свободно использовать и адаптировать [этот дизайн] для своих собственных нужд без необходимости спрашивать разрешения или платить за это». Производные платы работают с той же средой программирования, что и официальная Arduino, но часто их аппаратная часть настроена на лучшую производительность или модифицирована каким-либо иным образом.

Плата RedBoard компании SparkFun (рис. 1.3) является производной платой, совместимой с Arduino. Она основана на дизайне Arduino Uno, но имеет более стабильный интерфейс USB и оснащена мини-разъемом USB вместо разъема USB типа A. Во всем остальном она точно такая же, как и плата Arduino Uno, такой же формы и размеров.

Плата RedBoard содержит несколько ключевых компонентов, назначение которых необходимо знать, чтобы быть в состоянии разбираться с материалом нескольких первых глав книги. Эти компоненты обозначены на рис. 1.3, а далее приводится их описание:

- **микроконтроллер ATmega328** — черная квадратная микросхема посредине платы. Это мозг Arduino;
- **Штыревые разъемы** — металлические выводы на плате, позволяющие считывать с платы выходные сигналы и отправлять ей входные. Организованы они в виде четырех групп разъемов, по две на каждой стороне платы Arduino. Каждый штыревой разъем пронумерован или обозначен его функцией. Нас будут интересовать разъемы **Digital** (0–13), **Analog In** (A0–A5) и **Power**;
- **Порт мини-USB** — этот порт используется для взаимодействия с Arduino и для его программирования. Для большинства проектов этой книги этот порт мини-USB также служит для подачи на плату питания. Если для платы требуется внешний источник питания, это будет обязательно указано;
- **Индикатор питания** — этот светодиод служит индикатором подачи питания на плату. Если по какой-либо причине на плату не поступает питание, светодиод не будет гореть;
- **Светодиоды TX/RX** — эти светодиоды мигают при передаче (TX) или приеме (RX) платой каких-либо данных;

Рис. 1.3. Плата RedBoard компании SparkFun, совместимая с платой Arduino Uno. Обратите внимание, что форма платы совпадает с формой платы Arduino Uno, показанной на рис. 1.2

- **Встроенный светодиод 13** — применяется для отладки. При первом включении Arduino, когда он не содержит никаких пользовательских программ, светодиод 13 должен мигать с частотой один раз в секунду. Этот светодиод подключен к выводу 13 Arduino, отсюда и его название;
- **Разъем внешнего питания** — цилиндрический разъем (гнездо) рядом с портом USB. Для питания Arduino требуется напряжение 5 В, хотя на плату можно спокойно подавать напряжение величиной от 7 до 15 В, не опасаясь ее повредить. Специальная микросхема на плате понижает это напряжение до 5 В, требуемых для правильной работы электронных компонентов платы.

Как и в случае со всеми платами, совместимыми с Arduino, для программирования платы RedBoard используется интегрированная среда разработки Arduino IDE.

Установка Arduino IDE и драйверов

Прежде чем впервые подключать плату Arduino (или RedBoard) к порту USB компьютера, на компьютер необходимо установить среду разработки Arduino IDE.

> **Примечание**
>
> Если вы все-таки подключили плату до установки среды и драйверов, не волнуйтесь, это не станет проблемой. Придется после установки среды просто перезагрузить компьютер, чтобы драйверы работали должным образом.

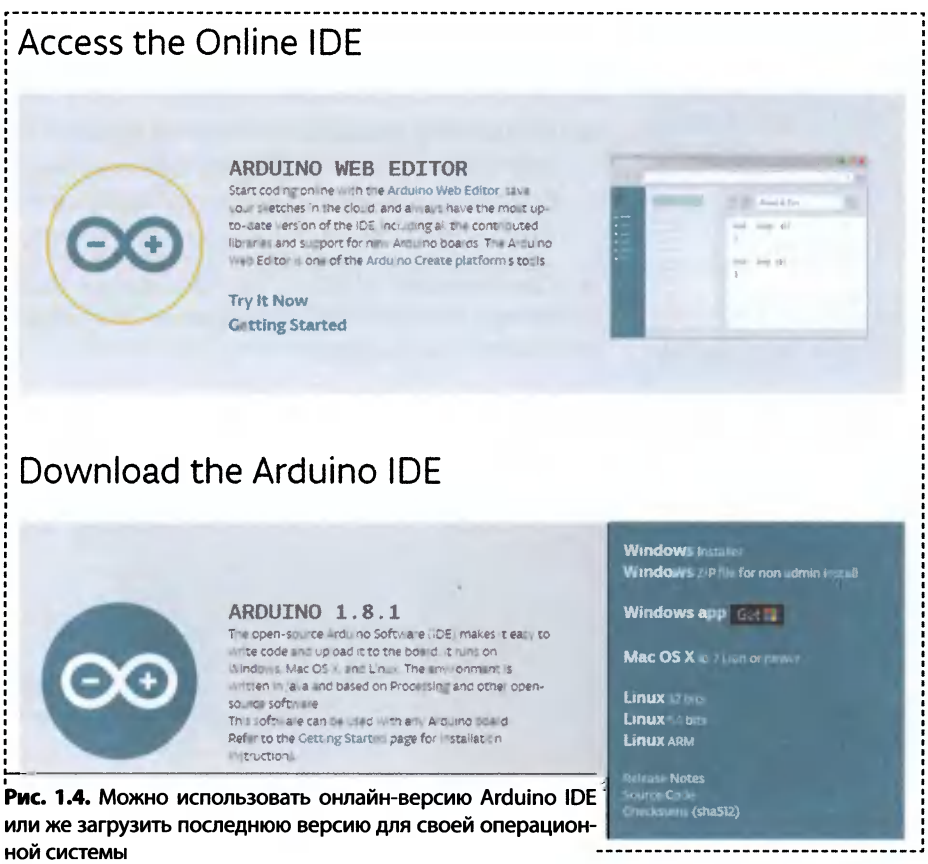


Рис. 1.4. Можно использовать онлайн-версию Arduino IDE или же загрузить последнюю версию для своей операционной системы

Среду разработки Arduino IDE можно загрузить по адресу: **www.arduino.cc/download**. На странице загрузки определитесь с версией среды, соответствующей операционной системе вашего компьютера, а затем щелкните необходимую ссылку (рис. 1.4). На следующей странице вас спросят, не желаете ли вы сделать взнос на развитие программного обеспечения Arduino (разработка и поддержка Arduino IDE выполняется за счет помощи и взносов сообщества ее пользователей).

Даже если у вас уже установлена Arduino IDE, рекомендуется загрузить и установить последнюю версию этой среды разработки. Среда Arduino IDE постоянно улучшается и обновляется, поэтому лучше пользоваться самым последним ее выпуском. Примеры в этой книге используют Arduino IDE версии 1.8.1 или более поздней.

На домашней странице веб-сайта Arduino также предоставляется возможность использования платформы Arduino Create, включая редактор кода с браузерным интерфейсом (см. рис. 1.4, *вверху*). Эта платформа позволяет программировать Arduino посредством веб-браузера, а также просматривать проекты и делиться ими с другими непосредственно в Интернете. На момент подготовки этой книги платформа поддерживается только операционной системой Windows и OS X.

Независимо от вашего выбора: использовать Arduino Create или загрузить среду разработки — следуйте инструкциям для установки.

Примечание

Если вы хотите использовать самую передовую версию программного обеспечения, на странице загрузок также предоставляются «ночные» сборки для следующего выпуска. Но для работы с этой книгой рекомендуется использовать только последний стабильный выпуск.

Установка под Windows

Для компьютеров на ОС Windows предлагается загрузить версию Arduino IDE Windows Installer. Загрузите этот установочный файл в подходящую папку на своем компьютере и запустите его на выполнение. В окне **Installation Options** (рис. 1.5) установите флажок **Install Arduino software** (Установить программное обеспечение Arduino) (если он еще не установлен), а также флажки всех других опций, чтобы не устанавливать драйверы отдельно. Нажмите кнопку **Next** (Далее) и в следующем окне укажите папку, в которую следует установить программное обеспечение Arduino (рекомендуется выполнить установку в предлагаемую папку по умолчанию), и нажмите кнопку **Install** (Установить).

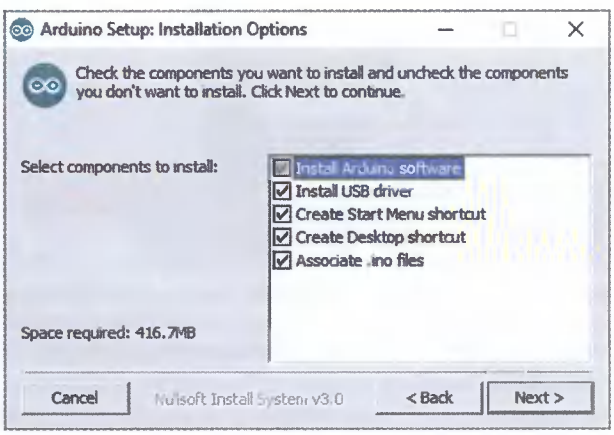


Рис. 1.5. Диалоговое окно **Installation Options** для установки программного обеспечения для Arduino. В обязательном порядке установите флажок для установки драйверов USB

Процесс установки может занять несколько минут, поэтому можете в это время сделать себе чашечку кофе. В зависимости от вашей версии Windows, в процессе установки может потребоваться дать разрешение на установку драйверов и, при желании, установить флажок, подтверждающий ваше доверие программному обеспечению от Arduino: **Always trust software from "Arduino srl"** (рис. 1.6).

Если вы не хотите подтверждать установку каждого драйвера по отдельности, установите этот флажок, разрешающий их установку по умолчанию. В любом случае нажмите кнопку **Install**, чтобы установить драйверы USB. Вот и все!

Установщик Arduino IDE обычно помещает ярлык среды программирования на рабочем столе. Щелкните на этом ярлыке двойным щелчком, чтобы запустить среду разработки Arduino IDE.

Рис. 1.6. Диалоговое окно разрешения установки драйверов

Рис. 1.7. Загруженный установочный пакет Arduino IDE будет сохранен в папке **Загрузки**. Щелкните опцию **Open in Finder**, чтобы переместить его в папку **Программы** (Applications)

Установка под OS X

Для компьютеров Mac щелкните ссылку на версию Arduino IDE для OS X и следуйте инструкциям на открывшейся странице.

Установка среды Arduino IDE

Загрузив установочный пакет, наведите курсор мыши на папку **Загрузки** и выберите опцию **Open in Finder** (Открыть в Finder) (рис. 1.7).

Затем выберите установочный файл Arduino IDE и перетащите его в папку **Программы** (Applications) (рис. 1.8). В большинстве случаев вам не придется устанавливать что-либо еще, и Arduino IDE можно будет запускать, как любую другую программу.

Ручная установка драйвера FTDI на OS X для платы RedBoard

Драйверов, установленных со средой разработки Arduino IDE, будет *достаточно для работы стандартной платы Arduino Uno*. Но для платы RedBoard от SparkFun необходимо вручную установить дополнительный драйвер. Для взаимодействия с компьютером в плате RedBoard

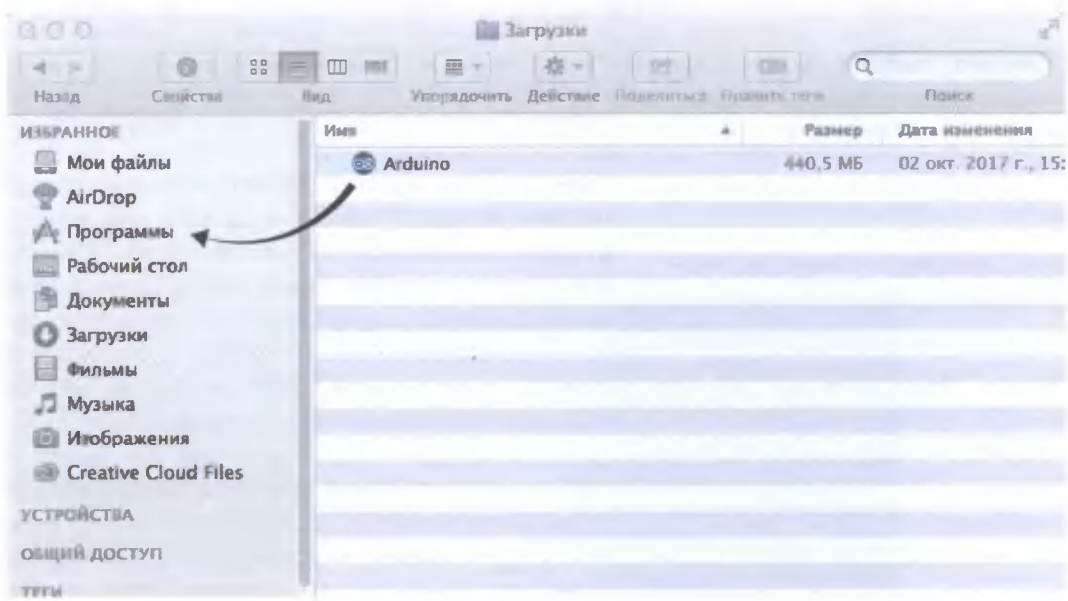


Рис. 1.8. Выберите установочный файл Arduino IDE и перетащите его в папку **Программы** (Applications)

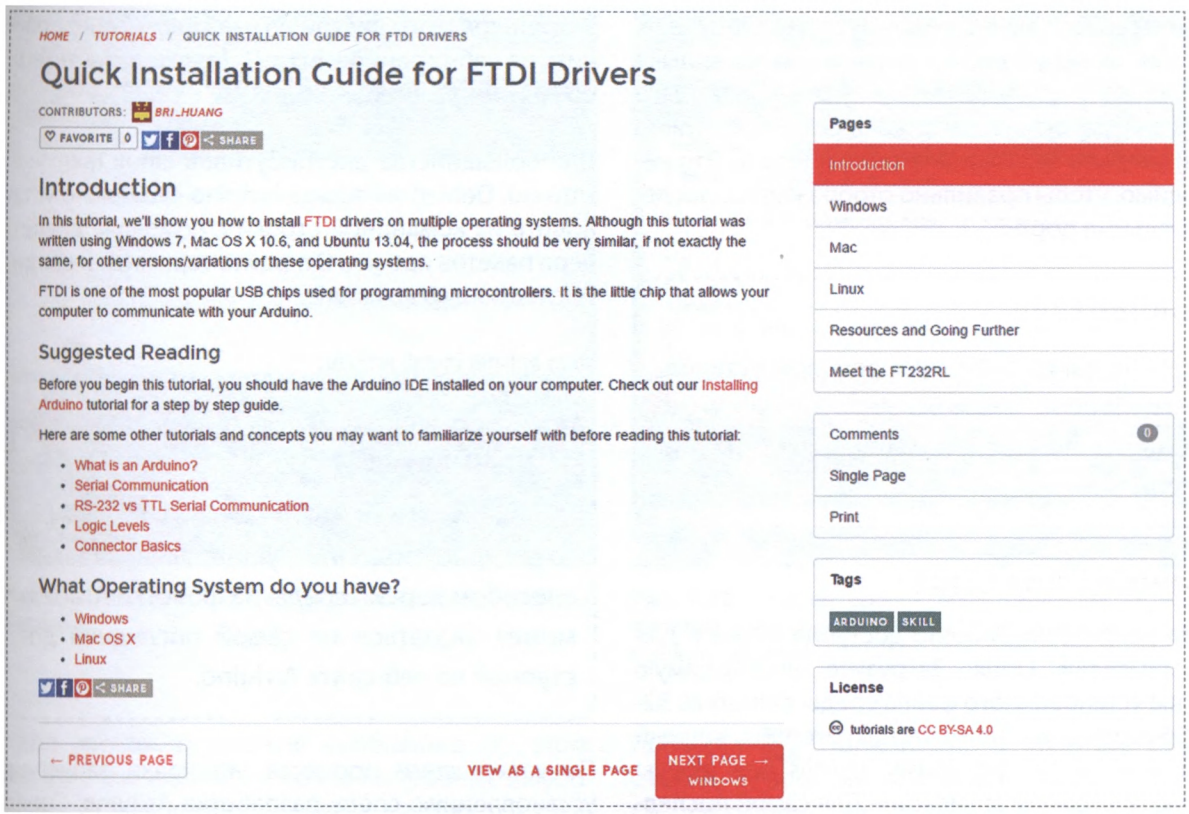


Рис. 1.9. Руководство по установке драйверов FTDI для платы RedBoard

от SparkFun используется микросхема USB компании Future Technology Devices International (FTDI). Вот для этой микросхемы и необходимо вручную установить драйвер. Для этого сначала откройте веб-страницу учебного пособия по установке драйверов FTDI (рис. 1.9) по адресу: **www.sparkfun.com/ftdi**.

Выберите ссылку для Mac OS X — будет выполнен переход в раздел с опциями выбора необходимого драйвера в зависимости от используемой версии OS X: один драйвер для версий операционной системы с Mac OS X 10.3 (Panther) по 10.8 (Mountain Lion) и другой — для версии OS X 10.9 (Mavericks) и более новых версий.

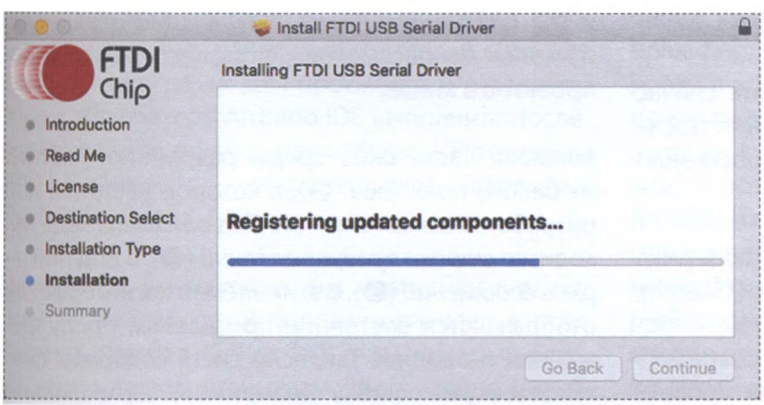


Рис. 1.10. Ручная установка драйвера FTDI на OS X

Загрузите необходимый драйвер, а затем щелкните на сохраненном файле двойным щелчком, чтобы запустить процесс установки. Откроется обычное окно установки программного обеспечения для Mac. Укажите жесткий диск, когда он будет обнаружен, и нажмите кнопку **OK**. Запустится процесс установки, статус которого будет отображаться в индикаторе выполнения (рис. 1.10). Драйверы будут установлены, когда индикатор дойдет до конца.

Вот и все! Теперь можно запустить среду разработки, щелкнув двойным щелчком на ярлыке Arduino в папке **Applications**. Если среда была запущена до установки драйверов FTDI, нужно будет закрыть ее, а затем снова запустить. Это необходимо, чтобы правильно отображались последовательные порты.

> **Примечание**
>
> В случае каких-либо проблем при установке драйверов, возможные решения можно найти по адресу: **www.sparkfun.com/macdriver**.

Установка под Linux

Среда разработки Arduino доступна также и для пользователей Linux. Загрузите необходимую версию установочного файла среды для Linux: 32- или 64-разрядную. Затем распакуйте загруженный файл, используя программу xz-utils или любую другую программу архивации. Для самой последней версии Arduino IDE для Linux, возможно, придется также установить некоторые другие зависимые программы. Информацию о зависимостях для конкретного дистрибутива Linux можно получить на веб-странице: **http://playground.arduino.cc/Learning/Linux**.

Для большинства дистрибутивов Linux (включая Ubuntu, Debian и Fedora) Arduino IDE можно установить из командной строки с помощью диспетчера пакетов apt-get. Откройте терминал и введите следующую команду:

```
sudo apt-get install arduino
```

> **Примечание**
>
> В зависимости от диспетчера пакетов вашего дистрибутива Linux, установленная таким способом версия среды разработки Arduino может оказаться не самой последней доступной на веб-сайте Arduino.

По завершении процесса установки запустите установленную среду разработки Arduino. Среда разработки Arduino работает на Java и должна запускаться в X Window или другой подобной среде пользовательского интерфейса.

Краткая экскурсия по среде разработки Arduino

Среда разработки Arduino используется для создания и отладки программ для Arduino. По терминологии Arduino программы называются *скетчами*. Среда разработки позволяет загрузить в Arduino скетч и управлять физическими устройствами.

Если вы этого еще не сделали, откройте сейчас установленную среду разработки Arduino. После начальной заставки откроется окно собственно IDE (рис. 1.11).

Панель меню (поз. ❶ на рисунке) содержит команды **Файл** (File), **Правка** (Edit), **Скетч** (Sketch), **Инструменты** (Tools) и **Помощь** (Help), с помощью которых можно открывать и сохранять файлы, загружать в Arduino код, редактировать параметры и т. п. Под строкой меню находится панель с кнопками команд (поз. ❷, слева направо): **Проверить** (Verify), **Загрузка** (Upload), **Новый** (New), **Открыть** (Open) и **Сохранить** (Save). Назначение этих меню и кнопок мы рассмотрим в процессе реализации проектов в книге.

Большая часть окна среды разработки состоит из белого поля (поз. ❸), в которое вводится код разрабатываемых скетчей. Под областью кода находится *строка предупреждений* (❹), а под ней — *окно сообщений* (❺). В этих элементах интерфейса отображаются состояние программы, предупреждения и ошибки. Так, если скетч содержит синтаксическую ошибку (например, неправильно набранную команду), в строке предупреждений

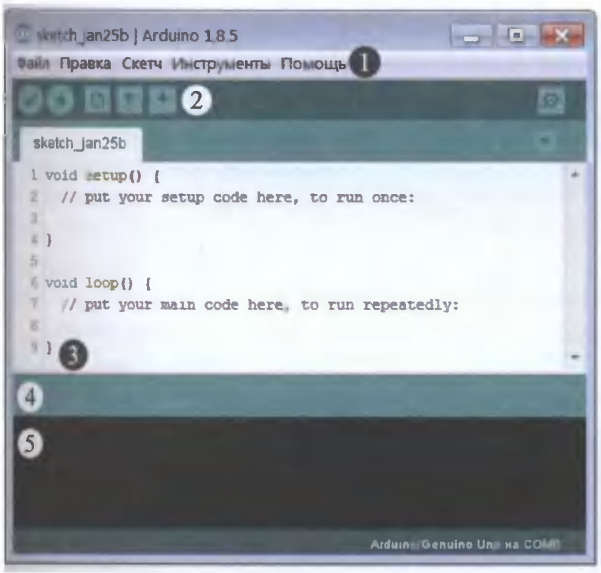


Рис. 1.11. Главное окно интегрированной среды разработки Arduino

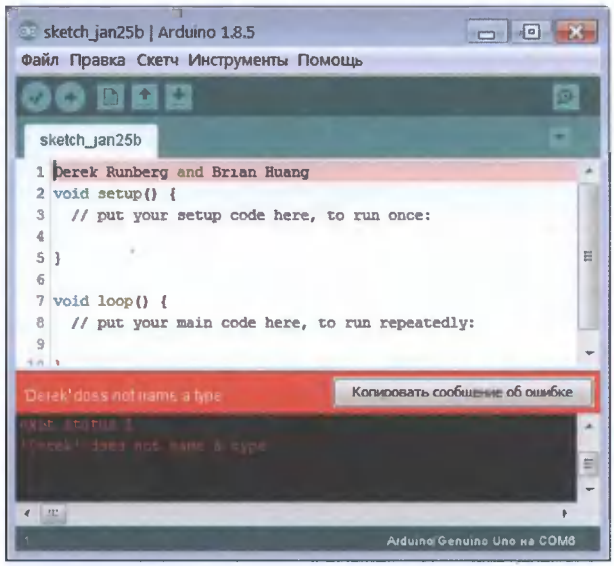


Рис. 1.12. Пример вывода предупреждения об ошибке в строке предупреждений и детальной информации о ней в окне сообщений

будет выведено краткое сообщение об этом, а в окне сообщений — более подробная информация, включая действия среды разработки касательно ошибки. Поясним это на более конкретном примере. Если в окне кода набрать какие-либо слова, например свое имя, а затем нажать кнопку проверки (**Проверить**), Arduino IDE задумается на мгновение, после чего выведет в строке предупреждений сигнал о наличии ошибки, а в окне сообщений — подробную информацию об этой ошибке (рис. 1.12).

Изменение настроек по умолчанию

Arduino IDE является полностью открытой и настраиваемой средой программирования. Некоторые из ее второстепенных параметров доступны для настройки пользователем, чтобы облегчить ему написание и отладку кода для создания программ. Для просмотра и настройки общих параметров Arduino IDE выполните последовательность команд меню **Файл | Настройки** (File | Preferences) — откроется окно **Настройки** (Preferences) (рис. 1.13).

Один из рекомендуемых для настройки параметров — размер шрифта редактора кода в поле **Размер шрифта** (Editor font size), который можно задать таким, чтобы вам было удобно читать содержимое редактора. Также будет полезным установить флажок для отображения нумерации строк **Показать номера строк** (Display line numbers) и снять флажок **Сохранять скетч при проверке или компиляции** (Save when verifying or uploading). Нумерация строк облегчает ориентировку в коде, а запрет на автоматическое сохранение файла скетча при каждой его проверке или загрузке в Arduino немного ускорит тестирование кода. Поскольку, как уже было отмечено ранее, платформа Arduino является полностью открытой, многие другие ее параметры также доступны для настройки посредством редактирования файла *preferences.txt* — см. опцию **Другие настройки можно редактировать непосредственно в файле** (More preferences can be edited directly in the file).

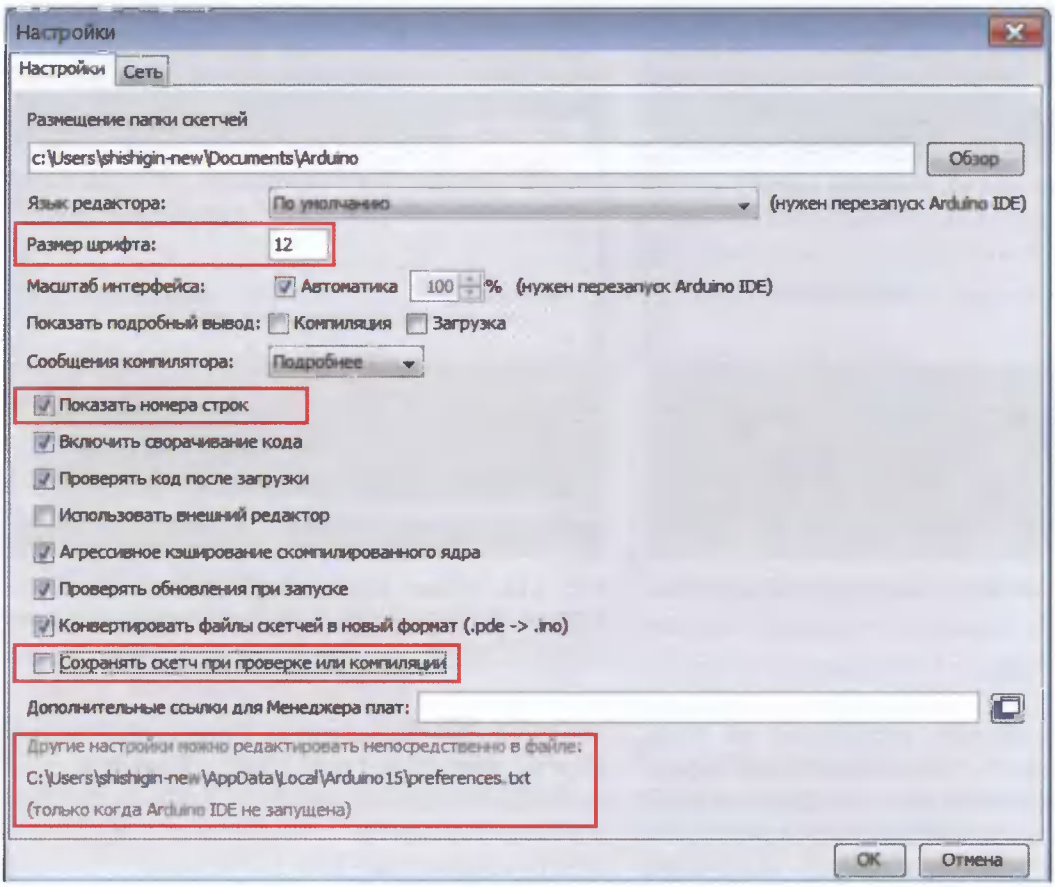


Рис. 1.13. Окно **Настройки** для настройки параметров Arduino IDE

Первое подключение Arduino к компьютеру

Установив среду разработки и драйверы, можно подключить плату Arduino к порту USB компьютера, используя соответствующий кабель. При подключении должен загореться светодиод питания (обозначен на плате меткой **ON**), а если плата не содержит никаких пользовательских программ, также должен мигать и светодиод, обозначенный меткой **13** (так называемый *светодиод 13*) (рис. 1.14).

> **Примечание**
>
> Если плата подключалась к компьютеру до установки среды разработки и драйверов, может понадобиться перезагрузить компьютер после установки этого программного обеспечения.

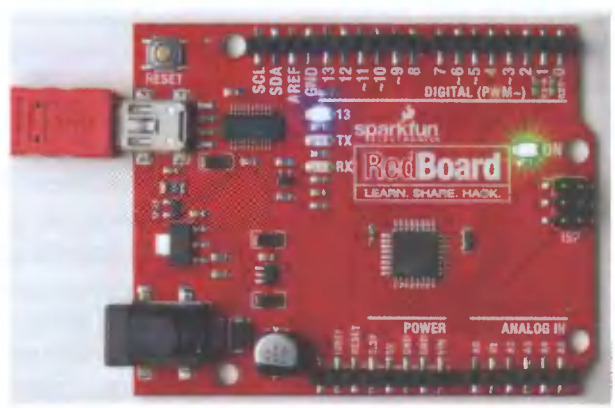


Рис. 1.14. При подключении к компьютеру платы, не содержащей пользовательских программ, кроме светодиода **ON**, мигает также светодиод **13**

В отличие от полноценного компьютера, Arduino может одновременно выполнять только одну программу (скетч). Плата Arduino получает питание от компьютера через кабель USB, которым она подключена к нему, и выполняет программу, загруженную производителем. Это стандартный скетч для проверки работоспособности платы, заключающийся в мигании светодиода. Когда плата Arduino подключена к компьютеру, в нее можно загружать другие скетчи, создаваемые в среде разработки.

Указание подключенной платы в IDE

В зависимости от компьютера или операционной системы, компьютеру может потребоваться некоторое время, чтобы определить подключенное к нему новое оборудование и сопоставить его с необходимыми драйверами. Когда компьютер определит подключенную плату, выберите команду меню **Инструменты** (Tools), а затем наведите курсор на опцию **Плата** (Board) — откроется список плат Arduino, поддерживаемых средой разработки (рис. 1.15).

Если вы используете стандартную плату Arduino Uno или плату SparkFun RedBoard, выберите опцию **Arduino/Genuino Uno**. В случае использования другой платы, выберите то название платы, которое прописано в ее документации. Как уже отмечалось ранее, в проектах этой книги предполагается использование платы Arduino Uno или производной от нее.

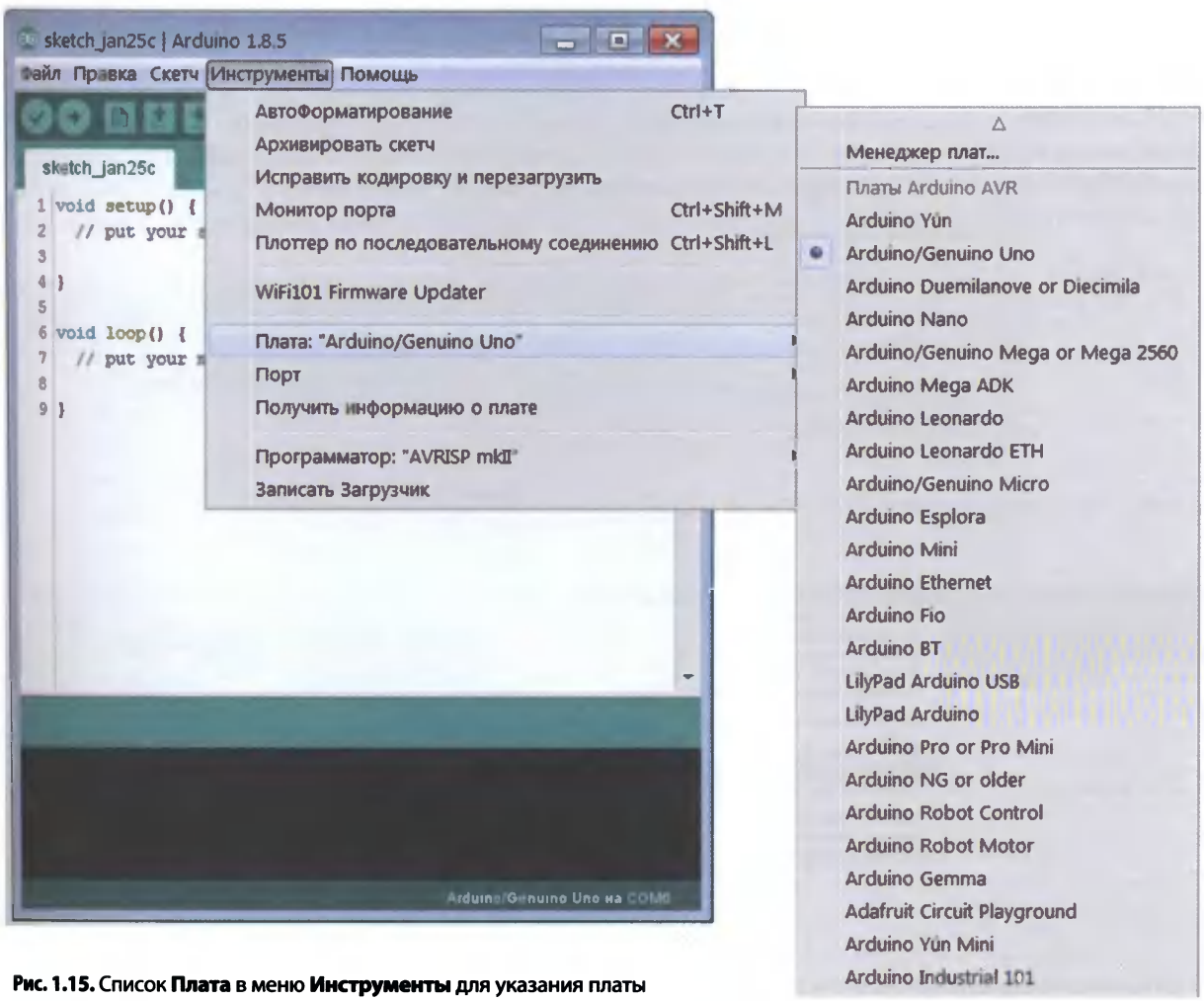


Рис. 1.15. Список **Плата** в меню **Инструменты** для указания платы

Выбор последовательного порта COM

Подключаемым к компьютеру устройствам присваивается уникальный идентификатор последовательного порта COM. Нам необходимо указать среде разработки Arduino номер последовательного порта, выделенного системой для связи с платой Arduino. Для этого сначала выполните последовательность команд меню **Инструменты | Порт** (Tools | Port). Доступные в результате выполнения этой команды опции будут зависеть от типа операционной системы.

Для Windows

Для компьютеров под Windows список будет содержать названия портов наподобие **COM3, COM4** и т. п. Выделенный для Arduino порт будет дополнительно обозначен (**Arduino/Genuino Uno**) (рис. 1.16). Выберите этот порт, щелкнув на нем мышью. В случае отсутствия опций выбора порта, обратитесь к *разд. «Поиск и устранение основных проблем с Arduino»* далее в этом проекте.

Для Mac OS X и Linux

Для компьютеров под управлением Mac или Linux последовательный порт будет отображаться в формате */dev/cu.usbserial-A<xxxx>*, где часть

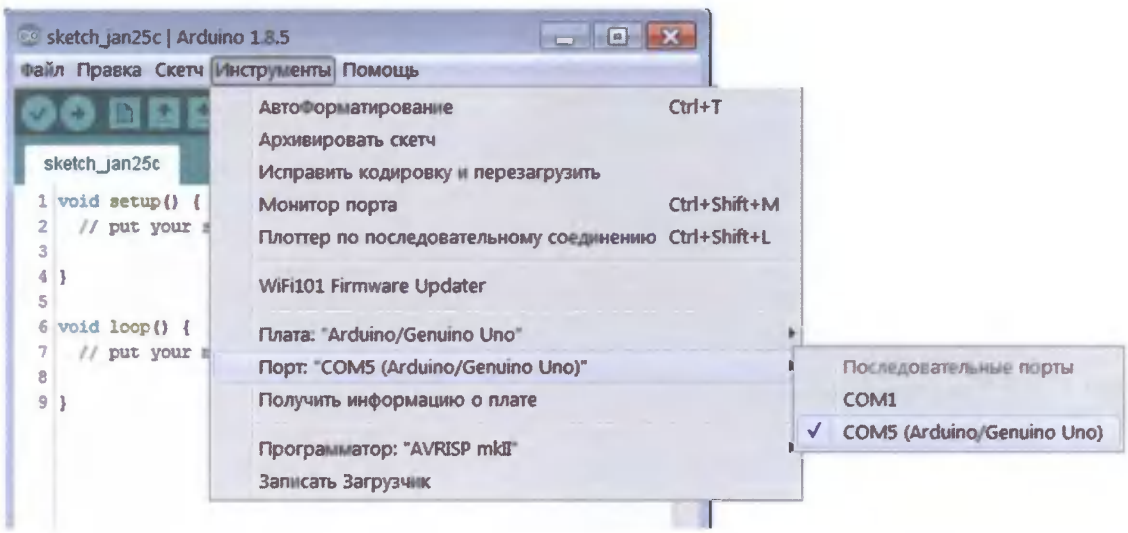


Рис. 1.16. Выбор последовательного порта COM для Arduino в компьютерах под управлением ОС Windows

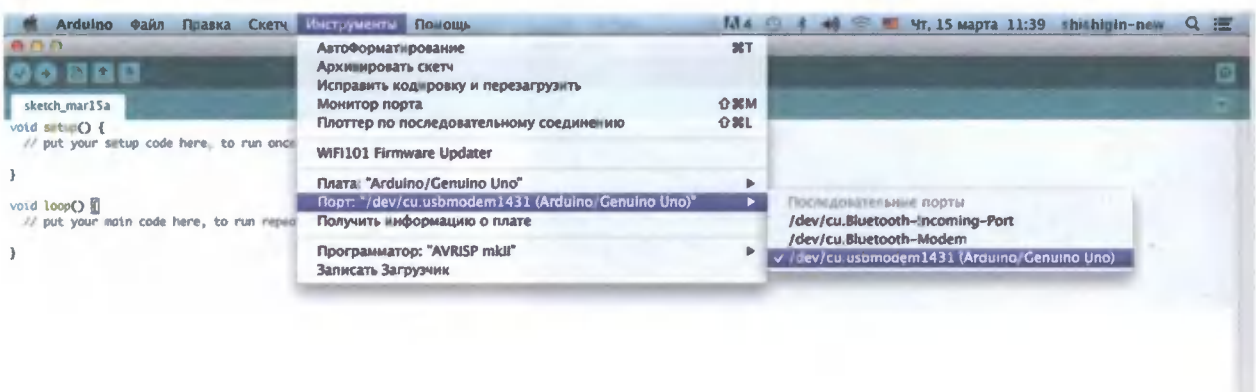


Рис. 1.17. Выбор последовательного порта COM для компьютеров под OS X и Linux

<xxxx> является строкой произвольных символов, уникальной для вашей платы Arduino. Выберите этот порт, щелкнув на нем мышью. Список может содержать несколько портов (рис. 1.17), но только порт с этой уникальной строкой ID будет сопоставлен Arduino. В случае отсутствия опций выбора порта, обратитесь к *разд. «Поиск и устранение основных проблем с Arduino»* далее в этом проекте.

Программа «Здравствуй, мир!» для Arduino

Программа «Здравствуй, мир!» является классической первой программой, создаваемой многими начинающими программистами. В большинстве языков программирования эта программа выводит на экран слова **Здравствуй, мир**. Но поскольку у Arduino нет экрана, то действием для него, сообщающим, что программа выполняется, станет мигание встроенного светодиода 13.

Для нашего первого скетча мы воспользуемся примером, входящим в состав Arduino IDE. Подключите плату к компьютеру, в меню **Файл** (File) выберите последовательность команд **Примеры | 01. Basics | Blink** (Examples | 01. Basics | Blink) (рис. 1.18) — откроется новое окно редактора скетчей с загруженным в него скетчем Blink.

В этом окне выполните последовательность команд меню **Скетч | Загрузка** (Sketch | Upload) или щелкните на значке **Загрузка** (Upload) (см. *разд. «Краткая экскурсия по среде разработки Arduino»*). Среда разработки преобразует сравнительно доступный для человеческого понимания код скетча в набор нулей и единиц, который понимает Arduino (этот процесс называется *компиляцией*), а затем загрузит этот набор в плату.

Итак, щелкнув на значке **Загрузка**, наблюдайте за выводимыми сообщениями в строке предупреждений. Сначала здесь должно выводиться сообщение **Компиляция скетча...** (Compiling sketch...), а в правой части строки будет отображен индикатор выполнения компиляции. После

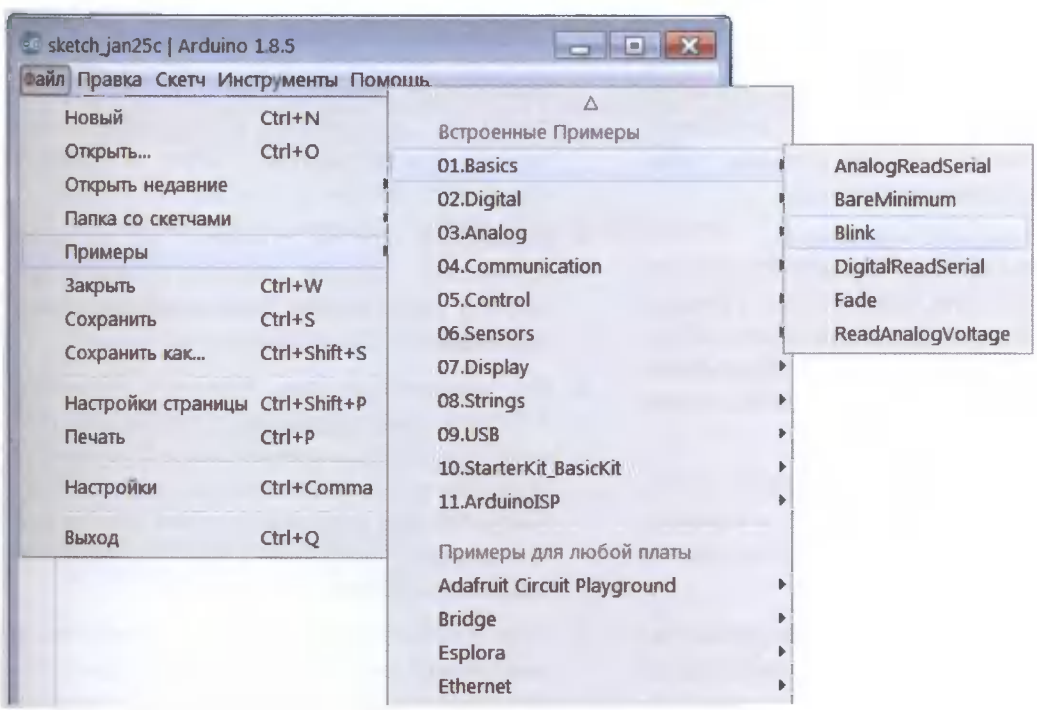


Рис. 1.18. Загрузка скетча Blink в редактор скетчей

завершения компиляции среда разработки станет загружать скомпилированный скетч в плату Arduino, и в строке отобразится сообщение **Загрузка** (Uploading). При этом на плате Arduino будут мигать светодиоды **RX** (прием) и **TX** (передача), подтверждая загрузку скетча в плату Arduino. Светодиод **RX** мигает, потому что Arduino принимает данные от компьютера, а светодиод **TX** — потому что Arduino отвечает компьютеру, подтверждая прием данных. По завершении загрузки скетча в плату в строке предупреждений выведется сообщение **Загрузка завершена** (Done uploading), светодиоды **TX** и **RX** мигать перестанут, и плата начнет исполнять загруженный скетч, мигая встроенным светодиодом 13 (рис. 1.19).

Рис. 1.19. Исполнение скетча: мигание светодиода 13

Если в процессе компиляции и загрузки что-либо происходит иначе, чем здесь описано, а в строке предупреждений выводятся какие-либо сообщения об ошибке, причиной может быть отсутствие взаимосвязи между компьютером и Arduino. Попытайтесь решить проблему, руководствуясь советами, изложенными в *разд. «Поиск и устранение основных проблем с Arduino»*, а затем попробуйте снова загрузить скетч в Arduino.

Поиск и устранение основных проблем с Arduino

Как и любое другое программируемое электронное устройство, плата Arduino может иногда вести себя не так, как от нее ожидается. Здесь мы приводим несколько советов для таких случаев, чтобы помочь решить возможные проблемы.

1. Убедитесь, что кабель USB надежно вставлен как в разъем на плате Arduino, так и в разъем компьютера. Довольно легко вставить кабель в разъем на плате лишь частично. Попробуйте извлечь кабель из гнезда и затем снова вставить его.
2. Убедитесь, что в пункте **Плата** указана плата, которая фактически подключена к компьютеру. Для примеров в этой книге указывается плата Arduino/Genuino Uno.
3. Проверьте, что в пункте меню **Инструменты | Порт** выбран правильный порт. Выбранный порт обозначается галочкой или точкой. Если вы не знаете, какой порт выделен для Arduino, извлеките кабель USB из разъема компьютера, обновите список портов (просто повторно выполнив последовательность команд **Инструменты | Порт**) и заметьте, какой порт пропал из списка.
4. Проверьте, что вы случайно не добавили какие-либо лишние символы в скетч примера. В таком случае скетч не будет компилироваться.
5. На компьютере под Windows проверьте в Диспетчере устройств правильность установки драйверов для Arduino. В частности, посмотрите, не обозначен ли порт COM для платы Arduino восклицательным знаком. В таком случае нужно будет вручную переустановить драйверы.
6. Если сообщения об ошибках продолжают выводиться, переустановите драйверы платы. Дополнительные инструкции доступны на веб-странице **www.sparkfun.com/ftdi/**.

Приведенные здесь шесть советов являются хорошей отправной точкой для новичков при решении наиболее распространенных проблем с Arduino. Если ни один из этих советов не поможет решить проблему, не паникуйте, сохраняйте спокойствие и помните, что не вы первый, кто испытывал проблемы с Arduino. Если вы не имеете ни малейшего понятия, как решить вставшую пред вами проблему, можно попытаться найти ее решение на официальном форуме Arduino по адресу: **http://forum.arduino.cc/**.

Анатомия скетча Arduino

Здесь мы пошагово рассмотрим скетч Blink, который загрузили в Arduino в *разд. «Программа "Здравствуй, мир!" для Arduino»*. Полный текст скетча приводится в листинге 1.1.

Листинг 1.1. Скетч примера Blink

```
❶ /*
    Blink
    Бесконечно включает светодиод на одну секунду,а затем выключает на одну секунду.

    Большинство плат Arduino оснащены встроенным светодиодом, который доступен
      для управления пользователем. На платах Arduino Uno и Zero этот светодиод
      подключен к цифровому выводу 13, а на платах MKR1000 – к цифровому выводу 6.
      Системной константе LED_BUILTIN присваивается правильное значение вывода
      подключения светодиода независимо от используемой платы.

    Информацию о том, какой вывод конкретной платы Arduino подключен
      к встроенному светодиоду, можно узнать в технических спецификациях
      платы на сайте https://www.arduino.cc/en/Main/Products.

    Этот код примера не защищен никакими авторскими правами.

    модифицирован 8 мая 2014 г.
    Скоттом Фитцджеральдом (Scott Fitzgerald)

    модифицирован 2 сентября 2016 г.
    Артуро Гвадалупи (Arturo Guadalupi)

    модифицирован 8 сентября 2016 г.
    Колби Ньюманом (Colby Newman)
  */

  //Функция 'setup' исполняется один раз при подаче питания на плату
  //или при нажатии кнопки сброса.
```

```
❷ void setup() {
    //инициализируем цифровой вывод LED_BUILTIN для вывода данных
    pinMode(LED_BUILTIN, OUTPUT);
  }

  //функция 'loop' исполняется в бесконечном цикле
❸ void loop() {
    digitalWrite(LED_BUILTIN, HIGH);        //включаем светодиод(высокий уровень напряжения)
    delay(1000);                            //ждем одну секунду
    digitalWrite(LED_BUILTIN, LOW);         //выключаем светодиод(низкий уровень напряжения)
    delay(1000);                            //ждем одну секунду
  }
```

При написании кода скетчей Arduino необходимо использовать специфические слова, знаки препинания и заглавные буквы. Эти элементы являются частью *синтаксиса* языка программирования. Чтобы среда разработки могла скомпилировать скетч должным образом, он должен быть написан словами и по правилам, которые эта среда понимает. Эти слова называются *ключевыми словами* — в зависимости от функции ключевых слов, в окне редактора среды разработки они отображаются шрифтами разного цвета: оранжевого, бирюзового или зеленого. Теперь давайте рассмотрим более подробно основные составные части нашего первого скетча.

Ключевые элементы скетча

Скетч начинается с объявления нового *глобального пространства имен* (поз. ❶ скетча). Это область, в которой описывается выполняемое скетчем действие и которая часто также содержит другую информацию, такую как инициализация переменных и библиотечные операторы. Практически все скетчи содержат пространство имен. Пространство имен этого скетча содержит *комментарии*, объясняющие на обычном человеческом языке, что именно этот скетч делает. В редакторе скетчей среды разработки Arduino текст комментариев отображается шрифтом серого цвета. Комментарии длиной в одну строку обозначаются двойной косой чертой (//) в начале комментария. Такие комментарии и ограничены одной строкой. Комментарии, занимающие более одной строки, заключаются в символы /* в начале комментария и */ в конце. Обратите внимание, что не все комментарии размещены на отдельных строках, — некоторые из них находятся на одной строке с текстом кода, который они поясняют. Это никоим образом не влияет на исполнение скетча, поскольку IDE игнорирует комментарии. В отличие от кода, в комментариях можно использовать любые слова, пунктуацию и орфографию.

Основу любого скетча Arduino составляет определение двух основных функций: setup() (поз. ❷) и loop() (поз. ❸). *Функция* — это просто группа инструкций или строк кода, выполняющих определенную задачу. Каждая функция имеет тип данных, имя и несколько инструкций. Определение функции начинается с объявления *типа данных*, возвращаемых функцией. Обе функции: setup() и loop() — имеют тип данных void, что означает, что они не возвращают никаких значений.

За типом данных следует *имя функции*, содержащее в конце открывающую и закрывающую скобки. В эти скобки помещаются параметры, которые передаются функции. Параметры представляют собой значения, требуемые функции для выполнения ее работы. Функции setup() и loop() не требуют

никаких параметров, но в дальнейших проектах мы увидим функции, которые используют их. Наконец, далее следуют строки кода, составляющие собственно функцию, которые заключены в фигурные скобки {}.

Каждый скетч Arduino должен содержать функции setup() и loop(). При подаче питания на плату или при нажатии кнопки сброса сначала один раз исполняется функция setup(), а затем начинает бесконечно исполняться функция loop(). Использование этих функций сродни выпеканию пирожков: инструкции в функции setup() приготавливают все ингредиенты и принадлежности, после чего функция loop() выпекает партию за партией пирожки, пока вы не выключите печь (т. е. Arduino).

Теперь давайте рассмотрим, что делает каждая строка кода в функциях setup() и loop().

Функция *setup()*

Код функции setup() для скетча Blink приводится в листинге 1.2.

Листинг 1.2. Код функции setup() для скетча Blink

```
void setup() {
  //инициализируем цифровой вывод
  //LED_BUILTIN для вывода данных
  pinMode(LED_BUILTIN, OUTPUT);
}
```

Эта функция содержит единственную строку кода, вызывающую функцию pinMode(). Контактные выводы 0–13 платы Arduino являются *контактами ввода/вывода общего назначения* — соответственно, их можно использовать как для ввода, так и для вывода данных. Режим работы любого из этих контактов задается функцией pinMode(). Для этого функции передаются два параметра: первый параметр указывает номер контакта — его значение может быть от 0 до 13, второй параметр определяет его конфигурацию.

Для обращения ко встроенному светодиоду в скетче Blink используется системная константа LED_BUILTIN, которая автоматически определяет необходимый контакт. Для большинства устройств Arduino константа LED_BUILTIN имеет значение 13. Обратите внимание, что в скетче эта константа отображается шрифтом сине-зеленого цвета заглавными буквами. Этот цвет обозначает, что LED_BUILTIN является ключевым словом с предопределенным значением, которое используется средой разработки.

Второй параметр (OUTPUT) — конфигурирует этот контакт для вывода данных. Обратите внимание, что ключевое слово OUTPUT также сине-зеленого цвета. Это означает, что это слово является еще одной константой, используемой в Arduino. Здесь есть еще несколько других вариантов, которые мы подробно рассмотрим в *проектах 4* и *9*, но на данном этапе будет достаточно знать, что этот код задает функционирование вывода в качестве OUTPUT для LED.

Простыми словами этот код можно выразить следующим образом: «контакт 13 будет выводить данные с Arduino».

Код вызова функции pinMode() завершается точкой с запятой, которая обозначает конец строки кода. Строки кода скетчей всегда должны заканчиваться точкой с запятой. Но если вы случайно забудете сделать это, не отчаивайтесь. Практически все программисты, одни чаще, другие реже, делают эту ошибку. Однако среда разработки Arduino всегда обнаружит ее и выдаст соответствующее сообщение, чтобы помочь вам определить ошибку и исправить ее.

Примечание

Имена функций пишутся с использованием так называемого *верблюжьего регистра*[1]. В этом формате имя переменной начинается со строчной буквы, а все остальные слова, составляющие имя переменной, начинаются с прописной буквы.

[1] От англ. camel case.

А ГДЕ ФУНКЦИЯ *MAIN()*?

Если вы немного знакомы с программированием и, в частности, с языками программирования C или C++, то можете задаться вопросом, где в скетчах Arduino функция main(). Загляните в папку программ Arduino — вы много узнаете о том, что в действительности происходит: после щелчка на значке **Verify/Compile** или **Upload** Arduino в фоновом режиме добавляет в код много других файлов, включая файл main.cpp. Далее приводится отрывок кода из файла main.cpp:

```
int main(void)
{
    init();
    initVariant();
#if defined(USBCON)
    USBDevice.attach();
#endif
❶   setup();
    for (;;)
    {
❷   loop();
    if (serialEventRun)
    serialEventRun();
    }
    return 0;
}
```

Видите, где происходит вызов функции setup() (поз. ❶)? Также обратите внимание, что функция loop() (поз. ❷) находится внутри бесконечного цикла, который Arduino реализует с помощью пустого цикла for(;;). Этим и объясняется ее бесконечное исполнение.

Функция *loop()*

Теперь давайте рассмотрим функцию loop(), которая бесконечно повторяет исполнение всех содержащихся в ней инструкций от первой до последней. Код функции приводится в листинге 1.3.

Листинг 1.3. Код функции loop() скетча Blink

```
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
            //включаем светодиод
            //(высокий уровень напряжения)
    delay(1000); //ждем одну секунду
    digitalWrite(LED_BUILTIN, LOW);
            //выключаем светодиод
            //(низкий уровень напряжения)
    delay(1000); //ждем одну секунду
}
```

Инструкция digitalWrite() позволяет нам включать и выключать выводы Arduino — такое действие называется *управлением состоянием вывода*. Эта функция также принимает два параметра: первый параметр указывает вывод (контакт), которым мы хотим управлять. В данном случае мы снова используем системную константу LED_BUILTIN. А второй параметр определяет состояние этого вывода. Чтобы включить подключенный к выводу светодиод, скетч Blink посылает на вывод сигнал высокого уровня (HIGH), а чтобы выключить — сигнал низкого уровня (LOW).

Вторая задействованная в функции loop() инструкция (delay()) приостанавливает исполнение скетча на количество миллисекунд, указанное в передаваемом этой инструкции параметре. Плата Arduino Uno и совместимые с ней платы, такие как плата SparkFun RedBoard, выполняют 16 миллионов инструкций в секунду. Это настолько быстро, что человеческий глаз не в силах заметить изменение состояния светодиода, если соответствующие инструкции просто следуют друг за другом. Инструкция *приостановления исполнения скетча* позволяет нам управлять длительностью текущего

состояния светодиода. Например, инструкция delay(1000) приостанавливает выполнение скетча на 1000 мс, прежде чем исполнять следующую команду.

Следующие две строки кода похожи на первые две: первая указывает Arduino выключить светодиод, а вторая создает задержку в 1000 мс, прежде чем исполнять следующую инструкцию. После исполнения последней инструкции функция loop() повторно исполняет все содержащиеся в ней инструкции, и так до бесконечности.

Примечание

Одним из лучших способов обучения с помощью кода примера будет его модификация. Попробуйте уменьшить значение времени задержки до 500 и щелкните на значке **Upload**. Как это повлияло на скорость мигания? А что, если передать функции delay() значение 5? Теперь интервал между включенным и выключенным состояниями светодиода будет всего лишь 5 мс. Можете ли вы видеть переход между этими состояниями? При какой самой меньшей задержке вы можете видеть мигание светодиода?

Наш первый аппаратный компонент

Итак, светодиод, встроенный в плату Arduino, мы запустили. На следующем шаге мы добавим в схему первый внешний компонент, которым в данном случае также станет светодиод. Как уже упоминалось, контактные выводы Arduino используются для ввода и вывода сигналов с микроконтроллера. Продемонстрировать вывод сигналов с микроконтроллера можно с помощью простого светодиода. Возьмите светодиод и внимательно рассмотрите его. Он должен выглядеть примерно так, как показано на рис. 1.20.

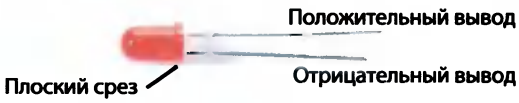


Рис. 1.20. Пример светодиода: длинный вывод — положительный, короткий — отрицательный

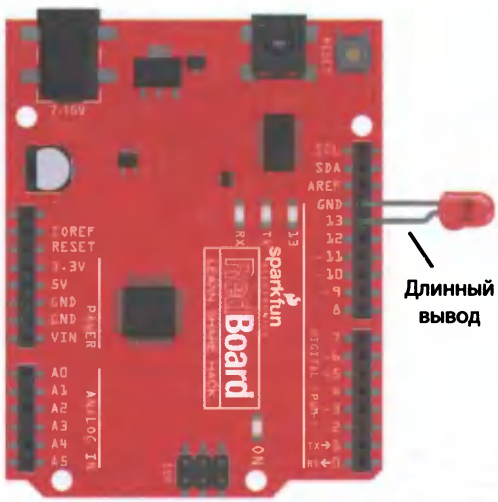


Рис. 1.21. Подключение внешнего светодиода к выводу 13 платы Arduino

Как можно видеть, выводы светодиода разной длины — один короче другого. Также при внимательном осмотре можно заметить, что фаска в основании светодиода срезана со стороны короткого вывода. Так указывается полярность выводов светодиода: длинный вывод является положительным (анодом), а короткий — отрицательным (катодом).

Вспомните, что константа LED_BUILTIN имеет значение 13, то есть обозначает вывод 13 на плате Arduino. Соответственно, чтобы подключить внешний светодиод к Arduino, нужно всего лишь вставить его длинный вывод в гнездо 13 Arduino, а короткий — в гнездо вывода «земли», которое на плате обозначено меткой **GND**[2] и расположено рядом с гнездом вывода 13. Светодиод можно вставить и при включенном питании платы. При правильном подключении (рис. 1.21) он сразу же должен начать мигать. Если светодиод не мигает, скорее всего вы перепутали выводы. Ничего страшного, просто извлеките его и снова вставьте, поменяв выводы местами.

[2] Сокращение от англ. Ground — земля.

Идем дальше...

Каждый проект этой книги завершается разделом с таким названием, в котором рассматриваются способы дальнейшего применения знаний, полученных в проекте. В этих разделах содержатся советы по использованию проекта в его текущем состоянии, экспериментированию с кодом, а также по модифицированию схемы проекта.

Экспериментируем с кодом

Попробуйте сделать так, чтобы светодиод мигал какими-либо занимательными последовательностями. Сначала скопируйте четыре строки кода функции loop() из листинга 1.1 и вставьте их после последней инструкции. В результате у вас будут две последовательности кода для мигания светодиодом, что даст больше возможностей для экспериментов. Так можно создавать последовательности мигания светодиода, изменяя время задержки как для включенного состояния светодиода, так и для выключенного. Например, последовательность, имитирующую биение сердца. Модифицированный соответствующим образом код исходного скетча Blink приводится в листинге 1.4.

Листинг 1.4. Код для последовательности мигания светодиода, имитирующей биение сердца

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(200);
  digitalWrite(LED_BUILTIN, LOW);
  delay(200);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(200);
  digitalWrite(LED_BUILTIN, LOW);
  delay(800);
}
```

Усложним задачу — попробуйте создать скетч Arduino для мигания букв своего имени с помощью сигналов азбуки Морзе, то есть последовательностями коротких (точка) и длинных (тире) миганий. Чтобы помочь вам с этой задачей, на рис. 1.22 приводится код Морзе для букв английского алфавита. Для начала попробуйте создать код для мигания «S-O-S»: точка, точка, точка — тире, тире, тире — точка, точка, точка.

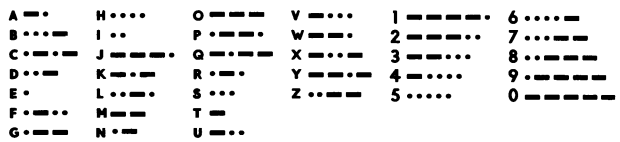


Рис. 1.22. Код Морзе для букв английского алфавита и цифр

Модифицируем схему

Мигающий светодиод дает нам целую кучу возможностей. Например, можно оснастить мигающими светодиодами какие-либо предметы у себя дома. В частности, хорошей идеей было бы украсить костюм для Хеллоуина каким-нибудь страшным существом с мигающими красными глазами. Для этого может потребоваться удлинить выводы светодиода, чтобы можно было спрятать плату Arduino где-либо в одежде (например, в кармане). Мы купили себе паука для Хеллоуина и встроили в него наводящие ужас мигающие красные глаза (рис. 1.23).

Рис. 1.23. Наводящий ужас паук с мигающими красными глазами

Управляемые мигающие светодиоды также хорошо подойдут и для масштабного моделирования. Использование светодиодов для фар автомобиля, освещения домов или уличных фонарей представляется прекрасным способом создать иллюзию реальности для любой масштабной модели или ландшафта (рис. 1.24).

Рис. 1.24. Масштабная модель автомобиля с фарами, управляемыми Arduino

Сохранение скетча

Кроме экспериментирования с исходным скетчем Blink, возможно, вы захотите продолжить экспериментировать и с его модификациями. Для этого сохраните их, присвоив им описательные имена, которые напомнят вам, что именно делает конкретный скетч. Имя файла скетча не должно содержать никаких пробелов — если вставить пробел, Arduino заменит его на символ подчеркивания: «_». По умолчанию среда разработки Arduino сохраняет скетч в папке Arduino, которая находится в папке Мои документы. Скетчи можно сохранять в любой другой папке, но гораздо удобнее, когда они все хранятся в одном месте.

Если вы уверены, что полностью освоили мигающий светодиодом скетч Blink и чувствуете теперь, что готовы расширить свои знания, переходите к *проекту 2*. В нем мы покажем вам, как создать светофор под управлением Arduino.

ДОМАШНИЙ СВЕТОФОР

Первым вашим большим шагом в сторону управления миром посредством встроенных электронных устройств была настройка среды разработки Arduino и создание в ней скетча для мигания светодиодом. Что ж, это был большой шаг для Вас лично, но на длинной дороге к овладению Arduino проект с одним светодиодом — всего лишь маленький шажок. Следующим шагом в этом направлении будет расширение нашего первого проекта и создание скетча и аппаратной инфраструктуры для работы с тремя мигающими светодиодами.

Если вы готовы к этому, тогда приступим к разработке светофора для регулировки оживленного движения в коридоре вашего дома (рис. 2.1).

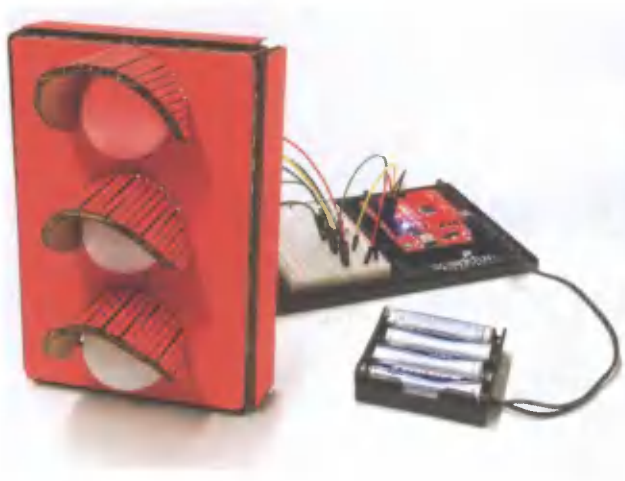


Рис. 2.1. Проект «Светофор» (Stoplight) в сборе

Рис. 2.2. Электронные компоненты для проекта «Светофор»

Необходимые компоненты, инструменты и материалы

Для этого проекта не требуется ничего сложного. Все электронные компоненты для него входят в стандартный состав набора изобретателя Inventor's Kit компании SparkFun, за исключением позиций, отмеченных звездочкой «*». Если вы используете свой набор компонентов или собираете их по отдельности, приведенный далее список необходимых электронных компонентов поможет вам определиться.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 2.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. (на рис. 2.2 не показан);
- беспаечная макетная плата (PRT-12002), 1 шт.;
- светодиоды (COM-12062): красный (1 шт.), желтый (1 шт.), зеленый (1 шт.);
- резисторы 330 Ом (COM-08377 или COM-11507 для пакета, содержащего 20 шт.), 3 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*;
- может пригодиться: держатель для четырех батареек типа AA (PRT-09835)* (на рис. 2.2 не показан).

Примечание

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

Прочие инструменты и материалы

Чтобы соорудить для электронной схемы корпус, чтобы он выглядел похожим на реальный светофор (см. рис. 2.1), вам потребуются дополнительные инструменты (рис. 2.3) и материалы (рис. 2.4).

Далее приводится краткий перечень этих инструментов и материалов:

- карандаш;
- макетный нож;
- металлическая линейка;
- плоскогубцы (бокорезы);
- инструмент для снятия изоляции;
- клей (клеевой пистолет или клей для моделирования);
- могут пригодиться: дрель и сверло диаметром 4,75 мм;
- может пригодиться: паяльник;
- может пригодиться: припой;
- может пригодиться: держатель «Третья рука»[1] (на рис. 2.3 не показана);
- гофрированный картон (приблизительно 30×30 см) или картонная коробка;
- мячик для настольного тенниса (2 шт.);
- шаблон корпуса (см. рис. 2.15 далее в этом проекте).

Рис. 2.3. Инструменты, рекомендуемые для проекта «Светофор»

[1] См. **https://ru.wikipedia.org/wiki/Третья_рука**.

Примечание

Для работы с проектами этой книги будет весьма полезно иметь хороший чистый картон. Рекомендуется приобрести несколько листов картона в магазине товаров для рукоделия.

Рис. 2.4. Материалы, рекомендуемые для проекта «Светофор»

Новый компонент: резистор

В предыдущем проекте (см. *проект 1*) мы подключили внешний резистор к плате Arduino напрямую, но в большинстве случаев светодиод лучше подключать через резистор, чтобы ограничить протекающий через него ток. Резисторы (рис. 2.5) используются практически во всех электронных устройствах и также потребуются для этого проекта.

Если вернуться к аналогии, представляющей электрический ток как воду, протекающую по трубе, то резистор можно представить в виде сужения в трубе, которое замедляет протекание воды (см. *разд. «Модель электрического тока: вода в трубе» на с. 4*). То есть резисторы ограничивают силу электрического тока.

Сопротивление измеряется в омах (и на схемах иногда представляется греческой буквой «омега»: Ω), разноцветные полосы на резисторах (см. рис. 2.5) обозначают номинальное значение сопротивления резистора. Описание системы разноцветных полос для обозначения номинального сопротивления резисторов и таблица значений полос приведены в *разд. «Полосатые резисторы» приложения*. Но для целей этой книги нам нужно определить только два разных значения резисторов: 330 Ом и 10 кОм. Резистор сопротивлением 330 Ом обозначается двумя оранжевыми полосами и одной коричневой (см. рис. 2.5), а резистор сопротивлением 10 кОм — коричневой, черной и оранжевой полосами. Четвертая полоса на резисторе обозначает погрешность его действительного значения по отношению к номинальному. При этом серебряная четвертая полоска означает, что действительное значение сопротивления резистора будет в пределах 5%, а золотистая — в пределах 10% погрешности от указанного номинального. Для проектов этой книги не требуется большая точность, поэтому для резисторов далее будет указываться только их номинальное значение, что будет достаточно при любой погрешности.

Некоторые электронные компоненты, включая светодиоды, можно повредить, пропуская по ним слишком большой ток. Чтобы защитить такие компоненты от повреждения, протекающий через них ток ограничивают с помощью резисторов. Поэтому, в целях повышения уровня безопасности наших проектов, мы всегда будем подключать токоограничивающие резисторы последовательно светодиодам, чтобы ограничить протекающий по ним ток до безопасного уровня и избежать их перегорания или даже, в худшем случае, взрыва. (Да, слишком большой ток может действительно заставить компоненты буквально взрываться.)

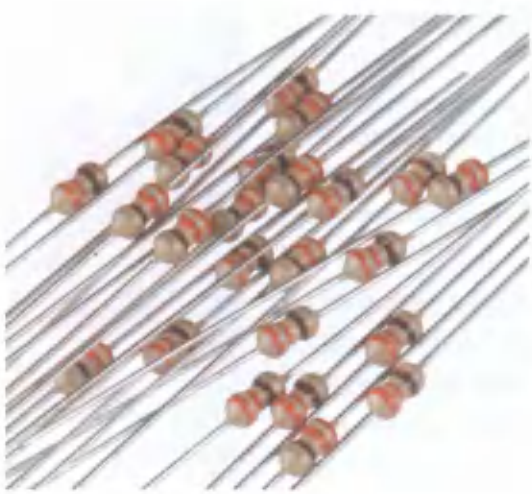


Рис. 2.5. Резисторы

ПОЧЕМУ В ПРОЕКТЕ «СВЕТОФОР» ИСПОЛЬЗУЮТСЯ РЕЗИСТОРЫ ЗНАЧЕНИЕМ 330 ОМ?

Средний светодиод красного цвета может выдержать нагрузку по току величиной около 20 мА. Обычно эти данные указываются в сопровождающей компонент документации. Чтобы защитить светодиод от токов больших, чем допустимые, последовательно с ним подключается резистор. Но откуда мы знаем, что нужно использовать резистор сопротивлением 330 Ом?

Включенные выводы Arduino выдают напряжение величиной 5 В. Для включения светодиодов разного цвета требуются немного разные напряжения обычно в диапазоне от 2,0 до 3,5 В. Так, для включения красного светодиода требуется 2 В, что делает 3 В (при напряжении питания 5 В) лишними. Эти 3 В рассеиваются в виде тепла на резисторе или на любом другом компоненте, подключенном последовательно в цепи. Рекомендуемой практикой является ограничение тока светодиода до примерно половины максимально допустимого, что для красного светодиода с максимальным током 20 мА будет составлять 10 мА. Значение величины сопротивления резистора для рассеивания 10 мА при напряжении 3 В можно рассчитать с помощью закона Ома (помните, что 10 мА = 0,01 А):

$$U = I \times R, \text{ откуда:}$$

$$R = \frac{V}{I} = \frac{3\,\text{В}}{0{,}01\,\text{А}} = 300\ \text{Ом}$$

Но значение 300 Ом не является стандартным значением резистора. Самым близким стандартным значением будет 330 Ом, и это вполне приемлемо. Использование резистора, подключенного последовательно в цепи светодиода, обеспечит долгое время беспроблемной работы светодиода. Поскольку этот резистор ограничивает ток, проходящий через светодиод, он и называется *токоограничивающим резистором*.

Если у вас есть под рукой резисторы разных значений, попробуйте использовать их со светодиодом и наблюдайте, что происходит. Резисторы с более высоким сопротивлением будут ограничивать ток в большей степени, а с меньшим — в меньшей. Что вы думаете будет, если вместо резистора сопротивлением 330 Ом использовать резистор 10 кОм?

Создаем прототип светофора

Теперь давайте приступим к созданию схемы светофора. Сначала рассмотрите его принципиальную схему, приведенную на рис. 2.6. Ее нужно смонтировать на беспаечной макетной плате, как показано на рис. 2.7.

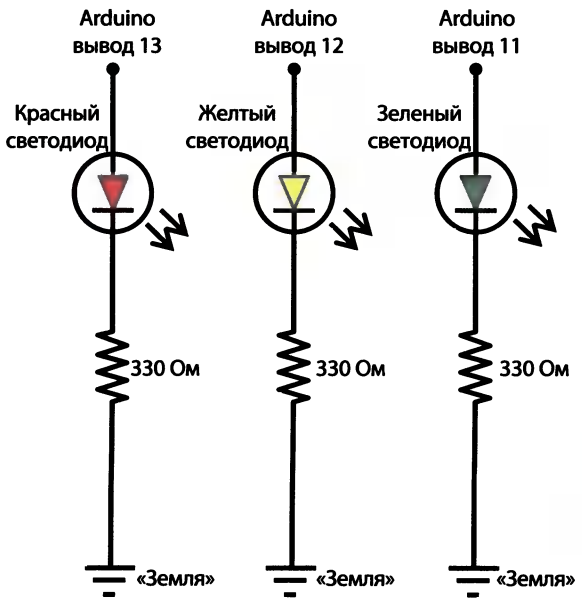


Рис. 2.6. Принципиальная схема проекта «Светофор»

Принципиальная схема (см. рис. 2.6) показывает электрическое подключение каждого компонента. Как можно видеть, выводы 13, 12 и 11 платы Arduino используются для управления отдельными светодиодами нашего проекта, при этом каждый светодиод подключен к отдельному резистору, который, в свою очередь, подключен к общему проводнику («земле»).

> **Примечание**
>
> Если вам необходимо освежить свои знания устройства и работы беспаечной макетной платы, вернитесь в *разд. «Создание прототипов схем»* на с. 5.

Подключаем красный светодиод

Теперь взглянем на монтажную схему проекта (см. рис. 2.7) и начнем преобразовывать принципиальную схему проекта в монтажную. В первом проекте мы заставляли мигать светодиод, встроенный в плату Arduino. Как мы помним, он подключен к выводу 13 на плате Arduino. Поскольку в этом проекте мы используем три светодиода, нам нужно самим выполнить их подключение к необходимым выводам платы. Следуя информации в принципиальной схеме (см. рис. 2.6) и в монтажной (см. рис. 2.7), подключите вывод 13 платы к положительному (длинному выводу) красного светодиода.

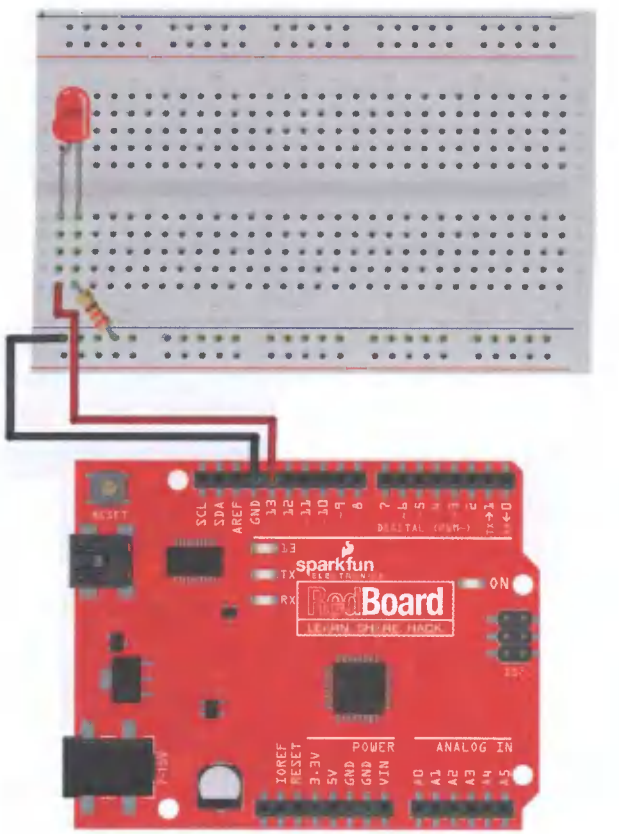


Рис. 2.7. Монтажная схема проекта «Светофор»: показано подключение красного светодиода, расположенного на монтажной плате, к выводу Arduino через токоограничивающий резистор

Чтобы выполнить это подключение, будет лучше сначала расположить плату Arduino рядом с беспаечной макетной платой именно так, как показано на рис. 2.7. (Такое размещение плат будет стандартным для всех проектов этой книги.)

Затем возьмите красный светодиод и резистор сопротивлением 330 Ом. Согните выводы резистора, как показано на рис. 2.8, чтобы его было легче вставить в гнезда беспаечной макетной платы. Рекомендуется при этом бокорезами обрезать выводы резистора наполовину, чтобы с ним было легче работать. В отличие от светодиодов, выводы резисторов не имеют полярности, поэтому не играет роли, какой вывод куда вставлять.

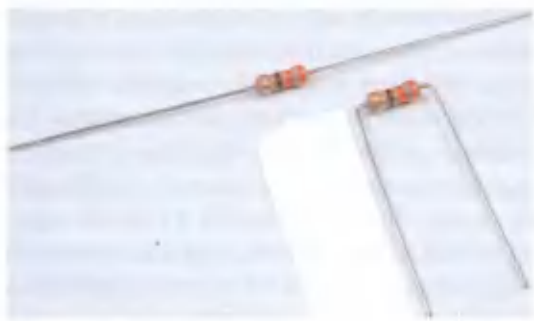


Рис. 2.8. Сгибание выводов резистора

Горизонтальные и вертикальные ряды большинства макетных плат для удобства обозначаются соответственно цифрами и буквами (рис. 2.9). Ориентируясь на эту маркировку, вставьте светодиод в беспаечную макетную плату, как показано на рис. 2.7. Длинный (положительный, или анод) вывод вставляется в гнездо столбца **e** ряда **1** (то есть, в сокращенной записи, в гнездо **e1**) макетной платы, а короткий (отрицательный, или катод) вывод — в гнездо столбца **e** ряда **2** (**e2**).

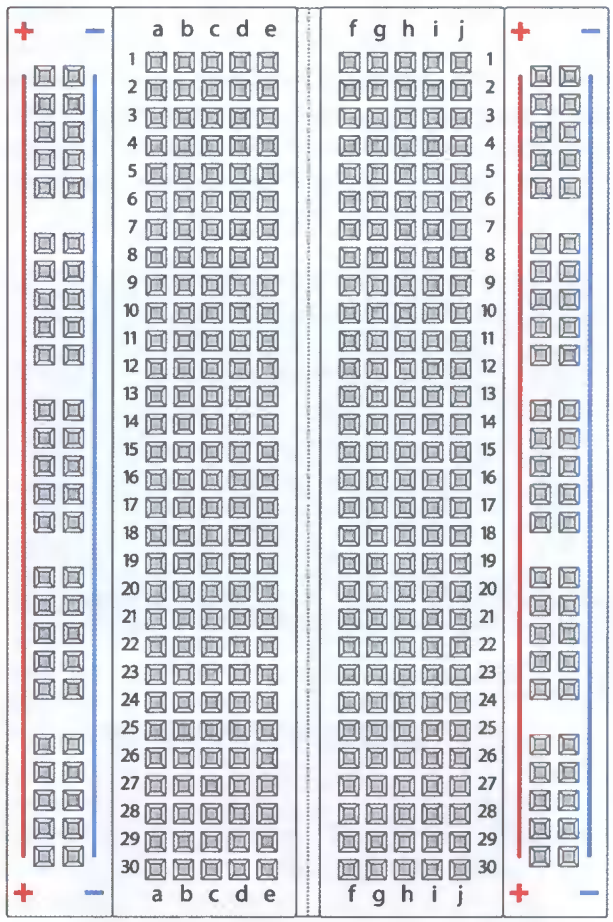


Рис. 2.9. Горизонтальные ряды гнезд беспаечной макетной платы обозначены цифрами, а вертикальные — буквами

Теперь очередь за резистором сопротивлением 330 Ом (с двумя оранжевыми и одной коричневой полосками). Вставляем один вывод резистора в любое гнездо в ряде **2** макетной платы, что подключает его к короткому (отрицательному) выводу светодиода (на рис. 2.7 этот вывод резистора вставлен в гнездо **a2** макетной платы). Помним при этом, что на всех стандартных беспаечных макетных платах гнезда столбцов **a** по **e** и **f** по **j** соединены между собой. Теперь вставьте другой вывод резистора в шину отрицательного питания макетной платы. Это будет столбец, обозначенный синей или черной линией и знаком минус (–).

Подаем питание на макетную плату

Возьмите две проволочные перемычки со штекерами на обоих концах. Рекомендуется, чтоб одна перемычка была черного цвета — для отрицательного напряжения, а другая, для положительного напряжения, красного цвета. Этот подход будет применяться во всех проектах этой книги.

Вставьте один штекер черной перемычки в гнездо GND («земля») платы Arduino (на плате Arduino есть три гнезда, обозначенные GND, — можно использовать любое из них), а второй — в гнездо шины отрицательного питания на макетной плате. Питание на каждый светодиод будет подаваться с цифровых выводов платы. Поскольку вывод **13** будет подавать питание на красный светодиод, вставьте один конец перемычки в гнездо **13** на плате Arduino, а второй — в гнездо **a1** на макетной плате.

Подключите плату Arduino к порту USB компьютера — должен начать исполняться загруженный в нее ранее скетч Blink из первого проекта, в результате чего красный светодиод начнет мигать раз в секунду. В принципе, мигать должны оба светодиода: и тот, что на макетной плате, и тот, который встроен в плату Arduino, поскольку они оба подключены к одному и тому же выводу 13.

Если светодиод на макетной плате не мигает, тогда как встроенный светодиод мигает, проверьте надежность и правильность монтажной проводки, а также полярность подключения светодиода. Убедитесь при этом, что короткий вывод светодиода вставлен в гнездо во втором ряду макетной платы, и что резистор вставлен одним выводом в гнездо в этом же ряду, а вторым — в гнездо шины отрицательного питания.

Когда схема заработает должным образом, то есть светодиод на макетной плате будет мигать, отключите плату Arduino от компьютера, чтобы можно было смонтировать остальную часть схемы. Рекомендуем вам обзавестить хорошей привычкой всегда обесточивать плату при выполнении монтажа компонентов на ней.

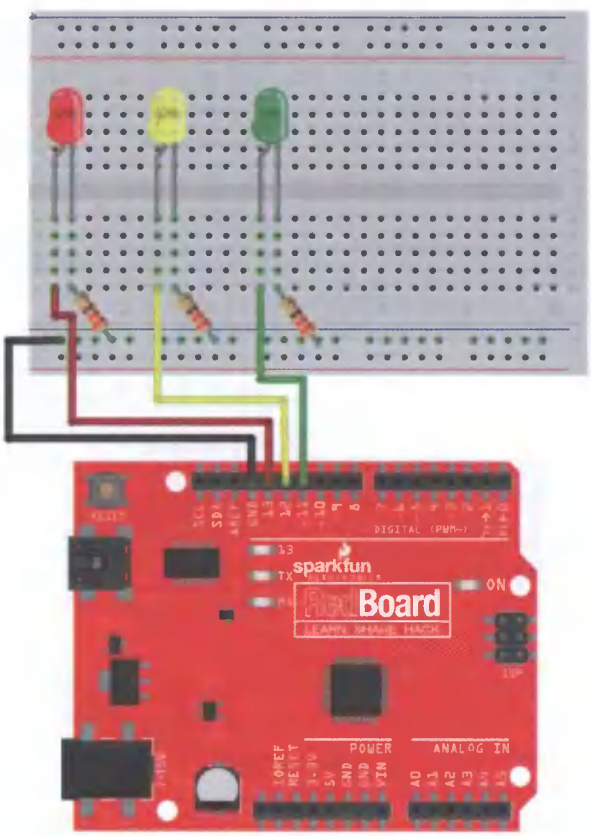


Рис. 2.10. Конечная монтажная схема прототипа проекта «Светофор»: светодиоды подключены к выводам 11, 12 и 13 платы Arduino

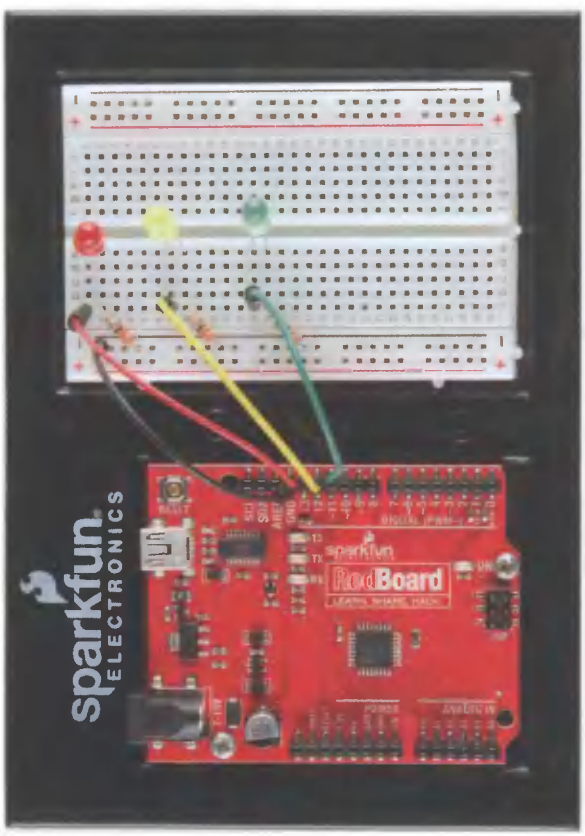


Рис. 2.11. Завершенный монтаж проекта «Светофор»: мы видим плату Arduino, светодиоды и резисторы

Добавляем желтый и зеленый светодиоды

Теперь подключите желтый светодиод к выводу 12 платы Arduino, а зеленый — к выводу 11. Следуйте тем же инструкциям, что и для подключения красного светодиода, но вставляйте каждый светодиод в отдельный ряд. К отрицательной шине питания макетной платы каждый светодиод, согласно принципиальной схеме, приведенной на рис. 2.6, подключайте через отдельный резистор. Конечный монтаж светодиодов показан на рис. 2.10. Обратите внимание, что возле каждого светодиода на макетной плате предусмотрено немного свободного пространства, чтобы компоненты не мешали выполнять монтаж.

Хотя здесь был изложен конкретный способ подключения компонентов на макетной плате, не обязательно следовать ему. Можно использовать любую часть макетной платы, при условии, что соединяемые компоненты вставляются в гнезда одного и того же ряда. Готовая схема проекта должна выглядеть наподобие показанной на рис. 2.11.

Впрочем, кроме внешнего подобия настоящему светофору, нам нужно реализовать и функциональное ему подобие, для чего следует включать каждый светодиод на определенное время, а затем выключать его и включать следующий, и так далее. Плата Arduino, разумеется, способна использовать в программах инструкции разных типов, включая команды синхронизации для управления схемами, поэтому она справится с этой задачей без каких бы то ни было проблем.

Программируем светофор

Снова подключите плату Arduino к компьютеру, чтобы начать программировать скетч светофора. Откройте среду разработки Arduino, чтобы создать новый скетч.

Проверьте параметры среды разработки

Прежде чем начинать создавать новый скетч, желательно всегда сначала выполнить определенные организационно-подготовительные работы. Итак, первым делом проверьте, что установлены правильные тип платы (**Board type**) и порт (**Port**). Для этого выполните последовательность команд меню **Tools | Board**. Если вы используете плату SparkFun RedBoard или стандартную плату Arduino Uno, выберите опцию **Arduino/Genuino Uno**. Затем выполните последовательность команд меню **Tools | Port**. В компьютере под Windows плате Arduino должен быть присвоен порт COM с самым большим значением. Для компьютеров Mac или Linux последовательный порт будет отображаться в формате */dev/cu.usbserial-A<xxxx>*, где фрагмент *<xxxx>* является строкой произвольных символов, уникальной для вашего Arduino.

Создаем переменные для номеров выводов

Удостоверившись в правильности параметров среды разработки, можно приступать к созданию скетча. Как пояснялось в *разд. «Анатомия скетча Arduino» проекта 1*, базовый скетч Arduino состоит из двух частей: функции `setup()` и функции `loop()`. Большинство простых скетчей обычно будут иметь такую базовую структуру, но более сложные скетчи содержат много разных составляющих. Вот и в скетче Stoplight («Светофор») присутствует новый раздел, называющийся *глобальным пространством имен*. Этот раздел расположен перед функцией `setup()` и не относится ни к одной функции. В этом разделе, среди прочего, определяются переменные, которые служат контейнерами для значений, предоставляя эти значения в любом месте скетча, где они требуются. В скетчах Arduino можно использовать значения (данные) нескольких типов.

Типы данных

В языке Arduino используются несколько разных типов данных, некоторые из них вам придется применять в своих скетчах довольно часто. Далее приводится список основных типов данных, их употребляемые в коде названия и краткое описание.

- **Целое число (int)** — целые числа в диапазоне от –32 768 до 32 767.
- **С плавающей запятой (float)** — числа с десятичной дробью в диапазоне от –3,4028235E+38 до 3,4028235E+38.
- **Байт (byte)** — целые числа в диапазоне от 0 до 255.
- **Символ (char)** — символ, обозначаемый заключением в одинарные кавычки, например, 'a'.
- **Строка (string)** — последовательность символов, обозначаемая заключением в двойные кавычки, например, «привет».
- **Булево (Boolean)** — значение, которое может быть или истиной или ложью, что соответствует значениям 1 или 0 в скетче и ВЫСОКИЙ или НИЗКИЙ в переводе на уровни напряжения выводов платы Arduino.

При определении переменной в Arduino необходимо указывать ее тип данных. Этот процесс рассматривается далее, в *разд. «Меняющиеся значения»*.

Меняющиеся значения

Большинство значений, используемых в скетчах, оформлены в виде *переменных*. Переменную можно представить как многоразовый контейнер для единицы данных, которая может быть цифрой, буквой или даже целым предложением.

Но прежде чем использовать переменную, ее необходимо *объявить*. Объявление переменной состоит в присвоении ей имени, указании ее типа данных и присвоении ей исходного значения. Рекомендуем вам присваивать переменной значение сразу при ее объявлении. Этот процесс выглядит следующим образом:

```
❶int ❷val = ❸10;
```

Объявление переменной состоит из трех частей: типа данных переменной ❶, ее имени ❷ и ее значения ❸. Строка кода объявления переменной завершается точкой с запятой, которая указывает среде разработки конец команды языка Arduino. Использование точки с запятой в конце команды является обязательным. Упущение этой детали часто является причиной многих ошибок компилятора, или, как они tot называются, багов[2]. Поэтому всегда будьте внимательны, чтобы не допускать эту ошибку.

В имени переменной можно использовать любую последовательность символов без пробелов, включая буквы и цифры. При этом, для первого символа имени переменной использовать цифры или специальные символы нельзя. Рекомендуется делать имена переменных как можно более описательными и в то же время как можно более короткими. При решении этой задачи следует, придумывая описания и сокращения, проявлять максимальную изобретательность. В данном примере переменная называется val (сокращение от английского value — значение), переменная инициализируется (ей присваивается начальное значение) значением 10. Инициализация переменной при ее объявлении не является обязательной, но выработать у себя такую привычку полезно.

В нашем проекте создаются три переменных для хранения номеров выводов, к которым будут подключены светодиоды, управляемые посредством Arduino. Работать с переменной, которая описывает цвет светодиода, намного легче и удобней, чем пытаться запомнить, какой светодиод подключен к какому выводу.

Итак, создайте новый скетч и вставьте код из листинга 2.1 в его глобальное пространство имен.

[2] От англ. bug — жучок. По легенде, от моли, застрявшей между контактами реле первого компьютера, что вызвало ошибку в его работе.

Листинг 2.1. Переменные для номеров выводов Arduino

```
byte redPin = 13;
byte ylwPin = 12;
byte grnPin = 11;
```

> **Примечание**
>
> Для удобства чтения при создании имен переменных используется верблюжий регистр, вследствие чего буква P в слове pin (вывод) пишется прописной. Верблюжий регистр позволяет разделять слова имени переменной без использования пробелов.

В этих трех объявленных нами переменных хранятся номера выводов, к которым подключаются светодиоды. Поскольку номера выводов в Arduino ограничены целыми числами в диапазоне от 0 до 13, мы используем тип данных byte. Использование этого типа данных возможно по той причине, что мы знаем, что номер вывода будет всегда меньше, чем 255. Обратите внимание, что имя каждой переменной описывает ее содержимое: redPin (красныйВывод) обозначает вывод Arduino для красного светодиода, ylwPin (желтыйВывод) — вывод для желтого светодиода и grnPin — вывод для зеленого светодиода. И, как можно видеть на рис. 2.10, номер вывода для красного светодиода — 13, для желтого — 12, а для зеленого — 11. Теперь в любое время, когда нам потребуется задействовать в нашем скетче номер вывода, мы можем использовать для этого описательное имя переменной.

Создаем функцию *setup()*

Продолжаем разработку скетча Stoplight («Светофор») добавлением в него функции setup(), текст которой приводится в листинге 2.2.

Подобно скетчу «Здравствуй, мир!» в *проекте 1* (см. функцию setup() в листинге 1.2), в скетче Stoplight («Светофор») функция setup() также выполняет конфигурирование цифровых выводов Arduino посредством функции pinMode().

Листинг 2.2. Код функции setup() для проекта Stoplight («Светофор»)

```
void setup()
{
  //красный светодиод
  pinMode(redPin ❶, OUTPUT ❷);
  //желтый светодиод
  pinMode(ylwPin, OUTPUT);
  //зеленый светодиод
  pinMode(grnPin, OUTPUT);
}
```

Поскольку в этом проекте задействуются три разных цифровых вывода, функция pinMode() вызывается в скетче три раза. В каждом отдельном вызове функции передаются в качестве параметров переменная номера вывода ❶ (redPin, ylwPin и grnPin) и константа функции вывода платы OUTPUT ❷. Константа режима вывода данных OUTPUT используется потому, что наш скетч управляет светодиодами, которые являются устройствами *вывода* данных. С устройствами *ввода* данных мы познакомимся в *проекте 4*.

Создаем функцию *loop()*

Далее создадим функцию loop(). Настоящие светофоры в цикле последовательно зажигают красный, зеленый, желтый огни, поэтому в этом проекте мы будем делать то же самое. Скопируйте код из листинга 2.3 и вставьте его в свой скетч на место функции loop().

Во избежание путаницы и аварий при проходе на светофор в коридоре, скетч Stoplight («Светофор») одновременно зажигает только один цвет светофора. Поэтому, когда включается очередной светодиод, все другие светодиоды должны быть выключенными. Например, если нужно зажечь красный свет, мы сначала вызываем функцию digitalWrite(redPin, HIGH), а затем функции digitalWrite(ylwPin, LOW) и digitalWrite(grnPin, LOW). Параметр HIGH в первом вызове функции

Листинг 2.3. Код функции loop() для проекта Stoplight («Светофор»)

```
void loop()
{
  //включаем красный свет
  digitalWrite(redPin, HIGH);
  digitalWrite(ylwPin, LOW);
  digitalWrite(grnPin, LOW);
  delay(2000);

  //включаем зеленый свет
  digitalWrite(redPin, LOW);
  digitalWrite(ylwPin, LOW);
  digitalWrite(grnPin, HIGH);
  delay(1500);

  //включаем желтый свет
  digitalWrite(redPin, LOW);
  digitalWrite(ylwPin, HIGH);
  digitalWrite(grnPin, LOW);
  delay(500);
}
```

переводит вывод redPin (вывод 13) платы в высокое состояние напряжения, включая подключенный к нему красный светодиод, а параметр LOW в двух последующих вызовах переводит выводы ylwPin и grnPin (выводы платы 12 и 11 соответственно) в низкое состояние напряжения, выключая желтый и зеленый светодиоды. Поскольку Arduino работает на рабочей тактовой частоте 16 МГц (исполняя приблизительно одну инструкцию каждые 16 миллионных секунды), задержка между этими командами составляет порядка нескольких микросекунд. То есть эти команды исполняются настолько быстро, что для всех практических целей можно считать, что они исполняются одновременно. Наконец, обратите внимание на функцию delay(2000). Эта функция приостанавливает исполнение скетча и удерживает красный светодиод включенным в течение 2000 мс или 2 секунд, прежде чем исполнится следующая инструкция.

Код для желтого и зеленого светодиодов работает по такому же принципу, устанавливая высокий уровень напряжения на соответствующем выводе платы и низкий на остальных и приостанавливая исполнение скетча на разные периоды времени. Попробуйте поэкспериментировать со значениями времени задержки в своем скетче Stoplight, соответствующими интенсивности движения в вашем коридоре. Помните, что значения, передаваемые функции delay(), устанавливают период времени в миллисекундах, в течение которого вы хотите удерживать светодиод во включенном состоянии.

Загружаем скетч в Arduino

Полный текст кода скетча для проекта Stoplight («Светофор») приведен в листинге 2.4. Вы до сих пор вводили в редактор код скетча частями (см. листинги 2.1–2.3). Теперь внимательно проверьте, что он соответствует коду в листинге 2.4, сохраните скетч, а затем загрузите его в Arduino, выполнив последовательность команд меню **Sketch | Upload** (Скетч | Загрузить) или нажав комбинацию клавиш <Ctrl>+<U>. В случае вывода средой разработки каких-либо сообщений об ошибках, снова проверьте свой код и убедитесь, что он в точности соответствует коду в листинге 2.4. Это означает, что он точь-в-точь повторяет написание каждого слова, использование заглавных букв и знаков препинания, не забудьте и о точке с запятой в конце каждой инструкции.

Когда все сделано правильно, светодиоды должны включаться и выключаться в цикле, подобном циклу работы настоящего светофора: сначала красный, за ним зеленый, а затем желтый на короткое время, после чего функция loop() начинает исполняться повторно, снова включая красный светодиод и т. д. Скетч должен работать таким образом бесконечно — до тех пор, пока на плату Arduino подается питание.

Листинг 2.4. Полный текст кода скетча для проекта Stoplight («Светофор»)

```
byte redPin = 13;
byte ylwPin = 12;
byte grnPin = 11;

void setup()
{
   pinMode(redPin, OUTPUT);
   pinMode(ylwPin, OUTPUT);
   pinMode(grnPin, OUTPUT);
}

void loop()
{
   //включаем красный свет
   digitalWrite(redPin, HIGH);
   digitalWrite(ylwPin, LOW);
   digitalWrite(grnPin, LOW);
   delay(2000);

   //включаем зеленый свет
   digitalWrite(redPin, LOW);
   digitalWrite(ylwPin, LOW);
   digitalWrite(grnPin, HIGH);
   delay(1500);

   //включаем желтый свет
   digitalWrite(redPin, LOW);
   digitalWrite(ylwPin, HIGH);
   digitalWrite(grnPin, LOW);
   delay(500);
}
```

Делаем светофор автономным

Когда плата Arduino подключена к компьютеру, питание на нее подается из порта USB компьютера. Но что, если мы хотим поместить наш проект в таком месте, куда не достает кабель подключения, или вообще взять его на какое-либо мероприятие, чтобы продемонстрировать свое творение друзьям? Для этого нам потребуется переносной источник питания, а именно портативный блок батареек. Вспомните, что плата Arduino оснащена цилиндрическим гнездовым разъемом питания, а также встроенным стабилизатором напряжения. Это делает возможным подключение платы к внешнему источнику питания напряжением в диапазоне от 6 до 18 В. На рынке предлагаются разнообразные держатели батареек, но авторам нравится показанный на рис. 2.12 держатель для четырех батареек типоразмера AA, который мы используем во многих наших проектах.

Рис. 2.12. Держатель на 4 батарейки AA с цилиндрическим разъемом

Извлеките кабель USB из разъемов компьютера и платы, вставьте батарейки в держатель батареек, а затем вставьте разъем держателя в разъем внешнего питания платы Arduino, как показано на рис. 2.13. Если батарейки заряжены, Arduino начнет исполнять загруженный в него скетч. При этом вы можете перемещать платы проекта куда угодно или же встроить их в модель светофора.

Рис. 2.13. Делаем проект «Светофор» автономным, используя держатель батареек в качестве источника питания

Теперь мы готовы перейти к реализации следующего уровня проекта, и далее, в *разд. «Создаем корпус для светофора»*, мы рассмотрим, как преобразовать этот прототип в модель светофора, который можно будет установить в местах с интенсивным движением в вашем доме.

Создаем корпус для светофора

Когда Arduino более не привязан к компьютеру, любой электронный проект на его основе можно встроить в постоянный корпус. Три мигающих светодиода на макетной плате лишь с трудом можно представить в виде светофора. Для большей реалистичности надо вставить платы проекта в соответствующий корпус и оснастить светодиоды линзами, которые сделают их лучше видимыми на расстоянии. Конечно, если вы хотите остановиться только на прототипе схемы, корпус и линзы делать не обязательно, но мы все же рекомендуем вам приобрести новый опыт изготовления корпусов для своих проектов.

Рис. 2.14. Корпус для модели светофора из картона и мячиков для настольного тенниса

Здесь мы покажем вам, как создать реалистично выглядящую модель светофора из гофрированного или открыточного картона, но можно использовать любой имеющийся под рукой материал. Проявляйте изобретательность! Наш пример, показанный на рис. 2.14, сделан из гофрированного картона, мячиков для настольного тенниса и некоторого умения работать руками.

Вы можете сделать корпус светофора сами, используя предоставленные здесь инструкции в качестве руководства. Впрочем, если вы хотите в точности воспроизвести этот проект, загрузите файл ZIP-архива, содержащий шаблоны и скетчи, по адресу: **https://www.nostarch.com/arduinoinventor/**. Все проекты в этой книге сопровождаются шаблонами, которые можно распечатать, перевести на картон и вырезать с помощью макетного ножа и металлической линейки.

Примечание

Если вам повезло, и у вас есть доступ к резальной машине наподобие Cricut, Silhouette Cameo или лазерному резаку, эти файлы можно легко приспособить под эти инструменты.

Распакуйте из сопровождающего книгу архива файлы *проекта 2* (папка ***P2_Stoplight***) и распечатайте файл шаблона ***P2_StopLightTemplate.svg*** в полный размер, чтобы его можно было использовать в качестве образца для вырезания корпуса. Затем вооружитесь предметами из *разд. «Прочие инструменты и материалы»* этого проекта и приступайте к работе над корпусом.

Делаем картонный корпус

Сначала вырежьте из распечатки шаблоны деталей корпуса, показанные на рис. 2.15. Корпус светофора вырезается из одного цельного куска картона, а затем сгибается по слегка прорезанным в картоне линиям.

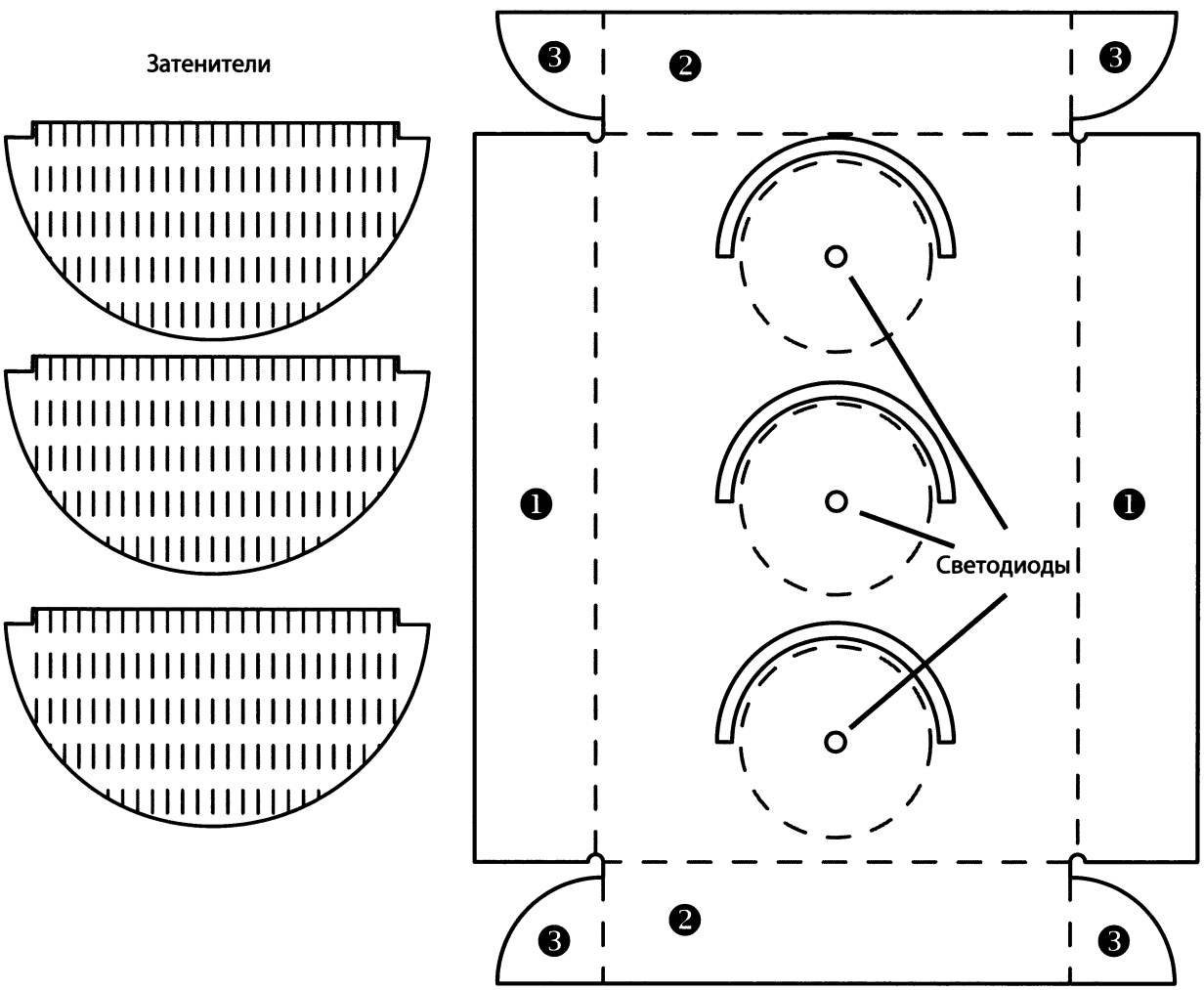


Рис. 2.15. Шаблон для корпуса модели светофора (в уменьшенном виде)

Рис. 2.16. Надрезание линий изгиба корпуса с помощью макетного ножа и металлической линейки

Рис. 2.17. Сверление отверстий под светодиоды

Рис. 2.18. Светодиоды, вставленные в заготовку корпуса модели светофора

Переведите сначала контуры шаблона на картон, а также наметьте пунктирные линии, используя для этого ручку или карандаш другого цвета. По этим линиям нужно будет сделать в картоне неглубокие прорези, чтобы затем по ним можно было согнуть корпус. Но пока не прорезайте их.

Обозначив контуры шаблона корпуса на картоне, вырежьте корпус по контуру с помощью макетного ножа и металлической линейки (рис. 2.16). Если вы никогда раньше не работали с макетным ножом, обязательно сначала прочитайте *врезку «Безопасная работа с макетным ножом»* чуть далее в этом проекте. Слегка надрежьте вырезанный корпус по пунктирным линиям на внешней стороне корпуса. Сделайте пару неглубоких проходов ножом, чтобы можно было с легкостью согнуть корпус по этим линиям, но не прорезайте картон насквозь. Также, пока не вырезайте затенители.

Вырезав контур корпуса и надрезав его линии изгиба, сделайте в нем отверстия для светодиодов в местах, обозначенных на шаблоне маленькими сплошными точками внутри больших пунктирных кругов. Можно, конечно, просто проколоть отверстия остро заточенным карандашом. Но чтобы получить более аккуратные отверстия, рекомендуется высверлить их с помощью электродрели и сверла диаметром чуть меньше 5 мм (рис. 2.17).

Дело в том, что диаметр светодиодов практически равен 5 мм. Но нам нужны отверстия чуть меньшего диаметра, в которые диоды можно легко вставить, причем так, чтобы они не выпадали из них. С этой задачей отлично справится сверло диаметром около 4,75 мм.

Будьте осторожны при сверлении отверстий под светодиоды — в частности, смотрите, чтобы ваши пальцы с тыльной стороны картона не располагались под сверлом. Дырка в одном из них вам будет совсем ни к чему. Если у вас нет дрели или вы не уверены в своих способностях использовать ее, отверстия можно также просверлить без дрели — одним сверлом, вращая его пальцами.

Просверлив отверстия, извлеките светодиоды из макетной платы и вставьте их в просверленные

БЕЗОПАСНАЯ РАБОТА С МАКЕТНЫМ НОЖОМ

Вам придется часто использовать макетный нож (рис. 2.19) при работе над проектами этой книги, поэтому важно знать, как с ним безопасно работать. Как и любой другой инструмент, при неправильном использовании макетный нож может нанести телесные повреждения.

Рис. 2.19. Макетный нож

Далее приводятся несколько советов по безопасной работе с макетным ножом.

- При резке листовых материалов всегда тяните лезвие на себя. При толкании лезвия от себя или в любом другом направлении оно может соскользнуть или сломаться.
- Не спешите! Не пытайтесь прорезать материал через всю его толщину за один проход. Сделайте несколько проходов, прилагая среднее давление на лезвие. Это будет более щадящим для лезвия, а также позволит получить более аккуратные резы.
- Используйте прямую металлическую направляющую для лезвия — например, металлическую линейку. Использование деревянной или пластмассовой направляющей чревато возможностью врезания лезвия в направляющую с последующим отскакиванием от нее, возможно, в направлении вашей руки и в саму руку.
- Не держите пальцы на пути лезвия. При всей очевидности этого совета, люди умудряются порезаться именно таким образом.
- Не пытайтесь поймать нож, который начал скатываться со стола, — позвольте ему упасть, а затем просто поднимите с пола. При попытке поймать его прежде, чем он упадет, вы подвергаете себя риску уколоть лезвием руку. Это может быть очень больно.
- Наконец, используйте новые, острые и неповрежденные лезвия. Немедленно заменяйте сломавшиеся, а также затупившиеся лезвия. Резка бумаги и картона затупляет лезвия очень быстро. Всегда имейте под рукой запасные лезвия, и когда резать становится трудно, замените лезвие новым.

отверстия с тыльной стороны заготовки корпуса модели светофора, как показано на рис. 2.18. Помните, что в настоящем светофоре цвета идут в порядке: красный, желтый, зеленый — сверху вниз. Запомните, как эти светодиоды были подключены к макетной плате, поскольку чуть позже их нужно будет снова к ней подключить.

Теперь согните заготовку корпуса модели светофора вдоль надрезанных линий, как показано на рис. 2.20: сначала согните внутрь вертикальные стороны ❶, а затем верхнюю и нижнюю стороны ❷ и язычки ❸ (нумерация частей заготовки корпуса светофора приведена на рис. 2.15).

Разместите язычки ❸ на внутренней стороне вертикальных сторон ❶, а затем приклейте их, как показано на рис. 2.21. Для этого можно использовать клеевой пистолет, скотч-ленту или обычный канцелярский клей. Клеевой пистолет будет предпочтительней, поскольку это более удобно — клей затвердевает быстрее и держится довольно крепко.

Рис. 2.20. Сгибание сторон корпуса светофора по надрезанным линиям

Когда вы приклеите все язычки, у вас должна получиться неглубокая прямоугольная коробка, открытая с тыльной стороны.

Делаем линзы для светофора

Линзы для светодиодов светофора делаются из половинок мячиков для настольного тенниса, но можно взять и любой другой более или менее прозрачный материал.

Если используются мячики для настольного тенниса, аккуратно разрежьте их пополам. Для этого разместите мячик на коврике для резки или на толстом куске картона и крепко держите его пальцами с обеих сторон. Осторожно проколите мячик лезвием ножа сверху вниз (при этом будьте внимательны, чтобы лезвие не было направлено на удерживающую мячик руку) и начните его разрезать. Для этого понемногу вращайте мячик, разрезая его в процессе поворота, повторяя вращение, пока полностью не разрежете мячик. Непременно держите пальцы вдали от лезвия и всегда режьте на коврике для резки или на куске картона.

Рис. 2.21. Гибка и склейка картонного корпуса модели светофора

Рис. 2.22. Безопасный способ разрезки мячика для настольного тенниса

Рис. 2.23. Приклеивание половинок теннисных мячиков к корпусу светофора

Рис. 2.24. Надрезание затенителей

Когда у вас образуются три половинки теннисного мячика (вообще-то, их будет четыре, но четвертую можно будет использовать для будущих проектов или сделать из нее шляпку для вашей любимой мягкой игрушки), приклейте их к корпусу светофора каплей клея, как показано на рис. 2.23.

Делаем затенители

Наконец, делаем и вставляем в светофор затенители. Чтобы получить хороший, правильный изгиб, надрежьте заготовки затенителей несколькими параллельными резами с интервалом между ними около 3 мм (рис. 2.24). На шаблоне затенителей (см. рис. 2.15) имеются соответствующие линии, по которым можно делать такие надрезы.

Согните надрезанные затенители в правильную дугу, как показано на рис. 2.25.

Вставьте согнутые затенители в корпус светофора — в отверстия над каждой линзой светодиода (рис. 2.26), а затем проклейте место соединения. Чтобы получить еще более реалистичную модель светофора, корпус можно окрасить пульверизатором в черный цвет. При этом обязательно удалите линзы или закройте их защитной липкой лентой, чтобы на них не попала краска.

Рис. 2.25. Сгибание затенителя в дугу

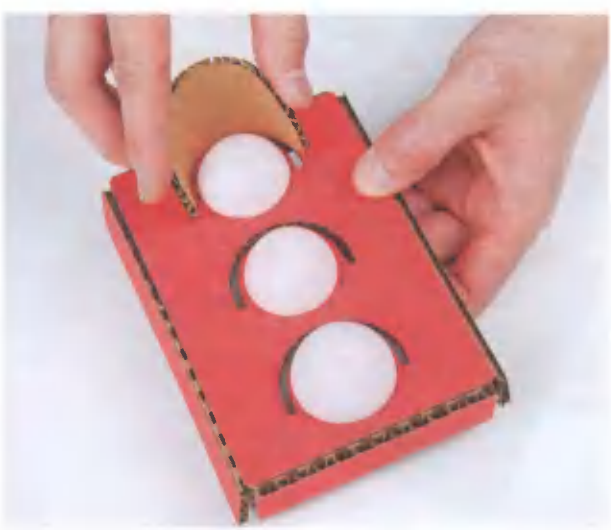


Рис. 2.26. Вставка затенителя в корпус

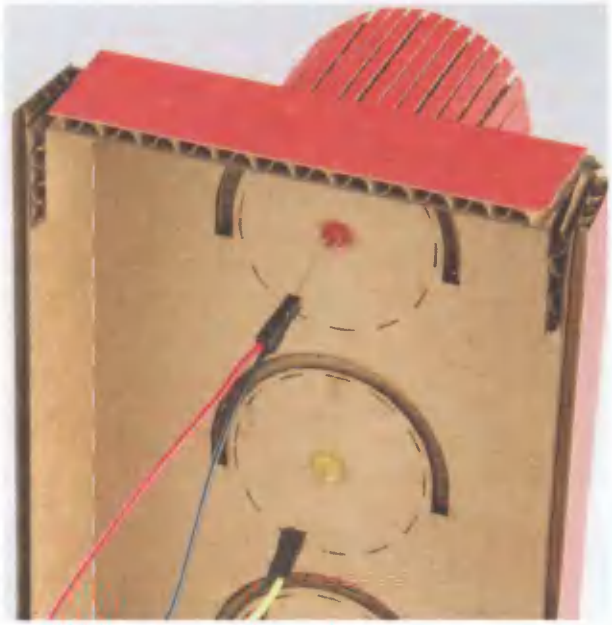


Рис. 2.27. Удлинение выводов светодиодов с помощью проволочных перемычек

Вставляем светодиоды и подключаем Arduino

Все, что нам осталось сделать, — это подключить ранее вставленные в корпус светофора светодиоды к Arduino. Чтобы подключать их было удобнее, необходимо удлинить выводы светодиодов. Для этого мы воспользуемся проволочными перемычками с гнездовым разъемом на одном конце и штыревым на другом (SparkFun PRT-09385 или самодельные). Всего нам нужно будет шесть таких перемычек. Выводы светодиодов мы удлиняем, просто надевая гнездовые разъемы перемычек на выводы светодиодов. Чтобы не было путаницы с проводами, используйте перемычки одного цвета (например, черного) для отрицательных (коротких) выводов светодиодов и какого-либо отличного от него цвета — для положительных (длинных) выводов (рис. 2.27).

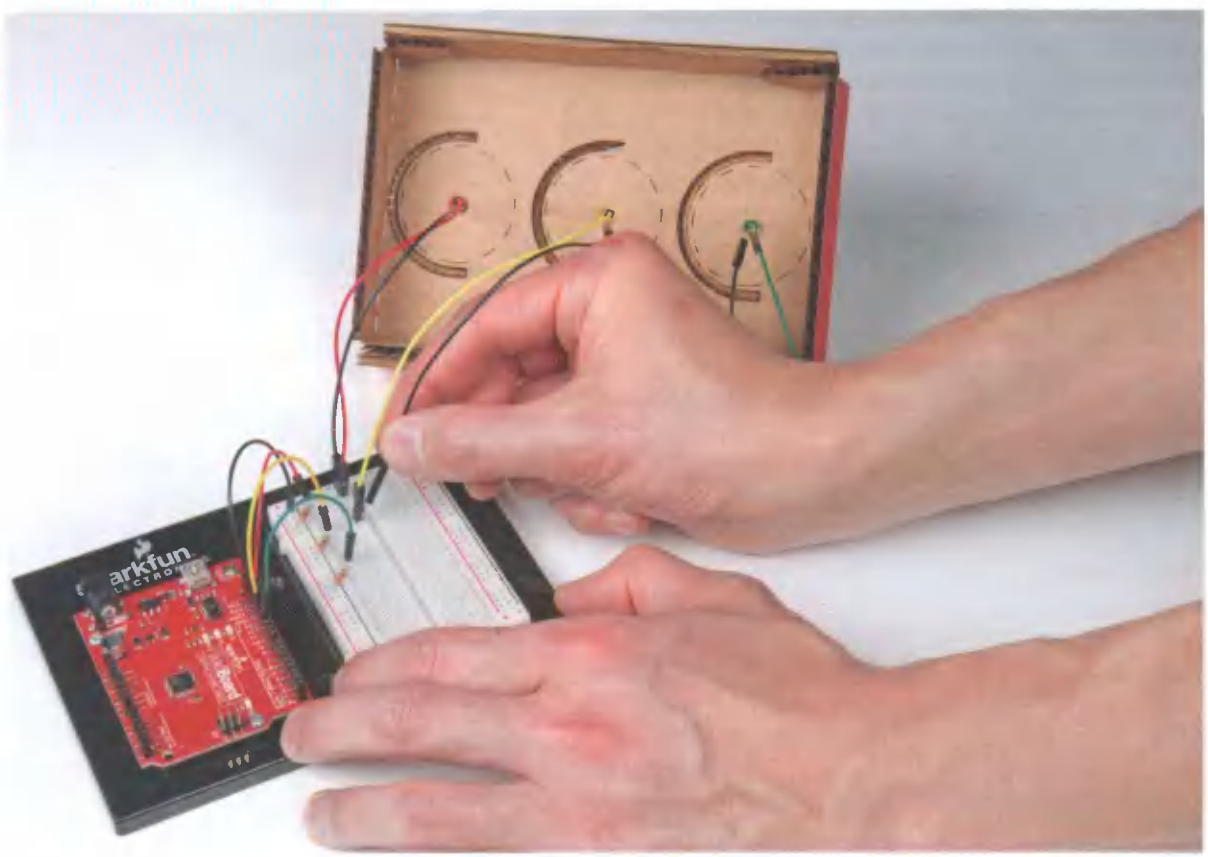


Рис. 2.28. Подключение удлинителей светодиодов к макетной плате

Затем подключаем другие концы проволочных перемычек к плате Arduino, вставляя штыревые разъемы от каждого светодиода в гнездо на плате, в которое этот светодиод был ранее вставлен (рис. 2.28). Опять же, будьте внимательны, чтобы не перепутать, какой светодиод подключать в какое гнездо. Если не помните, то сверьтесь с монтажной схемой на рис. 2.10.

Проверьте надежность соединений, подключив Arduino к компьютеру или к блоку батареек. Если какой-либо из светодиодов не включается, попробуйте пошевелить разъемы перемычек в местах подключения, а также еще раз убедитесь, что они вставлены в правильные гнезда на макетной плате. Модель светофора в ее завершенном виде показана на рис. 2.29.

Плату Arduino вместе с макетной платой можно или оставить вне корпуса светофора, или же прикрепить их внутри с помощью клея или двусторонней липкой ленты. В любом случае включите питание своего светофора и установите его на каком-либо перекрестке с интенсивным движением, чтобы сделать его безопасным.

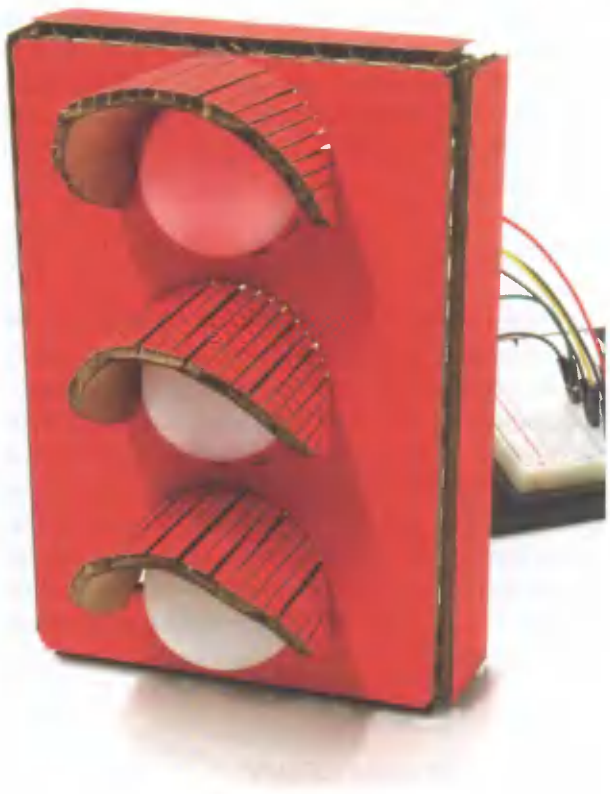


Рис. 2.29. Завершенный проект светофора

Идем дальше...

Возможностям, с которыми мы познакомились в процессе реализации проекта «Светофор» (Stoplight), — таким как синхронизация (тактирование) выходного сигнала для управления светодиодами, можно найти разнообразное применение как в доме в частности, так и в жизни в целом. Далее приводится несколько рекомендаций по другим применениям возможностей проекта «Светофор».

Экспериментируем с кодом

Основной возможностью, использованной нашим светофором, является отсчет времени с помощью таймера. Где еще можно применить таймер? Как насчет модифицирования кода, чтобы помочь с правильной выдержкой времени для варки яиц? В таком случае красный светодиод остается включенным, пока яйцо еще сырое, желтый зажигается, когда оно почти готово, а зеленый — когда полностью сварено.

Точные значения этих временны□х интерваловвам нужно будет установить самим, в зависимости от своих предпочтений касательно степени готовности вареных яиц. На длительность интервалов также будут оказывать влияние несколько переменных — таких как сила пламени, размер емкости для варки, количество воды в ней, а также размер яйца. Вам самому придется разобраться, как принять их все во внимание.

Для указания задержек в коде надо будет использовать большие числа, поскольку задаются они в миллисекундах. Вспомним, что 1 секунду

составляют 1000 мс, а в минуте таких секунд 60. Умножаем 1000 на 60 и получаем, что 1 минуту (60 секунд) составляют 60 000 мс. Чтобы задать задержку в 3 минуты, просто умножаем 60 000 на 3 непосредственно в функции delay() следующим образом:

```
delay(60,000 * 3);
```

Вас может интересовать, какое максимальное время задержки можно указать в функции delay(). Как нам известно, в функции delay() используется тип данных длинное целое (unsigned long). Этот тип данных охватывает целые числа в диапазоне от 0 до 4 294 967 295, так что максимальная задержка будет около 1193 часов. Вполне достаточно для любых практических целей, не так ли? Вооруженные этими знаниями, можете ли вы придумать, каким еще образом можно применить функцию delay()?

Модифицируем схему

Если вы желаете сделать этот проект более прочным и постоянно действующим, можно, вместо использования удлинителей с гнездами, просто припаять их к выводам светодиодов. Если вы никогда раньше не работали с паяльником, то прежде чем браться за эту задачу, прочитайте *разд. «Работа с паяльником» приложения*.

Возьмите проволочную перемычку со штыревыми разъемами на обоих концах и откусите один из разъемов. Затем снимите около 12 мм изоляции с откушенного конца, залудите оголенную часть провода и вывод светодиода, после чего припаяйте провод к выводу светодиода (рис. 2.30). То же проделайте и с другим его выводом. Обратите внимание, что на иллюстрации рабочий конец провода обмотан вокруг вывода светодиода, чтобы надежно удерживать его при пайке. Паяное соединение будет более надежным, чем соединение разъемом, а светодиод с припаянным удлинителем можно будет использовать в других проектах, поскольку другой конец провода имеет штыревой разъем.

Хотя этот проект выглядит довольно впечатляюще, его программная и аппаратная составляющие весьма просты. Мы рекомендуем вам при изучении датчиков и программной логики в следующих проектах возвращаться мысленно к этому проекту и пробовать придумывать способы, как его можно усовершенствовать, применяя свои новые знания.

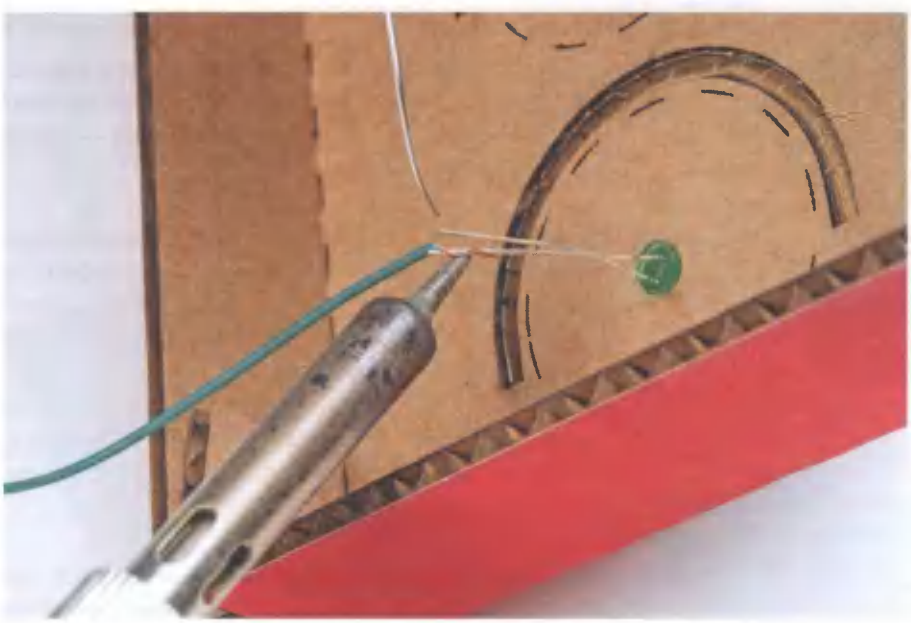


Рис. 2.30. Припаивание проволочного удлинителя к светодиоду

ДЕВЯТИПИКСЕЛЬНЫЙ АНИМАЦИОННЫЙ ДИСПЛЕЙ

Мы используем мониторы каждый день: в телефонах, с компьютерами, планшетами, телевизорами и т. п. Дисплеи большинства современных мониторов состоят из миллионов пикселов. *Пиксел* — это калька с английского *pixel*, что означает *picture element* — элемент изображения. Пиксел представляет собой крошечную точку дисплея, которая под управлением компьютерной программы может светиться разными цветами. Задавая множеству пикселов экрана различные комбинации, можно отображать на экране тексты, изображения и видео.

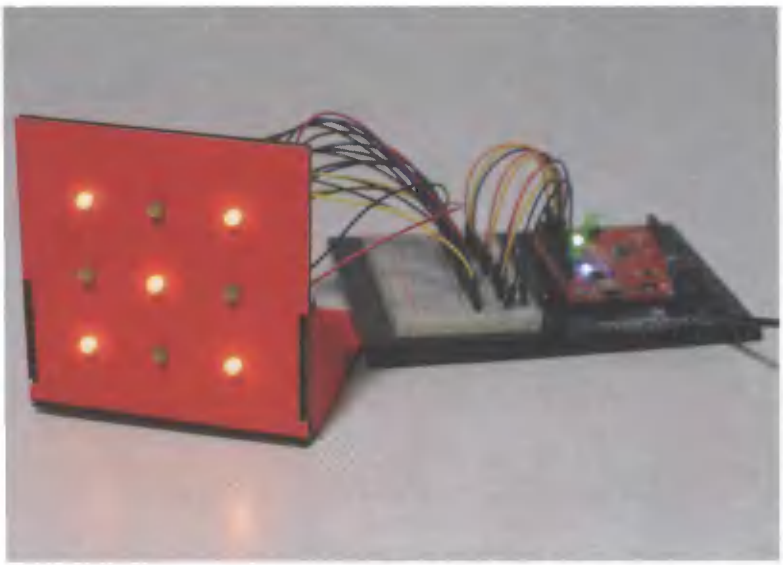


Рис. 3.1. Завершенный проект девятипиксельного дисплея

В этом проекте мы создадим простой монитор на светодиодах. В процессе работы с ним мы расширим наши предыдущие эксперименты с мигающими светодиодами и разберемся с пользовательскими функциями Arduino. Наконец, мы научимся отображать разные символы на созданном нами девятипиксельном дисплее, пример которого показан на рис. 3.1.

Как мы увидим дальше, на этом дисплее можно будет отображать буквы и цифры, рисовать простые геометрические фигуры, и вообще, воспроизводить множество другой занимательной пиксельной графики.

Необходимые компоненты, инструменты и материалы

Для этого проекта потребуется немного больше электронных компонентов, чем для предыдущих двух, и, в частности, больше светодиодов. Но создание корпуса для этого проекта будет легче, чем для *проекта 2*.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 3.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. (на рис. 3.2 не показан);
- беспаечная макетная плата (PRT-12002), 1 шт.;
- светодиоды предпочтительно одного цвета (COM-10049 для пакета, содержащего 20 шт. светодиодов красного и желтого цвета), 9 шт.;
- резисторы 330 Ом (COM-08377 или COM-11507 для пакета, содержащего 20 шт.), 9 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*;
- может пригодиться: держатель для четырех батареек типа AA (PRT-09835)* (на рис. 3.2 не показан).

Примечания

Использование всех светодиодов одного цвета поможет лучше различать отображаемые фигуры, но если у вас не найдется девяти светодиодов одного цвета, можно использовать и светодиоды разных цветов.

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

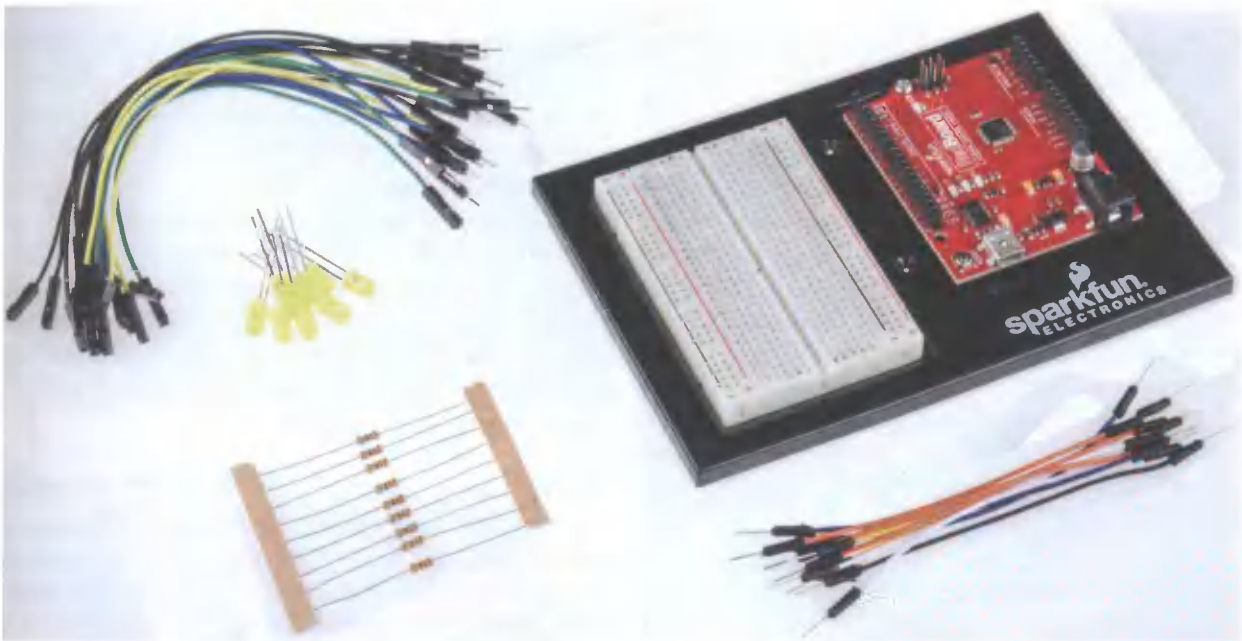


Рис. 3.2. Электронные компоненты для проекта девятипиксельного дисплея

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты и материалы (рис. 3.3):

- карандаш;
- макетный нож;
- металлическая линейка;
- плоскогубцы (бокорезы);
- инструмент для снятия изоляции;
- клей (клеевой пистолет или клей для моделирования);
- миллиметровая бумага (не показана);
- могут пригодиться: дрель и сверло диаметром 4,75 мм;
- может пригодиться: паяльник;
- может пригодиться: припой;
- может пригодиться: держатель «Третья рука»[1] (не показана);
- кусок картона 20×30 см (не показан);
- шаблон корпуса (см. рис. 3.13 далее в этом проекте).

[1] См. **https://ru.wikipedia.org/wiki/Третья_рука**.

Рис. 3.3. Инструменты, рекомендуемые для проекта девятипиксельного дисплея

Создаем прототип девятипиксельного дисплея

С помощью создаваемого нами дисплея для отображения простой пиксельной графики мы научимся работать со схемами с большим количеством проводов, что даст нам важные навыки такой работы, поскольку сложность наших схем будет постоянно возрастать. Сначала мы соберем на беспаечной макетной плате упрощенную схему — чтобы проверить ее в работе, протестировать скетч и набраться опыта работы с большим количеством проволочных перемычек. (Установка светодиодов в корпусе дисплея рассматривается в *разд. «Корпус для девятипиксельного дисплея»* далее в этом проекте.)

Обратите внимание, что приведенная на рис. 3.4 принципиальная схема нашего проекта во многом похожа на схему, приведенную на рис. 2.6. Это объясняется тем, что она основана на той же схеме подключения светодиодов, с той лишь разницей, что вместо одного или двух светодиодов в ней используются девять, каждый из которых управляется отдельным выводом платы Arduino.

Итак, для подключения к Arduino всех девяти светодиодов мы воспользуемся здесь беспаечной макетной платой. Приготовьте все необходимые компоненты и проволочные перемычки и соберите на беспаечной макетной плате схему, показанную на рис. 3.5. Если вы сначала хотите попрактиковаться на сборке меньшего количества подключаемых светодиодов, обратитесь к *разд. «Подключаем красный светодиод» проекта 2*, чтобы освежить свои знания.

Подключение девяти светодиодов с помощью проволочных перемычек может загромоздить макетную плату. Чтобы упорядочить монтаж, сначала подключите шину отрицательного питания («земли»), обозначенную знаком минус «–» с левой стороны макетной платы, к одному из выводов **GND** платы Arduino. На рис. 3.5 это черная перемычка. Затем подключите к этой шине отрицательного питания отрицательный (короткий) вывод первого светодиода через последовательный резистор сопротивлением 330 Ом. На рис. 3.5

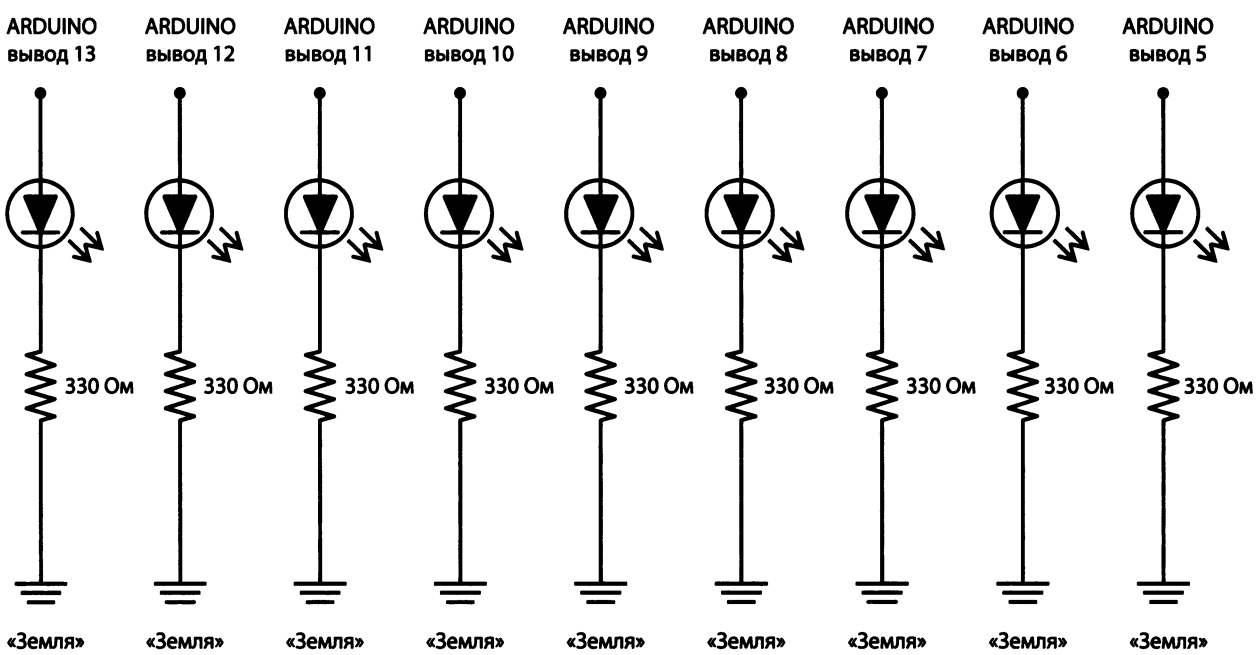


Рис. 3.4. Принципиальная схема девятипиксельного дисплея

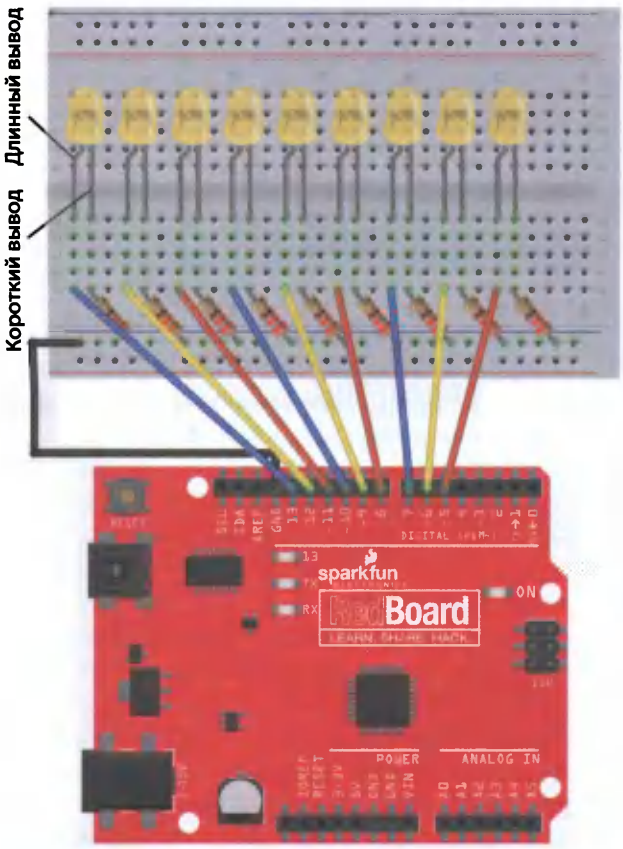


Рис. 3.5. Подключение к Arduino девяти светодиодов: светодиод, подключенный к выводу 13, находится вверху, а подключенный к выводу 5 — внизу

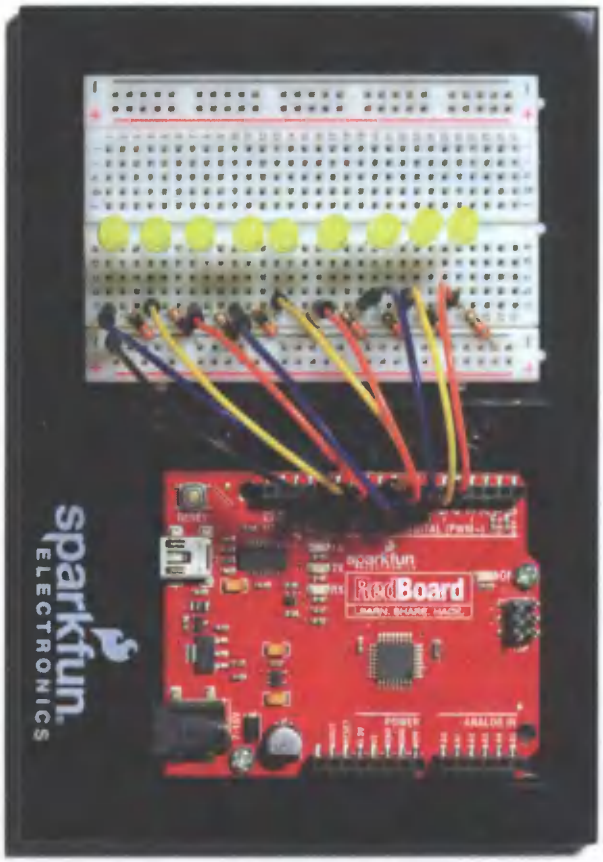


Рис. 3.6. Конечный монтаж схемы с девятью светодиодами, подключенными к Arduino: светодиод, подключенный к выводу 13, находится вверху, а подключенный к выводу 5 — внизу

длинный вывод первого светодиода вставлен в гнездо в первом ряду контактов макетной платы, а короткий — в гнездо во втором ряду. Наконец, подключите длинный вывод первого светодиода к выводу 13 Arduino. Для этого соедините проволочной перемычкой гнездо вывода 13 платы Arduino с любым гнездом первого ряда контактов макетной платы. Таким же образом подключите остальные светодиоды к выводам с 12-го по 5-й. При этом не забывайте, что короткий вывод светодиода имеет отрицательную полярность. В процессе создания этой схемы не забывайте, что подключать короткие выводы светодиодов к шине «земли» макетной платы следует через последовательный токоограничивающий резистор. Готовая схема должна выглядеть примерно так, как на рис. 3.6.

Подключив все девять светодиодов, запустите среду разработки Arduino и подключите плату Arduino к порту USB компьютера. Если в плату Arduino загружен скетч из предыдущего проекта, он начнет исполняться и некоторые светодиоды станут мигать. Теперь давайте рассмотрим, как запрограммировать все девять светодиодов, чтобы они включались, как нам требуется.

Программируем девятипиксельный дисплей

Для управления светодиодами в предыдущих проектах было достаточно использовать несколько функций digitalWrite() и delay(). Но с девятью диодами при таком подходе функция loop() получится чрезвычайно загроможденной. Чтобы избежать этого, мы создадим свою функцию для мигания одного светодиода, а затем используем эту функцию для управления всеми светодиодами.

Пользовательские функции

Язык Arduino содержит около 60 встроенных (или предопределенных) функций, что упрощает для разработчиков взаимодействие с аппаратными компонентами, позволяя использовать однострочные инструкции. Примерами таких предопределенных функций являются наши знакомые функции digitalWrite() и delay(). В действительности функция digitalWrite() содержит свыше 20 строк кода — этот код в большей части труднопонимаем, но сама функция digitalWrite() проблем с пониманием не представляет.

Но даже если внутренний код функции было бы понимать легко, вводить 20 или больше строк кода каждый раз, когда необходимо просто включить или выключить светодиод, очень утомительно, и к тому же чревато опасностью допустить ошибку при вводе. Встроенные функции Arduino охватывают выполнение распространенных задач, но когда в скетче требуется сделать что-то, присущее только данному скетчу, мы обращаемся к созданию своей, *пользовательской*, функции. Созданные нами для какого-либо скетча пользовательские функции можно также использовать и в других скетчах для уменьшения объема их кода, в результате чего функция loop() в них будет более удобочитаема.

Создаем пользовательскую функцию

С помощью пользовательской функции можно научить Arduino выполнять новые действия. Для практики мы сначала создадим пользовательскую функцию, которая включает и выключает светодиод, модифицировав для этого код в листинге 3.1.

Листинг 3.1. Простой код, включающий и выключающий светодиод

```
void setup()
  {
    pinMode(13, OUTPUT);
  }

  void loop()
  {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
  }
```

Этот код похож на код примера Blink из листинга 1.1. Но вместо использования системной константы LED_BUILTIN, в нем вывод 13 указывается явно. Этот код включает светодиод, приостанавливается на секунду, выключает светодиод, снова приостанавливается на секунду, после чего цикл повторяется. Нам придется часто выполнять операцию мигания светодиодом в дальнейших проектах, поэтому мы оформим его в пользовательскую функцию blink(). Чтобы эта функция была как можно более полезной, мы сделаем возможным указывать для нее любой вывод и любое время задержки.

Для начала создайте новый скетч, скопируйте в него код из листинга 3.1 и сохраните его. Затем за функциями setup() и loop() определите функцию blink(), как показано в листинге 3.2.

Листинг 3.2. «Скелет» для пользовательской функции blink()

```
❶ void ❷ blink(❸ int pinNumber, int delayTime)
{
  //сюда вставляется код пользовательской функции
}
```

Листинг 3.2 содержит лишь заготовку пользовательской функции. В определениях функций сначала всегда указывается тип данных, возвращаемых функцией ❶. Поскольку функция blink() только предлагает Arduino выполнить определенную задачу и не требует возврата никаких данных, то, подобно функциям setup() и loop(), ее тип данных будет void.

Далее следует имя функции ❷ — в данном случае это blink. Функциям Arduino можно присваивать практически любые имена, но они не должны начинаться с цифры и не могут содержать пробелов или специальных символов. А чтобы при просмотре скетча назначение функции было понятным, рекомендуется присваивать им описательные и легко запоминающиеся имена.

После имени функции в скобках определяются параметры ❸, необходимые для работы функции. Чтобы функцию blink() можно было использовать в разных ситуациях, в качестве параметров ей передаются номер вывода платы Arduino и время задержки. Посредством этих параметров мы указываем функции, каким светодиодом мигать, и период мигания. Оба параметра функции blink() имеют целочисленный тип данных int. Обратите внимание, что определение параметров функции подобно определению переменных. Это объясняется тем, что параметры, по сути, являются переменными, но которые можно использовать только внутри функции.

Наконец, пользовательская функция также содержит пару фигурных скобок, заключающих весь код, который мы хотим выполнить при вызове функции. Для функции blink() это будет набор функций digitalWrite() и delay() из листинга 3.1, как и показано в листинге 3.3. Добавьте код в фигурных скобках в свою функцию blink() в редакторе скетчей Arduino.

Листинг 3.3. Пользовательская функция blink()

```
void blink(int pinNumber, int delayTime)
{
  digitalWrite(pinNumber, HIGH);
  delay(delayTime);
  digitalWrite(pinNumber, LOW);
  delay(delayTime);
}
```

Обратите внимание, что в вызовах функций digitalWrite() и delay() функция blink() заменяет номер вывода 13 и время задержки в 1000 мс значениями параметров pinNumber и delayTime соответственно.

Применяем пользовательскую функцию

Теперь вместо нескольких строк кода в функции loop() можно использовать нашу пользовательскую функцию blink() — это показано в листинге 3.4.

Листинг 3.4. Полный текст скетча с использованием пользовательской функции blink()

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  blink(13, 1000);
}
void blink(int pinNumber, int delayTime)
{
  digitalWrite(pinNumber, HIGH);
  delay(delayTime);
  digitalWrite(pinNumber, LOW);
  delay(delayTime);
}
```

Модифицированная функция loop() вызывает функцию blink() и передает ей номер светодиода, которым нужно мигать (13), а также период времени в миллисекундах между включенным и выключенным состояниями (рекомендуется использовать 1000 мс, или 1 секунду). Вот и все! Разве этот код не выглядит намного аккуратней и более удобочитаемым?

После загрузки этого скетча в Arduino должен мигать светодиод, подключенный к выводу 13. Поздравляем! Вы только что научили Arduino понимать, что означает blink(), и сжали четыре строки кода в одну инструкцию.

ЭКСПЕРИМЕНТИРУЕМ С ГРАФИКОЙ

Прежде чем приступать к созданию конечной версии девятипиксельного дисплея, уделите некоторое время экспериментам с использованием нашей пользовательской функции blink() в скетчах. Например, модифицируйте функцию loop() следующим образом:

```
void loop()
{
  blink(13, 100);    //быстрое мигание на выводе 13
                     //с задержкой в 100 мс
  blink(13, 2000);   //медленное мигание на выводе 13
                     //с задержкой в 2000 мс
}
```

Этот код сначала быстро мигает светодиодом с периодом в 100 мс, а затем медленно с периодом в 2000 мс. Попробуйте мигать другими светодиодами, задавая различные периоды и узоры.

Однако применение пользовательских функций — это лишь одна ключевая часть нашего проекта. Весьма желательно предварительно придумать несколько девятипиксельных изображений, чтобы упростить их программирование. Давайте займемся этой задачей сейчас.

Разрабатываем графику

Достаньте свои цветные карандаши и наденьте сюртук художника — мы займемся разработкой пиксельной графики для отображения на нашем девятипиксельном дисплее.

Для начала нарисуйте несколько решеток как для игры в крестики-нолики или же распечатайте несколько экземпляров показанного на рис. 3.7 шаблона, который находится в архиве ресурсов, сопровождающих книгу (см. файл ***P3_AnimationMachineTemplate.pdf*** в папке шаблонов данного проекта). Этот шаблон поможет вам визуализировать свои идеи.

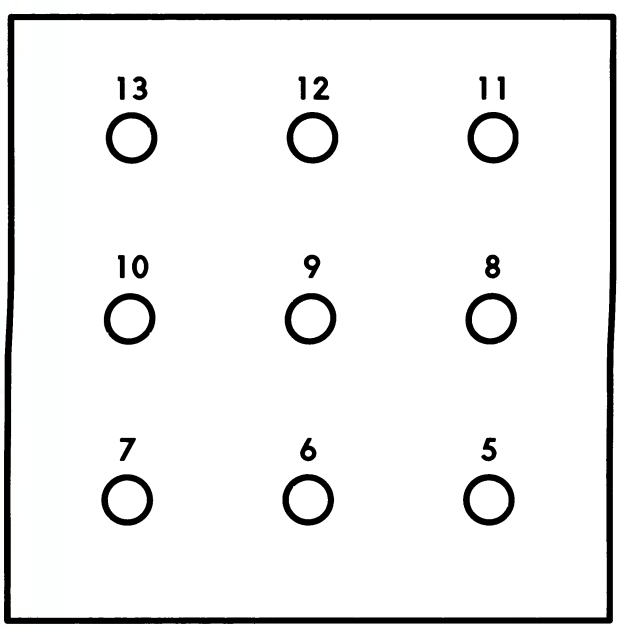


Рис. 3.7. Шаблон для визуализации идей девятипиксельной графики

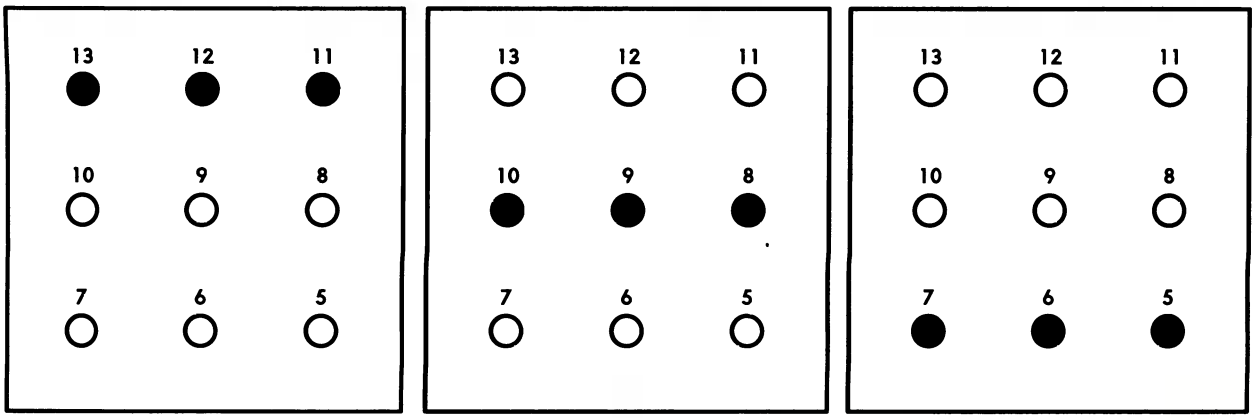


Рис. 3.8. Примеры узоров девятипиксельной графики

Позиции шаблона представляют пикселы нашего дисплея и пронумерованы, начиная с 13 в левом верхнем углу и кончая 5 в правом нижнем углу. Эти номера соответствуют номерам выводов Arduino, к которым подключены светодиоды, которыми мы будем управлять в нашем девятипиксельном дисплее. Приготовив шаблоны, займитесь творчеством: заполните их своими узорами. На рис. 3.8 показано несколько примеров, чтобы дать вам начальный толчок.

Последовательно отображая эти узоры, можно создавать анимацию. Сначала мы рассмотрим, как запрограммировать две простые фигуры, а затем перейдем к созданию анимации на их основе.

Тестовый скетч

Создайте новый скетч в среде разработки Arduino и вставьте в него функции setup() и loop() из листинга 3.5.

> **Примечание**
>
> Рекомендуется вставить в свой скетч также комментарий с диаграммой размещения светодиодов. Описание в скетче схемы, для работы с которой предназначен скетч, поможет другим воссоздать ее, а также поможет вам самим помнить структуру схемы, для которой вы разрабатываете этот скетч.

Листинг 3.5. Функция setup() для девятипиксельного дисплея

```
//Массив светодиодов конфигурируется таким образом:
// 13 ----12 ----11
// 10 ---- 9 ---- 8
//  7 ---- 6 ---- 5

void setup()
{
   pinMode(13, OUTPUT);
   pinMode(12, OUTPUT);
   pinMode(n, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(9, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(6, OUTPUT);
   pinMode(5, OUTPUT);
}

void loop()
{
}
```

В этом проекте используются девять светодиодов, подключенных к девяти (с 13-го по 5-й) цифровым контактам ввода/вывода общего назначения платы Arduino. Поэтому в функции setup() делается девять вызовов функции pinMode(), которая конфигурирует все эти контакты Arduino на вывод данных, поскольку они используются для управления светодиодами.

Создаем функцию для отображения фигуры X

Следующей задачей будет создать код для отображения заполненных нами ранее шаблонов графики. Для этого мы создадим пользовательскую функцию, в которой используются номера заполненных точек шаблонов. На рис. 3.9 показан пример графики, для которой мы создадим тестовый скетч, — простая фигура X.

Для отображения этой фигуры на нашем дисплее потребуется набор из девяти функций digitalWrite(). Соответствующий код приведен в листинге 3.6.

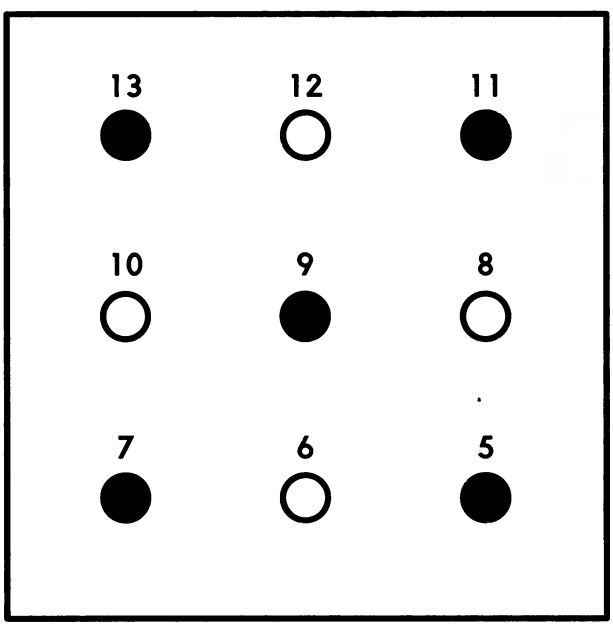


Рис. 3.9. Девятипиксельная фигура X

Листинг 3.6. Код для отображения фигуры X на девятипиксельном дисплее

```
digitalWrite(13, HIGH);
digitalWrite(12, LOW);
digitalWrite(11, HIGH);

digitalWrite(10, LOW);
digitalWrite(9, HIGH);
digitalWrite(8, LOW);

digitalWrite(7, HIGH);
digitalWrite(6, LOW);
digitalWrite(5, HIGH);
```

Эти вызовы функции digitalWrite() включают светодиоды, расположенные по диагонали матрицы, и выключают остальные. Но не кажется ли вам, что девять инструкций для отображения простой фигуры слишком много? Вместо того чтобы перечислять все эти инструкции каждый раз, когда нам необходимо отобразить эту фигуру, мы перечислим их всего лишь один раз в пользовательской функции. Затем, когда нам потребуется отобразить эту фигуру, мы сможем делать это с помощью только одной строки кода, вызывающей эту пользовательскую функцию.

В листинге 3.7 приводится код для этой пользовательской функции: xChar(). Вставьте его в свой скетч после закрывающей фигурной скобки функции loop().

Листинг 3.7. Пользовательская функция xChar()

```
void xChar()
{
    digitalWrite(13, HIGH);
    digitalWrite(12, LOW);
    digitalWrite(11, HIGH);

    digitalWrite(10, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);

    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
}
```

Функции присвоено имя xChar(), поскольку она отображает символ X. Так как функция не возвращает никаких данных, ей присваивается тип данных void. Помещение всех вызовов функции digitalWrite() (см. листинг 3.6) в эту одну пользовательскую функцию способствует упрощению кода в функции loop(). Вызов пользовательской функции xChar() из функции loop() осуществляется так, как показано в листинге 3.8.

Листинг 3.8. Вызов функции xChar() в функции loop()

```
void loop()
{
   xChar();
}
```

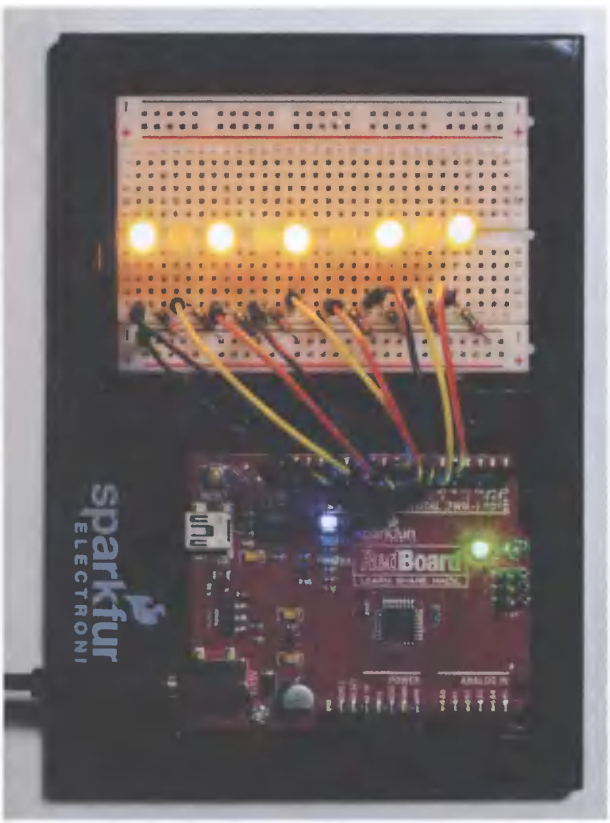


Рис. 3.10. Исполнение скетча для отображения фигуры X на прототипе схемы проекта

Это очень простое действие является, тем не менее, чрезвычайно важным. Если забыть выполнить вызов этой пользовательской функции, ее код никогда не исполнится, и фигура X не будет отображена на дисплее.

Теперь загрузите готовый скетч в Arduino. Хотя наши светодиоды расположены в одну вертикальную линию, мы все равно можем протестировать правильность нашего монтажа и программы. Вместо отображения фигуры X диоды должны светиться через один, начиная с верхнего, как показано на рис. 3.10. Если они не горят в этой последовательности, проверьте монтаж схемы, сверяясь с монтажной схемой, приведенной на рис. 3.5, проверьте также все функции digitalWrite() в пользовательской функции xChar() на отсутствие орфографических и синтаксических ошибок.

Когда светодиоды горят в правильной последовательности, можно приступать к созданию пользовательской функции для отображения другой фигуры.

Создаем функцию для отображения фигуры O

По нашему мнению, хорошо сочетаться с фигурой X сможет фигура O (рис. 3.11).

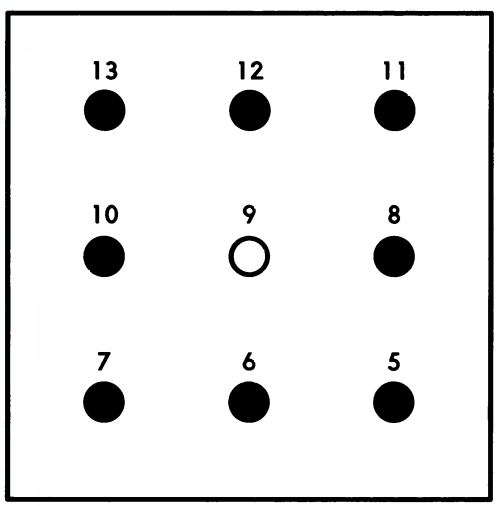


Рис. 3.11. Девятипиксельная фигура O

Профессиональный совет

Учитесь облегчать себе жизнь, повторно используя результаты уже выполненной работы. В частности, скопируйте всю пользовательскую функцию xChar(), вставьте ее в свой скетч после закрывающей фигурной скобки функции xChar(), а затем слегка откорректируйте ее, чтобы она выглядела так, как показано в листинге 3.9.

Листинг 3.9. Пользовательская функция oChar()

```
void oChar()
{
    digitalWrite(13, HIGH);
    digitalWrite(12, HIGH);
    digitalWrite(11, HIGH);

    digitalWrite(10, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(8, HIGH);

    digitalWrite(7, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(5, HIGH);
}
```

Единственная разница между функциями xChar() и oChar() состоит в разных включаемых и выключаемых светодиодах. Тогда как функция xChar() включает светодиоды через один, функция oChar() включает все светодиоды, за исключением светодиода в центре матрицы.

ПРАКТИКУМ: создайте пользовательскую функцию для отображения какой-либо фигуры

Но основе знаний, полученных в процессе создания пользовательских функций для отображения фигур X и O, создайте пользовательскую функцию для отображения какой-либо фигуры, придуманной вами самостоятельно. Сохраните эту функцию для отображения ее фигуры на девятипиксельном дисплее, когда мы закончим его создание.

Отображаем фигуры X и O

Нашей следующей задачей будет отобразить фигуру X в течение некоторого времени, потом фигуру O, а затем снова фигуру X и так далее в цикле. Для этого мы просто добавим вызов пользовательской функции oChar() в наш скетч после вызова функции xChar(), вставив после каждого из этих вызовов функцию delay(). Конечный код скетча должен выглядеть, как показано в листинге 3.10.

Листинг 3.10. Завершенный скетч для циклического отображения фигур X и O

```
//Массив светодиодов конфигурируется таким образом:
// 13 ---- 12 ---- 11
// 10 ----  9 ----  8
//  7 ----  6 ----  5

void setup()
{
    pinMode(13, OUTPUT);
    pinMode(12, OUTPUT);
    pinMode(11, OUTPUT);
    pinMode(10, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(5, OUTPUT);
}

void loop()
{
    //Попеременно отображаем фигуры X и O
    xChar();
```

```
    delay(500);
    oChar();
    delay(500);
}

void xChar()
{
    digitalWrite(13, HIGH);
    digitalWrite(12, LOW);
    digitalWrite(11, HIGH);

    digitalWrite(10, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);

    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
}

void oChar()
{
    digitalWrite(13, HIGH);
    digitalWrite(12, HIGH);
    digitalWrite(11, HIGH);

    digitalWrite(10, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(8, HIGH);

    digitalWrite(7, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(5, HIGH);
}
```

Функция loop() этого скетча похожа на функцию loop() скетча Blink, но вместо функции digitalWrite() в ней используются функции xChar() и oChar(). В результате функция отображает фигуру X в течение 500 мс, а затем фигуру O также в течение 500 мс. Загрузите обновленный скетч в плату Arduino и проверьте его работу. При отображении фигуры X светодиоды должны гореть через один (рис. 3.12, *вверху*), а при отображении фигуры O должны гореть все светодиоды, за исключением среднего (рис. 3.12, *внизу*).

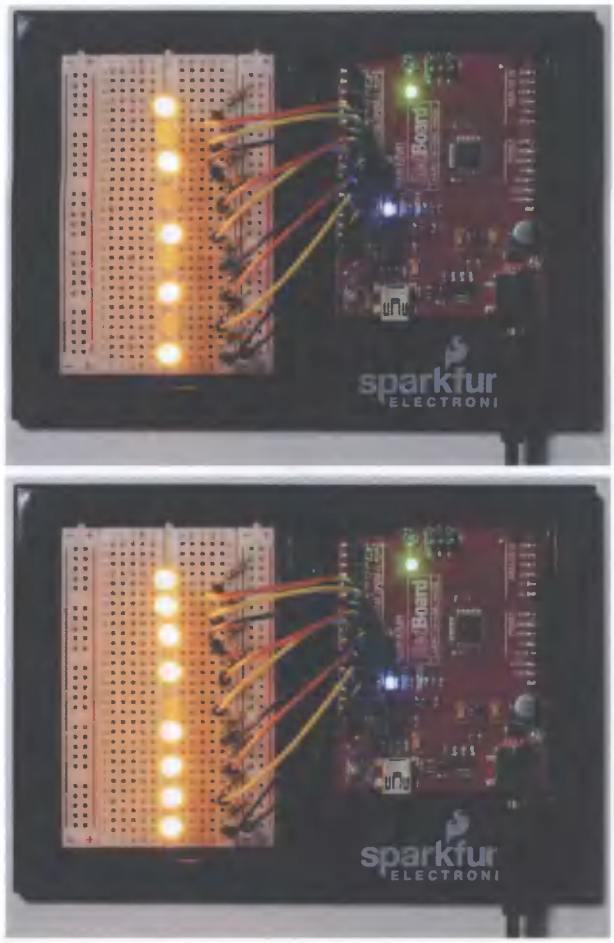


Рис. 3.12. Попеременное отображение фигур X (*вверху*) и O (*внизу*) на девятипиксельном дисплее

Сохраните скетч, поскольку мы будем использовать его в качестве основы для дальнейших разработок.

Прототип схемы девятипиксельного дисплея хорош тем, что он позволяет быстро проверить работу скетча. Но чтобы различить отображаемые на нем фигуры, необходимо проявить определенное воображение. Поэтому нашей следующей задачей станет создание настоящего девятипиксельного дисплея, на котором можно отображать как отдельные фигуры, так и незамысловатые анимации.

Создаем корпус для девятипиксельного дисплея

Корпусом для этого проекта будет служить простой лист картона с отверстиями в нем для светодиодов. Нам нужно будет также выполнить подключение светодиодов, после чего дисплей будет готов для отображения разнообразной пиксельной графики.

Делаем картонный корпус

Возьмите лист чистого картона без изгибов и вмятин. Мы ранее использовали гофрированный картон толщиной около 3 мм, но можно использовать любой подобный подходящий плоский материал. Некоторые материалы более легко поддаются обработке, чем другие, поэтому делайте выбор материала, исходя из имеющихся у вас инструментов.

Вырезаем детали

Распакуйте из сопровождающего книгу архива файлы *проекта 3* (папка P3_AnimationMachine), распечатайте файл P3_AnimationMachineTemplate.svg шаблона дисплея (рис. 3.13) и переведите его на картон. Старайтесь совместить края шаблона с краями листа картона, чтобы меньше резать.

Вырежьте переведенные на картон детали. Для этого настоятельно рекомендуется использовать острый макетный нож и металлическую линейку, чтобы детали имели опрятные и четкие края. Не забывайте о правилах безопасной работы с макетным ножом: всегда тяните лезвие на себя, а не толкайте, и делайте несколько неглубоких проходов, а не один рез на всю толщину материала.

Вырезав все детали, сделайте отверстия для светодиодов в лицевой части дисплея. Их можно просверлить, как показано на рис. 3.14, пробить с помощью дырокола или просто проткнуть остро заточенным карандашом. При этом следует иметь под рукой один из светодиодов и проверять размер каждого отверстия, чтобы светодиод сидел в нем плотно. Если отверстия получатся несколько большего размера, светодиоды можно в них вклеить, если вы не против, чтобы они остались там навсегда.

В результате у нас должно получиться четыре детали корпуса проекта (рис. 3.15). Основание дисплея имеет большое отверстие, вырезанное в нем. Вырезанный материал используется для

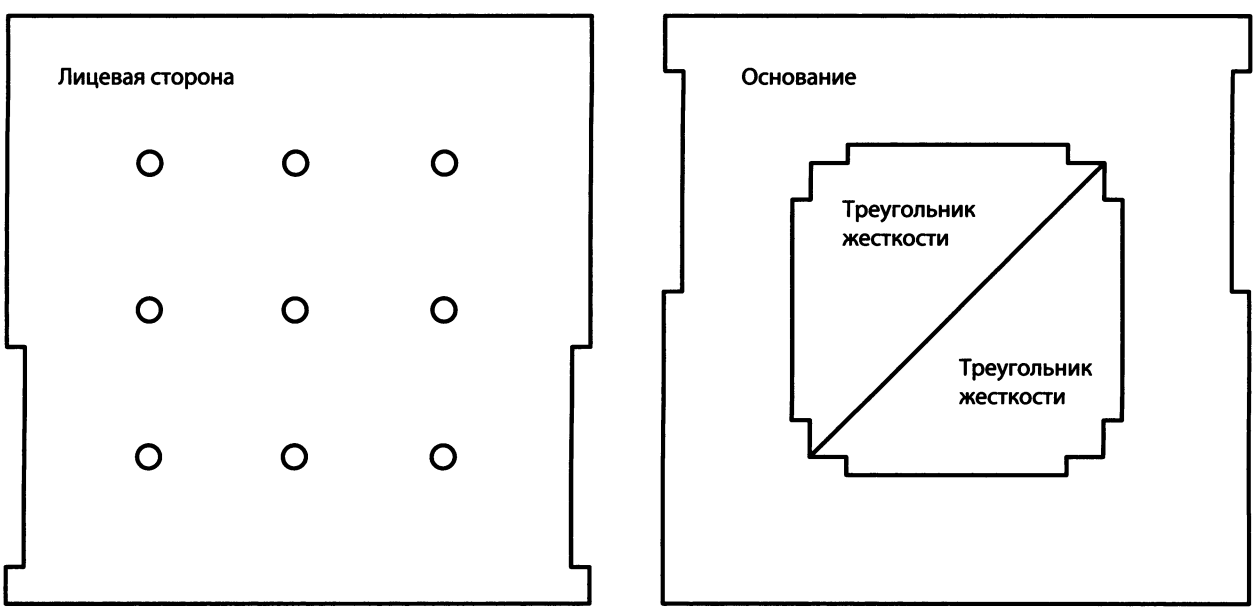


Рис. 3.13. Шаблон корпуса для девятипиксельного дисплея (в уменьшенном виде)

Рис. 3.14. Сверление отверстий под светодиоды. Будьте осторожны при сверлении или обратитесь к взрослым за помощью

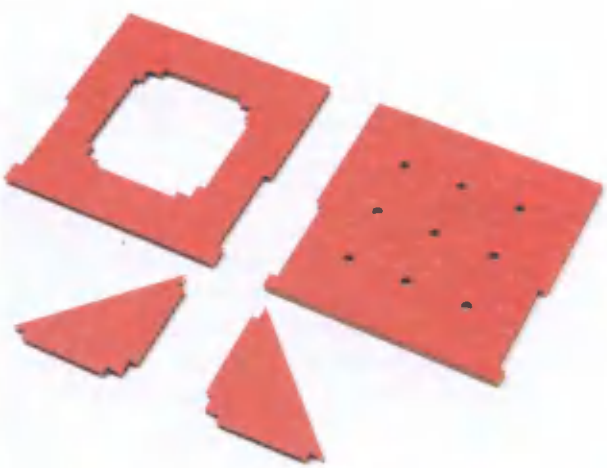


Рис. 3.15. Детали корпуса дисплея

укрепляющих треугольников. В треугольниках имеются выступы, которые вставляются в вырезы в основании и в лицевой части дисплея, соединяя тем самым дисплей в одно целое. Но прежде чем приступать к сборке дисплея, пронумеруйте отверстия под светодиоды, чтобы избежать путаницы при подключении их к схеме.

Нумеруем отверстия под светодиоды

Переверните лицевую часть дисплея тыльной стороной вверх и пронумеруйте отверстия под светодиоды, чтобы иметь схему их подключения под рукой. Начинаем с номера 13 для правого верхнего отверстия и продолжаем в убывающем порядке, перемещаясь влево и вниз. Левое нижнее отверстие должно иметь номер 5.

Вставляем светодиоды

На следующем шаге мы вставляем светодиоды в лицевую панель дисплея с ее тыльной стороны. Можно использовать светодиоды из прототипа, собранного на беспаечной макетной плате, или же взять новые. Вставляйте светодиоды в панель таким образом, чтобы их длинный вывод был справа (рис. 3.17), — это облегчит их подключение к плате Arduino. Как упоминалось ранее, светодиоды должны плотно входить в отверстия.

Рис. 3.16. Нумерация отверстий светодиодной матрицы

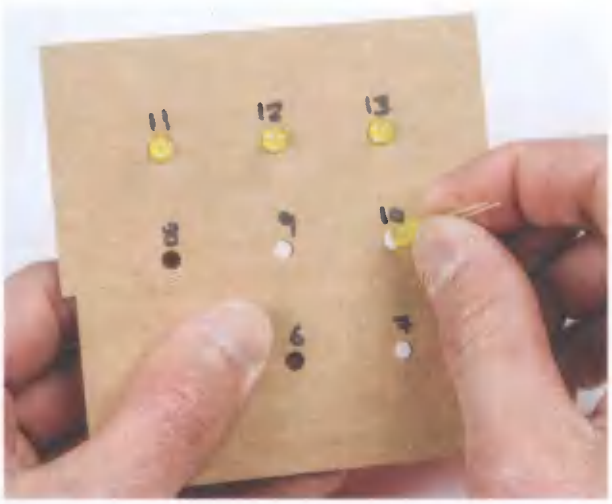


Рис. 3.17. Вставляем светодиоды в тыльную сторону лицевой панели дисплея

Если отверстие оказалось слишком большое, светодиод можно закрепить в нем каплей клея (но имейте в виду, что такой светодиод нельзя будет использовать повторно).

Собираем дисплей

Теперь все готово для сборки дисплея в единое целое (как правильно должны быть ориентированы треугольники при сборке, показано на рис. 3.18). Чтобы надежно скрепить все детали, нам понадобится клей ПВА или клеевой пистолет.

Сначала приклейте к основанию один из треугольников жесткости (рис. 3.19), затем приклейте другой. Прежде чем продолжать сборку дисплея, дайте клею высохнуть.

Рис. 3.18. Ориентация треугольников жесткости при установке лицевой панели на основание

Рис. 3.19. Приклеивание треугольников жесткости к основанию

Убедившись, что треугольники жесткости надежно приклеены к основанию, приступаем к приклеиванию лицевой панели. Вырезы лицевой панели должны плотно сесть на выступы в треугольниках жесткости, а сама панель всей кромкой прилегать к поверхности основания (см. рис. 3.18). Чтобы придать конструкции дополнительную жесткость, можно проклеить изнутри места стыка лицевой панели с основанием и с треугольниками жесткости.

Снова позвольте клеевым соединениям хорошо просохнуть, прежде чем приступать к подключению светодиодов к Arduino.

Подключаем электронику к дисплею

В этом проекте задействовано много проводов, поэтому при монтаже следует быть особо внимательным. Для подключения светодиодов мы используем макетную плату прототипа, созданного ранее, и просто соединяем светодиоды дисплея с помощью проволочных перемычек с соответствующими гнездами на макетной плате, которая уже подключена к Arduino.

Подключение светодиодов можно выполнить двумя способами. В первом, непостоянном, способе подключение выполняется с помощью проволочных перемычек со штекером на одном конце и гнездом на другом. Во втором, постоянном, способе используются перемычки со штекером на одном конце, второй конец которых припаивается к светодиодам. Мы рассмотрим первый способ, но если вы желаете припаять монтажные провода к светодиодам, прежде чем пытаться сделать это, ознакомьтесь с содержанием *разд. «Работа с паяльником» приложения*.

Слишком длинные выводы светодиодов можно немного укоротить, откусив лишнее, однако будьте при этом внимательны, отследив, какой из выводов положительный (длинный), а какой отрицательный (короткий). При укорачивании выводов оставьте положительный вывод чуть длиннее отрицательного, чтобы их можно было различить и после укорачивания. Можно также обозначить

положительный вывод точкой на картоне. При откусывании выводов обязательно наденьте защитные очки, поскольку откусываемые концы имеют свойство резко отскакивать и могут попасть в глаз.

Итак, наденьте гнездовые разъемы перемычек на выводы светодиодов, вставленных в лицевую панель. Чтобы не было путаницы с проводами, используйте перемычки черного цвета для отрицательных (коротких) выводов светодиодов и подключите их к коротким выводам светодиодов (рис. 3.20). Для положительных выводов светодиодов можно использовать перемычки любого другого цвета.

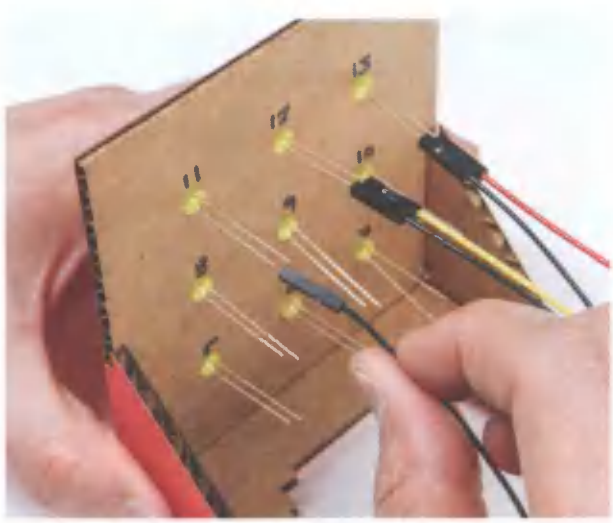


Рис. 3.20. Подключение гнездового конца проволочных перемычек к светодиодам с тыльной стороны лицевой панели дисплея

Подключив перемычки к светодиодам, вставляем штекеры на другом конце перемычек в гнезда на макетной плате, руководствуясь номерами выводов Arduino, обозначенными возле светодиодов на лицевой панели. Если в макетной плате остались светодиоды от прототипа, просто извлеките их и вставьте штекеры от светодиодов лицевой панели дисплея в соответствующие гнезда платы (рис. 3.21).

Рис. 3.21. Подключение светодиодов лицевой панели дисплея к макетной плате с помощью проволочных перемычек с гнездом на одном конце и штекером на другом

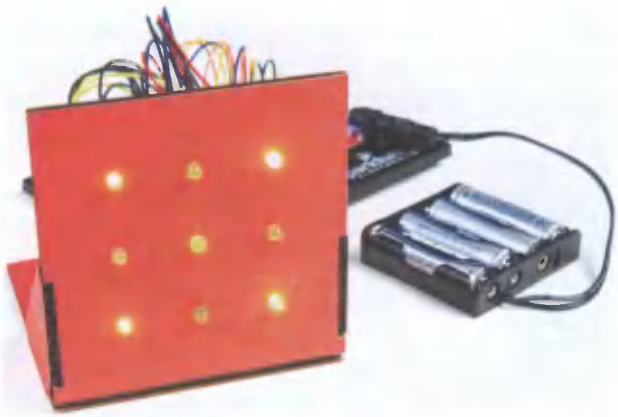


Рис. 3.22. Завершенный проект девятипиксельного дисплея, отображающий фигуру X

Перемычки, подключенные к отрицательным выводам светодиодов (провода черного цвета), подсоединяются другим концом к выводу резистора, другой вывод которого подключен к шине отрицательного питания. А перемычки, подключенные к положительным выводам светодиодов, подсоединяются другим концом к выводам платы Arduino, обозначенным возле соответствующих светодиодов.

Выполнив подключение светодиодов, подключите плату Arduino к порту USB компьютера. Если монтаж был выполнен правильно, на дисплее должны попеременно отображаться фигуры X и O. Питание на плату также можно подавать и от блока батареек (рис. 3.22).

Если тестовые фигуры не отображаются должным образом, еще раз проверьте правильность и надежность подключения светодиодов. Достигнув правильного отображения фигур на дисплее, можно будет передохнуть немного и полюбоваться на плоды своего труда. Но не расслабляйтесь, потому что у нас впереди есть еще неоконченная работа. Когда вы будете готовы оторваться от любования своим творением, попробуйте отобразить на нем более сложную анимацию.

Создаем пиксельную анимацию

Созданный нами дисплей может отображать любое изображение, которое можно представить в матрице размером 3×3. Анимация представляет собой просто набор отображаемых последовательно изображений. Таким образом, отображая последовательно набор изображений размером 3×3 пиксела, мы получим анимированную пиксельную графику. Для начала рассмотрим, как отображать на нашем дисплее вращающуюся линию.

Планируем последовательность анимации

Для начала преобразуем вращающуюся линию в последовательность составляющих ее изображений. Начнем в вертикальной линии, а для имитации ее вращения будем последовательно смещать ее на угол в 45° (рис. 3.23).

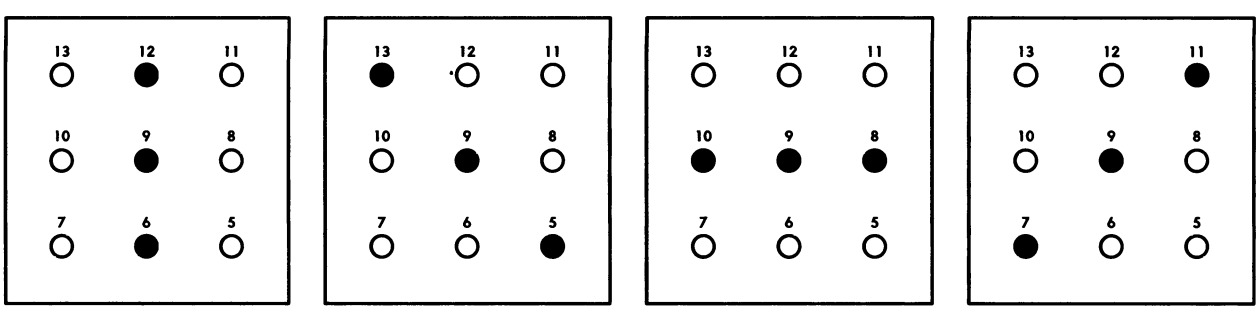


Рис. 3.23. Имитация вращения линии посредством отображения последовательности линий, смещенных на угол в 45°

Сохраните свой скетч из листинга 3.10, а затем создайте новый скетч. Вставьте в него функции setup() и loop() из листинга 3.11.

Листинг 3.11. Задаем режим вывода данных для всех девяти выводов платы Arduino

```
void setup()
{
   pinMode(13, OUTPUT);
   pinMode(12, OUTPUT);
   pinMode(11, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(9, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(6, OUTPUT);
   pinMode(5, OUTPUT);
}

void loop()
{
   //сюда вставляется вызов функции анимации
}

   //сюда вставляется код пользовательских функций
   для отображения кадров
```

Поскольку мы будем использовать те же выводы Arduino, что и в скетче для отображения фигур X и O, просто скопируйте код инициализации выводов Arduino из того скетча в этот.

Создаем пользовательские функции

Далее мы создаем пользовательскую функцию для отображения каждого составного кадра анимации. Поскольку анимация состоит из четырех кадров (линий), нам нужны будут четыре функции. Код для этих функций показан в листинге 3.12. Скопируйте его и вставьте в скетч после закрывающей фигурной скобки функции loop().

Листинг 3.12. Пользовательские функции для анимации вращающейся линии

```
❶ void verticalLine()
  {
     digitalWrite(13, LOW);
     digitalWrite(12, HIGH);
     digitalWrite(11, LOW);

     digitalWrite(10, LOW);
     digitalWrite(9, HIGH);
     digitalWrite(8, LOW);

     digitalWrite(7, LOW);
     digitalWrite(6, HIGH);
     digitalWrite(5, LOW);
  }

❷ void topLeftDiagonal()
  {
     digitalWrite(13, HIGH);
     digitalWrite(12, LOW);
     digitalWrite(11, LOW);

     digitalWrite(10, LOW);
     digitalWrite(9, HIGH);
     digitalWrite(8, LOW);

     digitalWrite(7, LOW);
     digitalWrite(6, LOW);
     digitalWrite(5, HIGH);
  }

❸ void horizontalLine()
  {
     digitalWrite(13, LOW);
     digitalWrite(12, LOW);
     digitalWrite(11, LOW);

     digitalWrite(10, HIGH);
     digitalWrite(9, HIGH);
     digitalWrite(8, HIGH);
```

```
    digitalWrite(7, LOW);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
  }

❹ void topRightDiagonal()
  {
    digitalWrite(13, LOW);
    digitalWrite(12, LOW);
    digitalWrite(11, HIGH);

    digitalWrite(10, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);

    digitalWrite(7, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(5, LOW);
  }
```

Примечание

Этот код также находится в архиве ресурсов по адресу: **https://nostarch.com/arduinoinventor/**.

Функция verticalLine() ❶ отображает первую линию из рис. 3.23, функция topLeftDiagonal() ❷ — вторую, функция horizontalLine() ❸ — третью и функция topRightDiagonal() ❹ — последнюю. Как и в случае с предыдущими пользовательскими функциями, эти пользовательские функции имеют тип данных void, поскольку они не возвращают никаких значений.

Пользовательские функции также могут вызывать другие пользовательские функции, поэтому давайте организуем вызов этих четырех функций одной пользовательской функцией spinningLine(). Код для этой функции показан в листинге 3.13. Скопируйте его и вставьте в скетч после закрывающей фигурной скобки функции topRightDiagonal().

Листинг 3.13. Код пользовательской функции для вызова четырех функций анимации линий

```
void spinningLine(int delayTime)
{
  verticalLine();
  delay(delayTime);

  topLeftDiagonal();
  delay(delayTime);

  horizontalLine();
  delay(delayTime);

  topRightDiagonal();
  delay(delayTime);
}
```

Исполнение этого кода будет в цикле последовательно отображать вертикальную линию, диагональную линию, горизонтальную линию и снова диагональную линию, но с противоположным наклоном, приостанавливая исполнение после вывода каждой линии. Все, что нам нужно сделать сейчас, чтобы вывести на наш дисплей анимацию вращающейся линии, — это сделать вызов функции spinningLine() изнутри функции loop().

Корректируем функцию *loop()*

Вставьте код для вызова нашей пользовательской функции spinningLine() в функцию loop(), как показано в листинге 3.14. Не забывайте, что нам по-прежнему требуются все команды pinMode() в функции setup().

Листинг 3.14. Вызов пользовательской функции spinningLine(200) из функции loop()

```
void loop()
{
  spinningLine(200);
}
```

В данном случае функции spinningLine() передается в параметрах время задержки: 200 мс. После загрузки скетча в Arduino на дисплее начнет отображаться анимация вращающейся линии.

Теперь, обогащенные знаниями способов создания пользовательских функций и их применения для отображения последовательности пиксельных фигур, вы можете сами создавать девятипиксельную анимацию.

Идем дальше...

Пользовательские функции полезны тем, что они позволяют упорядочить код и/или повторно использовать его. Попробуйте расширить этот проект, разрабатывая более сложные анимации. Например, можно создать свой алфавит и отображать с его помощью на дисплее секретные сообщения.

Экспериментируем с кодом

При работе со следующими несколькими проектами пытайтесь придумать способы использовать в них созданный в этом проекте дисплей. Например, можно задействовать какой-нибудь датчик для управления скоростью анимации или отображать на дисплее показания датчика каким-либо интересным способом. Шаблоны для разработки фигур можно загрузить по адресу: **https://www.nostarch.com/arduinoinventor/**.

Модифицируем схему

Мы только что научились управлять несколькими электронными компонентами (светодиодами), посылая с помощью пользовательских функций управляющие команды на цифровые выводы платы Arduino. Попробуйте использовать вместо отдельных светодиодов другие компоненты — например, можно попытаться управлять семисегментным светодиодным дисплеем (рис. 3.24).

Каждый сегмент дисплея представляет собой светодиод, которым можно управлять. Всего дисплей содержит семь отдельных сегментов для отображения цифр и дополнительный сегмент для отображения десятичной точки (рис. 3.25). Включая и выключая разные комбинации сегментов, на дисплее можно отображать все цифры десятичной системы и большинство букв английского алфавита.

Управлять одним таким дисплеем можно точно так же, как и созданным нами девятипиксельным дисплеем, — просто создав пользовательскую функцию для отображения каждой цифры. В качестве домашнего задания создайте пользовательскую функцию, в которую можно передавать цифру для отображения ее на семисегментном светодиодном дисплее.

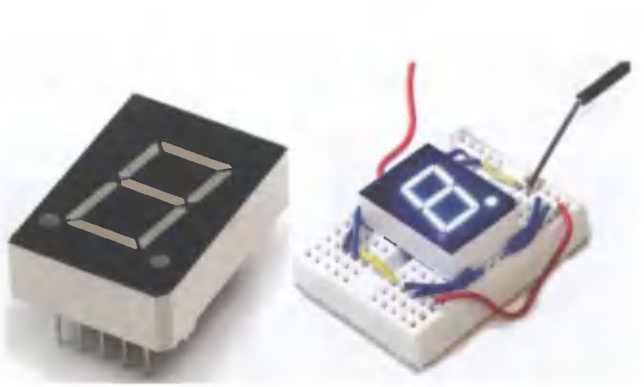


Рис. 3.24. Семисегментный светодиодный дисплей (*слева*) и вариант его установки на макетной плате (*справа*)

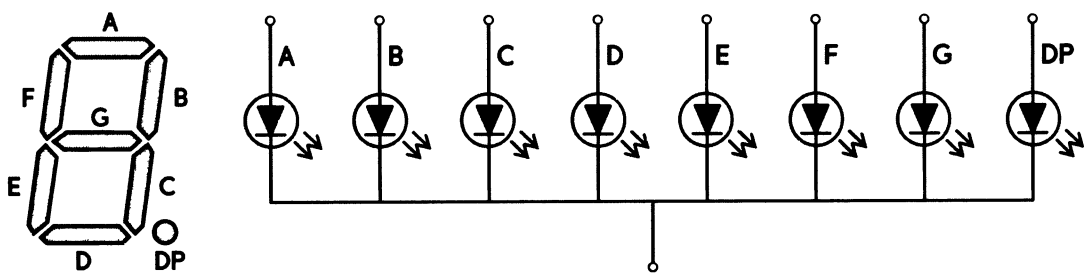


Рис. 3.25. Схематическое изображение семи отдельных сегментов и десятичной точки семисегментного светодиодного дисплея (*слева*) и соответствующая принципиальная схема (*справа*)

ИЗМЕРИТЕЛЬ СКОРОСТИ РЕАКЦИИ

Средний человек реагирует на визуальную стимуляцию — например, на включение света, — в течение приблизительно 215 миллисекунд. Это время, требуемое на то, чтобы зарегистрированный глазом сигнал дошел до мозга, был обработан мозгом, после чего команда из мозга дошла до части тела, которая должна отреагировать на этот сигнал. Создание проекта измерителя скорости реакции — прекрасный повод, чтобы зафиксировать и продемонстрировать нам эту задержку. Кроме того, он станет также занимательной игрой на соревнование в скорости реакции с вашими друзьями.

В этом проекте мы узнаем, как создать такой измеритель скорости реакции, используя плату Arduino. Проект в его завершенном виде показан на рис. 4.1.

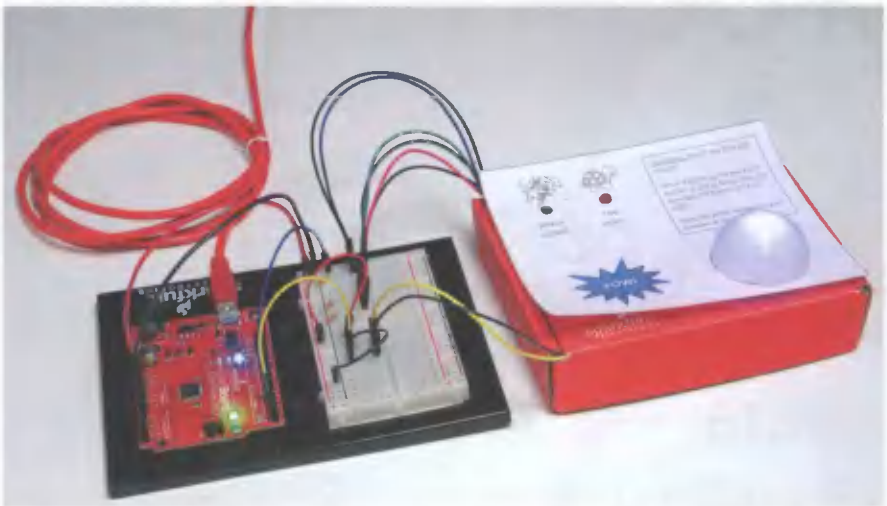


Рис. 4.1. Завершенный проект измерителя скорости реакции

Концепция проекта очень простая: Arduino включает светодиод, запускает таймер и ждет, пока пользователь не нажмет на кнопку. Пользователь нажимает на кнопку, как только видит загоревшийся светодиод, Arduino регистрирует время от включения светодиода до нажатия кнопки и передает его в компьютер. Рабочая тактовая частота Arduino в 16 МГц означает, что он может обрабатывать 16 миллионов инструкций в секунду. Такая скорость обработки инструкций делает Arduino идеальной платформой для нашего проекта.

Необходимые компоненты, инструменты и материалы

Подобно предыдущим проектам этой книги, в проекте измерителя реакции используются светодиоды, резисторы, монтажные провода и плата Arduino. От других проектов этот отличается наличием в нем кнопки, делающей его интерактивным. Кроме того, чтобы сделать игру более занимательной, рекомендуется оформить корпус проекта более нарядно.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 4.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. ;
- беспаечная макетная плата (PRT-12002), 1 шт.;
- светодиоды (COM-12062): красный (1 шт.), желтый (1 шт.), зеленый (1 шт.);
- резисторы 330 Ом (COM-08377 или COM-11507 для пакета, содержащего 20 шт.), 3 шт.;
- резистор 10 кОм (COM-08374 или COM-11508 для пакета, содержащего 20 шт.), 1 шт.;
- кнопка (COM-10302), 1 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*.

Примечание

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты (рис. 4.3) и материалы:

- карандаш;
- макетный нож;
- металлическая линейка;
- клей (клеевой пистолет или клей для моделирования);
- могут пригодиться: дрель и сверло диаметром 4,75 мм;
- могут пригодиться: кусачки (не показаны);
- гофрированный картон (размером примерно 30×30 см) или картонная коробка;
- шаблон корпуса (см. рис. 4.16 далее в этом проекте);
- может пригодиться: мячик для настольного тенниса.

Рис. 4.2. Электронные компоненты для проекта измерителя реакции

Рис. 4.3. Инструменты, рекомендуемые для проекта измерителя скорости реакции

Новый компонент: кнопка

В этом проекте мы сосредоточимся на использовании двух компонентов: светодиода и кнопки. Кнопка здесь играет роль устройства подачи сигнала на вывод Arduino, а это означает, что скетч может реагировать на изменение уровня напряжения на этом выводе. Таким образом, кнопки ввода позволяют создавать схемы, дающие возможность взаимодействия с людьми.

Принцип работы кнопок

Существует много разных типов кнопок, но все они работают сходным образом. На рис. 4.4 показаны несколько примеров кнопок — это небольшие подпружиненные устройства, которые создают электрический контакт в течение всего времени их нажатия, подобно нажатию клавиши клавиатуры. Кнопки используются практически всюду: в пультах дистанционного управления, кофеварках, радиоприемниках, игровых консолях и т. п. Легче перечислить, где они не используются. В действительности, кнопка — это вариант электрического переключателя. На рис. 4.5 показаны символы для кнопочного (*слева*) и обычного (*справа*) переключателей.

Рис. 4.4. Несколько примеров кнопок

Рис. 4.5. Схематический символ кнопочного (*слева*) и обычного (*справа*) переключателей

При включении обычного переключателя металлический язычок замыкает два контакта — наподобие моста — создавая постоянную цепь. При нажатии кнопки металлический язычок также замыкает два контакта, но в данном случае цепь не постоянная, а действующая только в течение времени, пока кнопка удерживается нажатой. Возьмите кнопочный переключатель для этого проекта и внимательно рассмотрите его. Хотя в схематическом символе кнопки (см. рис. 4.5) показано только два контакта, большинство стандартных кнопочных переключателей для использования в беспаечных макетных платах имеют четыре вывода. На рис. 4.6 изображено более точное представление контактов внутри кнопочного переключателя (*слева*) и показано, как может выглядеть такой переключатель, вставленный в макетную плату (*справа*). Обратите внимание — кнопочный переключатель размещается на беспаечной макетной плате так, чтобы он сидел над углублением в центре платы.

Кнопочные переключатели прекрасно подходят в качестве устройств ввода для проектов, поскольку все знают, как они работают. Кроме того, кнопочные переключатели весьма легко включить в схему на плате Arduino. В следующем разделе рассматривается, как это делать.

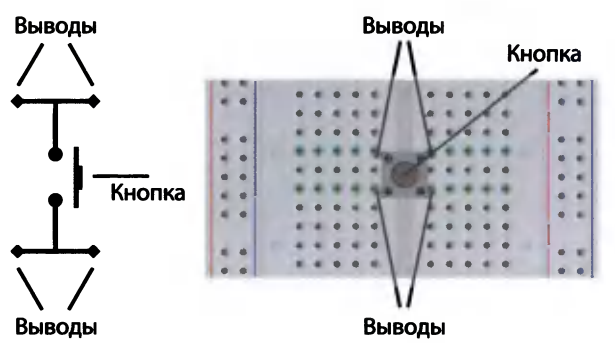


Рис. 4.6. Схема внутреннего устройства кнопочного переключателя (*слева*) и его правильная установка на беспаечной макетной плате (*справа*) — с выводами по обе стороны центрального углубления в плате

Использование резисторов с кнопками

Чтобы кнопочный переключатель можно было использовать в качестве устройства ввода для Arduino, вместе с ним необходимо подключить т. н. *нагрузочный* (pull-up) резистор (рис. 4.7).

Нагрузочный резистор подключается одним выводом к источнику питания, а другим — к устройству ввода (в данном случае — к кнопочному переключателю). Точка в схеме (в данном случае — вывод Arduino), на которую необходимо подать входной сигнал, подключается к точке подключения резистора к кнопке.

Рис. 4.7. Схема подключения кнопки к Arduino с использованием нагрузочного резистора

В схеме, показанной на рис. 4.7, резистор, подключенный одним выводом к источнику напряжения 5 В, повышает напряжение по умолчанию на выводе Arduino, к которому он подключен другим выводом, до 5 В. Этот уровень напряжения считается *высоким* (HIGH). Нажатие кнопки замыкает цепь между выводом Arduino и «землей», в результате чего на вывод Arduino подается напряжение *низкого* (LOW) уровня. Это объясняется тем, что ток всегда выбирает путь наименьшего сопротивления: при отпущенной кнопке ток может протекать только через резистор на вывод Arduino, но при нажатой кнопке создается путь с фактически нулевым сопротивлением.

Создаем прототип измерителя скорости реакции

Проект измерителя скорости реакции совмещает схему подключения светодиода, подобную схемам из предыдущих проектов, со схемой подключения кнопочного переключателя, показанной на рис. 4.7, в одну общую схему (рис. 4.8), которая включает светодиод и определяет нажатие кнопки.

Возьмите беспаечную макетную плату и смонтируйте на ней светодиод и кнопку, как показано на рис. 4.9 и 4.10. Прежде чем приступить к созданию конечной версии измерителя скорости реакции, мы воспользуемся этим прототипом для тестирования правильности работы кода проекта.

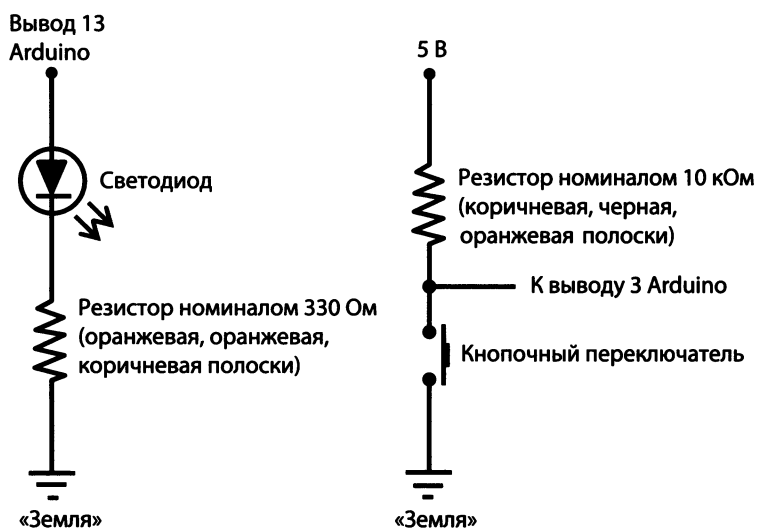


Рис. 4.8. Принципиальная схема прототипа измерителя скорости реакции

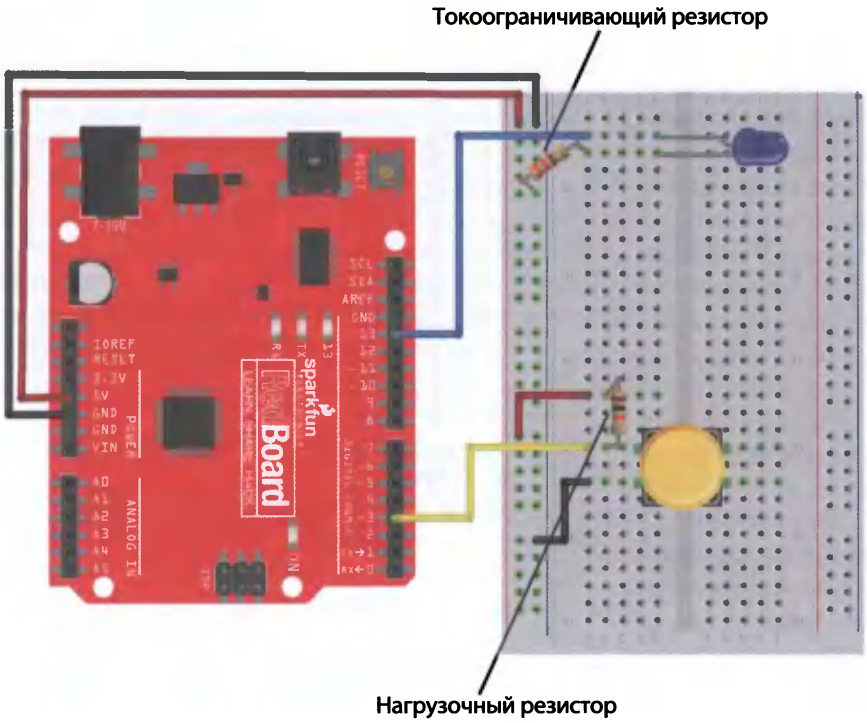


Рис. 4.9. Монтажная схема прототипа измерителя скорости реакции с одной кнопкой и одним светодиодом

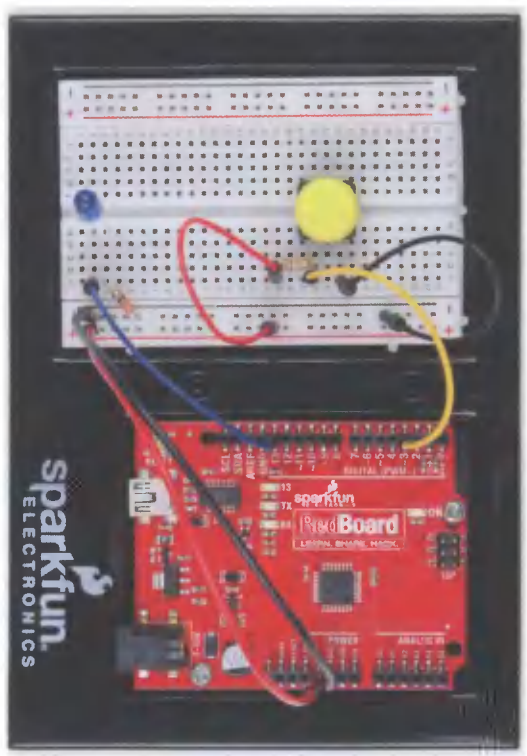


Рис. 4.10. Монтаж прототипа измерителя скорости реакции

При монтаже схемы обратите внимание на два разных номинальных значения резисторов: 330 Ом — для резистора светодиода и 10 кОм — для резистора кнопки (подробная информация о том, как расшифровывать номинальное значение резистора по его разноцветным полосам, приведена в *разд. «Полосатые резисторы» приложения*).

Резистор при светодиоде служит для ограничения тока светодиода — через него светодиод последовательно подключается к «земле». Резистор при кнопочном переключателе является нагрузочным резистором — он подключается одним выводом к выводу 3 платы Arduino, а другим — к источнику питания 5 В. Но собранная нами схема не может делать ничего без кода, который рассматривается в следующем разделе.

Программируем измеритель скорости реакции

С повышением сложности электронных схем и скетчей для них полезно упорядочивать свои мысли, перечисляя по порядку все действия скетча, которые мы хотим выполнить на Arduino. Такое словесное представление скетча обычно называется *псевдокодом*. Псевдокод для измерителя скорости реакции выглядит следующим образом:

1. Выдерживаем паузу произвольной длительности, прежде чем включать светодиод. Это необходимо для того, чтобы предотвратить нажатие кнопки игроком до начала замера времени.
2. Включаем светодиод.
3. Записываем время включения.
4. Запускаем таймер и ожидаем нажатия кнопки.
5. При нажатии кнопки вычисляем время реакции как текущее время таймера минус время запуска таймера.
6. Возвращаем полученный результат.

Довольно просто, не так ли? Тогда давайте откроем среду разработки Arduino и рассмотрим соответствующий скетч.

Создаем функцию *setup()*

Создайте новый скетч в среде разработки Arduino и вставьте в него функцию setup() из листинга 4.1.

Листинг 4.1. Код инициализации и функция setup() скетча измерителя реакции

```
❶ unsigned int waitTime;    //выдерживаем
                            //произвольную паузу,
                            //прежде чем включать
                            //светодиод
  unsigned int startTime;   //время начала отсчета
  unsigned int reactTime;   //вычисленное время реакции

  void setup()
  {
❷ Serial.begin(9600);       //устанавливаем последова-
                            //тельный режим связи
  pinMode(13, OUTPUT);      //конфигурируем вывод 13
                            //платы Arduino для вывода
                            //данных для управления
                            //светодиодом
❸ pinMode(3, INPUT);        //конфигурируем вывод 3
                            //платы Arduino для ввода
                            //данных с кнопки
  }
```

Примечание

Переменные типа int могут содержать целочисленные значения в диапазоне от 0 до 65 535 ($2^{16} - 1$).

Сначала в глобальном пространстве имен ❶ определяются три беззнаковые переменные типа int для хранения значений waitTime (время выдержки), startTime (время включения светодиода) и reactTime (время реакции).

Далее следует определение функции setup(), которая содержит новую инструкцию: Serial.begin(9600) ❷. Обратите внимание, что инструкция Serial пишется с заглавной буквы, далее сразу же без пробела идет точка, а затем опять без пробела идет begin. Эта инструкция несколько отличается от ранее используемых команд тем, что в ней точка разделяет *объект* и его *метод*. Объект — это используемый в программировании компонент наподобие особой переменной, которая обладает свойствами и функциями или действиями. В данной инструкции объект имеет имя Serial. Функции, которые объект может выполнять, в языке Arduino называются *методами*. Метод begin() инициирует (включает) последовательную связь между платой Arduino и компьютером, к которому она подключена. Это позволяет плате Arduino обмениваться данными

с компьютером по кабелю USB. Число в круглых скобках команды (9600) указывает скорость связи в битах в секунду (бодах). Последовательная связь требуется для того, чтобы Arduino мог сообщать компьютеру время реакции на зажигание светодиода. Объект Serial имеет много других методов для реализации обмена данными между Arduino и компьютером. Эти методы будут рассматриваться далее в книге по мере их применения.

Наконец, задаем режим функционирования выводов платы Arduino. В проекте используется один светодиод, подключенный к выводу 13 платы Arduino, включение которого служит начальной точкой отсчета времени реакции. Поэтому этот вывод конфигурируется для вывода данных функцией pinMode(13, OUTPUT). Затем вывод 3 конфигурируется функцией ❸ pinMode(3, INPUT) для работы в режиме ввода данных. Режим ввода данных необходим потому, что Arduino требуется *получать* данные от кнопки (в виде низкого уровня напряжения при ее нажатии), а не *отправлять* данные на кнопку.

Создаем функцию *loop()*

Далее создаем функцию loop() для скетча. Скопируйте код из листинга 4.2 и вставьте его в скетч после функции setup().

Листинг 4.2. Функция loop() для скетча измерителя реакции

```
void loop()
{
//Отображает инструкции для игры
❶ Serial.println("Когда загорится светодиод, нажмите кнопку!");
  Serial.println("Наблюдайте за светодиодом. Готовы?");
❷ waitTime = random(2000, 4000);  //произвольное
                                  //время выдержки
                                  //перед включением
                                  //светодиода от 2000
                                  //до 4000 мс
❸ delay(waitTime);    //выдерживаем заданное время
  //включаем светодиод
  digitalWrite(13, HIGH);

  startTime = ❹millis();//устанавливаем начальное время

  //выполняем ничего не делающий цикл, пока не будет
  //нажата кнопка
❺ while(digitalRead(3) == HIGH)
  {
  }

❻ reactTime = millis() - startTime;    //вычисляем время
                                       //реакции

  digitalWrite(13, LOW);    //выключаем светодиод

  //отображаем результат на мониторе порта (Serial Monitor)
❼ Serial.print("Отлично справились! Ваше время реакции:  ");
❽ Serial.print(reactTime);
❾ Serial.println("миллисекунд");
  delay(1000);    //выдерживаем короткую паузу
                  //прежде чем начинать снова
}
```

Сначала метод println() объекта Serial ❶ выводит в окне монитора порта сообщение с краткими инструкциями по игре. Метод println() отправляет текст на компьютер, добавляя в конце символ перевода строки для перевода курсора на новую строку. В частности, при вызове этого метода в окне **Serial Monitor** (Монитор порта) среды разработки Arduino отображается заключенный в кавычки текст, содержащийся в круглых скобках метода. Окно **Serial Monitor** (Монитор порта) похоже на простое окно чата или терминала, посредством которого можно выполнять обмен данными между платой Arduino и компьютером. Окно можно открыть, щелкнув на значке увеличительного стекла в правом краю панели инструментов среды разработки Arduino (рис. 4.11), выполнив последовательность команд **Tools | Serial Monitor** (Инструменты | Монитор порта)

Рис. 4.11. Открытие окна монитора порта (**Serial Monitor**) с панели инструментов

или нажав комбинацию клавиш быстрого вызова <Ctrl>+<Shift>+<M>. Мы рассмотрим окно монитора порта (**Serial Monitor**) чуть далее в этом проекте.

Задаем время выдержки

Если светодиод будет включаться каждый раз в одно и то же время, игрок может научиться предугадывать момент зажигания, что будет влиять на время его реакции на включение светодиода. Чтобы затруднить предугадывание игроком момента включения светодиода, мы станем включать его через произвольные моменты времени waitTime, которые генерируются с помощью функции random()❷. Функция random() берет в качестве аргументов минимальное и максимальное значения и возвращает *псевдослучайное число* в диапазоне этих значений. (Псевдослучайное число — это такое число, которое кажется случайным, но в действительности таковым не является. Этот вопрос рассматривается более подробно во *врезке «Практикум: делаем время выдержки "более случайным"»* далее в этом проекте.) В данном примере переменной waitTime присваивается псевдослучайное значение в диапазоне от 2000 до 4000, после чего она передается функции delay() ❸, которая задерживает включение светодиода на это количество миллисекунд. Таким образом, игрокам будет труднее предугадать момент зажигания светодиода. При включении светодиода вызывается функция millis() ❹, которая регистрирует время включения. Функция millis() считывает значение внутреннего таймера Arduino и возвращает количество миллисекунд со времени последнего включения или сброса Arduino. Эта функция хорошо подходит для проектов, в которых необходимо засекать время.

Проверяем состояние кнопки с помощью цикла *while()*

Получив значение начала отсчета, скетч начинает исполнять цикл while()❺, в котором ожидается нажатие кнопки. Подобно многим другим языкам программирования, в языке Arduino цикл while() исполняет код внутри фигурных скобок до тех пор, пока выполняется условие, указанное в круглых скобках. В данном случае указано следующее условие:

```
digitalRead(3) == HIGH
```

Функция digitalRead() считывает напряжение на указанном в скобках выводе платы и возвращает значение HIGH или LOW для напряжений 5 В и 0 В («земля») соответственно. В данном случае проверяется напряжение на выводе 3 платы. Снова взгляните на нагрузочный резистор (рис. 4.9), подключенный к кнопке. В состоянии кнопки по умолчанию — разомкнутом — на выводе платы присутствует высокий (HIGH) уровень. При нажатии кнопки вывод 3 платы подключается к «земле», в результате чего на нем устанавливается низкий (LOW) уровень. Оператор из двух знаков равенства (==) означает сравнение на равенство двух значений. (Этот и другие операторы сравнения рассматриваются во *врезке «Логические операторы сравнения»* далее в этом проекте.)

Пока кнопка не нажата, условие удовлетворяется, и цикл while() исполняется бесконечно, не допуская исполнения следующего за ним кода. Но обратите внимание на то, что цикл не содержит никакого кода. Такой цикл называется *блокирующим* — его назначением является не исполнение какого-либо кода, а просто предотвращение исполнения любого другого кода. В результате нажатия кнопки условие больше не удовлетворяется, и скетч начинает исполнять код, следующий за циклом while().

Вычисляем и отображаем время реакции

Далее скетч вычисляет время реакции ❻, отнимая значение startTime от текущего значения таймера, которое получается командой millis().

Наконец, скетч выключает светодиод и отображает значение скорости реакции reactTime в окне монитора порта. Чтобы эта информация была более заметной, она отображается с описательным текстом: **Отлично справились! Ваше время реакции:** посредством метода print() ❼ объекта Serial.

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ СРАВНЕНИЯ

Как следует из их названия, логические операторы сравнения выполняют операцию сравнивания двух значений. Например, оператор == (двойной знак равенства) проверяет, равны ли сравниваемые значения. Операция сравнения может возвратить только одно из двух значений: true (истина) — это означает, что условие сравнения удовлетворяется, или false (ложь) — это означает, что условие сравнения не удовлетворяется. Все операторы сравнения языка Arduino и соответствующие условия сравнения приводятся в табл. 4.1.

Таблица 4.1. *Операторы сравнения языка Arduino*

Оператор	Операция сравнения
==	Равно
!=	Не равно
>	Больше чем
>=	Больше чем или равно
<	Меньше чем
<=	Меньше чем или равно

Например, результатом выполнения операции сравнения 2 == 4 будет значение ложь, поскольку два не равняется четырем. Но результатом выполнения операции сравнения 2 <= 4 будет значение истина, поскольку два меньше, чем четыре. В скетчах операторы сравнения часто используются вместе с инструкциями if() или while() — для исполнения блока кода при удовлетворении определенного условия. Например, цикл while(digitalRead(3) == HIGH) в листинге 4.2 означает следующее: пока на выводе 3 платы сохраняется высокий уровень, что определяется инструкцией digitalRead(3), исполняем блокирующий цикл, в противном случае переходим к исполнению следующей строки кода.

При работе с операторами сравнения часто допускается ошибка использования только одного знака равенства вместо двух. Помните следующее: один знак равенства означает операцию присваивания значения переменной, тогда как двойной знак равенства означает операцию сравнения двух значений на равенство.

Мы будем часто использовать операторы сравнения в дальнейших проектах, где станем оценивать данные, получаемые с разных датчиков и других устройств ввода. Некоторые из них мы также используем дальше в этом проекте.

Подобно методу println(), метод print() отправляет текст в скобках на компьютер, но не переводит курсор на новую строку. В результате к отображенному тексту можно добавить дополнительную информацию — в данном случае время реакции, для чего снова же используется метод Serial.print(reactTime) ❽. Предложение завершается словом **миллисекунд** с переводом курсора на новую строку — эта операция выполняется с помощью метода println() ❾.

Обратите внимание, что необходимо явно указывать все выводимые на экран символы, включая количество пробелов между цифрами или буквами. С помощью методов print() и println() можно компоновать, форматировать и организовывать текст для отображения в окне монитора порта требуемым образом.

Последняя строка кода в функции loop() вызывает функцию delay(), которая приостанавливает исполнение скетча на короткое время, после чего скетч начинает исполняться в новом цикле.

СПЕЦИАЛЬНЫЕ УПРАВЛЯЮЩИЕ СИМВОЛЫ

При выводе текста на печать или экран, кроме всех видимых символов, необходимо также указывать символы форматирования, включая пробелы. Для выполнения специального форматирования применяется набор зарезервированных символов, называемых *управляющей последовательностью*, или *escape-последовательностью*. Например, последовательность символов \n переводит курсор на новую линию. Таким образом, команда Serial.print(«Здравствуй, Arduino!\n»); будет эквивалентна команде Serial.println(«Здравствуй, Arduino!»);.

Управляющие последовательности можно использовать в командах вывода для форматирования выводимого текста или для добавления в него других специальных символов. В табл. 4.2 приводится несколько основных управляющих последовательностей.

Таблица 4.2. *Несколько основных управляющих последовательностей*

Управляющая последовательность	Результат действия
\t	Табуляция
\n	Новая линия
\'	Одинарная кавычка
\"	Двойная кавычка
\xhh	Символ ASCII, где hh представляет шестнадцатеричный номер

Тестируем скетч измерителя скорости реакции

Теперь можно приступить к тестированию схемы измерителя скорости реакции. Сохраните скетч, а затем скомпилируйте его и загрузите в Arduino. Чтобы иметь возможность наблюдать за выводом данных скетчем на компьютер, откройте окно монитора порта. Это можно сделать, выполнив последовательность команд меню **Tools | Serial Monitor** (Инструменты | Монитор порта) или щелкнув на значке увеличительного стекла в правом краю панели инструментов среды разработки Arduino (см. рис. 4.11).

Обратите внимание на выпадающее меню в правом нижнем углу окна монитора порта (рис. 4.12). С помощью этого меню указывается скорость последовательной передачи в бодах (битах в секунду), которая по умолчанию установлена в 9600 бод. Эта же скорость последовательной связи указывается в команде Serial.begin(9600) для инициализации объекта Serial. В мониторе порта всегда должна быть установленная такая же скорость последовательной передачи, как и указанная в скетче в команде инициализации Serial.begin(XXXX). В противном случае в окне монитора порта будет выводиться искаженный текст вплоть до бессмысленного набора символов.

В большинстве случаев можно устанавливать любую скорость последовательной передачи: от 300 до 250 000 бод, но наиболее часто используется скорость в 9600 бод. В общем случае, более медленная скорость обеспечивает более надежную передачу данных и потребляет меньше мощности и ресурсов Arduino, но при этом замедляется исполнение цикла loop(). Если требуется быстрая реакция и повышенное энергопотребление не столь важно, можно установить более высокую скорость передачи.

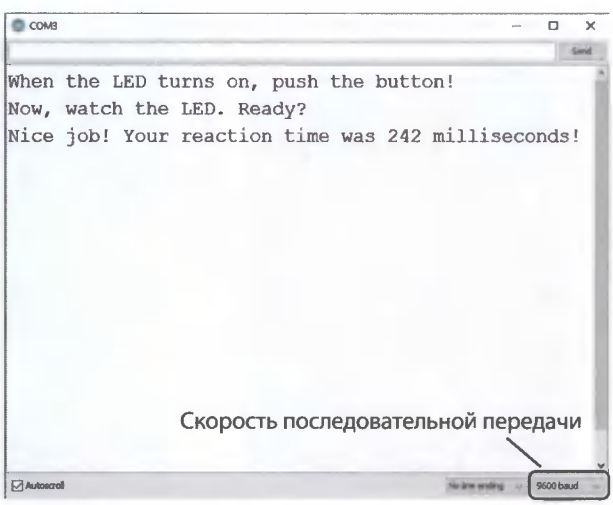


Рис. 4.12. Вывод скетча тестирования измерителя скорости реакции в окне монитора порта

Теперь приступим собственно к тестированию прототипа измерителя скорости реакции. Запустите скетч на исполнение и внимательно наблюдайте за светодиодом. Когда светодиод загорится, нажмите кнопку. Старайтесь сделать это как можно быстрее после включения светодиода. После нажатия кнопки скетч выведет в окно монитора порта время вашей реакции в миллисекундах, как показано в примере на рис. 4.12. Какая ваша скорость реакции? Вызовите на соревнование кого-либо из своих друзей. Средняя скорость реакции составляет 215 миллисекунд. А какова ваша скорость по сравнению со средней?

Примечание

Некоторые компоненты, например устройства GPS или жидкокристаллические дисплеи, применяют разные скорости обмена данными с Arduino. Поэтому при использовании нового компонента следует проверить допустимую им скорость передачи данных и установить эту скорость в скетче.

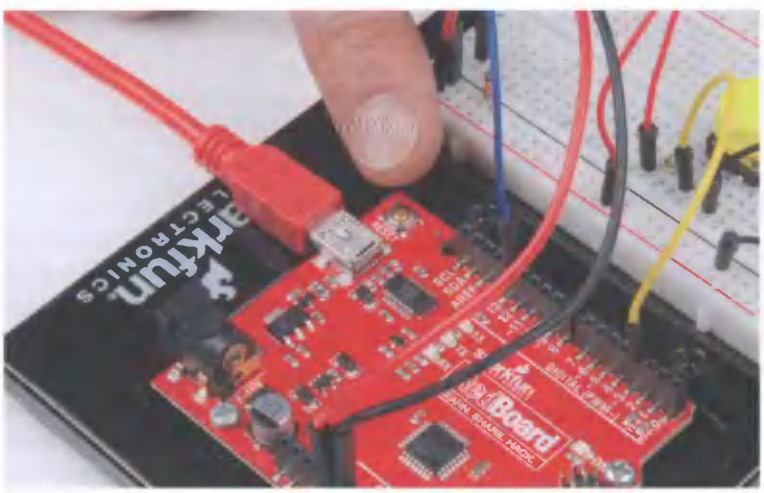


Рис. 4.13. Повторный запуск скетча нажатием кнопки сброса

Следующий раунд

Чтобы попробовать улучшить свое время реакции, просто нажмите кнопку сброса Arduino — это кнопка желтоватого цвета рядом с разъемом кабеля USD (рис. 4.13). Скетч перезапустится, и можно будет играть снова. Внимательно наблюдайте за светодиодом. Ну как, получилось быстрее в этот раз?

Добавляем аркадный элемент

Чтобы придать этому проекту подобие аркадной игры, в него можно добавить простые индикаторы, сравнивающие вашу скорость реакции с определенным значением. Для начала это значение можно установить равным среднему значению скорости реакции: 215 миллисекунд.

Итак, добавьте в схему прототипа еще два светодиода: зеленый, означающий более быстрое, чем среднее, время реакции, и красный, означающий более медленное. Подключим эти светодиоды к выводам 11 и 12 платы Arduino. Монтажная схема подключения показана на рис. 4.14, а модернизированный проект — на рис. 4.15.

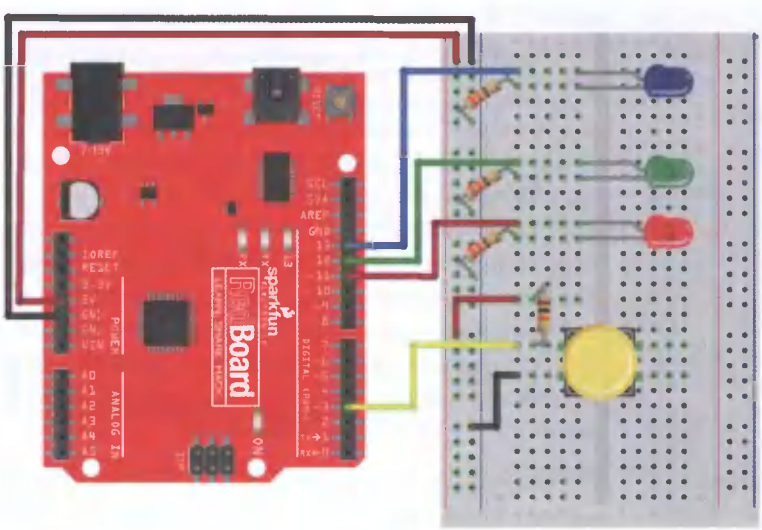


Рис. 4.14. Монтажная схема подключения дополнительных светодиодов

ПРАКТИКУМ: ДЕЛАЕМ ВРЕМЯ ВЫДЕРЖКИ «БОЛЕЕ СЛУЧАЙНЫМ»

В большинстве цифровых устройств, одним из которых является Arduino, задача создания по-настоящему случайных чисел представляет большую трудность, поскольку для этого требуются сложные математические вычисления. В случае с Arduino каждый последующий вызов функции random() выполняет вычисления на основе результата предыдущего вызова этой функции. Поскольку следующее якобы случайное число вычисляется из предыдущего результата, выдаваемая последовательность чисел будет всегда одинаковой.

Например, на плате Arduino авторов (и, скорее всего, на вашей также) первый вызов функции random(2000, 4000) всегда присваивает переменной waitTime значение 2807. Второе число последовательности всегда будет 3249, третье — 2073 и т. д.

Значение переменной waitTime можно сделать «более случайным» с помощью вызова функции randomSeed() в функции setup(). Эта функция генерирует начальное число, называющееся *затравочным*, для последовательности псевдослучайных чисел, генерируемых функцией random(). Когда Arduino начинает исполнять скетч, затравочное число по умолчанию равно 1, вследствие чего первый вызов функции random(2000, 4000) будет всегда возвращать число 2807.

Чтобы время выдержки перед включением светодиода было «более случайным», добавьте следующую строку кода в функцию setup() непосредственно перед закрывающей фигурной скобкой:

```
randomSeed(analogRead(A5));
```

Этот код генерирует затравочное число, полученное на основании текущего значения напряжения на аналоговом выводе A5 платы Arduino. Функция analogRead() рассматривается более подробно в *примере 5*, но на данном этапе будет достаточно знать, что она считывает уровень напряжения на выводе платы Arduino, номер которого ей передается в качестве параметра. Поскольку в схеме измерителя скорости реакции вывод A5 не подключен, напряжение на нем «плавает» или скачет вверх и вниз непредсказуемо.

Если в скетч добавить эту линию кода, тогда при каждом его исполнении первый вызов функции random(2000, 4000) должен возвращать другое число. Чтобы удостовериться в этом, добавьте следующие две строки кода в скетч после вызова функции delay(waitTime):

```
Serial.print("waitTime = ");
Serial.println(waitTime);
```

Далее закомментируйте вызов функции randomSeed(), добавив две косые черты в начале строки, выполните скетч несколько раз и обратите внимание на выводимые начальные значения переменной waitTime. Затем удалите комментарий с вызова randomSeed() и повторите процесс.

Тем не менее, даже при использовании функции randomSeed() в последовательности чисел, генерируемых функцией random(), будет наблюдаться определенная закономерность. Но это уже тема для другой книги или, возможно, для диссертации в области компьютерных вычислений.

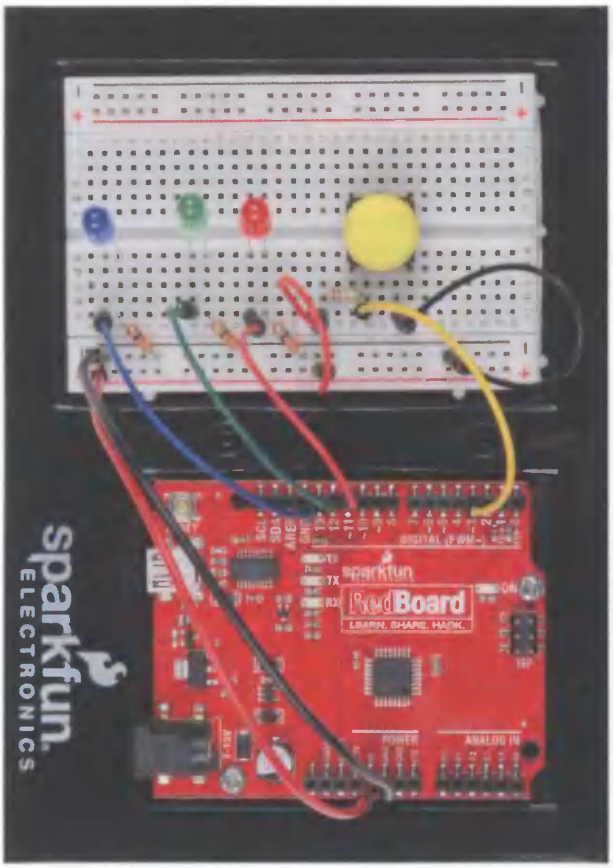


Рис. 4.15. Модернизированный проект с двумя дополнительными светодиодами

Обновляем код с учетом дополнительных светодиодов

Сами по себе дополнительные светодиоды ничего делать не будут. Нам нужно добавить в скетч код, чтобы включать зеленый светодиод при времени реакции более быстром, чем среднее, и красный — при более медленном.

В частности, в функцию setup() нужно добавить инструкции pinMode() для установки режима работы OUTPUT (вывод данных) для выводов 11 и 12 платы Arduino. Отредактированная таким образом функция setup() показана в листинге 4.3, где добавленный код выделен полужирным шрифтом.

Листинг 4.3. Функция setup() скетча измерителя реакции, модифицированная для работы с двумя дополнительными светодиодами

```
void setup()
{
  Serial.begin(9600);          //устанавливаем
                               //последовательный режим
                               //связи
  pinMode(13, OUTPUT);         //конфигурируем вывод 13
                               //платы Arduino для вывода
                               //данных для управления
                               //контрольным светодиодом
  pinMode(12, OUTPUT);         //конфигурируем вывод 12
                               //платы Arduino для вывода
                               //данных для управления
                               //зеленым светодиодом
  pinMode(11, OUTPUT);         //конфигурируем вывод 11
                               //платы Arduino для вывода
                               //данных для управления
                               //красным светодиодом
  pinMode(3, INPUT);           //конфигурируем вывод 3
                               //платы Arduino для ввода
                               //данных с кнопки
}
```

В общем, мы просто вставили в функцию setup() две дополнительные команды pinMode(), конфигурирующие выводы 11 и 12 платы Arduino для работы в режиме OUTPUT для управления двумя дополнительными светодиодами.

Управление потоком выполнения скетча с помощью условного оператора *if ...else*

Далее нам нужно добавить в исходный скетч немного логики решений. В языке программирования Arduino оператор if...else позволяет управлять потоком исполнения скетча. Суть этого оператора можно выразить следующим образом: если указанное в круглых скобках условное выражение удовлетворяется (имеет значение ИСТИНА), тогда выполняем код в фигурных скобках после части if. В противном случае (выражение имеет значение ЛОЖЬ) исполняем код в фигурных скобках части else. Общий синтаксис оператора if...else приводится в листинге 4.4.

Листинг 4.4. Синтаксис оператора if..else в языке Arduino

```
if(❶ expression)  //если условное выражение
                  //в скобках имеет значение ИСТИНА,
                  //выполняем код в этих фигурных
                  //скобках
{
❷
}
❸ else   //в противном случае выполняем код в этих
         //фигурных скобках
{
❹
}
```

Листинг 4.5. Код оператора if..else для игры измерителя скорости реакции

```
❶ if (❷ reactTime <= 215)
  {
❸   digitalWrite(12, HIGH);  //включаем зеленый
                             //светодиод
❹   digitalWrite(11, LOW);   //выключаем красный
                             //светодиод
  }
❺ else
  {
    digitalWrite(12, LOW);   //выключаем зеленый
                             //светодиод
    digitalWrite(11, HIGH);  //включаем красный
                             //светодиод
  }
```

Условное выражение ❶ является булевым выражением, которое может иметь одно из двух значений: ИСТИНА или ЛОЖЬ, подобно значениям, рассматриваемым во *врезке «Логические операторы сравнения»* ранее в этом проекте. Если выражение имеет значение ИСТИНА, выполняется код, заключенный в фигурные скобки ❷. Если же выражение имеет значение ЛОЖЬ, исполнение переходит к коду после фигурных скобок. Оператор if() может использоваться один или же с дополнительной частью else ❸. В последнем случае, если выражение имеет значение ЛОЖЬ, код в фигурных скобках части if ❷ также пропускается и выполняется код в фигурных скобках части else ❹.

В игре измерителя скорости реакции с помощью оператора if...else мы будем включать зеленый светодиод, если время реакции меньше или равно 215 миллисекундам, и красный в противном случае (то есть когда время реакции больше, чем 215 миллисекунд). Это эталонное значение впоследствии можно будет сделать меньшим или большим, чтобы сделать игру более трудной или легкой, но на данном этапе это хорошее начальное значение. Соответствующий код приводится в листинге 4.5.

Скетч начинает исполнять логику включения зеленого или красного светодиода в точке ❶ с помощью оператора if...else. Булево выражение reactTime <= 215 ❷ проверяет значение переменной reactTime — равно ли оно или меньше 215. При положительном результате исполняется код для включения зеленого светодиода ❸. Когда включен зеленый светодиод, красный светодиод должен быть выключен, поэтому в данном блоке также исполняется код ❹ для выполнения этой операции. Если же условие оператора if не удовлетворяется, выполняется код блока else ❺ для включения красного светодиода и выключения зеленого.

Полный код скетча для измерителя скорости реакции

Код оператора if...else нужно вставить в функцию loop() после кода для выключения красного контрольного светодиода, как показано полужирным шрифтом в листинге 4.6. (Троеточия в начале и в конце листинга означают, что предшествующий и последующий коды были опущены для краткости.)

Листинг 4.6. Вставка в основной скетч кода оператора if..else для управления зеленым и красным светодиодными индикаторами скорости реакции

```
--. . .--
  //выполняем ничего не делающий цикл, пока не будет
  //нажата кнопка
  while(digitalRead(3) == HIGH)
  {
  }

  reactTime = millis() - startTime;    //вычисляем время
                                       //реакции

  digitalWrite(13, LOW);    //выключаем контрольный
                            //светодиод

  if (reactTime <= 215)
  {
  digitalWrite(12, HIGH);   //включаем зеленый
                            //светодиод
  digitalWrite(11, LOW);    //выключаем красный
                            //светодиод
  }

  else
  {
  digitalWrite(12, LOW);    //выключаем зеленый
                            //светодиод
  digitalWrite(11, HIGH);   //включаем красный
                            //светодиод
  }

  //отображаем результат на Мониторе порта
  //(Serial Monitor)
--. . .--
```

НАСТРОЙКА ИГРЫ: УПРАВЛЯЕМ СЛОЖНОСТЬЮ

Наш измеритель скорости реакции можно улучшить еще больше, оснастив его возможностью настройки уровня сложности. Поскольку программируете ее вы сами, то это не будет представлять никакой сложности. Эталонное значение времени реакции можно настраивать, изменяя число в коде if (reactTime<=215).

Хотите сделать так, чтобы игру стало невозможно выиграть? Измените это число на меньшее — типа 100 миллисекунд. Хотите, чтобы все чувствовали себя победителями? Измените это число на большее — типа 500 миллисекунд. Вы программист, так что правила устанавливаете вы!

Добавив новый код в скетч, загрузите его в плату Arduino. Откройте окно монитора порта и сыграйте несколько раз, чтобы убедиться, что все работает должным образом. Удается ли вам зажечь зеленый светодиод? Если да, то у вас отличная реакция.

Создаем корпус для измерителя скорости реакции

Убедившись, что наш прототип работает должным образом, можно приступать к созданию для него постоянного корпуса. Чтобы сделать этот проект как можно проще, мы установим компоненты измерителя скорости реакции в небольшую картонную коробку с размером поверхности около 100×130 мм. Но вы можете вставить ее в любой подходящий корпус, который у вас имеется под рукой, — пустую коробку из-под хлопьев или овсянки или в любую другую коробку из жесткого картона.

Опять же, рассматриваемый здесь корпус выполнен в виде старомодной аркадной игры с наклеенными на него картинками, но вы можете оформить свой корпус, как вам захочется. На рис. 4.16 показан шаблон для корпуса игры. В нем сделаны отверстия для кнопки, для контрольного светодиода и для светодиодов-индикаторов скорости реакции. Шаблон можно загрузить вместе с остальными ресурсами книги по адресу: **https://www.nostarch.com/arduinoinventor/**, впрочем, вы можете сделать отверстия для светодиодов и кнопки, где вам более удобно.

Вырезаем отверстия в корпусе

Если вы используете наш шаблон, распечатайте его и наклейте на лицевую сторону корпуса игры. Независимо от того, используете ли вы наш шаблон или нет, в лицевой панели корпуса нужно сделать всего четыре отверстия: три — для светодиодов и одно — для кнопки. Отверстия можно сделать с помощью макетного ножа (рис. 4.17) или просверлить их. Диаметр светодиодов 5 мм, так что — для плотной посадки светодиодов — отверстия для них можно просверлить с помощью сверла диаметром 4,75 мм. Отверстие для кнопки рекомендуется просверлить сверлом диаметром 8 мм или же проткнуть с помощью острого карандаша.

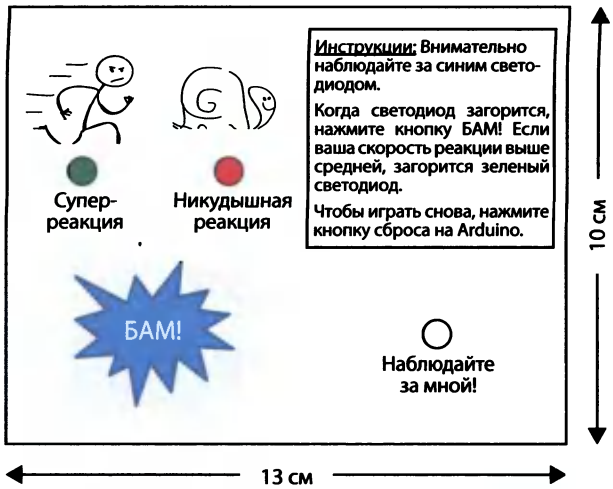


Рис. 4.16. Шаблон и графика для корпуса игры измерителя скорости реакции (в уменьшенном виде)

Собираем электронную часть

Вырезав отверстия в корпусе, вставляем в них соответствующие детали. Нам нужно будет переместить три светодиода и кнопку с беспаечной макетной платы на лицевую панель корпуса игры.

Вставляем светодиоды и кнопку в корпус

Сначала с тыльной стороны лицевой панели корпуса вставляем в приготовленные отверстия светодиоды, проверяя при этом, чтобы они плотно сидели в отверстиях. Если отверстия слишком большие, и светодиоды не сидят в них плотно, их можно закрепить каплей клея из клеевого пистолета, как показано на рис. 4.18.

Рис. 4.17. Вырезание отверстий в корпусе с помощью макетного ножа

Затем вставляем в корпус кнопку. У кнопок, поставляемых в наборе изобретателя SparkFun, насадка снимается. Снимите насадку, вставьте кнопку в корпус с тыльной стороны панели, и приклейте ее, как показано на рис. 4.19.

Когда клей застынет, наденьте насадку обратно на кнопку. В своем стремлении получить наилучший результат игроки будут бить по этой кнопке довольно сильно, поэтому не скупитесь с клеем, чтобы она была надежно закреплена. Завершив установку кнопки, испытайте ее работу. Кнопка должна нажиматься до конца,

Рис. 4.18. Закрепление в корпусе светодиодов с помощью клеевого пистолета

поэтому убедитесь, что при нажатии кнопки ее насадка не упирается в лицевую панель.

Рис. 4.19. Закрепление в корпусе кнопки с помощью клеевого пистолета

> **Примечание**
>
> Будьте осторожны при закреплении деталей проекта с помощью горячего клея из клеевого пистолета. Горячий клей не зря называется горячим!

Подключаем компоненты к Arduino

Подключим теперь светодиоды к макетной плате с помощью проволочных перемычек с гнездовым разъемом на одном конце и штыревым на другом. Не забывайте, что короткий вывод светодиодов через последовательный резистор сопротивлением 330 Ом подключается к отрицательной шине питания («земле») макетной платы, а длинный — к соответствующему выводу платы Arduino. Поскольку выводы светодиодов несколько длинноваты, их следует слегка укоротить с помощью кусачек. При этом укорачивать их нужно так, чтобы сохранить относительную длину выводов, — то есть короткий должен оставаться коротким, а длинный — длинным (рис. 4.20).

Если по какой-либо причине определить полярность выводов светодиода по их длине не удается, это можно сделать по метке на пластиковой линзе светодиода. В частности, на круглом фланце в нижней части линзы светодиода обычно сделана плоская грань со стороны отрицательного вывода. Размер грани весьма небольшой, но при близком рассмотрении ее можно увидеть.

Рис. 4.20. Укорачивание выводов светодиодов с сохранением их относительной длины

Рис. 4.21. Подключение к светодиодам гнездового конца проволочных перемычек

Наденьте гнездовые разъемы проволочных перемычек на выводы светодиодов (рис. 4.21). Рекомендуется для подключения отрицательного (короткого) вывода светодиодов использовать перемычки черного цвета. Так вам будет легче разбираться в монтаже.

Другие разъемы перемычек, штыревые, вставляются в соответствующие гнезда макетной платы: разъемы для контрольного, синего, светодиода — в гнезда **e2** и **e3**, зеленого индикаторного — в гнезда **e8** и **e9**, а красного индикаторного — в гнезда **e12** и **e13**.

Далее вводим в схему кнопку. Из четырех выводов кнопки нам нужно подключить только два — с одной стороны кнопки. Подключите гнездовые разъемы перемычек к выводам кнопки, как показано на рис. 4.22.

Затем вставьте один штыревой разъем в шину отрицательного питания («землю») на макетной плате, а другой — в ряд на макетной плате, к которому подключены один вывод резистора сопротивлением 10 кОм и вывод 3 платы Arduino. (Если монтаж вашего прототипа выполнен, как в схеме на рис. 4.14, тогда эти перемычки подключаются к гнездам **e20** и **e22** макетной платы.) Поскольку выводы кнопки не имеют полярности, нет разницы, какой вывод кнопки подключается к какому гнезду макетной платы.

Собрав всю схему, подключите плату Arduino к компьютеру и откройте окно монитора порта, чтобы убедиться, что работоспособность схемы не была нарушена. Если все в порядке, в окне монитора порта должно отобразиться сообщение с инструкцией игры. Когда загорится синий светодиод, как можно быстрее нажмите кнопку. В окне монитора порта должно отобразиться ваше время реакции, а также должен загореться один из индикаторных светодиодов.

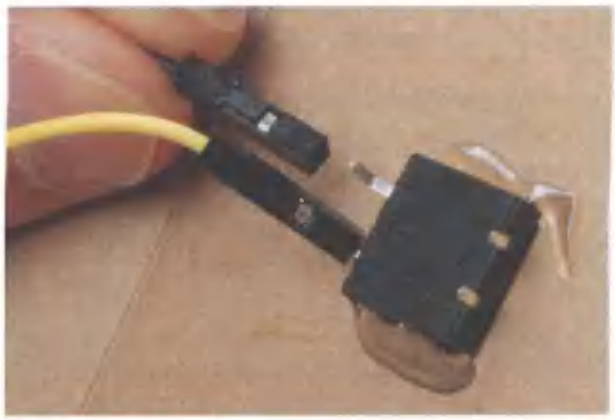


Рис. 4.22. Подключение гнездовых разъемов монтажных перемычек к выводам кнопки

Если схема проявляет признаки неправильной работы, проверьте надежность всех соединений, а также сверьте свой монтаж со схемами на рис. 4.14 и 4.15, чтобы убедиться в отсутствии ошибок.

Декорируем корпус

В завершение немного украсьте свою новую игру. Дайте волю воображению! Например, корпус игры можно украсить наклейками или покрасить. Авторам нравится использовать в своих проектах мячики от настольного тенниса, и, поскольку у нас осталась половинка мячика после *проекта 2*, мы наклеили ее поверх синего контрольного светодиода, как показано на рис. 4.23.

Рис. 4.23. Завершенный проект измерителя скорости реакции

Идем дальше...

На следующем шаге попробуйте объединить знания, полученные в результате реализации первых трех проектов, со знаниями с этого проекта, чтобы сделать его более интересным. Например, добавьте еще больше светодиодов или расширьте игру для двух игроков.

Экспериментируем с кодом

Добавьте в игру еще два светодиода, чтобы сделать четырехсветодиодный индикатор для более точного отображения скорости реакции, — чем быстрее время реакции, тем больше включается светодиодов. Для этого нам потребуется вложенный оператор if...else if. Условные операторы можно организовать в стек таким образом, чтобы дать указание коду, что делать в разных ситуациях. В частности, если первое условие не выполняется, проверяем второе, если и это не выполняется, проверяем следующее. И так до тех пор, пока не дойдем до последней части else, код в которой исполняется, если все предыдущие условия не были удовлетворены. Пример такой вложенной условной логики приводится в листинге 4.7. В коде подразумевается, что в схему добавлены два дополнительных светодиода, подключенные к выводам 10 и 9 платы Arduino. Не забудьте настроить эти выводы на работу в режиме OUTPUT, добавив две соответствующие инструкции pinMode() в функцию setup().

Листинг 4.7. Фрагмент кода вложенного оператора if...else if

```
❶ if (reactTime <= 215)
  {
     //включаем все индикаторные светодиоды
     digitalWrite(12, HIGH);
     digitalWrite(11, HIGH);
     digitalWrite(10, HIGH);
     digitalWrite(9, HIGH);
  }
❷ else if (reactTime <= 250)
  {
     //включаем три индикаторных светодиода
     digitalWrite(12, LOW);
     digitalWrite(11, HIGH);
     digitalWrite(10, HIGH);
     digitalWrite(9, HIGH);
  }
❸ else if (reactTime <= 300)
  {
     //включаем два индикаторных светодиода
     digitalWrite(12, LOW);
     digitalWrite(11, LOW);
     digitalWrite(10, HIGH);
     digitalWrite(9, HIGH);
  }
❹ else
  {
     //включаем один индикаторный светодиод
     digitalWrite(12, LOW);
     digitalWrite(11, LOW);
     digitalWrite(10, LOW);
     digitalWrite(9, HIGH);
  }
```

Оператор if() в точке ❶ проверяет время реакции: равно или меньше оно 215 миллисекунд, и при положительном результате включает все четыре индикаторных светодиода. Следующие два оператора else if обрабатывают результаты в диапазонах 215–250 мс ❷ и 250–300 мс ❸ и включают три и два индикаторных светодиода соответственно. Наконец, оператор else ❹ обрабатывает результаты, превышающие 300 мс, и включает один индикаторный светодиод.

Если вам требуется дополнительная информация по этому коду, см. скетч этого примера в архиве ресурсов, сопровождающих книгу, по адресу: **https://www.nostarch.com/arduinoinventor/**.

Модифицируем схему

Какой бы ни была интересна наша игра, она будет еще интересней, если в нее можно играть вдвоем. Для этого нужно просто добавить еще одну кнопку и изменить назначение индикаторных светодиодов, чтобы они указывали победителя. В предлагаемой здесь модификации победа игрока 1 указывается включением зеленого светодиода, а игрока 2 — включением красного.

Модификацию начнем с добавления второй кнопки. На рис. 4.24 показана добавленная вторая кнопка внизу макетной платы. Обратите внимание, что схема подключения этой кнопки просто дублирует схему подключения первой, лишь с той разницей, что другой вывод кнопки подключен к выводу 2 платы Arduino.

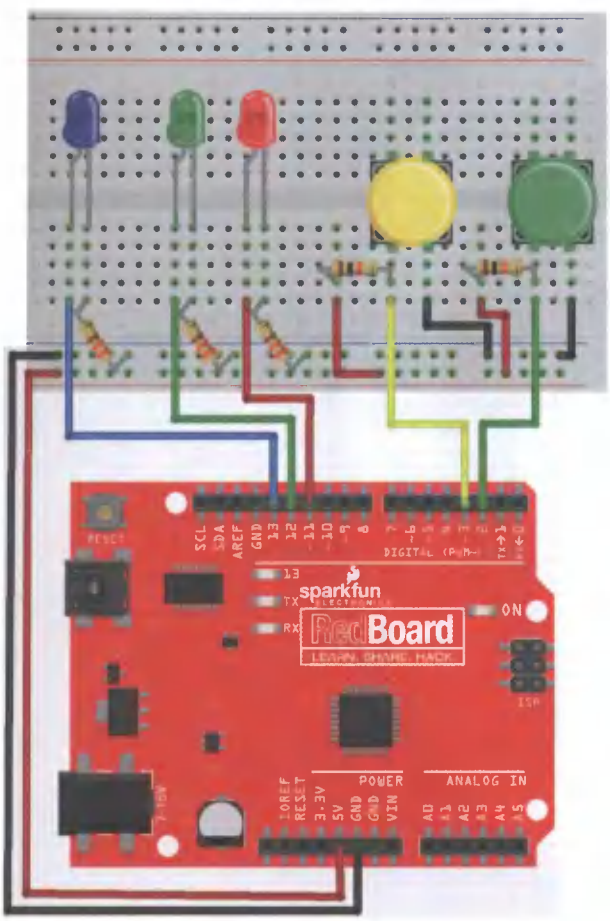


Рис. 4.24. Добавление второй кнопки для игры вдвоем

Примечание

Если оба игрока нажмут кнопки точно одновременно, победа будет присуждена игроку 1 (зеленый светодиод). Хотя такое событие может произойти достаточно редко, но это все равно недостаток игры. Попробуйте модифицировать код таким образом, чтобы при одновременном нажатии кнопок зажигались оба индикаторных светодиода.

Принципиальная схема модификации и скетч для игры доступны в архиве ресурсов, сопровождающих книгу, по адресу: **https://www.nostarch.com/arduinoinventor/**.

Теперь можно пригласить сразиться в новую игру членов семьи и друзей. Кто в вашей семье самый быстрый? А из друзей? Кто самый быстрый изо всех?

РАЗНОЦВЕТНЫЙ НОЧНИК

Одним из замечательных свойств современной цифровой электроники в целом и микроконтроллеров в частности является их определенный уровень интеллектуальности. Микроконтроллеры могут считывать показания датчиков и на основании полученных данных принимать определенные решения. Датчики представляют собой электронные устройства, которые собирают данные о тех или иных аспектах окружающей среды и преобразовывают эти данные в представление, понимаемое микроконтроллерами.

С помощью датчиков проектам можно придать способность реагировать на всякого рода управляющие воздействия типа температуры, звука, близости объектов и т. п. Но мы начнем наше знакомство с этими устройствами с датчика освещенности, создав на его основе ночник, реагирующий на разные уровни освещенности. Проект в его завершенном виде показан на рис. 5.1.

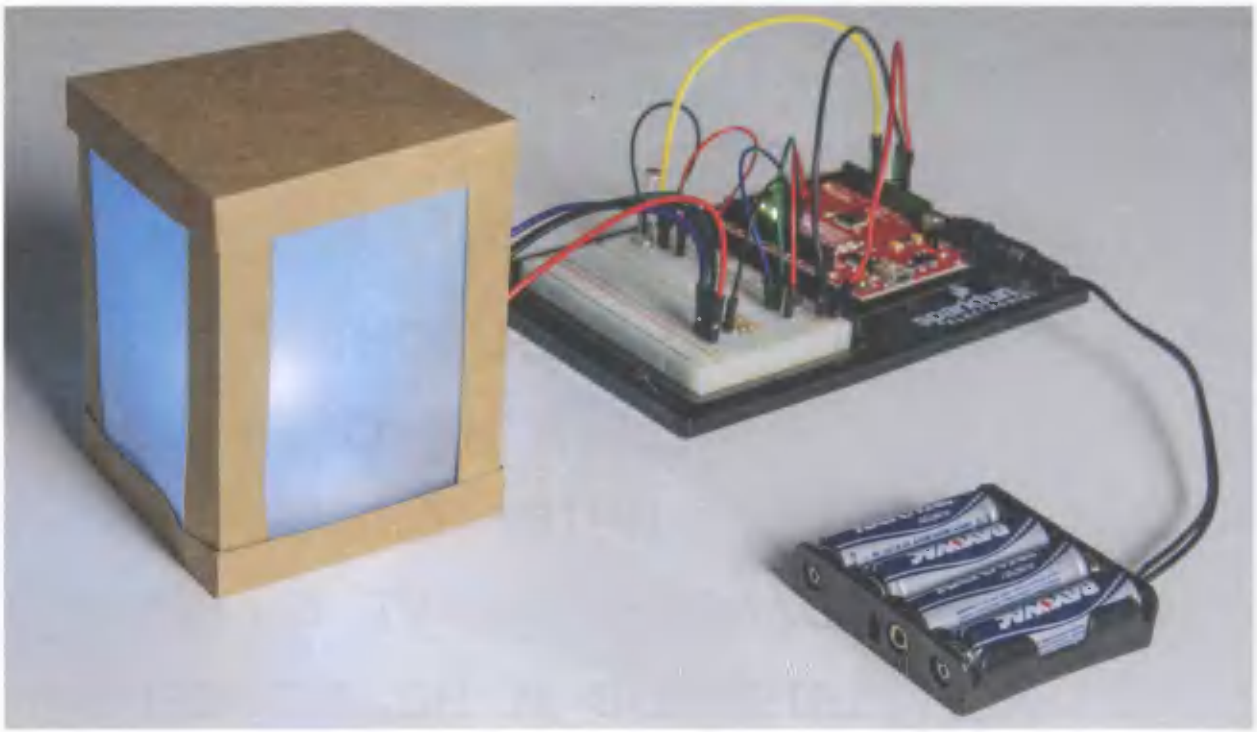


Рис. 5.1. Завершенный проект ночника

Необходимые компоненты, инструменты и материалы

В этом проекте используются новый тип светодиода, а также новый электронный компонент — *фоторезистор*, представляющий собой особый вид резистора, сопротивление которого зависит от уровня его освещенности. Кроме создания электронной части ночника, рекомендуется также сделать для него и абажур, изготовление которого даст новый толчок развитию ваших дизайнерских способностей. Впрочем, если вы предпочтете пропустить эту часть работы, ничего страшного не произойдет. Однако, если же вы все-таки решите заняться абажуром для ночника, в архиве ресурсов, сопровождающем эту книгу, вы найдете шаблон, который можно использовать в качестве отправной точки для его изготовления.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 5.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. (на рис. 5.2 не показан);
- беспаечная макетная плата (PRT-12002), 1 шт.;
- миниатюрная беспаечная макетная плата (PRT-12043)*, 1 шт. (на рис. 5.2 не показана);

- трехцветный светодиод (красный, зеленый, синий) с общим катодом (COM-9264), 1 шт.;
- резисторы 330 Ом (COM-08377 или COM-11507 для пакета, содержащего 20 шт.), 3 шт.;
- резистор 10 кОм (COM-08374 или COM-11508 для пакета, содержащего 20 шт.), 1 шт.;
- фоторезистор (SEN-09088), 1 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- короткие (10 см) проволочные перемычки со штекерами на обоих концах (PRT-13870)*;
- могут пригодиться: проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*;
- может пригодиться: держатель для четырех батареек типа AA (PRT-09835)* (на рис. 5.2 не показан).

Рис. 5.2. Электронные компоненты для проекта ночника

Примечание

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты и материалы (рис. 5.3):

- макетный нож;
- металлическая линейка;
- клей (клеевой пистолет или клей для моделирования);
- лист открыточного картона (не гофрированного) размером примерно 22×28 см;
- лист кальки или обычной копировальной бумаги размером примерно 22×28 см;
- шаблон корпуса (см. рис. 5.20 далее в этом проекте).

Рис. 5.3. Инструменты и материалы, рекомендуемые для проекта ночника

Два новых компонента

В этом проекте мы задействуем два новых для нас электронных устройства: *трехцветный светодиод* и *фоторезистор*. Давайте познакомимся с этими устройствами поближе.

Трехцветный (RGB) светодиод

Из предыдущих проектов этой книги нам уже знакомы обычные светодиоды. Разноцветные, или RGB[1], светодиоды (рис. 5.4) работают подобным образом. В действительности они представляют собой блок из трех светодиодов разных цветов (красного, зеленого и синего), смонтированных в одном корпусе. Каждый из этих светодиодов имеет отдельный положительный вывод — анод, но все они используют один общий отрицательный вывод — катод, который так и называется — *общий катод*.

[1] От англ. Red, Green, Blue — красный, зеленый, синий.

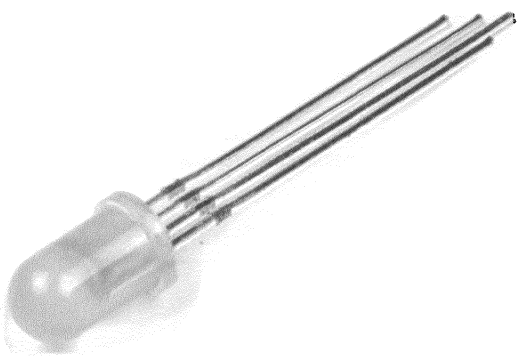


Рис. 5.4. Светодиод RGB с общим катодом

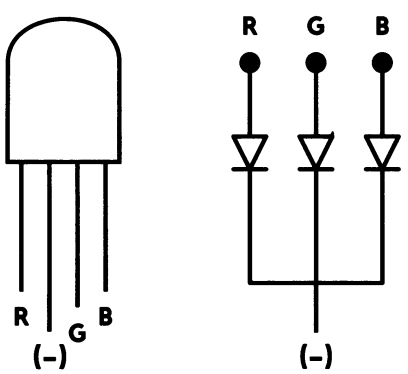


Рис. 5.5. Принципиальная схема трехцветного светодиода RGB

Если внимательно посмотреть на светодиод, показанный на рис. 5.4, то можно увидеть, что все его три вывода имеют разную длину. В отличие от обычных одноцветных светодиодов, имеющих короткий отрицательный вывод, в светодиодах RGB отрицательным является самый длинный вывод.

Принципиальная схема трехцветного светодиода показана на рис. 5.5. Как можно видеть, это действительно три отдельных светодиода, катоды (отрицательные выводы) которых соединены вместе.

Чтобы определить, какой анод принадлежит светодиоду какого цвета, расположите трехцветный светодиод, как показано на рис. 5.5, *слева*. В таком положении самый левый анод принадлежит красному светодиоду. Следующий вывод является общим катодом, а следующие два вывода являются анодами зеленого и синего светодиодов соответственно.

Исходя из сказанного, цветные светодиоды, составляющие трехцветный светодиод, можно подключать в схему, как отдельные светодиоды, — просто подключите положительные выводы, которые вы хотите использовать, к источнику питания или к выводу платы Arduino через последовательный токоограничивающий резистор, а общий катод подключите к отрицательному источнику питания («земле»).

Трехцветные светодиоды предоставляют больше творческих возможностей, поскольку позволяют создавать множество других цветов. Красный, зеленый и синий цвета являются первичными цветами *аддитивной* (или RGB) системы цветовоспроизведения, и с помощью трехцветного светодиода можно смешивать эти три цвета, чтобы получать из них другие цвета. Аддитивный цветовой диск на рис. 5.6 показывает, как посредством сочетания первичных цветов можно создавать все цвета радуги.

Примечание

В другой системе цветовоспроизведения — субтрактивной (или CMYK) — используются три других первичных цвета: голубой (Cyan), пурпурный (Magenta) и желтый (Yellow), из которых также можно создавать все остальные.

Так, если одновременно включить синий и красный светодиоды, мы получим свет пурпурного цвета. А одновременное включение красного и зеленого светодиода даст свет желтого цвета. Если же включить все три составляющие светодиода, получится белый свет. На этом принципе — смешения первичных цветов — основана работа светодиодных телевизоров и мониторов: каждый пиксел экрана, по сути, является светодиодом RGB.

Рис. 5.6. Аддитивный цветовой диск

Фоторезистор

Создаваемый в этом проекте ночник включается, когда в комнате становится темно, и выключается, когда в ней светлеет. Это означает, что ему необходимо как-то определять, когда в комнате темно, а когда светло. Для решения этой задачи служит светочувствительный датчик, который и определяет уровень освещенности в зоне ночника. На рынке предлагаются разные типы светочувствительных датчиков, но мы используем простой фоторезистор, пример которого показан на рис. 5.7.

Рис. 5.7. Фоторезистор

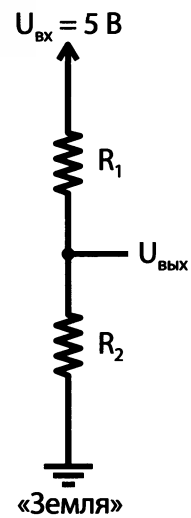


Рис. 5.8. Принципиальная схема делителя напряжения

Это устройство также может называться *фотодатчиком* или *фотоэлементом*. Кроме того, подобно многим другим типам датчиков, фоторезистор иногда называют *датчиком с переменным сопротивлением*.

В зависимости от интенсивности воздействующего на него освещения, сопротивление используемого в этом проекте фоторезистора варьируется от 80 Ом до около 1 МОм (1 000 000 Ом). Фоторезистор имеет низкое сопротивление при освещении интенсивным светом и высокое — в отсутствие освещения.

Чтобы фоторезистор можно было использовать для измерения уровня освещенности, его необходимо включить в схему, называющуюся *делителем напряжения* (рис. 5.8). Делитель напряжения представляет собой два соединенных последовательно резистора, у которых свободный вывод одного подключен к положительному источнику питания (+5 В), а другого — к «земле». В результате между точкой соединения резисторов и «землей» будет присутствовать напряжение меньшее, чем напряжение источника питания.

Общее напряжение на противоположных выводах резисторов равно 5 В, а напряжения на резисторах R_1 и R_2 зависят от соотношения сопротивления этих резисторов. Величина напряжения $U_{вых}$ будет при этом находиться в диапазоне между 5 В и 0 В, поскольку общее напряжение разделено между этими двумя резисторами. Взаимоотношение между напряжением $U_{вых}$ и сопротивлениями резисторов R_1 и R_2 можно выразить следующим уравнением:

$$U_{вых} = \frac{R_2}{R_1 + R_2} \times U_{вх} \qquad (5.1)$$

Не спешите возмущаться, что вам снова предъявляют математическую формулу, — математика в электронике играет важную роль. Но это не означает, что она должна быть сложной. Мы все рассмотрим в деталях и не спеша, чтобы всем все стало понятно. Это простое уравнение особенно полезно при работе с фоторезистором или резистивным датчиком любого другого типа. Так, если в схеме делителя напряжения заменить резистор R_1 на фоторезистор, мы получим схему, показанную на рис. 5.9. Сопротивление фоторезистора здесь повышается по мере падения интенсивности его освещения.

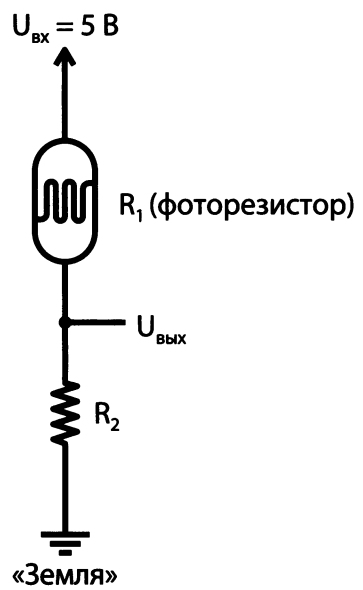


Рис. 5.9. Схема делителя напряжения с фоторезистором

Теперь снова обратимся к уравнению делителя напряжения (5.1). С повышением сопротивления резистора R_1 знаменатель дроби увеличивается, в результате чего значение дроби уменьшается. Это означает, что по мере уменьшения интенсивности освещения выходное напряжение $U_{вых}$ уменьшается.

Таким образом, с помощью этой схемы можно точно измерять интенсивность освещения фоторезистора, подавая напряжение $U_{вых}$ на один из аналоговых выводов платы Arduino (выводы, обозначенные буквой A). Аналоговые сигналы представляют собой непрерывные сигналы, которые могут принимать любое значение в определенном диапазоне значений. До настоящего времени мы использовали в наших проектах только цифровые выводы платы Arduino. В отличие от кнопки, выходной сигнал которой имеет только два состояния, выходной сигнал фоторезистора может иметь диапазон значений в зависимости от интенсивности освещения и характеристик схемы делителя напряжения. В этом заключается разница между цифровыми и аналоговыми сигналами.

Это все, что вам в действительности нужно знать, чтобы работать со схемой делителя напряжения. Но если вы хотите иметь более подробную информацию, чтобы производить вычисления, вы найдете ее во *врезке «Математика для делителей напряжения»* далее в этом проекте.

МАТЕМАТИКА ДЛЯ ДЕЛИТЕЛЕЙ НАПРЯЖЕНИЯ

Сопротивление фоторезистора при разных уровнях освещенности можно измерить с помощью мультиметра (инструкции по использованию мультиметра приведены в *разд. «Электрические измерения с помощью мультиметра» приложения*). Авторы провели эксперимент по измерению сопротивления фоторезистора, в процессе которого фоторезистор то освещался фонариком, то закрывался рукой. При освещении фоторезистора фонариком его измеренное сопротивление составляло около 100 Ом. А при затенении фоторезистора рукой его сопротивление увеличивалось до приблизительно 200 кОм. При сопротивлении постоянного резистора R_2 равным 10 кОм можно ожидать следующие значения выходного напряжения делителя напряжения для этих двух случаев:

$$U_{\text{вых}} = \frac{R_2}{R_1 + R_2} \times U_{\text{вх}}$$

$$U_{\text{вых (освещ.)}} = \frac{10000}{100 + 10000} \times 5\ \text{В} = 0{,}990 \times 5\ \text{В} = 4{,}95\ \text{В}$$

$$U_{\text{вых (затен.)}} = \frac{10000}{200000 + 10000} \times 5\ \text{В} = 0{,}048 \times 5\ \text{В} = 0{,}24\ \text{В}$$

Как можно видеть, в диапазоне освещенности фоторезистора от полностью затененного до ярко освещенного напряжение на фоторезисторе варьируется в диапазоне от 0,24 до 4,95 В. Круто, не так ли? Все-таки математика довольно-таки полезная вещь! Далее в этом проекте мы рассмотрим, как считывать эти напряжения с помощью Arduino.

Создаем прототип ночника

Теперь давайте объединим в одной схеме разноцветный светодиод и делитель напряжения, чтобы получить схему подключения ночника к плате Arduino (рис. 5.10). При этом сначала создадим схему делителя напряжения с фоторезистором, а затем добавим в нее RGB-светодиод. Готовая схема должна выглядеть наподобие показанной на рис. 5.11.

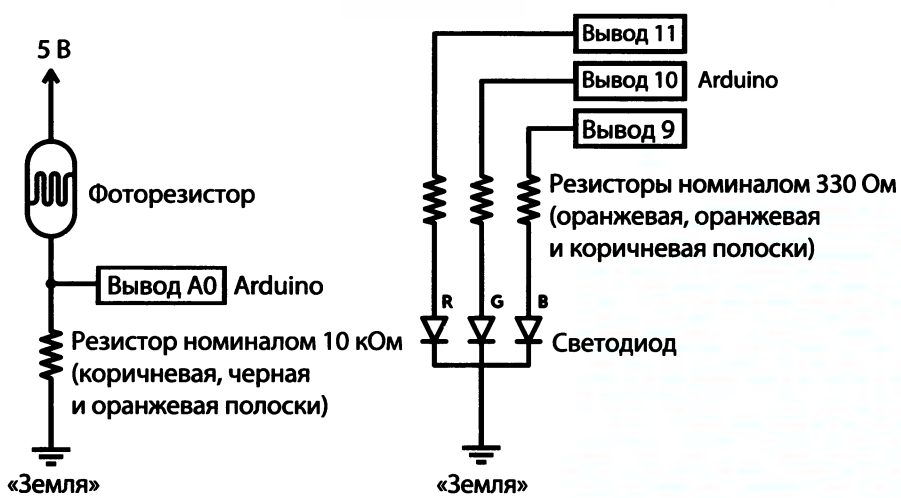


Рис. 5.10. Принципиальная схема прототипа ночника

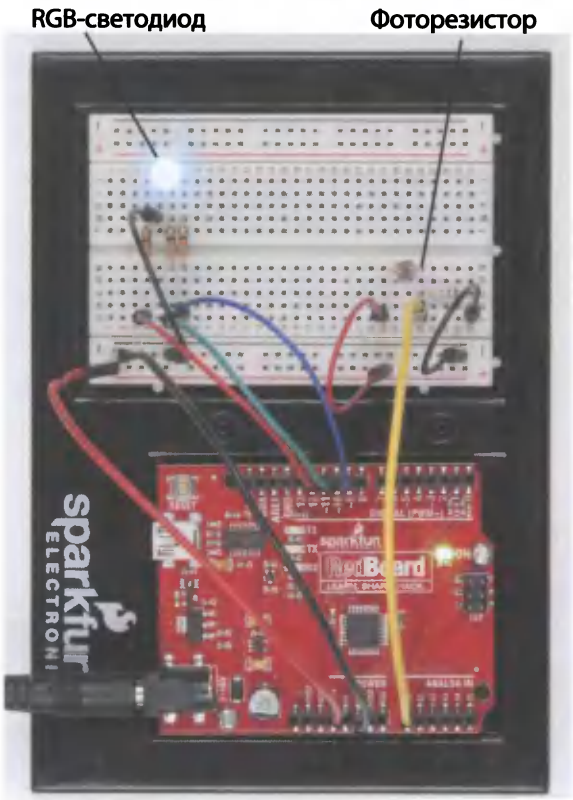


Рис. 5.11. Монтаж схемы прототипа ночника

Рис. 5.12. Резистор с сопротивлением 10 кОм (коричневая, черная и оранжевая полоски)

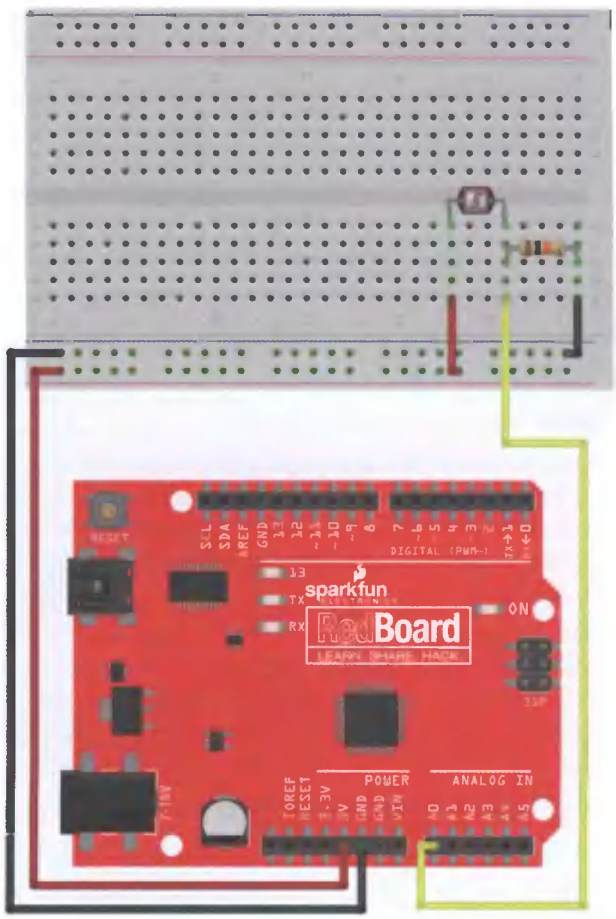


Рис. 5.13. Монтажная схема делителя напряжения с использованием фоторезистора

Собираем схему делителя напряжения

Приготовьте фоторезистор (он должен выглядеть примерно так, как показано на рис. 5.7) и резистор с сопротивлением 10 кОм. Вспомните, что резистор сопротивлением 10 кОм обозначается коричневой, черной и оранжевой полосками (рис. 5.12). Более подробную информацию о том, как расшифровывать номинальное значение резистора по его разноцветным полоскам, вы найдете в *разд. «Полосатые резисторы» приложения*.

Соберите из подготовленных деталей схему делителя напряжения, показанную на рис. 5.13. При подготовке макетной платы для сборки схем рекомендуется в первую очередь подключать к ней питание, поэтому первым делом так и сделайте. Для этого проекта мы воспользуемся шинами питания с левой стороны макетной платы. Так что соедините перемычкой гнездо 5 V на плате Arduino с положительной шиной питания макетной платы, а гнездо GND — с отрицательной.

Затем вставьте фоторезистор в нижнюю часть макетной платы, причем обратите внимание, что выводы фоторезистора вставляются в разные ряды платы. Вставьте один вывод резистора сопротивлением 10 кОм в гнездо в том же ряду макетной платы, что и один из выводов фоторезистора (соединив таким образом эти компоненты электрически). Другой вывод резистора вставляется в гнездо в ряду макетной платы, к которому пока больше ничего не подключено. Проволочной

перемычкой подключите ряд, в котором находится свободный вывод фоторезистора (то есть вывод, не соединенный с резистором), к положительной шине питания макетной платы, после чего проволочной перемычкой подключите ряд со свободным выводом резистора сопротивлением 10 кОм к отрицательной шине питания макетной платы.

Наконец, подключите фоторезистор к Arduino, соединив проволочной перемычкой гнездо аналогового ввода A0 на плате Arduino с рядом на макетной плате, в гнезда которого вставлены один вывод резистора сопротивлением 10 кОм и один вывод фоторезистора. Этот проводник часто называется *проводником выходного напряжения* фоторезистора, или *сигнальным проводником*. В принципе, любой из аналоговых выводов платы Arduino — с A0 по A5 — можно использовать для измерения какого-либо диапазона напряжений.

Обратите внимание, что монтаж на макетной плате соответствует принципиальной схеме, показанной на рис. 5.9. Это одна из базовых схем с датчиком, используемых в проектах на Arduino. Переменными резисторами являются также многие другие аналоговые датчики, такие как датчики изгиба, температуры, давления и т. п. Чтобы экспериментировать с каким-либо из этих датчиков, просто вставьте его вместо фоторезистора.

Подключаем трехцветный светодиод

Помните схему светофора из *проекта 2*? В ней использовались три светодиода. Трехцветный светодиод, по сути, представляет собой те же три светодиода, но в одном корпусе. У него три вывода, самый длинный из которых является общим катодом (отрицательным выводом). Расположите свой трехцветный светодиод, как показано на рис. 5.5, и, следуя инструкциям к этому рисунку, определите вывод для красного светодиода.

Беспаечная макетная плата разделена на две группы рядов продольным углублением. Вставьте трехцветный светодиод в в правую часть макетной платы таким образом, чтобы красный вывод оказался в верхнем (четвертом) ряду, а самый длинный вывод (общий катод) — в следующем нижнем ряду (рис. 5.14).

Затем приготовьте три резистора сопротивлением 330 Ом (они обозначены двумя оранжевыми и одной коричневой полосками) и соедините посредством этих резисторов ряды правой стороны, в которые вставлены красный, зеленый и синий выводы светодиода, с соответствующими рядами на противоположной стороне углубления, как показано на рис. 5.14. Таким образом, выводы резисторов не будут закорочены, поскольку ряды противоположных сторон макетной платы не соединены друг с другом. С помощью проволочной перемычки подключите общий катод

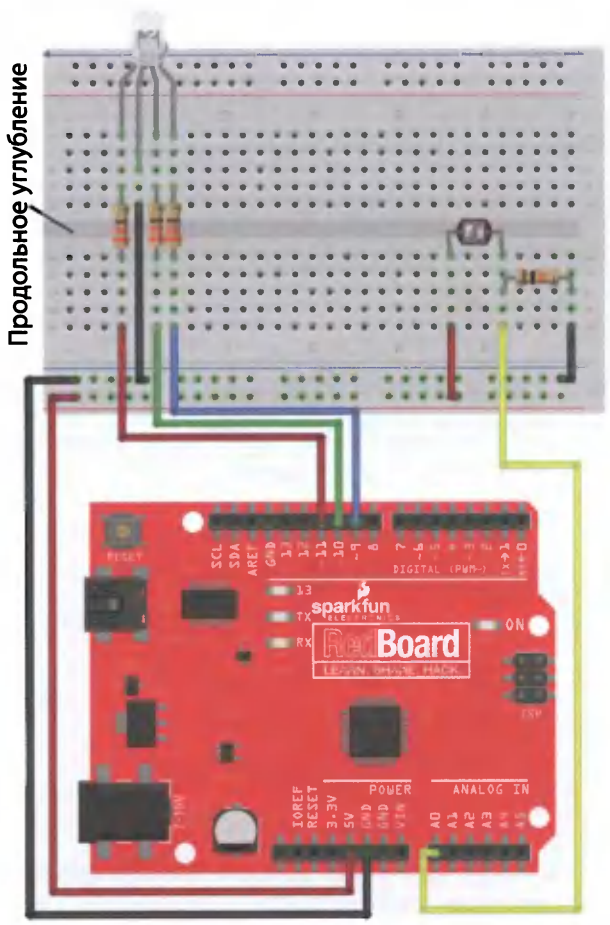


Рис. 5.14. Подключение трехцветного светодиода к схеме делителя напряжения

(отрицательный вывод) светодиода к шине отрицательного питания («земле») на левой стороне макетной платы. Наконец, с помощью проволочных перемычек подключите вывод 11 платы Arduino к резистору красного вывода светодиода, вывод 10 платы — к резистору зеленого вывода светодиода, а вывод 9 платы — к резистору синего вывода светодиода. Готовый монтаж должен выглядеть так, как показано на рис. 5.14. Обратите внимание, что красный, зеленый и синий монтажные провода соответствуют здесь красному, зеленому и синему выводам светодиода.

В подключенном таким образом трехцветном светодиоде можно управлять — подобно управлению отдельным светодиодом — каждым его отдельным цветом, используя выводы платы Arduino 9, 10 и 11. Теперь все готово, чтобы запустить среду разработки Arduino и экспериментировать с этой возможностью.

Тестируем ночник с простым смешением цветов

В этом проекте нам нужно разобраться с несколькими новыми подходами, начиная с понятия о том, как смешивать цвета в трехцветном светодиоде. Поскольку трехцветный светодиод, по сути, является тремя светодиодами в одной упаковке, в коде нам нужно обращаться с ним, как с тремя отдельными светодиодами. Создайте новый скетч в среде разработки Arduino и вставьте в него функции setup() и loop() из листинга 5.1.

Листинг 5.1. Простой пример кода для излучения трехцветным светодиодом голубого цвета

```
void setup()
{
❶  pinMode(11, OUTPUT);  //красный
   pinMode(10, OUTPUT);  //зеленый
   pinMode(9, OUTPUT);   //синий
}

void loop()
{
❷  digitalWrite(11, LOW);   //красный
❸  digitalWrite(10, HIGH);  //зеленый
❹  digitalWrite(9, HIGH);   //синий
}
```

Здесь мы сначала конфигурируем выводы Arduino, управляющие трехцветным светодиодом, для функционирования в режиме вывода данных (OUTPUT) ❶, используя для этого три вызова функции pinMode() и передавая каждой из них соответствующий номер вывода в качестве параметра. В предыдущих проектах мы управляли отдельными светодиодами с помощью функции digitalWrite() — эта же функция используется и для управления отдельными цветами трехцветного светодиода. Выберем один из составных цветов на цветовом диске (см. рис. 5.6). Пусть это будет голубой цвет, получаемый смешением зеленого и синего первичных цветов. Таким образом, с помощью функции digitalWrite() нам нужно включить зеленый ❸ и синий ❹ светодиоды. При этом красный светодиод должен быть выключен, что мы и делаем с помощью соответствующей функции digitalWrite() ❷.

Загрузите этот скетч в Arduino, и если вы не допустили никаких ошибок при монтаже схемы и в коде, трехцветный светодиод должен светиться мягким голубым цветом. Если светодиод светится другим цветом или не светится вообще, проверьте правильность и надежность монтажа, и, в частности, ориентацию выводов трехцветного светодиода.

Обратите внимание, что хотя смешиваем мы цвета в трехцветном светодиоде, код для этого весьма похож на код для управления светодиодами в других проектах: каждый цвет управляется отдельным выводом Arduino, выводы конфигурируются для работы в режиме вывода данных вызовами функции pinMode(), а управление цветами осуществляется так же, как и управление несколькими обычными светодиодами, — посредством вызовов функции digitalWrite().

ПРАКТИКУМ: СМЕШИВАЕМ ДРУГИЕ ЦВЕТА

Попробуйте самостоятельно изменить излучаемый светодиодом цвет, используя первичный красный цвет. Пусть, например, светодиод засветится пурпурным или желтым цветом. А какой цвет получится, если включить все три первичных цвета? Для помощи используйте цветовой диск, показанный на рис. 5.6.

Программируем ночник

В схеме ночника в качестве датчика освещенности используется фоторезистор. Выход делителя напряжения на фоторезисторе подключается в аналоговому выводу A0 платы Arduino, для которого задана конфигурация работы в режиме ввода данных (INPUT). Вспомните, что любой из аналоговых выводов платы Arduino — с A0 по A5 — можно использовать для измерения какого-либо диапазона напряжений. Определить значение подаваемого на этот вывод напряжения можно с помощью функции analogRead(). Эта функция считывает поступаемое на аналоговый вывод платы напряжение и возвращает значение в диапазоне от 0 до 1023, представляющее напряжение в диапазоне от 0 до 5 В. Например, если на вывод A0 подать напряжение величиной 2,5 В, функция analogRead() возвратит значение 512, что практически равно половине 1023.

Сопротивление фоторезистора изменяется соответственно изменению его освещенности, в результате чего также изменяется выходное напряжение делителя напряжения, которое подается на аналоговый вход платы. Давайте посмотрим, как все это выразить в коде для Arduino. Чтобы определить, включить или выключить ночник, нам нужно считать напряжение на фоторезисторе и сравнить его со значением, указывающим степень освещенности в комнате. У нас уже есть готовая схема для этого с трехцветным светодиодом и фоторезистором, подключенным к аналоговому выводу A0 платы Arduino. А в листинге 5.2 приводится весь необходимый код для выполнения этой задачи. Можете или вставить этот код в скетч из листинга 5.1, заменив старый код, или же вставить его в новый скетч.

Листинг 5.2. Полный код для скетча ночника

```
int calibrationValue;
int lightValue;

void setup()
{
   pinMode(9, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(n, OUTPUT);
❶  calibrationValue = analogRead(A0);
}

void loop()
{
❷  lightValue = analogRead(A0);
   if(lightValue < calibrationValue - 50)
{
   digitalWrite(11, LOW);  //красный
   digitalWrite(10, HIGH); //зеленый
   digitalWrite(9, HIGH);  //синий
}
❸ else
{
   digitalWrite(11, LOW); //красный
   digitalWrite(10, LOW); //зеленый
   digitalWrite(9, LOW);  //синий
}
}
```

Загрузите этот код в Arduino и создайте в комнате яркое освещение. Светодиод не должен гореть. Теперь затените фоторезистор рукой — светодиод должен загореться голубым цветом. Если светодиод не загорается, попробуйте обеспечить лучшее затенение, вплоть до полного закрытия светодиода ладонью или книгой, чтобы на него совсем не попадало света. Теперь, если убрать затенение и осветить фоторезистор, трехцветный светодиод снова должен выключиться. Довольно занимательно, не так ли? Давайте рассмотрим, каким образом скетч делает это.

Подготовка к проверке уровня освещенности

Сначала в скетче создаются глобальные переменные calibrationValue и lightValue, но значения им не присваиваются. Далее, подобно коду из листинга 5.1, в функции setup() вызывается функция pinMode() для настройки выводов 9, 10 и 11 Arduino на работу в режиме вывода данных (OUTPUT). Затем переменной calibrationValue присваивается калибровочное значение, полученное функцией analogRead() ❶ в результате считывания напряжения фоторезистора. С этим значением скетч будет сравнивать последующие уровни освещенности, чтобы решить, включать ли светодиод или нет.

Затем управление исполнением скетча переходит в функцию loop(). Код в этой функции считывает текущий уровень освещенности и сохраняет его в переменной lightValue ❷. Значение переменной lightValue обновляется при каждом последующем исполнении цикла loop().

Рис. 5.15. Структурная схема оператора if...else

Управляем ночником в зависимости от уровня освещенности

Arduino сравнивает калибровочное значение освещенности, сохраненное в переменной calibrationValue, с текущим значением освещенности в переменной lightValue, чтобы решить, включить или выключить ночник, то есть зажечь или погасить два цвета трехцветного светодиода. Для этого используется условный оператор if...else, позволяющий Arduino принять решение на основе истинности логического выражения, которое может иметь только два значения: true (истина) или false (ложь). На рис. 5.15 показана схема алгоритма принятия решения в операторе if...else.

Оператор if...else проверяет истинность условия lightValue < calibrationValue – 50. Символ < означает «меньше чем», поэтому условие читается как «меньше ли значение lightValue, чем значение calibrationValue минус 50?» Если условие удовлетворяется, то есть результат оценки выражения есть значение ИСТИНА, выполняется код в фигурных скобках сразу же после слова if.

По мере уменьшения освещенности снижается и уровень напряжения на фоторезисторе. Наше условное выражение проверяет, не значительно ли меньше значение lightValue, чем значение calibrationValue, и будет выполняться, пока уровень освещения датчика значительно не понизится. Когда условие выполняется, скетч устанавливает высокий (HIGH) уровень на выводах 9 и 10 и низкий (LOW) на выводе 11, в результате чего трехцветный светодиод начинает светиться голубым светом.

Если же условие не выполняется, управление исполнением скетча передается коду в блоке else ❸, который устанавливает низкий уровень на всех трех выводах платы Arduino, в результате чего трехцветный светодиод выключается.

Предотвращение ложных срабатываний

Если нам нужно всего лишь проверить, изменился ли уровень освещенности, зачем тогда отнимать 50 от значения calibrationValue в условном выражении? Это делается для того, чтобы понизить порог срабатывания скетча. Если использовать просто условие lightValue < calibrationValue, светодиод будет включаться и выключаться при малейших изменениях освещенности (о символе < упоминается также во *врезке «Логические операторы сравнения»* в *проекте 4*). Вычитая 50 из калибровочного значения, мы устанавливаем пороговое значение для включения светодиода на 50 единиц меньше калибровочного (начального) значения уровня освещенности.

Рекалибровка ночника

Последнее, что мы рассмотрим перед тем, как приступать к декорированию ночника, — это переустановка калибровочного значения для разных уровней освещенности. Значение переменной calibrationValue устанавливается в функции setup(), которая исполняется всего лишь один раз, после чего исполняется цикл loop(). Перезапустить исполняющийся скетч можно двумя способами. Первый состоит в выключении и последующем включении питания платы, но это довольно-таки неудобно и грубо. Второй способ более элегантен. Как и в *проекте 4*, скетч можно перезапустить, просто нажав кнопку сброса платы Arduino (рис. 5.16). Эта кнопка работает так же, как и кнопка сброса компьютера или игровой консоли. При каждом нажатии этой кнопки скетч начинает исполняться сначала, переустанавливая при этом значение calibrationValue.

Таким образом, датчик можно перекалибровать для новых условий освещенности, просто нажав кнопку сброса. При этом необходимо следить за тем, чтобы датчик фоторезистора измерял фактическую освещенность окружения, то есть чтобы он не был случайно каким-либо образом затенен.

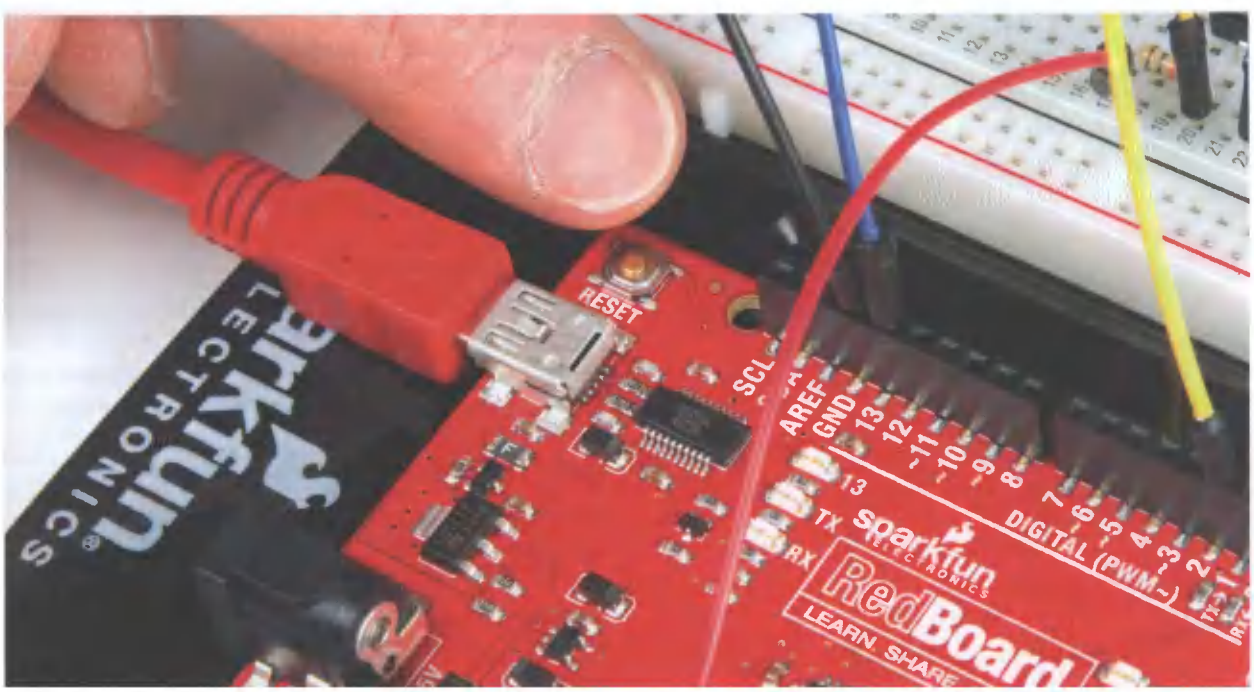


Рис. 5.16. Выполнение сброса платы Arduino

Создаем другие цвета

Кроме упомянутых ранее цветов, на трехцветном светодиоде можно создавать и другие цвета. Для этого вместо смешения простых первичных цветов нужно смешивать их градации. Так можно получить лазурный, оранжевый, светло-розовый или любой другой из тысяч возможных цветов.

Но, как уже говорилось, для этого недостаточно просто включить ту или иную комбинацию первичных цветов, а нужно включать эти цвета с разной интенсивностью.

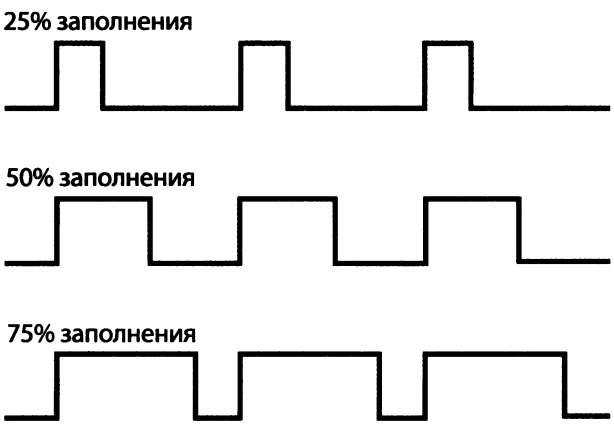


Рис. 5.17. Цифровые сигналы с разными коэффициентами заполнения.

Рис. 5.18. Выводы ШИМ платы RedBoard

Создание аналоговых сигналов посредством ШИМ

Чтобы регулировать интенсивность включенного первичного цвета, нужно использовать аналоговый управляющий сигнал вместо цифрового. Разница между аналоговыми и цифровыми сигналами рассматривалась в *главе «Основы электроники»* в начале книги. Цифровой сигнал, подобно обычному выключателю, может иметь только два состояния: включенное или выключенное. А аналоговый сигнал похож на реостат плавного регулирования освещения и может иметь бесконечное множество состояний в границах определенного диапазона.

Но Arduino является цифровым устройством, а это означает, что его сигналы могут быть только включенными или выключенными. Однако аналоговый сигнал в диапазоне всех его промежуточных состояний можно эмулировать с помощью цифрового сигнала, используя метод, называемый *широтно-импульсной модуляцией* (ШИМ). Метод заключается в попеременном включении и выключении цифрового сигнала с высокой частотой с одновременным варьированием продолжительности высокого (HIGH) уровня сигнала относительно низкого (LOW) уровня, в результате чего выходной сигнал кажется аналоговым. Чем больше продолжительность высокого уровня, тем больше аналоговое значение сигнала. Этот метод называется варьированием *коэффициента заполнения*, или *скважности*. В этой книге мы будем использовать термин *коэффициент заполнения* или просто *заполнение*, как более описательный. На рис. 5.17 показаны три цифровых сигнала с разным заполнением (то есть с разной длительностью высокого состояния) — аналоговое значение сигнала с заполнением 75% выше, чем сигнала с заполнением 25%.

Однако не все выводы Arduino могут работать с сигналами ШИМ. На стандартной плате Arduino это могут делать только определенные выводы общего назначения, а именно выводы 3, 5, 6, 9, и 11. На плате эти выводы помечены символом тильды (~), как показано в выделенном фрагменте платы на рис. 5.18.

Когда нам нужно управлять каким-либо устройством, требующим аналогового управляющего сигнала, — например, яркостью светодиода, скоростью двигателя, громкостью зуммера и т. п., мы может использовать один из выводов ШИМ платы Arduino, чтобы эмулировать аналоговый сигнал.

В этом проекте мы будем использовать выводы ШИМ для управления яркостью первичных цветов трехцветного светодиода, чтобы создавать составные цвета. Поскольку мы уже используем здесь для управления трехцветным светодиодом выводы 9, 10 и 11, нам в схеме ничего изменять не придется, а надо будет только внести необходимые изменения в код.

Смешение цветов посредством функции *analogWrite()*

Чтобы воспользоваться функциональностью ШИМ, нам нужно использовать функцию analogWrite(), которая подает значение ШИМ на указанный вывод. Функция принимает два параметра: номер вывода, которым требуется управлять, и значение ШИМ для подачи на этот вывод. Значение ШИМ может быть любым в диапазоне от 0 до 255. Значение 0 соответствует постоянно отсутствующему сигналу (низкий уровень) на выводе, а 255 — постоянно присутствующему (высокий уровень). В листинге 5.3 приводится простой скетч для демонстрации работы функции analogWrite().

Листинг 5.3. Скетч для демонстрации работы функции analogWrite()

```
void setup()
{
❶  pinMode(9, OUTPUT);
}

void loop()
{
❷  analogWrite(9, 2);
}
```

Сначала, как и для любого другого вывода, нам нужно сконфигурировать режим работы требуемого вывода с помощью функции pinMode(). Ей передаются в качестве параметров номер конфигурируемого вывода и значение режима работы. Поскольку светодиод является устройством вывода, то это значение будет OUTPUT ❶, в результате чего для указанного вывода платы устанавливается режим вывода данных. Значение ШИМ для вывода устанавливается с помощью функции analogWrite(), которой в качестве параметров передаются номер управляемого вывода и значение ШИМ в диапазоне от 0 до 255. В данном случае для вывода 9 (управляющего синим компонентом трехцветного светодиода) устанавливается значение 2 ❷. Загрузите этот скетч в Arduino — трехцветный светодиод должен засветиться тусклым синим цветом. Значение ШИМ 2 включает этот светодиод в течение 0,7% (или $^{2}/_{255}$) времени. Прежде чем двигаться дальше, попробуйте изменять значение ШИМ несколько раз и исполнять скетч с этими значениями, чтобы получить представление о разных значениях и соответствующих интенсивностях свечения светодиода.

Когда вы почувствуете, что уверенно управляетесь с использованием функции analogWrite() при разных значениях ШИМ, попробуйте использовать ее для смешения цветов. В листинге 5.4 приводится скетч для мигания трехцветным светодиодом разными цветами, более интересными, чем использованные ранее.

Листинг 5.4. Многоцветное мигание

```
void setup()
{
❶  pinMode(11, OUTPUT);  //красный
   pinMode(10, OUTPUT);  //зеленый
   pinMode(9, OUTPUT);   //синий
}

void loop()
{
```

```
❷   analogWrite(11, 153); //темно-фиолетовый
    analogWrite(10, 50);
    analogWrite(9, 204);
    delay(1000);
❸   analogWrite(11, 155); //бледно-голубой
    analogWrite(10, 196);
    analogWrite(9, 226);
    delay(1000);
❹   analogWrite(11, 255); //ярко-желтый
    analogWrite(10, 246);
    analogWrite(9, 0);
    delay(1000);
}
```

Сначала для управления еще двумя цветами конфигурируем два дополнительных вывода с помощью функции pinMode() ❶. Затем в цикле заставляем светодиод светиться тремя разными цветами, устанавливая для этого разные значения на выводах Arduino, управляющих первичными цветами. Так мы задаем интенсивность каждого первичного цвета, что определяет его долю в составном цвете.

Первый набор первичных цветов создает составной темно-фиолетовый цвет ❷, второй — бледно-голубой ❸, а третий — ярко-желтый ❹. Попробуйте экспериментировать с другими значениям красного, зеленого и синего (RGB) цветов, чтобы создать еще какие-либо цвета.

Определение значений цветов RGB с помощью цветоподборщика

Задавая значения первичных цветов наобум, трудно предсказать, каким получится составной цвет. Эту проблему можно решить с помощью одного из онлайновых инструментов подбора цветов. Из великого множества таких сайтов мы рекомендуем использовать **https://www.colorpicker.com/**. Вы просто указываете понравившийся вам цвет на палитре, а значения его первичных цветов выводятся в полях справа (рис. 5.19).

Нужные нам поля значений цветов обозначены **R**, **G** и **B**, что соответствует красному (red), зеленому (green) и синему (blue) цветам. Игнорируйте поля, обозначенные H, S и B, — в них выводятся

Рис. 5.19. Инструмент подбора цветов на сайте **https://www.colorpicker.com/**

значения первичных цветов для выбранного цвета, но в другой системе подбора цветов, называющейся HSB[2]. Эта система применяется во многих приложениях, требующих смешения цветов, но она не так полезна, когда у нас есть прямой контроль над тремя первичными цветами.

Ночник с задаваемым цветом

Обладая информацией о получении разных цветов путем смешения первичных, можно с легкостью модифицировать код ночника, чтобы он светился любым задаваемым цветом. Для этого нужно всего лишь заменить вызов функции digitalWrite() вызовом функции analogWrite(). В листинге 5.5 приводится модифицированный таким образом код для свечения светодиода цветом, выбранным на рис. 5.19. Измененный код выделен полужирным шрифтом.

Листинг 5.5. Код для свечения ночника любым заданным цветом

```
int calibrationValue;
int lightValue;

void setup()
{
   pinMode(9, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(11, OUTPUT);
   calibrationValue = analogRead(A0);
}

void loop()
{
   lightValue = analogRead(A0);
   if(lightValue < calibrationValue - 50)
   {
      analogWrite(11, 66);   //красный
      analogWrite(10, 166);  //зеленый
      analogWrite(9, 199);   //синий
   }
   else
   {
      analogWrite(11, 0);    //выключаем красный
      analogWrite(10, 0);    //выключаем зеленый
      analogWrite(9, 0);     //выключаем синий
   }
}
```

На этом создание прототипа схемы ночника завершено. Если вам не нравится выбранный здесь цвет, с помощью инструмента подбора цветов выберите цвет, который вам более по душе, отредактируйте функции analogWrite() полученными значениями, а затем снова загрузите скетч в Arduino. Если вам интересно, на сайте **http://99colors.net/color-names** можно найти некоторые занимательные предложения цветов.

Теперь, когда у нас готов прототип ночника и скетч для его работы, можно проявить свои творческие способности и создать для нашего ночника абажур.

[2] От англ. Hue — оттенок, Saturation — насыщенность, Brightness — яркость.

Создаем абажур для ночника

Для корпуса абажура рекомендуется использовать открыточный, а не гофрированный картон, поскольку с ним легче работать, и сгибы на нем получаются более аккуратными. Отверстия для света можно закрыть каким-либо прозрачным материалом, например калькой. Приступим к работе.

Делаем картонный корпус

В качестве отправной точки можно использовать предлагаемый в архиве ресурсов базовый шаблон корпуса ночника, но авторы настоятельно рекомендуют, оформляя этот корпус, дать волю своему воображению. Более того, если вы уверены

в своих силах, то можете вообще не пользоваться нашим шаблоном, а самостоятельно разработать абажур для ночника с нуля.

Вырезаем детали

Проект содержит два шаблона: один для корпуса абажура, а другой для прозрачных панелей. Панели можно сделать из любого материала, который по толщине похож на бумагу для ксерокса, но авторы рекомендуют использовать какой-либо прозрачный материал наподобие вощеной бумаги или кальки. Шаблоны для корпуса и панелей показаны на рис. 5.20 (они находятся в соответствующей папке архива ресурсов для этой книги). Если у вас есть принтер, можно распечатать их прямо на материале, с которым будем работать.

Обратите внимание, что в основании корпуса ночника имеется квадратный вырез для доступа к проводке. Авторы также обнаружили, что удобно не закрывать одну из сторон панелью, чтобы было легче подключать провода к большой беспаечной макетной плате. Но вы сами решайте, устанавливать ли четвертую панель или нет.

Переведите шаблоны корпуса и панелей на рабочий материал и вырежьте их. Для этого настоятельно рекомендуется использовать острый макетный нож и металлическую линейку (рис. 5.21), чтобы детали имели опрятные и четкие края. Не забывайте правила безопасной работы с макетным ножом: всегда тяните лезвие на себя, а не толкайте его в какую-либо сторону, и делайте рез в несколько проходов.

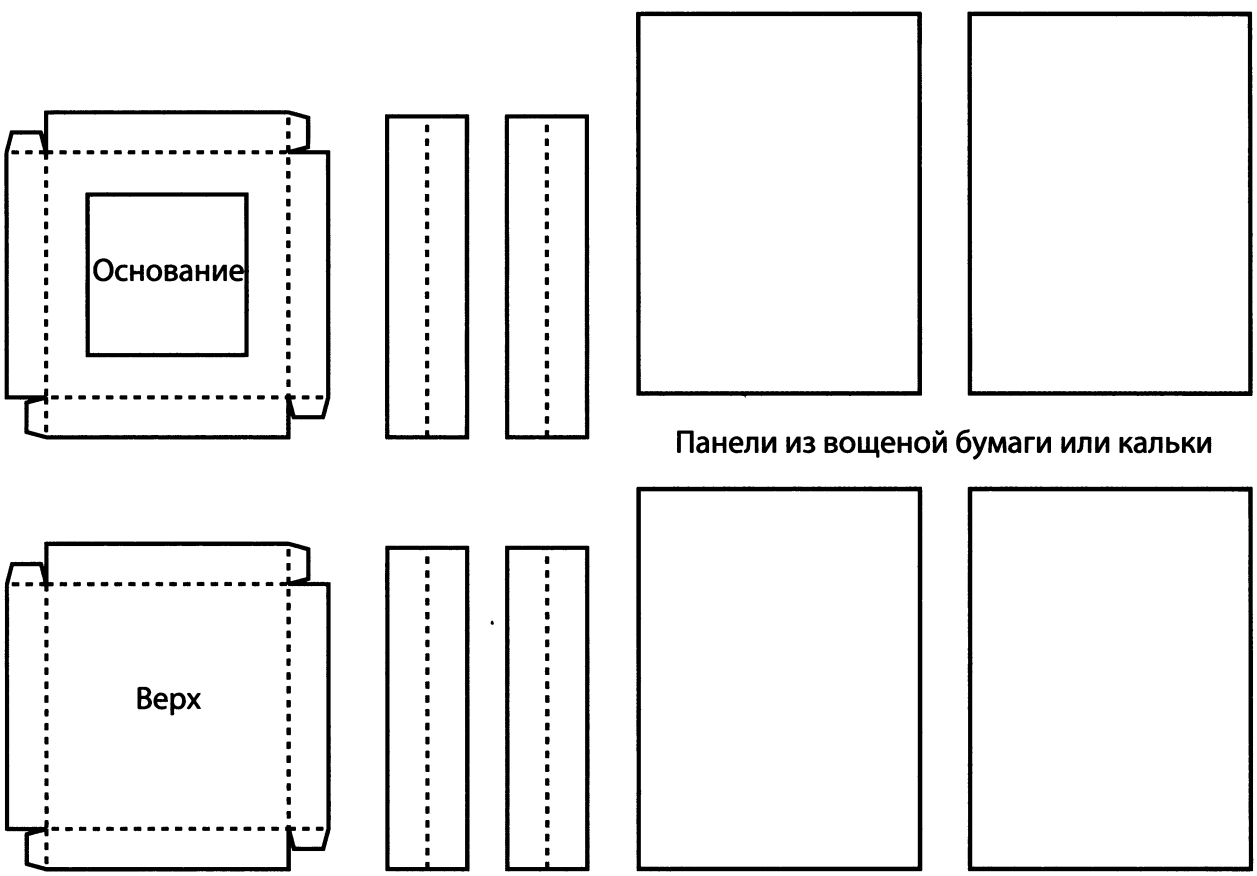


Рис. 5.20. Шаблоны для абажура ночника (в уменьшенном виде)

Собираем абажур

Выложите все вырезанные детали абажура перед собой. У вас должно быть шесть деталей для корпуса и четыре панели, как показано на рис. 5.22.

Возьмите деталь основания (с вырезанным в центре квадратным отверстием) и согните его левый и верхний края по направлению к себе под прямым углом к основанию. Согните язычок левой грани внутрь и приклейте ее к верхней грани каплей клея, как показано на рис. 5.23. Повторите эту процедуру для остальных двух граней основания, а затем таким же способом соберите верх корпуса.

Примечание

В зависимости от толщины используемого картона, может быть, полезно сделать макетным ножом неглубокие надрезы по линиям изгиба. Это будет содействовать получению более аккуратных изгибов.

Рис. 5.21. Вырезание корпуса из открыточного картона

Рис. 5.23. Сборка основания корпуса абажура

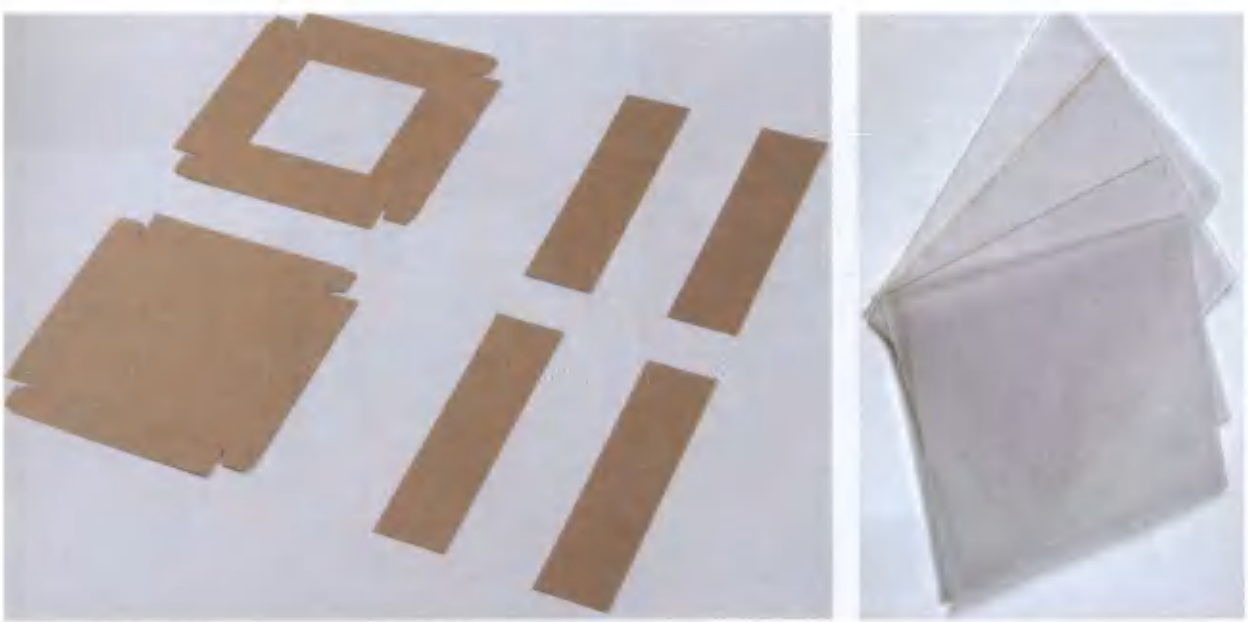


Рис. 5.22. Детали корпуса абажура (*слева*) и панели (*справа*) в ожидании сборки

Затем согните вдоль все четыре боковые стойки под углом 90 градусов. Если вы распечатали шаблон, линии изгиба стоек обозначены пунктирной линией. Собранные основание и корпус и согнутые боковые стойки должны выглядеть так, как показано на рис. 5.24.

Наконец, приклейте все боковые стойки к основанию, как показано на рис. 5.25.

Авторы обнаружили, что световые панели проще приклеивать до того, как будет приклеен верх. Поэтому нанесите немного клея на внутренние стороны боковых стоек и приклейте к ним прозрачные панели, как показано на рис. 5.26. Как уже упоминалось, имеет смысл использовать только три панели, чтобы было более удобно проложить провода к макетной плате.

Панели приклеили? Очередь за верхом. Просто нанесите капельку клея вверху каждой боковой панели и наденьте на них верх, как показано на рис. 5.27.

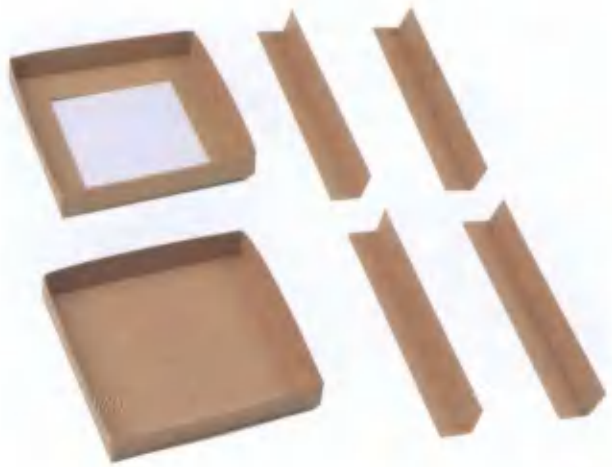


Рис. 5.24. Детали корпуса, подготовленные к конечной сборке

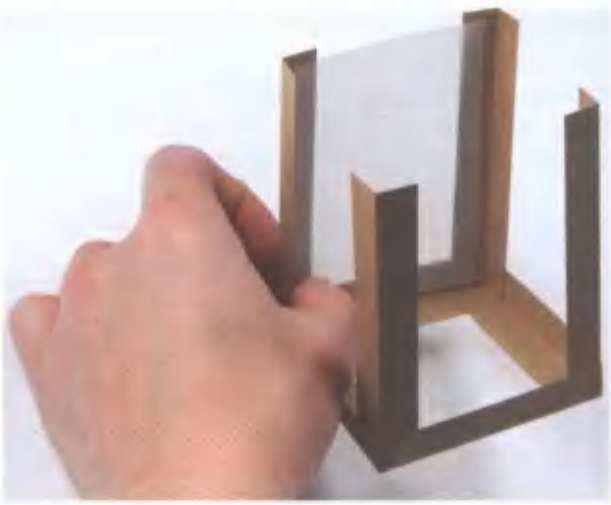


Рис. 5.26. Приклеиваем панели, прежде чем приклеивать верх

Рис. 5.25. Приклеивание боковых стоек к основанию

Рис. 5.27. Устанавливаем верх абажура

Готовый абажур должен выглядеть так, как показано на рис. 5.28.

Рис. 5.28. Готовый абажур для ночника

Вставляем в абажур электронные компоненты

Электронные компоненты можно вставить в абажур двумя способами: вставить под абажур макетную плату и плату Arduino или же вставить только светодиод. Авторы пошли вторым путем.

Сначала отсоедините Arduino от компьютера, а затем перенесите трехцветный светодиод на миниатюрную беспаечную макетную плату. Миниатюрная беспаечная макетная плата устроена точно так же, как и ее большая сестра, с той лишь разницей, что она короче и не имеет шин питания. Вставьте проволочные перемычки в ряды на миниатюрной макетной плате, в которые вставлен трехцветный светодиод (рис. 5.29). Обратите внимание, что каждый из четырех выводов трехцветного светодиода: красный, общий катод (самый длинный), зеленый и синий — вставлен в отдельный ряд на миниатюрной макетной плате.

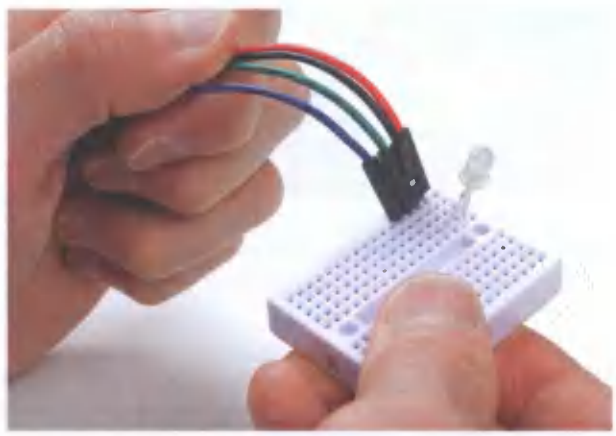


Рис. 5.29. Установка и подключение трехцветного светодиода на миниатюрной макетной плате для помещения под абажур

Затем вставьте другие разъемы каждой проволочной перемычки в ряд большой макетной платы, в который раньше был вставлен соответствующий вывод трехцветного светодиода (рис. 5.30).

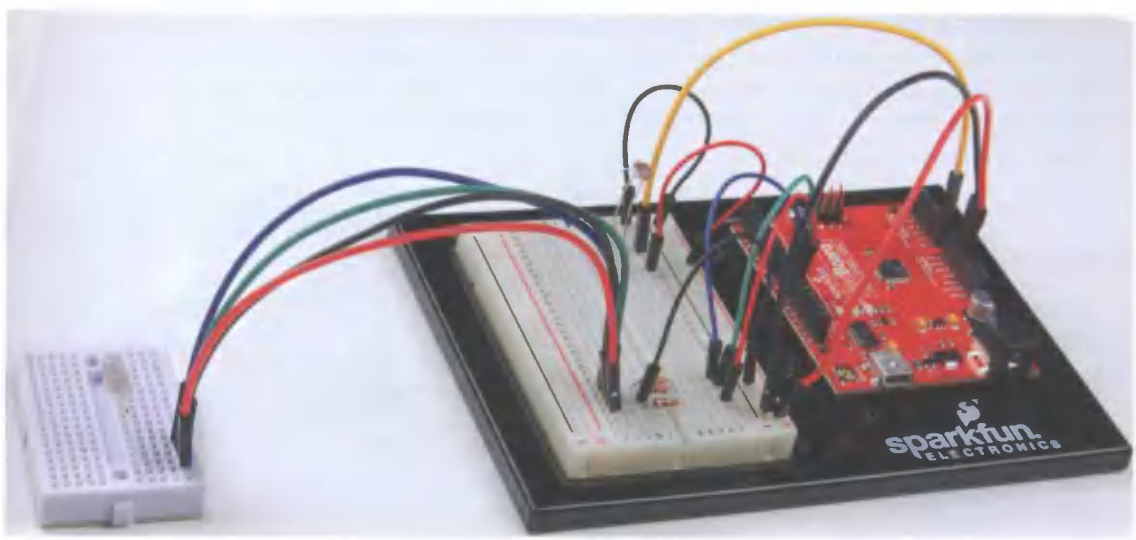


Рис. 5.30. Подключение трехцветного светодиода, установленного на миниатюрной макетной плате, к схеме на большой макетной плате

Если вы оставили одну из сторон абажура открытой, просто вставьте миниатюрную макетную плату в абажур через это отверстие, как показано на рис. 5.31. В противном случае осторожно разместите абажур над миниатюрной макетной платой, чтобы он стоял ровно, или прикрепите клейкой лентой проволочные перемычки к столу, или же сделайте пару вырезов внизу корпуса, чтобы пропустить через них провода.

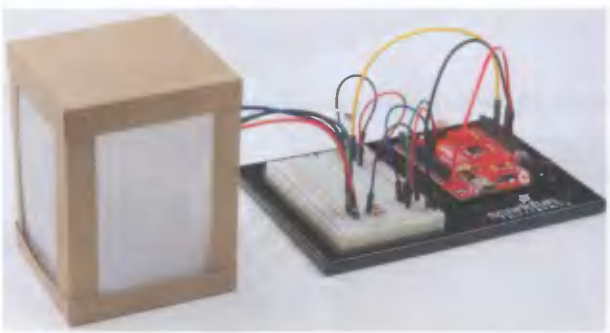


Рис. 5.31. Абажур с установленной в него миниатюрной макетной платой

Да будет свет!

Подключите плату Arduino к внешнему источнику питания на батарейках и выключите в комнате свет — ночник должен засветиться мягким синеватым цветом (рис. 5.32).

Рис. 5.32. Завершенный проект ночника: Бабай в ночи нам больше не страшен!

Идем дальше...

Для этого проекта потребовалось применение многих новых навыков и знаний, но он все еще имеет большой потенциал для дальнейшего развития. И много чем можно его дополнить, как в плане оформления, так и по части кода, по пути наращивая свои знания платформы Arduino.

Экспериментируем с кодом

В качестве одного из возможных направлений экспериментирования можно предложить периодическую смену цвета включенного ночника. Некоторую информацию о том, как это реализовать, можно получить, вернувшись к коду для проекта светофора (см. *проект 2*). Так, можно добавить в оператор if...else простой код для мигания вместо использования только функций digitalWrite и реализовать цветовую анимацию.

Можно также для разных уровней освещенности включать различные цвета, используя для этого оператор if...else if вместо простого оператора if... else. Базовый код для реализации этого подхода может выглядеть так (листинг 5.6):

Листинг 5.6. Код для трехцветного ночника

```
if (lightValue < calibrationValue - 200)
{
   //исполняем этот код при полной темноте
   digitalWrite(11, HIGH);
   digitalWrite(10, LOW);
}

else if (lightValue < calibrationValue - 50)
```

```
{
   //исполняем этот код при небольших сумерках
   digitalWrite(11, LOW);
   digitalWrite(10, HIGH);
}
else
{
   //исполняем этот код при полной освещенности
   digitalWrite(11, LOW);
   digitalWrite(10, LOW);
}
```

В этом коде оператор if...else if используется для установки категорий освещенности. Для каждой категории нужно потом добавить код установки цвета для этой категории.

Модифицируем корпус

Вы полностью свободны в разработке дизайна своего ночника, так что вместо предложенного здесь абажура можете создать абажур полностью своей конструкции. Для проектирования и последующего изготовления абажура можно использовать разнообразные средства и материалы; например, бальзу, нарезанную на лазерном резаке, 3D-печать такими материалами, как пластмасса ABS или HIPS, или даже токарные и фрезерные станки с ЧПУ. Управляемое с помощью компьютеров оборудование позволит получить суперточные и аккуратные детали, что, в свою очередь, выльется в более утонченный продукт. Авторы настоятельно рекомендуют исследовать эти возможности, чтобы создать произведение, удовлетворяющее самым изысканным вкусам. Если у вас нет доступа к инструментам такого рода, попробуйте связаться с местными сообществами мастеров-любителей. Часто у них найдется оборудование и инструменты, которые вы сможете использовать.

Файл ресурсов для этой книги (**https://www.nostarch.com/arduinoinventor/**) содержит несколько разнообразных шаблонов абажуров, которые вы можете приспособить под свои желания. На рис. 5.33 приводится пример необычной конструкции одного из таких абажуров.

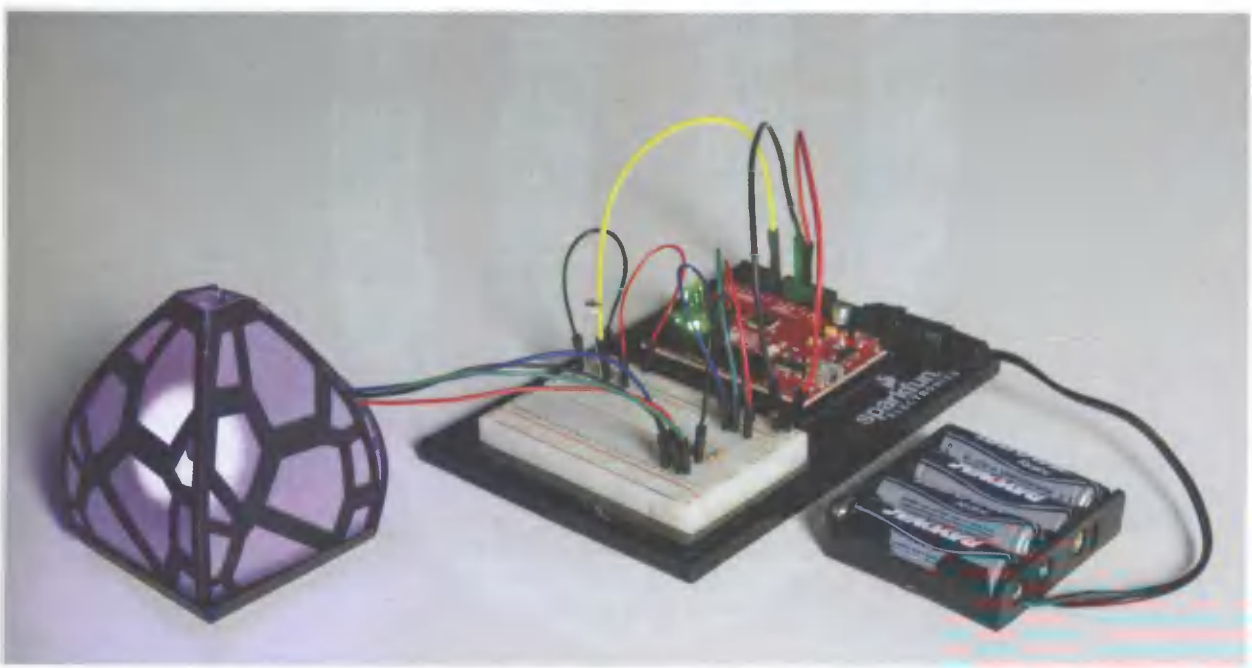


Рис. 5.33. Занятная конструкция абажура ночника, выполненная по дополнительным шаблонам из архива ресурсов для этой книги

БАЛАНСИРНАЯ БАЛКА

В этом проекте мы создадим настольную балансирную игру, используя для этого потенциометр и серводвигатель (небольшой электромотор, способный поворачивать свой вал на точно заданный угол). Игра заключается в том, чтобы катать мячик от настольного тенниса по балансирной балке, не допуская его падения. Для этого мы будем с помощью потенциометра управлять серводвигателем, который в свою очередь будет управлять положением балансирной балки. Готовы начать?

Проект в его завершенном виде показан на рис. 6.1. Это простая игрушка, выполненная полностью из упаковочного картона и некоторых других материалов, которые всегда можно найти у себя дома.

Рис. 6.1. Завершенный проект балансирной балки

Необходимые компоненты, инструменты и материалы

Электронная часть проекта включает сравнительно небольшое количество компонентов, хотя в их состав входят и два новых: серводвигатель и потенциометр.

> **Примечание**
>
> Все электронные компоненты, используемые в этом проекте, входят в стандартный состав набора изобретателя Inventor's Kit компании SparkFun.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 6.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. (на рис. 6.2 не показан);
- беспаечная макетная плата (PRT-12002), 1 шт.;
- потенциометр 10 кОм (COM-09806), 1 шт.;
- миниатюрный серводвигатель (ROB-09065), 1 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026).

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты (рис. 6.3) и материалы (рис. 6.4):

- карандаш или маркер;
- макетный нож;
- металлическая линейка;
- длинногубцы (игольчатые плоскогубцы);
- кусачки;
- клей (клеевой пистолет или клей для моделирования);
- небольшая отвертка;
- ножницы (не показаны);
- могут пригодиться: дрель и сверла диаметром 1,6 мм, 3 мм и 6,4 мм;
- два листа картона размером примерно 22×28 см;
- шаблон балансирной балки (см. рис. 6.16 далее в этом проекте);
- бамбуковая палочка, 1 шт.;
- соломинка для питья (бамбуковая палочка должна свободно входить в соломинку), 1 шт.;
- мячик для настольного тенниса, 1 шт.;
- скрепка для бумаг среднего размера, 1 шт.

Рис. 6.2. Электронные компоненты для проекта балансирной балки

Рис. 6.3. Инструменты, рекомендуемые для проекта балансирной балки

Рис. 6.4. Материалы, рекомендуемые для проекта балансирной балки

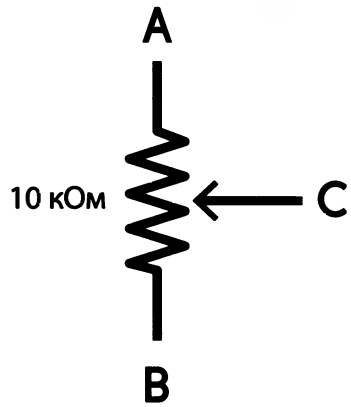


Рис. 6.5. Условное обозначение потенциометра

Новые компоненты

В предыдущих проектах мы использовали плату Arduino в основном для управления светодиодами, но настало время расширить область ее применения и исследовать возможности управления с ее помощью другими компонентами. В этом проекте мы познакомимся с новым схемным элементом, называемым *потенциометром*, и электродвигателем, или, более конкретно, *серводвигателем*.

Потенциометр

В этом проекте мы воспользуемся потенциометром для управления движением балансирной балки. Потенциометр представляет собой переменный резистор, то есть резистор, чье сопротивление можно изменять.

Потенциометр обычно имеет три вывода (контакта), символ потенциометра показан на рис. 6.5.

Потенциометры бывают разных типов и размеров. Наиболее распространены потенциометры роторного типа, приводимые в действие ручкой, установленной на его валу (рис. 6.6, *слева*), или небольшой отверткой (рис. 6.6, *посередине*), но достаточно часто применяются и ползунковые потенциометры (рис. 6.6, *справа*). Независимо от типа потенциометра, все они имеют одинаковый

Рис. 6.6. Потенциометры разных типов: с ручкой на валу (*слева*) — и его мы будем использовать в этом проекте; под отвертку (*посередине*); ползунковый (*справа*)

принцип работы. И мы буквально окружены потенциометрами — дома, на работе, в общественных местах. Практически каждый из нас когда-либо использовал потенциометр, поскольку они применяются, например, для регулировки громкости звука в аудиосистемах. А в старых телевизорах на электронно-лучевых трубках с их помощью также регулировали яркость и выполняли ряд других настроек.

Потенциометр имеет постоянное сопротивление между его противоположными выводами A и B (рис. 6.7). Это сопротивление у разных потенциометров может иметь различную величину, но для нашего проекта мы используем потенциометр сопротивлением 10 кОм. При вращении ручки или перемещении ползунка, третий, скользящий, вывод потенциометра (на рисунках обозначен C), который называется *движком*, перемещается вдоль его резисторной части, вследствие чего сопротивление между выводами B и C меняется. Значение этого сопротивления и считывается схемой, в которой потенциометр применяется.

При вращении вала потенциометра по часовой стрелке движок перемещается по направлению к выводу A, и сопротивление между выводами C и B увеличивается, а при вращении вала против часовой стрелки движок перемещается по направлению к выводу B, и это сопротивление уменьшается. На рис. 6.7 как раз и показаны фазы этого перемещения.

Если вывод A потенциометра подключить к напряжению +5 В, вывод B — к «земле», а вывод C — к аналоговому вводу на плате Arduino, мы получим схему, похожую на схему делителя напряжения, которую мы использовали в *проекте 5*. Вращая вал потенциометра, мы можем варьировать напряжение на его выводе C в диапазоне значений от 0 до +5 В. Такая схема иногда называется *регулируемым делителем напряжения*.

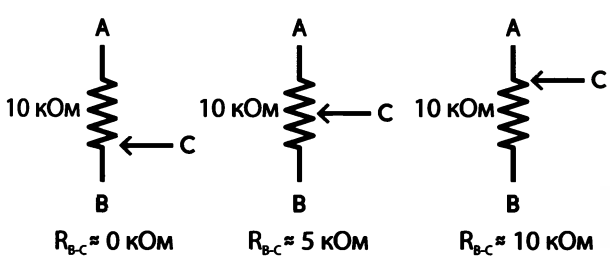


Рис. 6.7. Разные позиции движка потенциометра

Серводвигатель

Серводвигатель (или сервопривод) представляет собой особый вид электродвигателя, вал которого вращается не постоянно, а лишь поворачивается на определенный заданный угол. Большинство серводвигателей могут передавать вращение в диапазоне 180 градусов, но диапазон вращения некоторых составляет все 360 градусов, — такие сервоприводы называются *серводвигателями непрерывного вращения*. В этом проекте мы используем стандартный серводвигатель для моделирования с рабочим диапазоном 180 градусов (их обычно называют *сервомашинками*). Пример подобной сервомашинки показан на рис. 6.8.

Серводвигатели устанавливаются во многие различные агрегаты — от моделей самолетов или автомашин до спидометров автомобилей и роботизированных манипуляторов на сборочных конвейерах.

Чтобы лучше понимать принцип работы сервомашинки, не помешает знать, что у нее внутри. С этой целью мы вскрыли одну из них (рис. 6.9).

Внутри сервомашинка содержит три основные части: электродвигатель, редуктор и плату

Рис. 6.8. Стандартная сервомашинка для моделирования

Рис. 6.9. Внутреннее устройство сервомашинки

управления. Когда на электродвигатель подается напряжение, его ротор начинает вращаться, и это вращение передается через редуктор на вал с качалкой. Угол поворота вала определяется платой управления. Одна из шестерен редуктора насажена на вал потенциометра, вследствие чего при вращении вала электродвигателя он также приводится в движение. Вспомним, что потенциометр представляет собой простой компонент, сопротивление которого меняется в зависимости от величины угла поворота его вала. При включении потенциометра в схему делителя напряжения в качестве одного из его резисторов, выходное напряжение делителя будет варьироваться соответственно повороту вала потенциометра. Плата управления сравнивает значение входного сигнала, поступающего на сервомашинку (в данном случае от Arduino), и значение, поступающее от потенциометра. Когда оба эти значения становятся равны, электродвигатель останавливается и удерживается в этом положении.

Для управления сервомашинкой используется сигнал ШИМ, с которым мы познакомились ранее, в *разд. «Создаем другие цвета» проекта 5*. В частности, плата Arduino подает на вход платы управления машинки сигнал ШИМ с периодом импульсов 20 мс. Длительность самого импульса и определяет угол поворота сервомашинки. Этот процесс показан на рис. 6.10, где минимальная длительность импульсов соответствует нулевому углу поворота, средняя — углу 90°, а максимальная — углу 180°. Точно так же, как мы использовали плату Arduino для мигания светодиодом, ее можно настроить и на создание последовательности импульсов длительностью 1 мс и периодом 20 мс (то есть каждые 20 мс создавать импульс длительностью 1 мс), чтобы установить вал сервомашинки на угол 0°.

Таким образом, установить угол поворота сервомашинки в 0° можно с помощью кода, показанного в листинге 6.1.

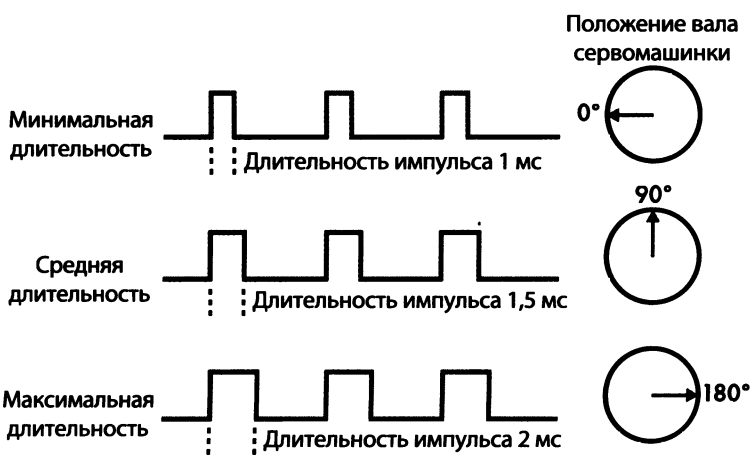


Рис. 6.10. Длительность сигнала ШИМ для разных положений сервомашинки

Листинг 6.1. Код для установки сервомашинки на угол 0°

```
void setup()
{
    pinMode(9, OUTPUT);
}
void loop()
{
    digitalWrite(9, HIGH);
    delay(1);
    digitalWrite(9, LOW);
    delay(19);
}
```

Этот код выставляет высокий уровень сигнала на выводе 9 в течение 1 мс, после чего переключает сигнал на нижний уровень и удерживает его в течение 19 мс, повторяя эту последовательность уровней в бесконечном цикле. Но код, занятый удерживанием такого тактирования последовательности уровней сигнала, не может заниматься более ничем другим, поскольку это собьет тактирование сигналов и, соответственно, точность управления сервомашинкой. К счастью, эта проблема решается с помощью облегчающей управление сервомашинкой уловки, которая заключается в использовании специальной библиотеки. Упрощенно, *библиотека* представляет собой файл, содержащий код, который можно включить в свой скетч для выполнения определенных действий. В данном случае мы воспользуемся библиотекой Servo, которая и станет заниматься задачей тактирования импульсов для вращения вала сервомашинки на определенный угол.

В этом проекте с помощью Arduino мы будем управлять движением балансирной балки в ответ на выходные сигналы датчика, которым в нашем случае станет потенциометр. Для этого наш код должен считывать показания датчика и на их основании устанавливать требуемую длительность импульса, чтобы провернуть вал сервомашинки на требуемый угол, а вал в свою очередь повернет на соответствующий угол прикрепленную к нему балансирную балку.

Эта задача легче, чем может казаться, поскольку вся работа по созданию основного необходимого кода уже выполнена в только что упомянутой библиотеке Servo. Понимать принцип управления сервомашинкой посредством варьирования длительности импульсов полезно, но трудную работу по реализации этого управления лучше оставить программе.

Создаем прототип схемы управления балансирной балкой

Теперь мы, подкованные теорией, можем приступить к созданию схемы для управления балансирной балкой. Ничего сложного в этой схеме нет — сначала подключим к Arduino сервомашинку, а затем потенциометр. Готовая схема показана на рис. 6.11.

Обратите внимание, что три соединительных провода сервомашинки имеют гнездовой разъем, вследствие чего для подключения сервомашинки к схеме потребуются проволочные перемычки со штыревыми разъемами на обоих концах. Возьмите три такие перемычки и вставьте их в разъем сервомашинки, как показано на рис. 6.12.

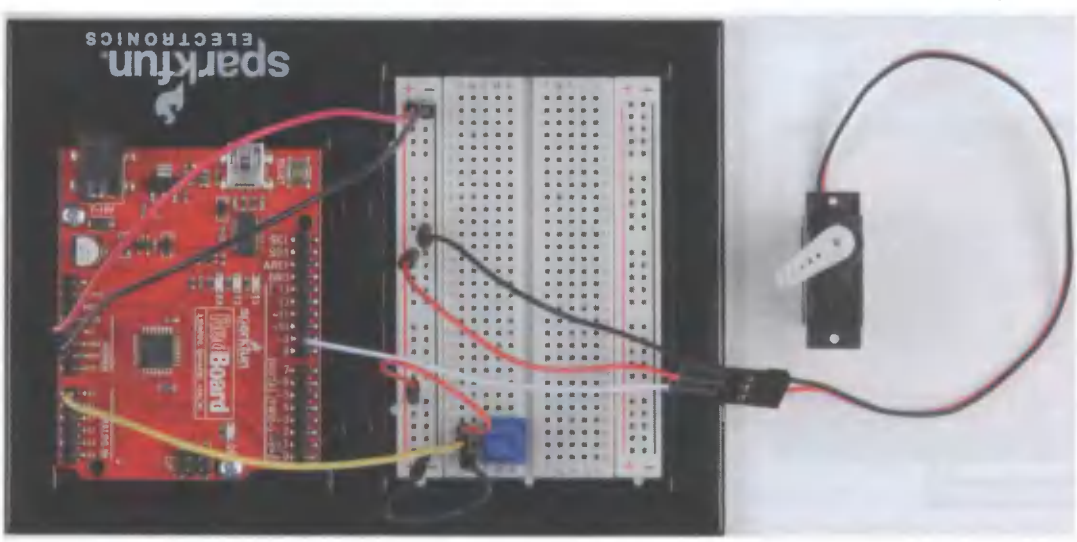


Рис. 6.11. Прототип схемы для управления балансирной балкой

Рис. 6.12. Подключение проволочных перемычек к разъему сервомашинки

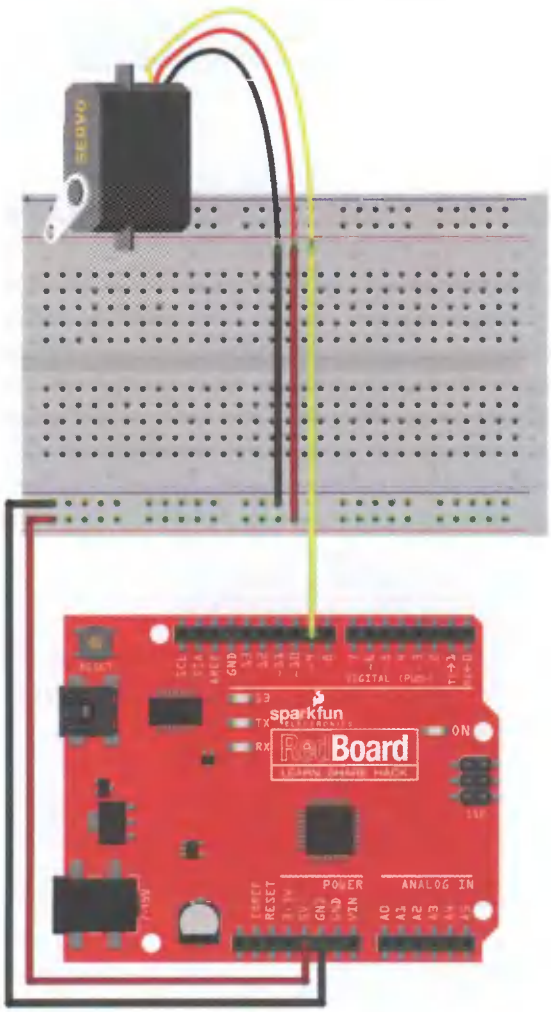


Рис. 6.13. Монтажная схема подключения сервомашинки к плате Arduino

Как уже неоднократно упоминалось, мы рекомендуем вам использовать разноцветные провода, чтобы было легче разбираться со схемой: красный — для положительного питания, черный — для отрицательного («земля») и белый — для сигнальной линии.

Далее подключаем перемычки к Arduino. Здесь тоже нет ничего сложного: соединяем перемычками разъемы +5 В и GND платы Arduino с шинами положительного и отрицательного питания с левой стороны макетной платы. Затем подключаем перемычку отрицательного питания сервомашинки (черный провод) к отрицательной шине макетной платы, а перемычку положительного питания — к шине положительного (+5 В) питания макетной платы. Сигнальный провод подключаем непосредственно к плате Arduino в гнездо вывода 9. Монтажная схема подключения сервомашинки к плате Arduino показана на рис. 6.13.

Теперь наденем на вал сервомашинки качалку. *Качалка сервомашинки* — это рычаг (в комплекте сервомашинки могут поставляться рычаги разной формы), который передает движение вала сервомашинки на управляемую ею деталь устройства. Выберем из поставляемого комплекта одностороннюю качалку и насадим ее на шпиндель сервомашинки, как показано на рис. 6.14. Для рабочего проекта мы применим качалку другой формы, но сейчас воспользуемся этой, чтобы было легче наблюдать за вращением шпинделя сервомашинки.

Рис. 6.14. Надевание односторонней качалки на шпиндель сервомашинки

Если после подключения сервомашинки ее вал начнет беспорядочно двигаться, просто отсоедините черный провод от отрицательной шины питания, чтобы остановить его. Вообще, рекомендуется не подключать на схему отрицательное питание до тех пор, пока в Arduino не будет загружен код.

Наконец, включим в схему потенциометр. На макетной плате у нас полно места, поэтому потенциометр можно поставить на ней где угодно, но не забывайте при этом, что каждый вывод потенциометра нужно вставить в отдельный ряд отверстий макетной платы. Подключите крайние выводы потенциометра к шинам питания макетной платы, а средний вывод — к аналоговому выводу A0 платы Arduino, как показано на рис. 6.15.

Таким образом, на данном этапе у нас есть датчик в виде потенциометра, подключенный к сервомашинке. Чтобы датчик мог управлять сервомашинкой, которая в свою очередь будет управлять балансирной балкой, нам требуется соответствующая программа.

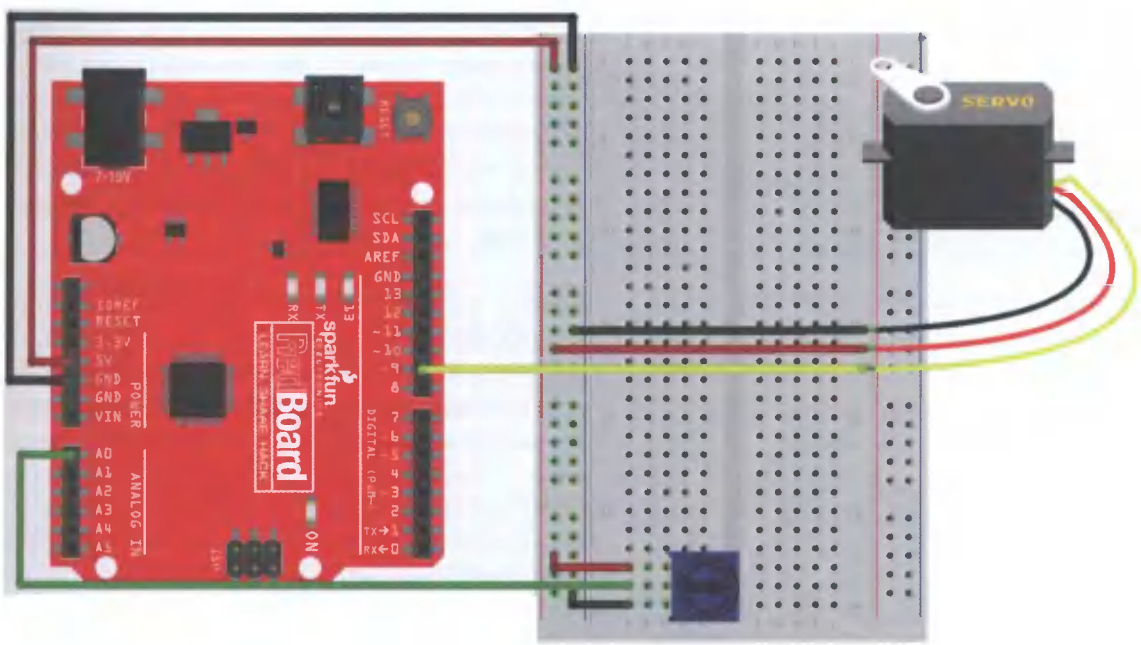


Рис. 6.15. Полностью собранная монтажная схема для управления балансирной балкой

Программа для управления балансирной балкой

Чтобы управлять сервомашинкой с помощью Arduino, нам требуется внешняя библиотека Servo. Как упоминалось ранее, внешняя библиотека представляет собой набор функций, которые можно вставлять в код скетча, чтобы расширить его возможности. Таким образом, внешняя библиотека облегчает работу программиста по разработке кода для управления разнообразными аппаратными устройствами с помощью Arduino. В данном случае библиотека Servo содержит код, конфигурирующий вывод Arduino для управления сервомашинкой, вращения вала сервомашинки на заданный угол, а также даже для переконфигурирования вывода, используемого сервомашинкой, для других функций.

Прежде чем приступать к программированию проекта, загрузим в Arduino небольшой скетч, чтобы проверить работоспособность сервомашинки.

Проверяем работоспособность машинки

Листинг 6.2 содержит простой скетч для управления сервомашинкой. Создайте новый скетч и скопируйте в него код из этого листинга.

Листинг 6.2. Программа «Здравствуй, мир!» для сервомашинки

```
❶ #include<Servo.h>
❷ Servo myServo;
  void setup()
  {
❸ myServo.attach(9);
  }

  void loop()
  {
❹   myServo.write(90);
  }
```

Чтобы библиотеку Servo можно было использовать в коде разрабатываемых скетчей, ее нужно сначала подключить, что и делается вызовом специальной функции #include<Servo.h> ❶. Эта функция дает указание Arduino подключить к скетчу код из файла Servo.h, который содержит команды для управления сервомашинкой. Таким способом все определения и функции библиотеки включаются в скетч. Обратите внимание на отсутствие точки с запятой в конце этой инструкции — это один из таких редких случаев. В программировании Arduino символ # указывает компилятору, что следующий за ним код является *директивой препроцессора*. Это специальный код, который необходимо обработать, прежде чем обрабатывать остальной код скетча. При компилировании скетча сначала исполняется препроцессор, который ищет в коде строки, начинающиеся символом # и не заканчивающиеся точкой с запятой, и сначала исполняет эти команды. Директива #include дает указание препроцессору включить в скетч весь код из указанного файла, прежде чем выполнять компиляцию скетча.

Библиотеку также можно подключить из меню среды разработки Arduino. Для этого нужно выполнить последовательность команд меню **Sketch | Include Library**, а затем выбрать в открывшемся списке требуемую библиотеку (в данном случае **Servo**). Это автоматически вставит в скетч инструкцию #include. Такая возможность полезна тогда, когда вы забыли точный синтаксис команды #include или не знаете точное имя подключаемой библиотеки, — например, когда работаете с ней впервые.

Библиотека Servo позволит нам создать особый вид структуры данных, называемой *объектом*. Объект представляет собой контейнер для предопределенных переменных и функций. Функции, связанные с объектом, называются *методами*. В данном скетче строка кода Servo myServo; создает объект типа Servo и присваивает ему имя myServo ❷.

Подобно переменной, объектам можно присваивать любые имена, но рекомендуется использовать описательные имена, чтобы вам самим было легче понимать функцию объекта. Через имя объекта можно обращаться ко всем командам управления сервомашинкой, содержащимися в библиотеке Servo. Например, метод myServo.attach() указывает Arduino, какой вывод использовать для управления сервомашинкой. В случае необходимости управлять несколькими сервомашинками, потребуется создать нужное количество объектов типа Servo, присвоив им однозначные имена, чтобы можно было независимо управлять каждой из них.

Предположим, например, что нам требуется управлять плечом (shoulder), локтем (elbow) и кистью (wrist) роботизированной руки, используя отдельные сервомашинки, встроенные в соответствующие суставы. Для этого нужно создать три объекта типа Servo, назвав их, соответственно, shoulderServo, elbowServo и wristServo, которые и будут независимо управлять каждой соответствующей сервомашинкой. Для каждого из этих объектов Servo их методы можно будет использовать независимо друг от друга.

Но для игры в балансирную балку нам требуется всего одна сервомашинка. И посредством метода myServo.attach(9) ❸ функция setup() скетча в листинге 6.2 сообщает Arduino, что для управления сервомашинкой будет использоваться вывод 9. Затем посредством метода myServo.write(90) ❹ в функции loop() дается инструкция повернуть вал сервомашинки на угол 90°. Библиотека Servo преобразовывает величину угла вращения в градусах в сигнал ШИМ с соответствующей длительностью импульса. Операция преобразования встроена в метод write(), и нам не нужно заботиться об этом.

Примечание

Хотя полный диапазон вращения вала сервомашинки составляет 0°–180°, рекомендуется использовать для метода write() значения в диапазоне 10°–170°, особенно для сервомашинок с пластмассовыми шестернями редуктора. Выход за границы рабочего диапазона сервомашинки может вызвать необратимые повреждения шестерен.

Загрузите этот скетч в Arduino, а затем вставьте черный провод (отрицательное питание) сервомашинки в шину отрицательного питания макетной платы. Вал сервомашинки должен провернуться на угол 90°. Теперь можно спокойно оставить питание сервомашинки подключенным.

Чтобы провернуть вал на другой угол в пределах рабочего диапазона сервомашинки (10°–170°), вставьте соответствующее значение угла в функцию write() и снова загрузите скетч в Arduino. Поэкспериментируйте немного с сервомашинкой, поворачивая ее вал на разные углы.

На данном этапе мы знаем, как провернуть вал сервомашинки всего лишь один раз. Теперь попробуем создать код для поворота вала сервомашинки на разные углы (листинг 6.3).

Листинг 6.3. «Мигание» сервомашинкой

```
#include<Servo.h>
Servo myServo;
void setup()
{
  myServo.attach(9);
}

void loop()
{
  myServo.write(10);
  delay(1000);
  myServo.write(170);
  delay(1000);
}
```

Этот скетч является версией скетча мигания светодиодом из *проекта 1*, слегка модифицированного под «мигание» сервомашинкой. Он выставляет вал сервомашинки в позицию 10°, приостанавливает исполнение на 1 секунду, затем выставляет вал в позицию 170°, снова приостанавливает исполнение, а затем повторяет эти действия в цикле.

Загрузите скетч в Arduino и понаблюдайте за его работой. Если он работает должным образом, можно переходить к следующему этапу — интерактивному управлению сервомашинкой, без необходимости перепрограммировать Arduino каждый раз для новой позиции вала. Для этого нам и пригодится потенциометр.

Финальная версия скетча для игры в балансирную балку

В финальной версии скетча мы будем управлять вращением вала сервомашинки с помощью потенциометра. Модифицируйте скетч, как показано в листинге 6.4.

Листинг 6.4. Использование функции map() для управления сервомашинкой посредством потенциометра

```
#include<Servo.h>
Servo myServo;
❶ int potVal;
int anglePosition;
void setup()
{
  myServo.attach(9);
}

void loop()
{
  potVal = analogRead(A0);
❷ anglePosition = map(potVal, 0, 1023, 10, 170);
  myServo.write(anglePosition);
  delay(20);
}
```

Этот скетч считывает значение сопротивления потенциометра, преобразовывает его в соответствующее значение угла, а затем подает соответствующий сигнал управления на вывод управления сервомашинкой. В скетче используются некоторые новые команды, поэтому мы рассмотрим его пошагово.

Верхняя часть кода такая же, как и в первых двух листингах. В частности, строка кода Servo myServo; создает объект типа Servo и присваивает ему имя myServo. Далее объявляются две глобальные переменные: potVal и anglePosition ❶. Первая переменная используется для хранения значения положения потенциометра, а вторая — для угла положения вала сервомашинки.

В функции loop() переменной potValue присваивается значение, полученное преобразованием функцией analogRead(A0) аналогового значения на потенциометре в цифровое значение. При вращении вала потенциометра напряжение на его скользящем контакте может варьироваться в диапазоне значений от 0 до +5 В. Вспомним, что функция analogRead() преобразовывает напряжение в диапазоне 0–5 В в числа в диапазоне 0–1023. Но этот диапазон значений не очень полезен для управления сервомашинкой. Как упоминалось ранее, безопасный диапазон положений вала сервомашинки ограничен диапазоном углов 10–170°.

Ничего страшного. Встроенная функция Arduino map() позволяет преобразовывать числа в одном диапазоне в соответствующие числа в другом диапазоне. Значение угла, вычисленное этой функцией для значения potVal, сохраняется в переменной anglePosition ❷. Функция map() принимает пять параметров: map(значение, изНижнего, изВерхнего, вНижнее, вВерхнее). В данном случае она преобразовывает значение переменной potVal в диапазоне значений 0–1023 в диапазон значений 10–170. Это очень изящная функция, которая позволяет с большой легкостью выполнять масштабирование и преобразование из одного диапазона значений в другой.

В скетч также встроена короткая пауза длительностью 20 мс, чтобы дать сервомашинке достаточно времени на выполнение поворота вала, прежде чем считывать значение потенциометра снова. Длительность 20 мс является минимальной, требуемой сервомашинкой. Вспомните, что период импульсов сигнала ШИМ, применяемого для установки угла поворота, составляет 20 мс.

Загрузите модифицированный скетч в Arduino. Проверьте работу схемы и скетча, вращая вал потенциометра. Вал сервомашинки должен поворачиваться в положения, соответствующие положениям вала потенциометра. Если все работает должным образом, можно приступать к созданию игры в балансирную балку на этой основе.

Собираем игру в балансирную балку

Нашу новоприобретенную способность управления сервомашинкой можно использовать для создания простой настольной игры. А именно балансирной балки, положение которой управляется сервомашинкой, которая в свою очередь управляется потенциометром, который в свою очередь управляется игроком. Суть игры следующая — на балансирную балку ставится мячик от настольного тенниса, и игрок должен постараться подкатить его как можно ближе к концу балки, не дав ему упасть с нее.

Вырезаем детали

Распечатайте шаблон для игры (рис. 6.16) и переведите его на картон. Напомним, что все необходимые для проектов ресурсы, включая скетчи и файлы шаблонов, доступны для загрузки по адресу: **https://www.nostarch.com/arduinoinventor/**. При разработке проекта мы прилагали усилия, чтобы разместить шаблон на кусок картона как можно меньшего размера.

С помощью макетного ножа вырежьте каждую деталь по сплошным линиям контура, вырежьте также отверстие для установки сервомашинки.

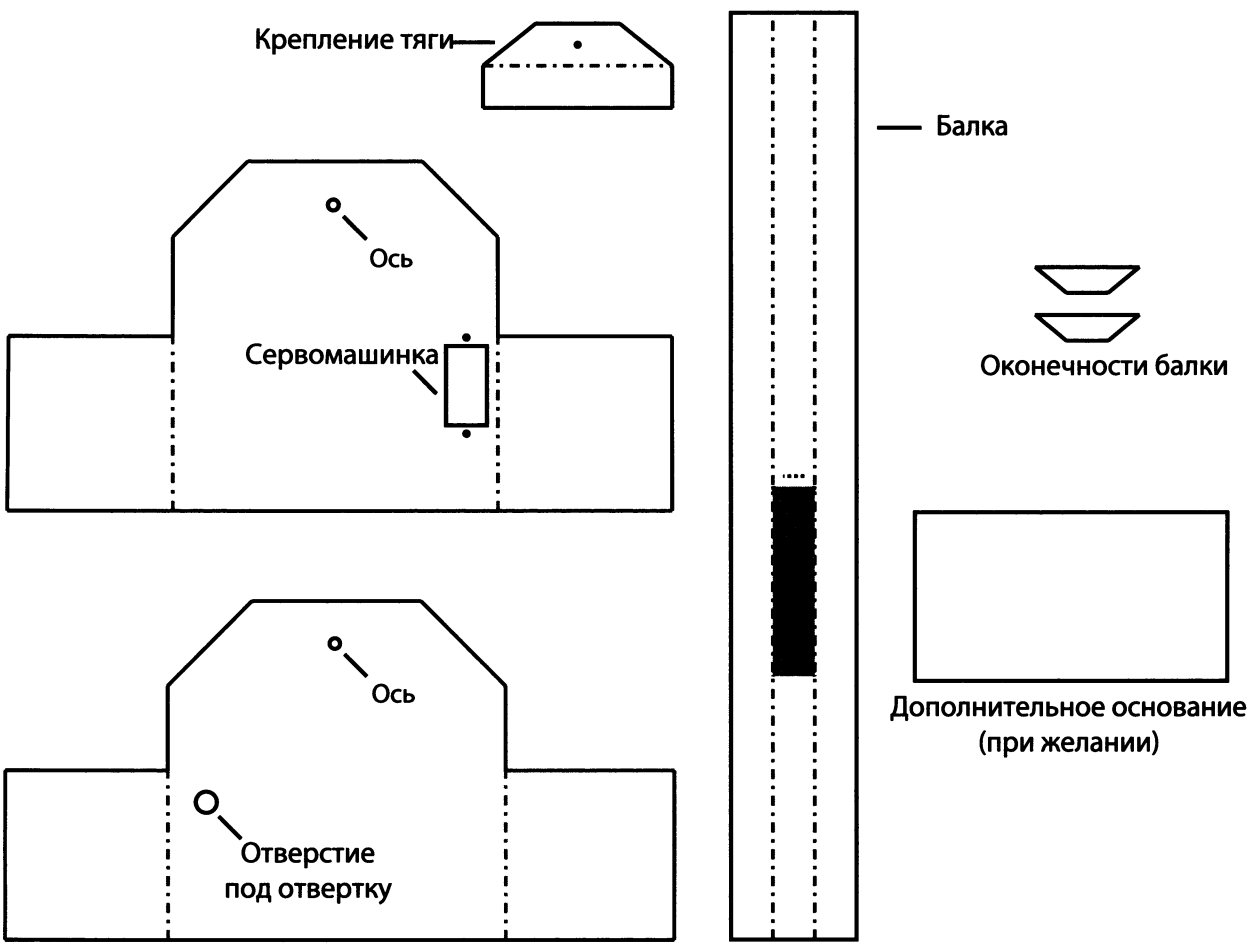


Рис. 6.16. Шаблоны деталей (боковых стоек и собственно балки) для игры в балансирную балку (в уменьшенном виде)

Пока не делайте надрезов по пунктирным линиям — это нужно будет сделать позже, в процессе сборки. Как всегда, соблюдайте правила безопасности при вырезании шаблонов. В частности, используйте острый макетный нож и металлическую линейку (рис. 6.17) и работайте не торопясь.

С помощью макетного ножа или сверла сделайте все обозначенные отверстия. Для отверстия под отвертку используйте сверло диаметром 6,3 мм, для отверстий под ось — сверло диаметром 3 мм, а для отверстий в креплении тяги и под крепление сервомашинки — диаметром 1,6 мм.

Рис. 6.17. Вырезание деталей корпуса игры

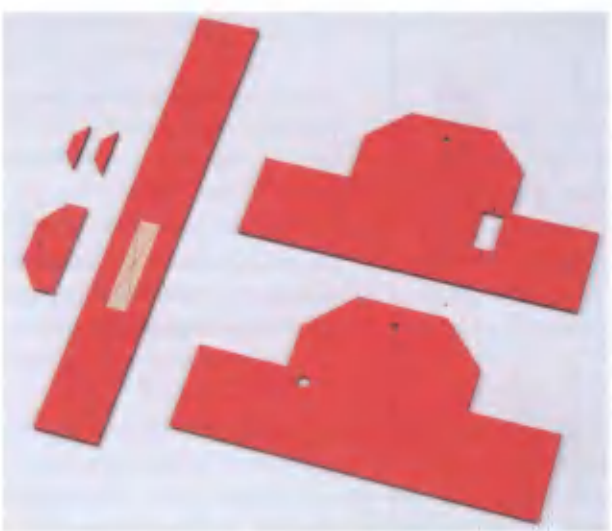


Рис. 6.18. Все вырезанные детали корпуса игры

В результате у вас должно получиться шесть деталей корпуса (рис. 6.18).

Собираем собственно балансирную балку

Возьмите самую длинную вырезанную деталь, которая станет собственно балансирной балкой, и аккуратно сделайте неглубокие надрезы по пунктирным линиям вдоль нее. Это позволит нам согнуть балку, чтобы создать в ней желобок для удерживания мячика. Длина балки должна быть 11 дюймов (28 см) — такой же, как и длина стандартного листа бумаги для писем[1].

Далее подготовим крепление для тяги. Это небольшая деталь трапециевидной формы, приблизительно 60 мм длиной и 25 мм шириной. С помощью этой детали качалка сервомашинки будет соединяться с балкой. Сделайте в ней неглубокий надрез по пунктирной линии (рис. 6.19, *слева*), а потом согните по надрезу под углом 90°, как показано на рис. 6.19, *справа*.

Затем отрежьте от соломинки для питья отрезок длиной около 45 мм и приклейте его поперек балки к центральной трети балки (рис. 6.20).

Далее приклейте к балке крепление для тяги. Крепление приклеивается прямоугольной частью слева от соломинки для питья, как показано на рис. 6.21. В этом положении трапециевидная часть с отверстием для тяги должна быть ориентирована в нашу сторону. Это важный аспект, так как нам нужно, чтобы положение крепления совпадало с положением качалки сервомашинки.

Затем слегка согните стороны балки вдоль сделанных ранее надрезов, чтобы создать желобок для удержания мячика (рис. 6.22).

На концах балки приклеиваем трапециевидные оконечники, чтобы балка удерживала форму

[1] Не путать с размером A4, который составляет 297×210 мм. А размер «для писем» (letter size) составляет 11×8,5 дюйма или 279×216 мм.

Рис. 6.19. Подготавливаем крепление для тяги: делаем надрез (*слева*) и сгибаем по надрезу (*справа*)

Рис. 6.20. Приклеиваем соломинку для питья по центру балки

Рис. 6.21. Приклеивание к балке крепления для тяги

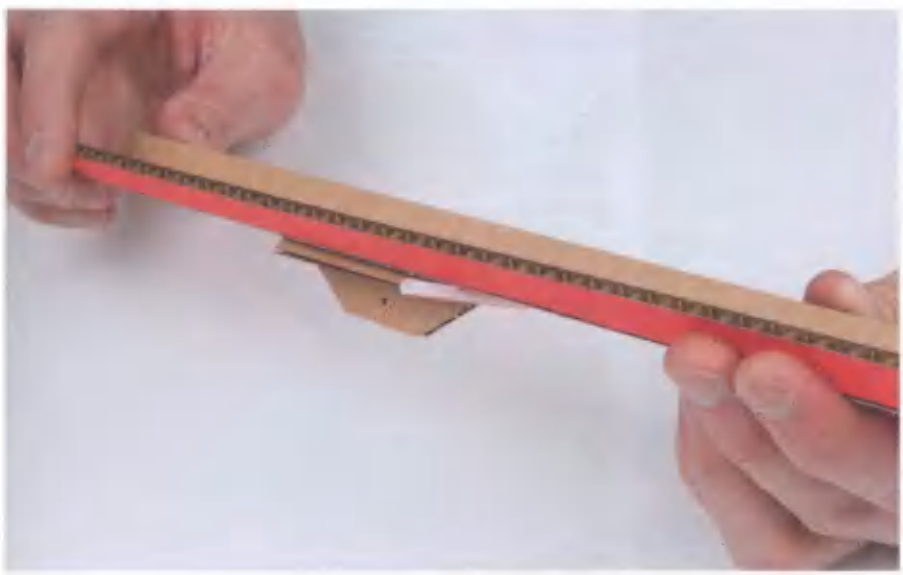


Рис. 6.22. Подгибаем стороны по надрезам, чтобы создать желобок для мячика

желобка и чтобы закрыть его концы. Для надежности крепления рекомендуется использовать клеевой пистолет (рис. 6.23).

С помощью кусачек откусите от бамбуковой палочки отрезок длиной около 85 мм. Вставьте полученный отрезок в соломинку для питья, чтобы получить ось для балансирной балки (рис. 6.24).

Собираем основание и прикрепляем сервомашинку

На следующем этапе мы соберем основание для балансирной балки. Сделайте неглубокие надрезы по пунктирным линиям одной из боковых стоек, как показано на рис. 6.25, и согните ее вдоль надрезов, чтобы она приобрела П-образную

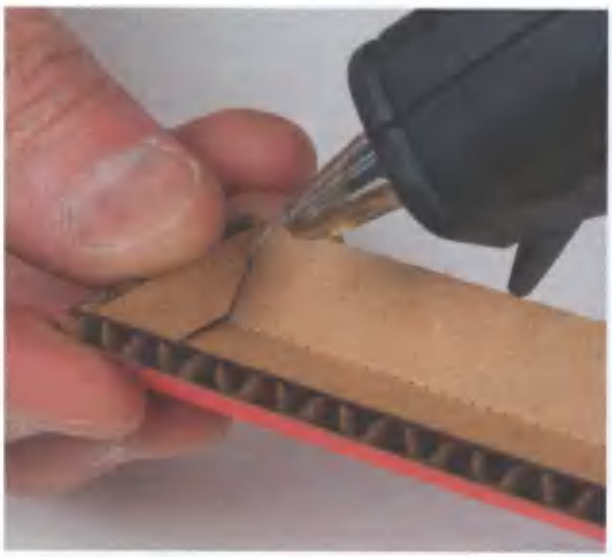


Рис. 6.23. Приклеиваем оконечники балки

Рис. 6.25. Надрезание боковых стоек

Рис. 6.24. Ось качания из бамбуковой палочки: длина выступающих частей оси одинаковая с обеих сторон

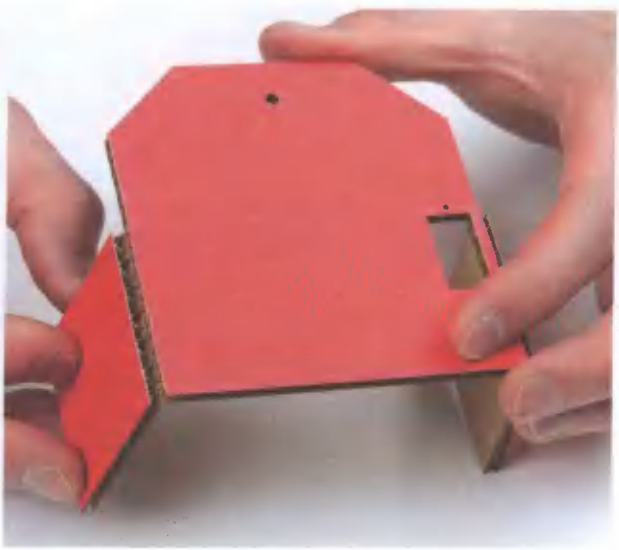


Рис. 6.26. Сгибаем боковые стойки буквой «П»

форму в плане (рис. 6.26). Выполните эти же операции и для второй боковой стойки.

Прежде чем склеивать боковые стойки вместе, на одну из них необходимо установить сервомашинку. Удалите сервомашинку с макетной платы. В одной из боковых стоек у вас имеется сделанный ранее вырез, в который должна входить сервомашинка. Вставьте сервомашинку в этот вырез валом вовнутрь, как показано на рис. 6.27.

В комплекте с сервомашинкой должны идти три крепежных винта: один короткий и два чуть подлиннее. С помощью длинных винтов прикрепите сервомашинку к боковой стойке, как показано на рис. 6.28. Если винты затерялись, сервомашинку можно закрепить и клеем.

Теперь из набора качалок, поставляемых в комплекте сервомашинки, выберите одностороннюю качалку длиной около 12,5 мм. Аккуратно насадите качалку на вал сервомашинки, как показано на рис. 6.29.

Надев качалку, установите вал в положении 0°. Для этого осторожно вращайте вал за качалку против часовой стрелки, пока он не остановится. При этом должен слышаться звук вращения шестеренок редуктора сервомашинки. Обязательно вращайте вал медленно, поскольку шестеренки изготовлены из пластмассы, и грубое обращение может их повредить.

Выставив вал в крайнее положение, снимите с него качалку, а затем снова наденьте ее, но так, чтобы она была направлена вдоль сервомашинки, как показано на рис. 6.30. Это облегчит крепление тяги к балке.

Закрепите качалку на валу с помощью короткого винта, идущего в комплекте с сервомашинкой, чтобы она случайно не соскочила с вала. При затягивании винта качалка может слегка проворачиваться. Не волнуйтесь, ничего не будет повреждено. Но лучше, конечно, при затягивании крепящего винта удерживать качалку пальцами, чтобы она не вращалась. (Опять же, если крепежный винт

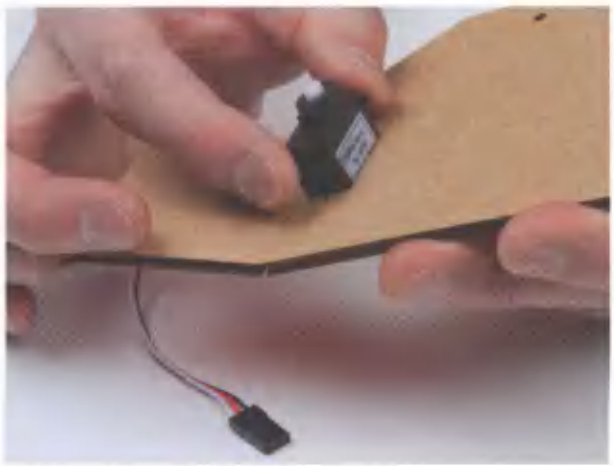


Рис. 6.27. Установка сервомашинки в боковую стойку

Рис. 6.28. Крепим сервомашинку к боковой стойке

Рис. 6.29. Надеваем одностороннюю качалку на вал сервомашинки

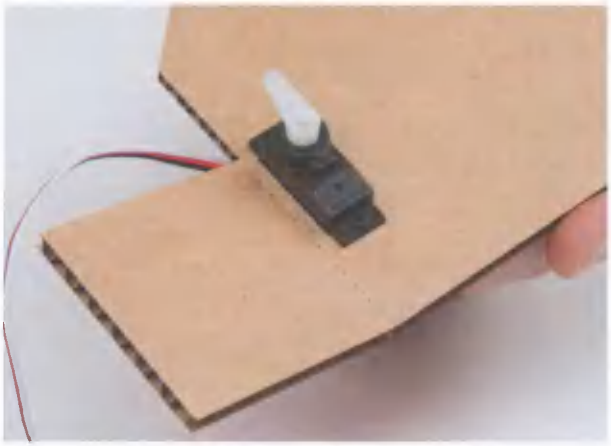


Рис. 6.30. Сервомашинка выставлена на 0°, качалка сориентирована вдоль сервомашинки

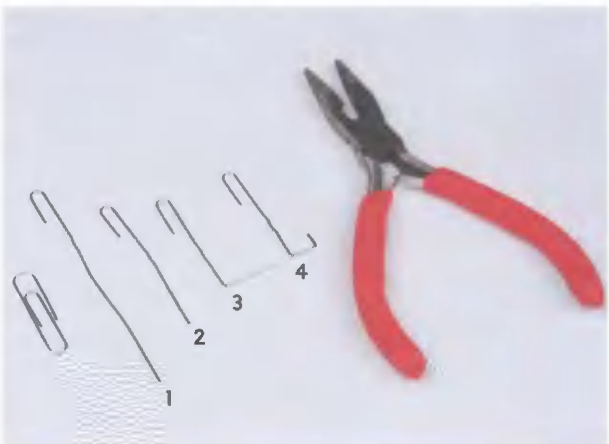


Рис. 6.31. Этапы создания тяги для качалки из канцелярской скрепки

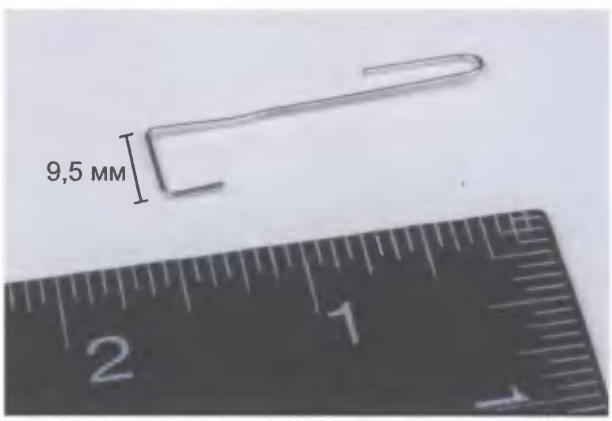


Рис. 6.32. Размеры готовой тяги

качалки затерялся, не проблема, можно работать и без него, — если качалка соскочит с вала, ее надо будет просто надеть обратно.) Если в будущем понадобится изменить положение качалки на шпинделе, вам нужно будет вывернуть крепежный винт. Для этого предусмотрено отверстие под отвертку на противоположной боковой стойке.

Теперь нам нужно сделать тягу, чтобы соединить качалку с балкой. Для этого возьмем канцелярскую скрепку для бумаг, выпрямим ее, а затем согнем в требуемую форму с помощью длинногубцев. Все шаги этого процесса показаны на рис. 6.31.

1. С помощью длинногубцев разогните скрепку, оставив только внутренний сгиб на одном конце.
2. Откусите скрепку с прямого конца, чтобы ее длина была около 5 см.
3. Согните прямой конец под прямым углом на расстоянии около 38 мм от согнутого конца.
4. Согнутую часть опять согните под прямым углом на расстоянии около 9 мм от первого сгиба. Готовая тяга должна быть около 1,5 дюйма (38 мм) длиной (рис. 6.32). Эта длина рассчитана на размеры и геометрию используемого шаблона корпуса. Если вы отклонились от этих размеров, вам придется поэкспериментировать с длиной тяги, чтобы ее можно было правильно присоединить к балке.

Финальная сборка

Настало время заняться финальной сборкой игры. Сначала склеим вместе две боковые стойки. Для этого склейте противоположные квадратные створки, начиная со стороны, обратной той, на которой установлена сервомашинка (рис. 6.33 и 6.34). Таким образом у нас останется место для рук внутри основания, чтобы подсоединить тягу.

Возьмите сделанную из скрепки тягу и вставьте ее согнутым концом, который уже был на ней

Рис. 6.33. Для склеивания створок основания нанесите на одну из них клей зигзагом

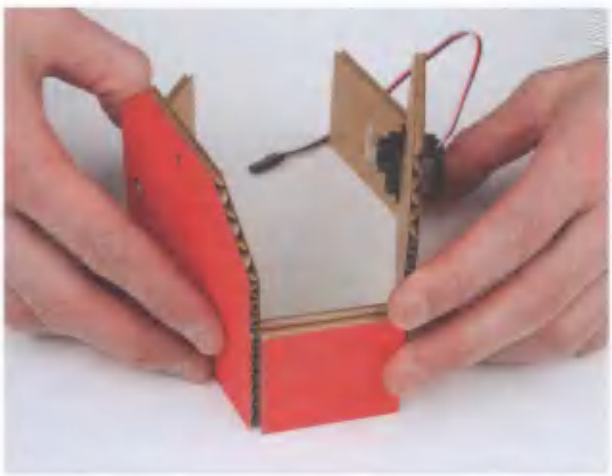


Рис. 6.34. Сначала склейте сторону, которая дальше от сервомашинки

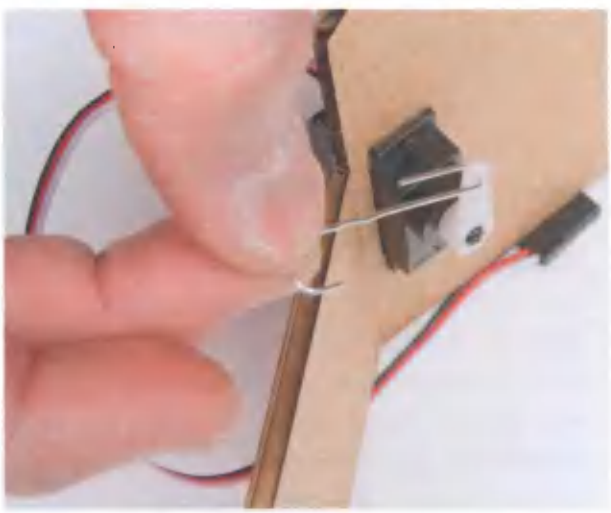


Рис. 6.35. Вставляем тягу в качалку сервомашинки: согнутый конец тяги (*слева*) продет в последнее отверстие в качалке (*справа*)

исходно (рис. 6.35, *слева*), в последнее отверстие в качалке сервомашинки, как показано на рис. 6.35, *справа*.

Вставьте другой конец тяги в отверстие крепления тяги на балке, как показано на рис. 6.37.

Теперь вставьте ось тяги в соответствующее отверстие в одной боковой стойке (рис. 6.38), совместите другой конец оси с отверстием в другой боковой стойке и вставьте ось в него. Теперь можно приклеить и другую боковую стойку, которую мы пока оставляли открытой (рис. 6.39).

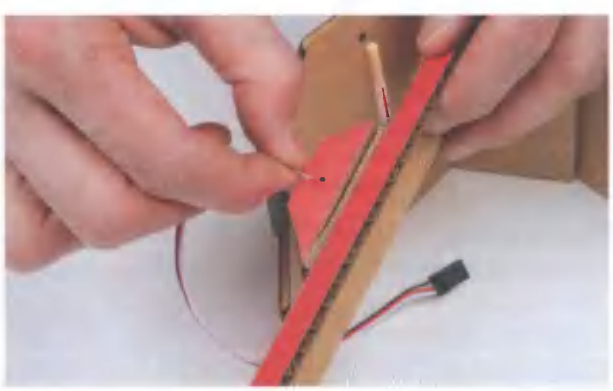


Рис. 6.37. Вставляем другой конец тяги в крепление тяги на балансирной балке

ЧЕТЫРЕХЗВЕННЫЕ ШАРНИРНЫЕ МЕХАНИЗМЫ И ИСПОЛЬЗОВАНИЕ СЕРВОПРИВОДОВ ДЛЯ ВЫПОЛНЕНИЯ СЛОЖНЫХ ДВИЖЕНИЙ

Механизм, с помощью которого вращательное движение качалки сервомашинки преобразовывается в вертикальное возвратно-поступательное движение балансирной балки, называется *четырехзвенным шарнирным механизмом*. Шаблон этого проекта был разработан таким образом, чтобы длина тяги составляла около 38 мм, полагая длину качалки сервомашинки 13 мм. Перемещения вала сервомашинки и балансирной балки рассчитывались, исходя из этих размеров. Так что вам самим не придется заниматься никакими сложными геометрическими вычислениями, поскольку это уже было сделано за вас. На рис. 6.36 показан четырехзвенный шарнирный механизм в действии.

Четырехзвенные шарнирные механизмы предоставляют искусный способ преобразования вращательного движения (например, вращения вала сервомашинки) в движение другого типа (например, вертикальное возвратно-поступательное движение балансирной балки). Инженеры-механики и разработчики роботов используют механизмы этого типа вместе с разными тягами для придания своим творениям способность двигаться требуемым образом.

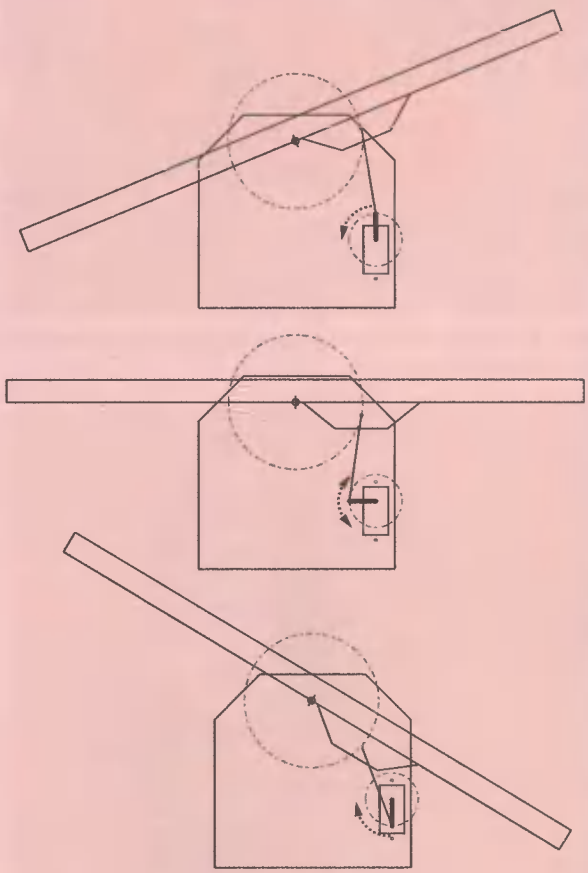


Рис. 6.36. Четырехзвенный шарнирный механизм в действии

В завершение подключите сервомашинку обратно к схеме на макетной плате (рис. 6.40).

Подключите питание к плате Arduino — вал сервомашинки должен стать в исходное положение. Проверните вал потенциометра, чтобы проверить, что наш шарнирный механизм работает должным образом. Если что-то не так, убедитесь, что все подсоединено должным образом и что ничего не выпало из креплений.

Мы рекомендуем вам сделать также дополнительное основание, устанавливаемое в нижнюю часть корпуса игры. На распечатке шаблона

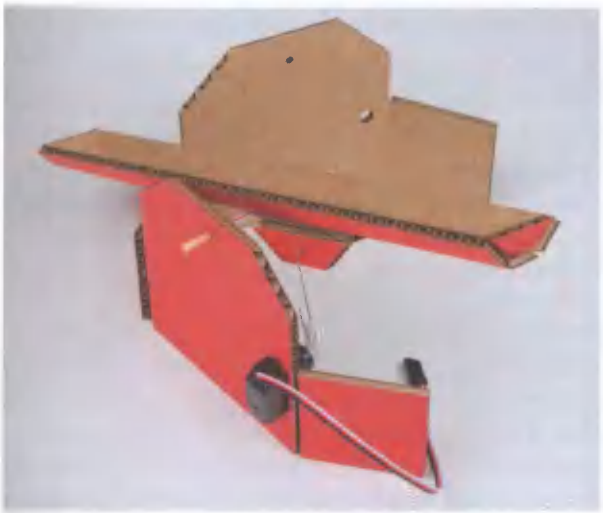


Рис. 6.38. Установка балансирной балки в корпус

Рис. 6.39. Проект балансирной балки в сборе

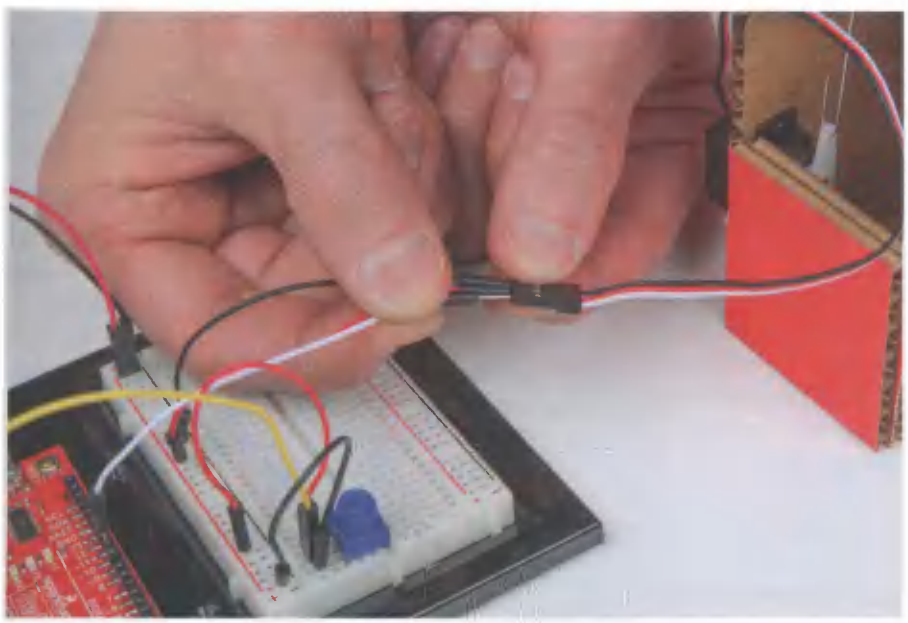


Рис. 6.40. Подключение сервомашинки к схеме на макетной плате

(см. рис. 6.16) оно снабжено пометкой «при желании». Размер этого основания должен быть около 5×9,5 см. Вставьте его в корпус, чтобы обеспечить ему дополнительную жесткость.

Вот теперь наш проект полностью готов! Положите в желобок на балке мячик для настольного тенниса или небольшой шарик и проверьте свой глазомер и реакцию. Теперь у вас есть с чем поиграть, когда вам не нужно делать что-либо более полезное. Сколько раз вы сможете прокатить мячик туда и обратно, не дав ему упасть с балки? Вызовите на соревнование кого-нибудь из своих друзей и выясните, кто из вас лучше в этой игре.

Идем дальше...

В этом проекте мы окунулись в мир сервоприводов и библиотек Arduino. Эти две области содержат большие потенциальные возможности, и мы бы хотели дать вам несколько советов по экспериментированию с сервоприводами.

Экспериментируем со схемой и кодом

Замените потенциометр схемой фоторезистора из *проекта 5*. При этом нужно будет включить в схему резистор номиналом 10 кОм и откорректировать используемые значения масштабирования. Теперь попробуйте управлять движением балансирной балки, перемещая руку вверх-вниз над фоторезистором. С чем у вас получается лучше — с фотодатчиком или потенциометром?

Модифицируем проект

В этот проект можно включить режим «автопилота», чтобы он балансировал мячиком самостоятельно. Для этого в схему следует добавить переключатель. Как мы видели в *проекте 5*, переключатель похож на кнопку в том смысле, что он замыкает или размыкает электрическую цепь, но с той разницей, что установленное переключателем состояние сохраняется до тех пор, пока он не будет переключен в другое положение.

В данном случае нам потребуется однополюсный двухпозиционный переключатель, наподобие показанного на рис. 6.41. *Однополюсный двухпозиционный* означает, что переключатель имеет один общий вывод и два других вывода, к которым общий вывод может быть подключен. Когда ползунок переключателя находится в левом положении, к общему центральному выводу подключается левый вывод, а когда в правом положении — правый вывод.

Рис. 6.41. Однополюсный двухпозиционный переключатель

Переключатель подключается в схему таким образом, чтобы его центральный вывод можно было подключать или к положительному питанию (+5 В), или к отрицательному («земля»). Вставьте переключатель в макетную плату, расположив каждый его вывод в своем ряду гнезд. В рассматриваемом примере переключатель установлен в верхней части макетной платы (рис. 6.42). Как и в случае с потенциометром, с помощью коротких проволочных перемычек подключите один крайний вывод переключателя к положительной шине питания макетной платы, а другой — к отрицательной. Центральный вывод переключателя подключите к гнезду вывода 12 платы Arduino. Монтаж завершенного проекта показан на рис. 6.43.

В зависимости от положения переключателя на его общем выводе появится или высокий, или низкий уровень сигнала, подаваемый на вывод 12 платы Arduino. В зависимости от этого уровня Arduino будет переключать управление балансирной балкой между ручным режимом (управлением игрой посредством потенциометра) и автоматическим, в котором вал сервомашинки станет вращаться вперед-назад самостоятельно.

В архиве ресурсов для этой книги (**https://www.nostarch.com/arduinoinventor/**) найдите скетч P6_AutoBalanceBeam.ino и загрузите его в Arduino. Потратьте несколько минут на ознакомление с имеющимся в нем комментарием, объясняющим его работу.

Помните, что если при включении автоматического режима балка не отцентрирована, мячик, скорее всего, с нее скатится. Может потребоваться несколько попыток, чтобы уловить правильный момент, но когда все получится, то результат будет выглядеть как какое-то волшебство!

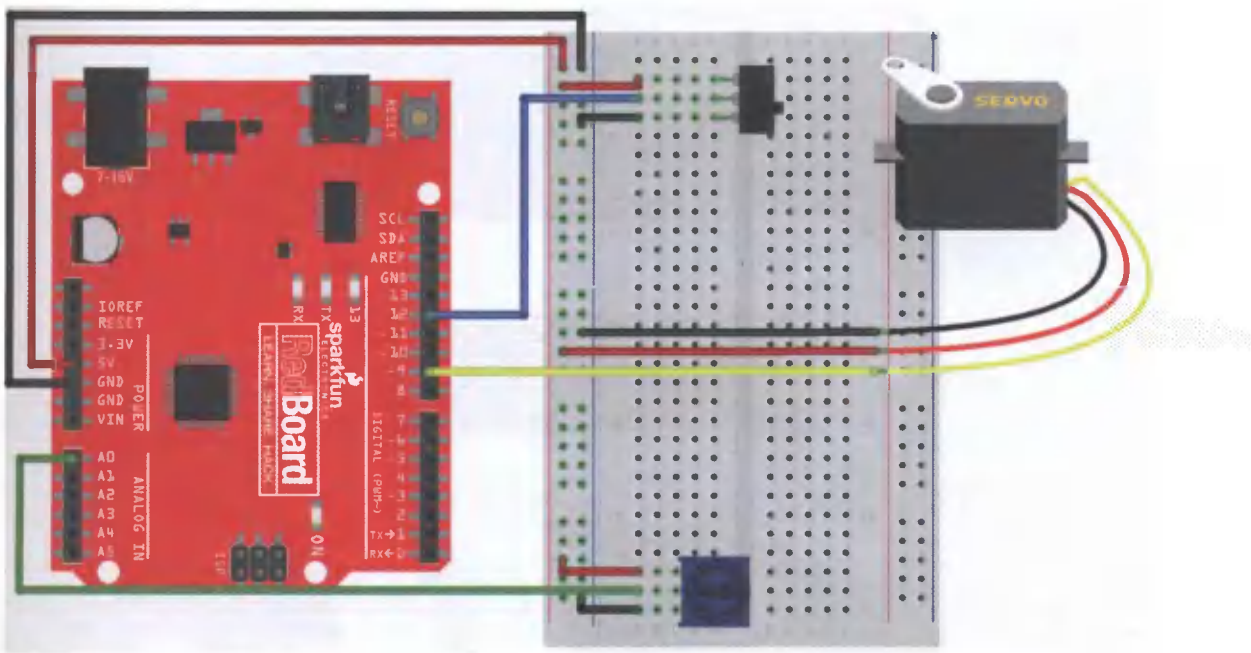


Рис. 6.42. Монтажная схема варианта проекта балансирной балки с добавленным в нее переключателем выбора режима

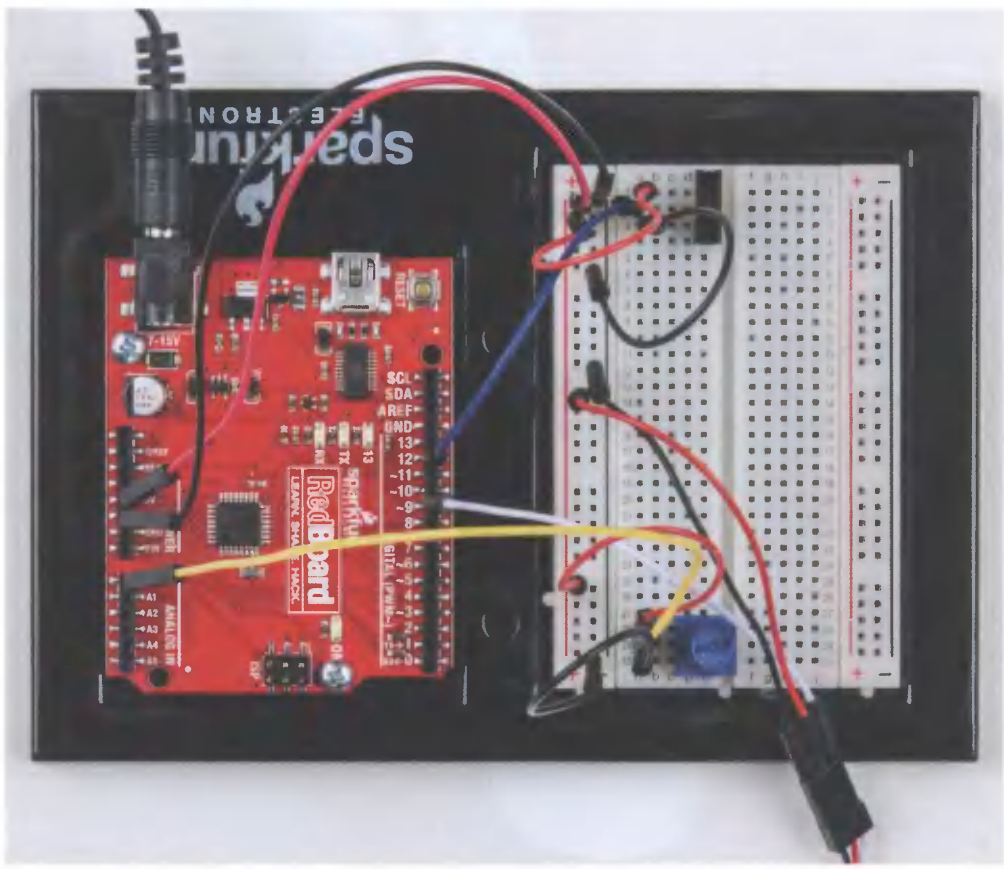


Рис. 6.43. Монтаж варианта проекта балансирной балки, способного переключаться в режим «автопилота»

МИНИАТЮРНАЯ НАСТОЛЬНАЯ ТЕПЛИЦА

Теплицы бывают разных форм и размеров — от маленьких комнатных тепличек из целлофана до больших промышленных теплиц площадью в тысячи квадратных метров. Но не у каждого есть средства и время на содержание большой теплицы, и для них мы соорудим в этом проекте миниатюрную тепличку, которую можно будет разместить на столе (рис. 7.1).

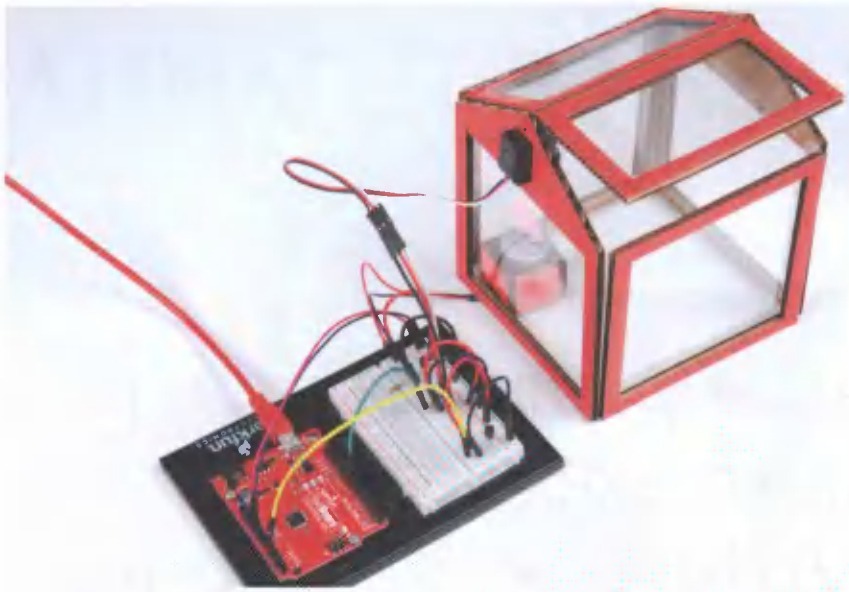


Рис. 7.1. Настольная тепличка

В обычных теплицах световая энергия проникает внутрь через прозрачные стеклянные или пластиковые панели, прогревая содержащийся там воздух. Но при определенных условиях температура в теплице может подняться выше допустимой. Чтобы с этим бороться, многие теплицы оборудуются вентиляторами и автоматически открываемыми окнами, чтобы проветривать теплицу и таким образом понижать температуру внутри нее.

Нашу тепличку мы также оборудуем подобным образом — создадим тепличный контроллер, отслеживающий температуру внутри теплички и, при ее повышении выше определенного уровня, открывающий окно и включающий вентилятор для проветривания и охлаждения теплички.

ТЕПЛИЧНЫЙ ЭФФЕКТ

Условия, позволяющие круглогодично выращивать овощи, создаются благодаря способности теплицы улавливать и накапливать энергию. Земная атмосфера тоже работает подобным образом. Лучи солнца сквозь нее нагревают земную поверхность, но излучаемое поверхностью тепло отражается атмосферой, не выпуская его обратно в космос. Это явление называется *тепличным эффектом*, и благодаря ему на Земле удерживается температура, позволяющая жить. Без этого эффекта температура на нашей планете была бы около –18 °C.

Другой пример тепличного эффекта можно наблюдать в автомобиле, оставленном летом под солнцем. При закрытых окнах температура внутри автомобиля может подняться на 5–15 °C выше, чем снаружи. Вот поэтому никогда нельзя оставлять домашних питомцев, не говоря уже о маленьких детях, в закрытой машине, особенно летом.

Необходимые компоненты, инструменты и материалы

Для открытия/закрытия окна в этом проекте мы задействуем сервомашинку наподобие той, что использовали в *проекте 6*, когда создавали игру в балансирную балку. Предстоит нам познакомиться и с тремя новыми компонентами: с электромотором постоянного тока (для вентилятора), с транзистором (для управления этим электромотором), а также с датчиком температуры (для определения температуры внутри теплички).

Два последних компонента внешне весьма похожи друг на друга: оба смонтированы в пластмассовых цилиндрических корпусах черного цвета с плоской гранью, и оба имеют по три вывода (рис. 7.2).

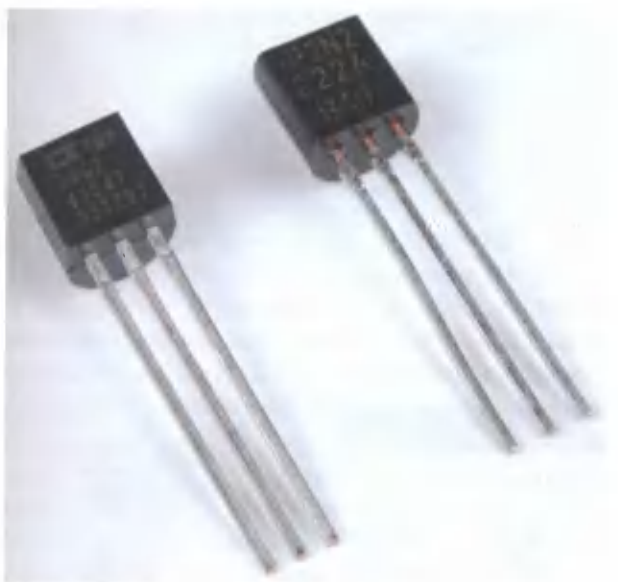


Рис. 7.2. Датчик температуры TMP36 (*слева*) и транзистор 2N2222 (*справа*)

Чтобы отличить эти компоненты друг от друга, наклоните корпус, чтобы свет падал на плоскую грань таким образом, чтобы можно было разобрать сделанную на ней надпись. Датчик температуры должен иметь маркировку **TMP**, сопровождаемую набором цифровых и буквенных обозначений, а транзистор — маркировку **2N2222** и также еще некоторые буквенные и цифровые обозначения после нее.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 7.3):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. ();
- беспаечная макетная плата (PRT-12002), 1 шт.;
- резисторы 330 Ом (COM-08377 или COM-11507 для пакета, содержащего 20 шт.), 3 шт.;
- диод (COM-08588), 1 шт.;
- NPN-транзистор 2N2222 или BC337 (COM-13689), 1 шт.;
- датчик температуры TMP36 (SEN-10988), 1 шт.;
- электромоторчик (ROB-11696), 1 шт.;
- миниатюрный серводвигатель (ROB-09065), 1 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*.

Примечание

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

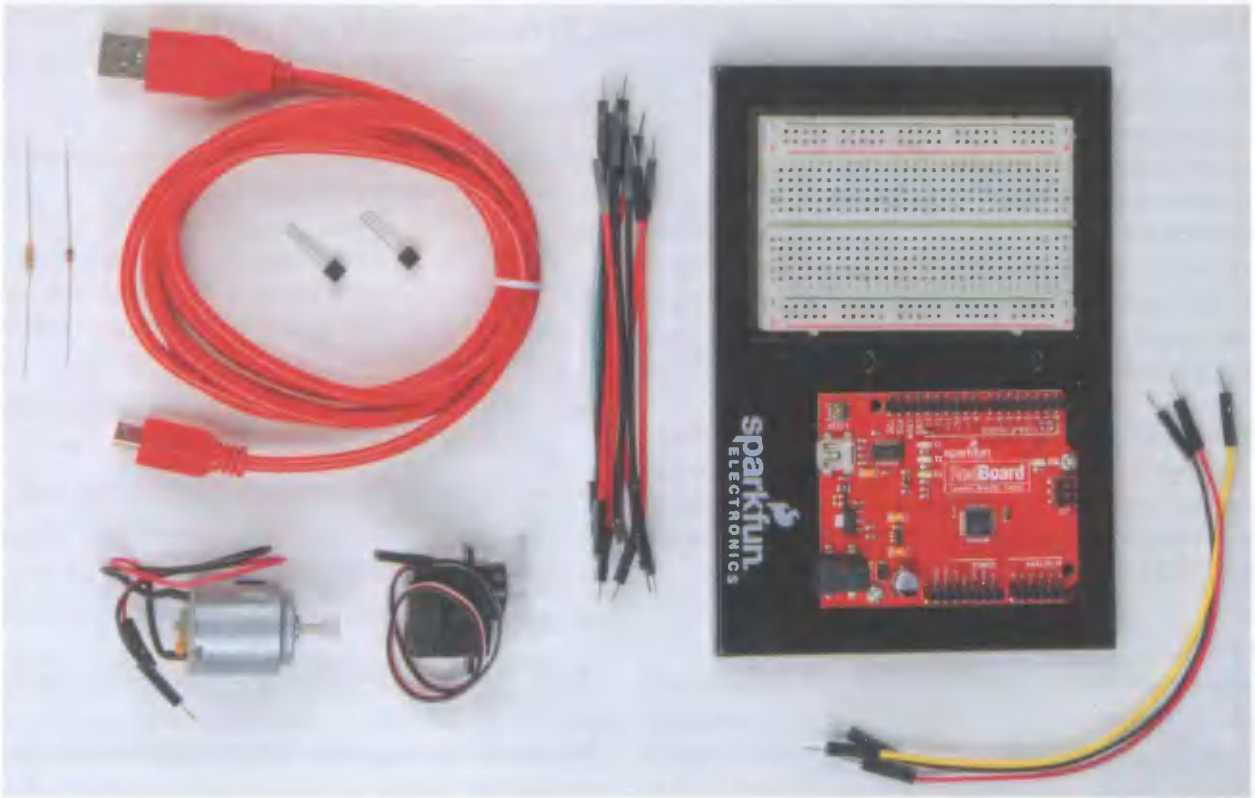


Рис. 7.3. Электронные компоненты для проекта настольной теплички

Рис. 7.4. Инструменты и материалы, рекомендуемые для проекта настольной теплички

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты и материалы (рис. 7.4):

- карандаш (на рис. 7.4 не показан);
- макетный нож;
- металлическая линейка;
- линейка;
- длинногубцы (игольчатые плоскогубцы);
- кусачки;
- клей (клеевой пистолет или клей для моделирования);
- клейкая лента типа «скотч» (на рис 7.4 не показана);
- картон: один лист размером примерно 30×45 см или три листа размером не менее 25×30 см;
- шаблон корпуса (см. рис. 7.18 далее в этом проекте);
- лист прозрачной пленки размером примерно 25×30 см (на рис. 7.4 не показан);
- канцелярская скрепка для бумаг (на рис. 7.4 не показана), 1 шт.

Новые компоненты

Давайте сначала рассмотрим новые компоненты, и начнем мы с датчика температуры.

Датчик температуры TMP36

Мы уже научились измерять интенсивность освещения. А теперь с помощью другого изящного электронного устройства мы сможем измерять и температуру. Термодатчик TMP36 представляет собой одно из самых легких в использовании устройств этого рода. Датчик заключен в пластмассовый цилиндрический корпус со срезанной гранью и имеет всего лишь три вывода. (Не забудьте посмотреть надпись на плоской грани, чтобы не спутать термодатчик с транзистором. Термодатчик должен иметь надпись TMP, а если там написано 2N2222 или надпись вовсе отсутствует, то это не тот компонент.)

Когда на термодатчик TMP36 должным образом подается питание, его выходное напряжение прямо пропорционально измеряемой температуре. Подобно тому, как мы использовали функцию analogRead() для измерения выходного напряжения фоторезистора в *проекте 5* или потенциометра в *проекте 6*, в этом проекте мы воспользуемся ею для измерения выходного напряжения термодатчика, чтобы впоследствии преобразовать это напряжение в соответствующую температуру.

Электромотор

Удалять горячий воздух из теплички нам поможет вентилятор, который мы соберем на электромоторчике постоянного тока для моделирования (рис. 7.5).

Рис. 7.5. Электромоторчик постоянного тока для моделирования

Это один из самых простых электромоторчиков: при подключении его к источнику питания он начинает вращаться в одну сторону, а при смене полярности питания — в противоположную. В отличие от сервомашинки из *проекта 6*, электромоторчик вращается постоянно. Для питания электромоторчика требуется напряжение постоянного тока в диапазоне 3–6 В, так что он идеально подходит для проектов с Arduino.

NPN-транзистор

Изобретение транзистора сделало возможным создание различного рода цифровых устройств. Например, микроконтроллер Arduino в действительности состоит из миллионов транзисторов. Транзисторы принадлежат к семейству электронных компонентов, называющихся *полупроводниковыми*. *Полупроводник* — это материал, в одних случаях обладающий свойствами проводника, позволяя протекание электрического тока, а в других — изолятора, не пропускающего электрический ток.

В этом проекте транзистор используется в качестве ключа, повышающего выходной ток Arduino. Для работы электромоторчика требуется ток питания величиной около 200–300 мА, однако выводы Arduino при работе в выходном режиме могут выдавать ток величиной только порядка 40 мА. В этой ситуации мы воспользуемся простой электронной схемой, которая поможет нам использовать низкий ток с вывода Arduino для управления транзистором, который и будет предоставлять ток требуемой для электромоторчика величины.

Применяем системный подход

Чтобы упростить организацию проекта, мы разобьем его на три отдельные части, или подсистемы. Такой способ, называющийся *системным подходом*, используется инженерами, чтобы разделить сложные проекты на управляемые части, каждую из которых можно собрать и протестировать независимо от других. Основными компонентами трех разных частей нашего проекта являются термодатчик, сервомашинка (для управления окном) и электромоторчик постоянного тока (для вентилятора). На рис. 7.6 показаны принципиальные схемы этих трех частей, а на рис. 7.7 — монтажная схема всего проекта.

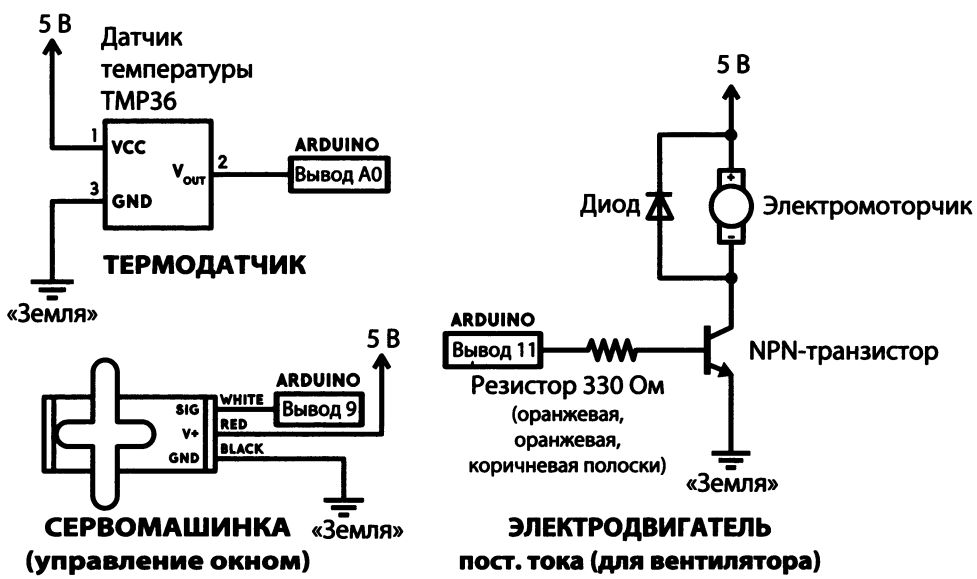


Рис. 7.6. Принципиальные схемы трех основных частей проекта: термодатчика (*вверху слева*), сервомашинки (*внизу слева*) и электродвигателя постоянного тока (*внизу справа*)

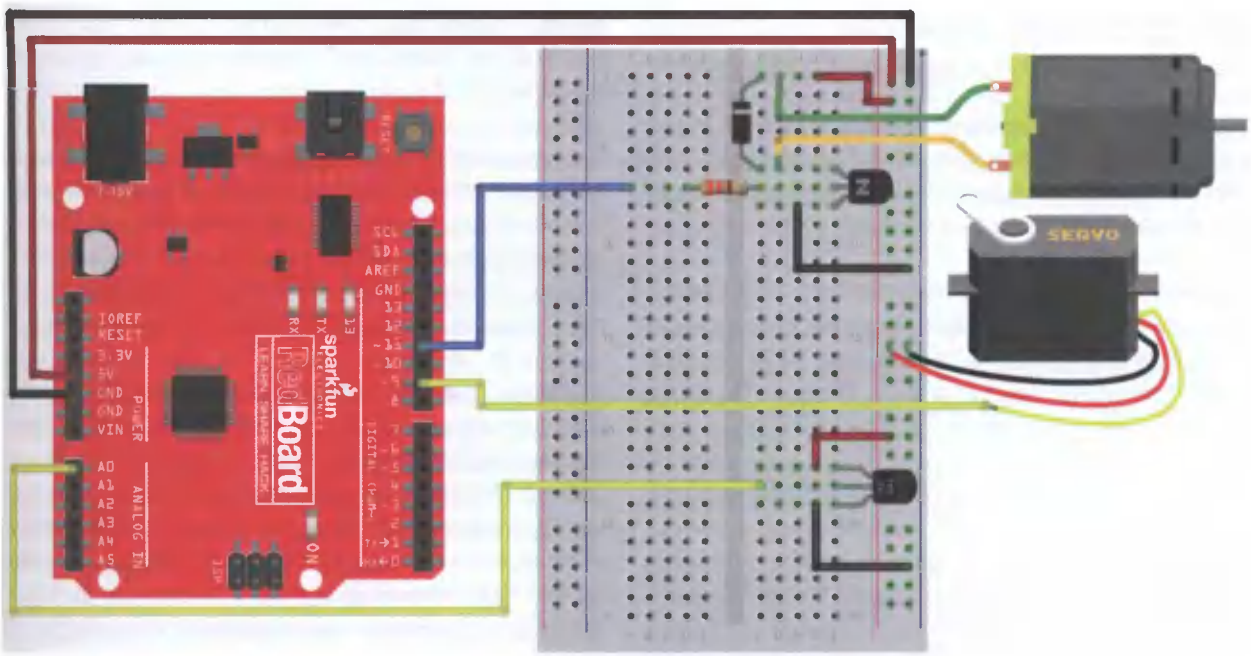


Рис. 7.7. Монтажная схема проекта

Собираем систему управления температурой

Сначала давайте рассмотрим часть системы управления тепличкой, которая ответственна за измерение температуры внутри нее. На рынке предлагается большой выбор датчиков температуры разных типов. Самыми распространенными, с которыми вам, возможно, придется иметь дело, являются *термисторы* и *термопары*. Первые изменяют свое сопротивление в зависимости от измеряемой температуры, а вторые выдают небольшое напряжение, порядка менее чем 10 мВ, поэтому для работы с ними требуется усилитель. Термодатчик TMP36 представляет собой третий тип датчиков температуры — он просто выдает напряжение, откалиброванное на 0,75 В для 25 °C. При отклонении измеряемой температуры от опорной в ту или иную сторону выдаваемое напряжение изменяется линейно. На рис. 7.8 приводится график выдаваемого термодатчиком TMP36 напряжения в зависимости от измеряемой температуры.

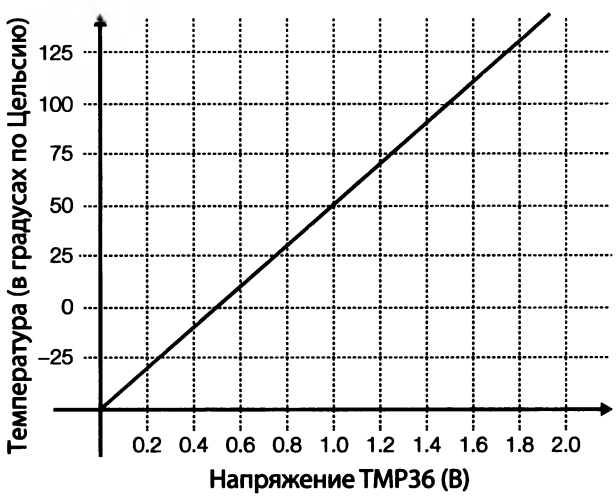


Рис. 7.8. График линейной зависимости выходного напряжения термодатчика TMP36 от измеряемой температуры

Измерение температуры с помощью термодатчика TMP36

Термодатчик TMP36 является одним из самых легких в использовании устройств этого рода — он заключен в пластмассовый цилиндрический корпус со срезанной гранью и имеет всего лишь три вывода.

Как уже упоминалось ранее, внешне термодатчик TMP36 очень похож на транзистор 2N2222, который также используется в этом проекте. Поэтому, прежде чем монтировать его в схему, убедитесь в том, что это именно термодатчик, а не транзистор. Для этого проверьте наличие маркировки TMP на грани устройства, наклонив его, если потребуется, должным образом, чтобы получить необходимое освещение грани. Если на грани написано 2N2222, или надпись отсутствует, или вообще там написано что-либо другое, то это не тот компонент.

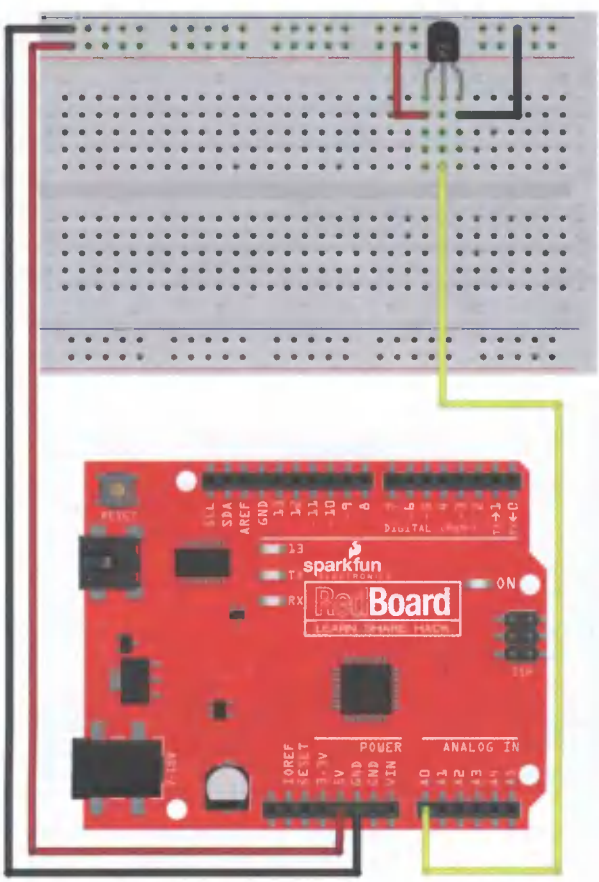


Рис. 7.9. Монтажная схема подключения термодатчика к плате Arduino

Термодатчик TMP36 выдает напряжение, величина которого прямо пропорциональна измеряемой температуре. Поскольку мы уже знаем, как считывать напряжение с помощью функции analogRead(), использование этого датчика не должно представлять для нас какой бы то ни было сложности.

Внешние выводы датчика служат для подачи на него питания, а центральный вывод выдает сигнальное напряжение. Чтобы использовать термодатчик TMP36, мы подаем питание на выводы питания, а сигнальный вывод подключаем к гнезду аналогового вывода Arduino. Но питание датчика нужно подключить правильно, поскольку выводы питания устройства имеют разную полярность.

Для определения полярности выводов питания термодатчика расположите его перед собой таким образом, чтобы концы выводов были направлены к вам, а срез на корпусе — направлен вверх. В таком положении левый вывод подключается к положительному питанию (+5 В), а правый — к отрицательному («земле»). При температуре 25 °C на сигнальном выводе должно присутствовать напряжение величиной 0,750 В (750 мВ). С изменением температуры напряжение на этом выводе будет изменяться в пропорции 0,010 В (10 мВ) на каждый 1 °C.

Не переживайте, если вам покажется, что здесь опять слишком много математики. Мы разберемся, как использовать эту информацию в коде, чтобы получить показания температуры в градусах, как по Цельсию, так и по Фаренгейту. Но сначала давайте подключим датчик к Arduino и запустим его в работу.

Подключаем датчик температуры

На рис. 7.9 показана монтажная схема подключения датчика температуры к плате Arduino. Большинство других компонентов полной схемы проекта будут размещаться с правой стороны

макетной платы, поэтому к гнездам питания платы Arduino подключите шины питания платы именно этой стороны. Затем вставьте термодатчик TMP36 в гнезда в нижней части макетной платы так, чтобы плоский срез корпуса датчика был направлен влево.

После чего возьмите две короткие монтажные перемычки и подключите верхний вывод термодатчика к положительной шине питания макетной платы, а нижний — к отрицательной. Опять же, убедитесь, что датчик вставлен в макетную плату плоским срезом корпуса влево. Средний вывод датчика — сигнальный, выдающий напряжение показаний температуры. Подключите его к гнезду вывода A0 на плате Arduino. Вот и весь монтаж термодатчика!

Теперь давайте взглянем на пример кода для получения показаний температуры с этого датчика.

Программируем снятие показаний датчика температуры

Как уже отмечалось, выходное сигнальное напряжение термодатчика TMP36 прямо пропорционально измеряемой температуре. Сопроводительная документация на термодатчик TMP36 предоставляет пару опорных точек для преобразования выходного напряжения датчика в показания температуры. В частности, там имеется информация, что выходное напряжение изменяется на величину 0,010 В на каждый 1 °C изменения температуры, и что при температуре 25 °C выходное напряжение датчика составляет 0,750 В. Этой информации достаточно, чтобы пересчитать выходное напряжение датчика при любой температуре в показания температуры в градусах по Цельсию.

Вспомним из *проекта 5*, что функция analogRead() возвращает целое число 1023 для считываемого напряжения величиной 5 В и целое число 0 для напряжения 0 В. Для наших целей нам нужно будет преобразовать возвращаемое функцией число в напряжение, затем преобразовать полученное напряжение в градусы по Цельсию и, наконец, преобразовать градусы по Цельсию в градусы по Фаренгейту. Чтобы не загромождать наш код всеми этими преобразованиями, мы сначала создадим пользовательскую функцию для преобразования возвращаемых функцией analogRead() целых чисел в соответствующие напряжения.

Создаем пользовательскую функцию преобразования

В листинге 7.1 приводится пример пользовательской функции volts() для преобразования возвращаемых функцией analogRead() целых чисел в соответствующие напряжения и возвращения полученных результатов.

Листинг 7.1. Пользовательская функция volts() для преобразования целочисленных эквивалентов напряжения в значения напряжения

```
❶ float ❷volts(❸int rawValue)
  {
❹   const float AREF = 5.0;
❺   float calculatedVolts;
❻   calculatedVolts = rawValue * AREF / 1023;
❼   return ❽calculatedVolts;
  }
```

Работая над *проектом 3*, мы узнали, как создавать пользовательские функции, чтобы сокращать объем кода скетча и не загромождать функцию loop(). В тех примерах для пользовательских функций всегда задавался тип данных void, поскольку они не возвращали никаких значений. Но в нашем случае — для пользовательской функции volts() — мы хотим, чтобы она возвращала значение вольт, полученное из преобразования целочисленного представления напряжения, поэтому нам нужно указать для нее иной тип возвращаемых данных. Так что укажем для функции volts() тип данных float ❶, поскольку мы хотим, чтобы возвращаемое функцией значение напряжения содержало как можно больше десятичных позиций — для получения большей точности. Назовем функцию volts ❷, чтобы имя было и описательное, и, в то

же самое время, короткое. Затем определим параметр ❸, который будет передаваться функции. В нашем примере — это исходное целочисленное представление напряжения, возвращаемое функцией analogRead().

Математика для преобразования целочисленного представления напряжения в реальное значение напряжения весьма простая, поскольку, как уже упоминалось ранее, мы знаем, что функция analogRead() возвращает целочисленное значение 1023 для входного напряжения 5 В и значение 0 для напряжения 0 В. Это означает, что целочисленное представление напряжения 1023 будет равно значению напряжения 5 В. Пользовательская функция volts() использует это отношение для преобразования целочисленных представлений напряжения, возвращаемых функцией analogRead(), в соответствующие значения напряжения.

Сначала объявим опорную переменную, которую назовем AREF ❹, и присвоим ее значение эталонного напряжения, которое равно 5,0 В. Поскольку это значение не будет изменяться нигде в коде, это будет даже не переменная, а константа, на что указывает ключевое слово const.

> **Примечание**
>
> По общепринятой практике имена констант указываются ПОЛНОСТЬЮ ЗАГЛАВНЫМИ БУКВАМИ.

Далее определяем переменную для хранения результатов преобразования, назвав ее calculatedVolts ❺. Обратите внимание, что для этой переменной также указан тип данных float. Это делается с тем, чтобы результаты наших математических вычислений были более точными, чем можно выразить целыми числами. Для вычисления напряжения просто умножаем значение rawCount на отношение AREF (5,0 В) к 1023 ❻.

Команда return ❼ в предыдущих проектах не использовалась. Смысл ее в том, что когда исполнение скетча доходит до инструкции return, управление исполнением скетча выходит из функции volts() и возвращается в точку, из которой был осуществлен вызов этой функции. Если после инструкции return следует значение, которое может быть переменной, как здесь, функция возвращает это значение при возвращении управления в точку ее вызова. Тип данных возвращаемого функцией значения должен быть таким же, как и тип данных самой функции. Во всех ранее используемых функциях инструкция return не употреблялась, поскольку эти функции не возвращали никаких значений, что указывалось их типом данных void. В нашем же примере за инструкцией return следует переменная calculatedVolts ❽. Таким образом, при возвращении управления исполнением скетча обратно в точку вызова функции возвращается и значение calculatedVolts.

Следует заметить, что инструкция return может использоваться также функциями с типом данных void и без возвращаемого значения. В таких случаях функции просто дается указание прекратить исполнение и возвратить управление в точку ее вызова. Пример такого использования инструкции return показан в листинге 7.2.

Листинг 7.2. Пользовательская функция с типом данных void и инструкцией return

```
void blink()
{
   digitalWrite(13, HIGH);
   delay(500);
   digitalWrite(13, LOW);
   delay(500);
   return;
}
```

Тестируем функцию *volts()*

Теперь давайте протестируем нашу пользовательскую функцию volts() в специальном скетче. Для этого расширим код из листинга 7.1 функциями setup() и loop() с кодом для считывания напряжения на выводе A0 и вывода полученных данных в окно монитора порта. Полный код скетча тестирования приводится в листинге 7.3.

Листинг 7.3. Скетч для тестирования функции преобразования целочисленных представлений напряжения в значения напряжения

```
//Скетч для тестирования — считывает аналоговое
//значение на выводе A0 и выводит целочисленное
//представление напряжения и его соответствующее
//значение в вольтах в окне монитора порта.
int rawSensorValue;
float rawVolts;

void setup()
{
  Serial.begin(9600);  //Инициализируем
                       //последовательную связь
  Serial.print("raw");
  Serial.print("\t");       //Символ табуляции
  Serial.print("volts");
  Serial.println();        //Символ новой строки
}

void loop()
{
  rawSensorValue = analogRead(A0); //Считываем
                                   //показание датчика
  rawVolts = volts(rawSensorValue);  //Преобразовываем
                                   //полученное целочисленное
                                   //значение напряжения в вольты
  Serial.print(rawSensorValue); //Выводим на экран
                                //значение датчика
  Serial.print("\t");
  Serial.print(rawVolts);    //Выводим на экран
                             //соответствующее значение в вольтах
  Serial.println();          //Символ новой строки
}
/*********************************************/
float volts(int rawCount)
{
  float AREF = 5.0;
  float calculatedVolts;
  calculatedVolts = rawCount * AREF / 1023;
  return calculatedVolts;
}
```

Подключите плату Arduino к компьютеру, загрузите в нее этот скетч, а затем откройте окно монитора порта (**Serial Monitor**). В окне монитора порта должен непрерывно выводиться текст наподобие показанного на рис. 7.10.

Согласно выводимым данным, целочисленному представлению напряжения 156 соответствует значение напряжения 0,76 В. Эти данные можно проверить простым вычислением: 156 × (5,0 / 1023) = 0,762 В. С математикой не поспоришь!

Без использования команды delay() в функции loop() Arduino считывает показания датчика примерно 80–90 раз в секунду, отправляя полученные данные на компьютер. Это объясняет вывод непрерывного потока показаний вместо вывода показаний через какой-либо интервал времени. Чтобы замедлить скорость вывода данных на экран, просто добавьте инструкцию delay(1000); в конце кода в функции loop(), сразу же после последней инструкции Serial.println();. Это замедлит вывод данных до одного раза в секунду, что гораздо практичнее. Эта модификация будет добавлена в код позже.

А пока нам нужно сделать еще один шаг, чтобы отображать температуру, а не вольты.

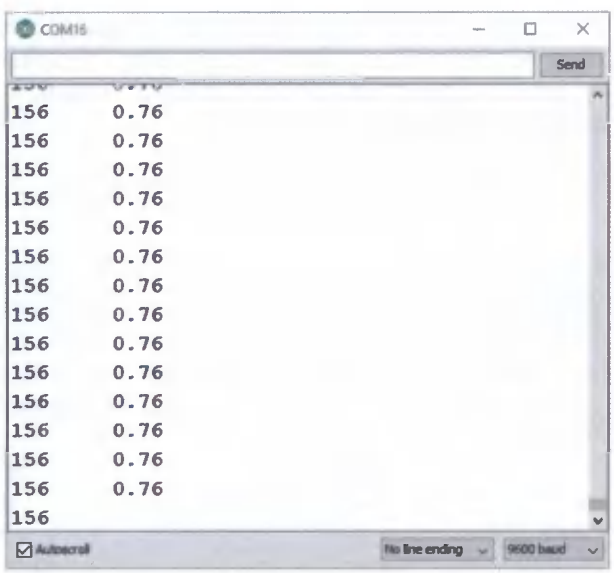


Рис. 7.10. Вывод целочисленного представления напряжения и соответствующего значения в вольтах в окне монитора порта

Преобразовываем вольты в градусы

Теперь нам требуется формула для преобразования вольт в градусы. Мы знаем, что выходное напряжение термодатчика TMP36 изменяется прямо пропорционально измеряемой температуре. Поэтому для создания формулы пересчета мы используем уравнение прямой с угловым коэффициентом:

$$y = mx + b$$

Это уравнение описывает прямую линию или, в более общем смысле, взаимоотношение между двумя переменными *x* и *y*, где *m* обозначает крутизну линии (то есть величину изменения *y* относительно изменения *x*), а *b* обозначает точку, в которой линия пересекает ось *y*. В нашем случае переменные *x* и *y* представляют напряжение и температуру соответственно, и мы используем известную величину *x* (напряжение в вольтах) для вычисления неизвестной величины *y* (температура в градусах).

Как уже упоминалось ранее, в документации на термодатчик TMP36 указывается, что выходное напряжение датчика изменяется на величину 0,010 В на каждый 1 °C изменения температуры, или на 1 В на каждые 100 °C. Эта величина изменения напряжения в результате изменения температуры определяет *крутизну линии* графика уравнения, то есть его величину *m*. Заменив *x* и *y* соответствующими переменными, получим следующее уравнение:

$$\text{температура} = \frac{100\ °C}{1\ В} \times \text{напряжение} + b$$

В документации на термодатчик TMP36 также указывается, что при температуре 25 °C выходное напряжение датчика составляет 0,750 В. Подставляем эти данные в уравнение, чтобы вычислить значение *b* — точки, в которой прямая пересекает ось *y*:

$$25\ °C = \frac{100\ °C}{1\ В} \times 0{,}750\ В + b$$

$$25\ °C = 75\ °C + b$$

$$b = -50\ °C$$

Имея значения для *m* и *b*, конечную версию уравнения можно представить в следующем виде:

$$\text{температура} = \frac{100\ °C}{1\ В} \times \text{напряжение} - 50\ °C$$

Такой подход можно применять к разным датчикам, чье выходное напряжение меняется линейно при изменениях измеряемой величины. Это также еще одно напоминание о важности и полезности математики. Но если вы предпочитаете не вдаваться во все эти математические тонкости, вполне можно обойтись и без них. Все что нужно знать, чтобы определить температуру по показаниям датчика, — это просто вставить выходное напряжение датчика в это уравнение.

Теперь используем это уравнение в коде, чтобы преобразовать значение вольт rawVolts в температуру. Необходимый код для преобразования напряжения в температуру добавим в код листинга 7.3. Получившийся скетч показан в листинге 7.4, добавленный в него код выделен полужирным шрифтом.

Листинг 7.4. Преобразовываем вольты в градусы

```
//Скетч для тестирования — считывает аналоговое
//показание термодатчика TMP36 на выводе A0,
//вычисляет по нему температуру и выводит
//целочисленное представление напряжения и его
//соответствующее значение в вольтах и температуры
//в окне монитора порта.

int rawSensorValue;
float rawVolts;
❶ float tempC;
  float tempF;

  void setup()
  {
  Serial.begin(9600);   //Инициализируем
                        //последовательную связь
  Serial.print("raw");
  Serial.print("\t");    //Символ табуляции
  Serial.print("volts");
```

```
❷ Serial.print("\t");
   Serial.print("degC");
   Serial.print("\t");
   Serial.print("degF");
   Serial.println();
}

void loop()
{
   rawSensorValue = analogRead(A0); //Считываем
                                    //показание датчика
   rawVolts = volts(rawSensorValue); //Преобразовываем
              //полученное целочисленное значение
              //напряжения в вольты
❸ tempC = 100 * rawVolts - 50; //Преобразовываем
                               //вольты в °C
❹ tempF = 1.8 * tempC + 32;    //преобразовываем °C в °F
   --пропущено для краткости--
   Serial.print(rawVolts);          //Выводим на экран
              //соответствующее значение в вольтах
❺ Serial.print("\t");
   Serial.print(tempC);
   Serial.print("\t");
   Serial.print(tempF);
   Serial.println(); //Символ новой строки
❻ delay(1000);
   }
   /*****************************************/
   float volts(int rawCount)
   --пропущено для краткости--
```

Здесь мы сначала добавляем в глобальном пространстве имен в верхней части скетча две строки кода, в которых объявляем две переменные для хранения значений температуры в градусах по Цельсию (tempC) и в градусах по Фаренгейту (tempF) ❶. Затем вставляем в функцию setup() код для отображения названий заголовков столбцов данных, выводимых в окно монитора порта, разделенных символом табуляции, который представлен управляющим символом \t ❷. Теперь можно использовать уравнение прямой с угловым коэффициентом для вычисления температуры в градусах по Цельсию ❸. Добавляем также строку кода для преобразования градусов по Цельсию в градусы по Фаренгейту ❹.

Наконец, для дополнительной обратной связи добавляем несколько команд Serial.print() для вывода новых значений в окно монитора порта ❺. Обратите внимание, что последняя команда здесь — команда Serial.println(), которая завершает вывод символом новой строки, переводя курсор на новую строку. Таким образом, следующие показания будут выводиться в новой строке. Наконец, добавляем код для задержки в 1 секунду ❻, чтобы считывать показания датчика только раз в секунду и иметь достаточно времени для чтения выводимых данных.

Загрузите модифицированный код в Arduino и откройте окно монитора порта (**Serial Monitor**). В окне монитора порта должен непрерывно выводиться текст наподобие показанного на рис. 7.11.

Как можно видеть, в дополнение к данным о целочисленном представлении выходного напряжения термодатчика (raw) и соответствующим им значениям напряжения (volts), выводятся также соответствующие значения температуры в градусах по Цельсию (tempC) и по Фаренгейту (tempF). Если удерживать датчик температуры пальцами, его температура должна повыситься, что отобразится в выводимых данных. Поздравляем! У нас теперь есть рабочий монитор температуры — первая большая часть этого проекта.

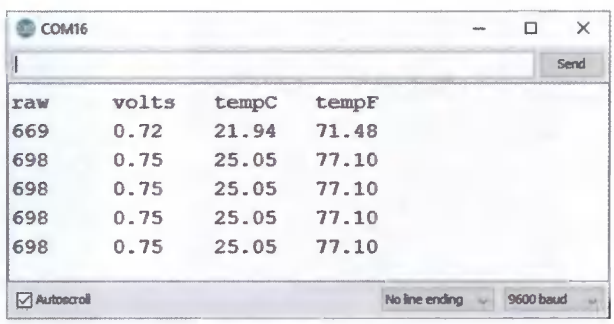


Рис. 7.11. Вывод в окно монитора порта данных замеров температуры, полученных из показаний термодатчика TMP36

Собираем схему сервомашинки для управления окном

Для открытия и закрытия окна теплички мы применим сервомашинку наподобие той, которую использовали в *проекте 6*. Сервомашинка представляет собой простой электродвигатель с двумя проводами питания (красным — для положительного и черным — для отрицательного) и одним проводом управления (желтым или белым).

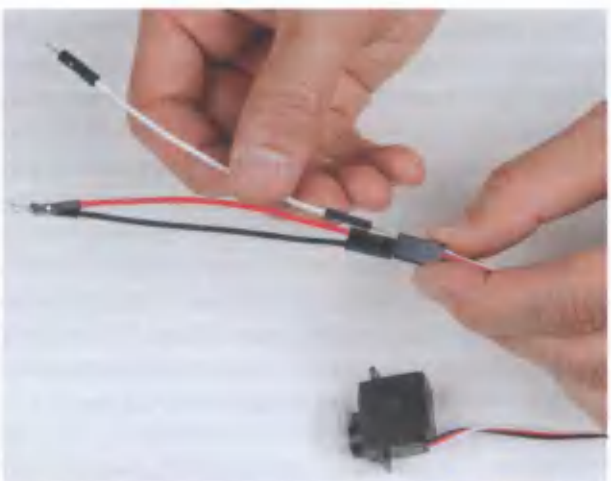


Рис. 7.12. Подсоединение проволочных перемычек со штыревыми разъемами к разъему сервомашинки

Большинство стандартных сервомашинок имеют трехконтактный гнездовой разъем. Поэтому, чтобы подключить сервомашинку к схеме, возьмите три проволочные перемычки со штыревыми разъемами на обоих концах и вставьте их одним концом в разъем сервомашинки, как показано на рис. 7.12. По мере возможности постарайтесь использовать перемычки такого же цвета, как и провода разъема сервомашинки. Затем подключите сигнальный вывод сервомашинки (желтый или белый провод) к гнезду вывода 9 платы Arduino, вывод положительного питания сервомашинки (красный провод) подключите к шине положительного питания макетной платы, а вывод отрицательного (черный провод) — к шине отрицательного питания («земле»).

Подключение сервомашинки к макетной плате показано на рис. 7.13.

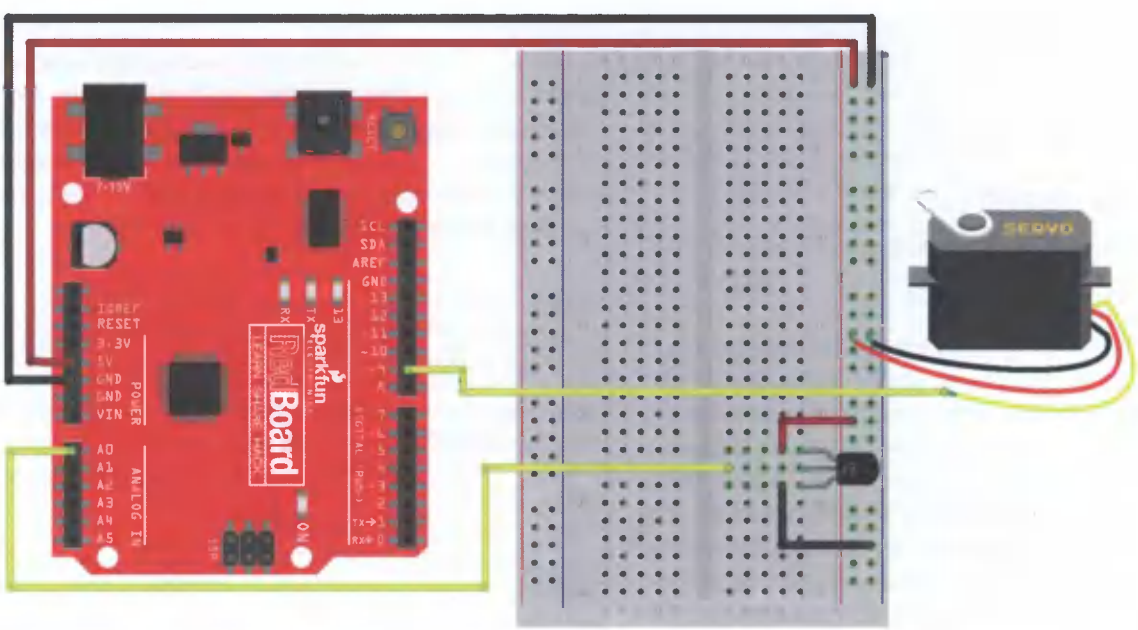


Рис. 7.13. Подключение сервомашинки к схеме на макетной плате

Разрабатываем код для управления сервомашинкой

Добавляем код, выделенный в листинге 7.5 полужирным шрифтом, в код листинга 7.3.

Листинг 7.5. Код для управления сервомашинкой

```
❶ #include<Servo.h>
❷ Servo myServo;

   // Скетч для тестирования — считывает аналоговое
   //показание термодатчика TMP36 на выводе A0,
   //вычисляет по нему температуру и выводит
   //целочисленное представление напряжения и его
   //соответствующее значение напряжения
   //и температуры в окне монитора порта.

   int rawSensorValue;
   float rawVolts;
   float tempC;
   float tempF;
❸ int setPoint = 85;
   int returnPoint = 83;

   void setup()
   {
   myServo.attach(❹9, 1000, 2000); //Инициализируем
                                   //объект myServo
   Serial.begin(9600);        //Инициализируем
                              //последовательную связь
   --пропущено для краткости--
   }

   void loop()
   {
   --пропущено для краткости--
   Serial.println();          //Символ новой строки

❺ if(tempF > setPoint)
   {
   myServo.write(180);
   }
   else if(tempF < returnPoint)
   {
   myServo.write(0);
   }
   delay(1000);
}

/*********************************************/
   float volts(int rawCount)
   --пропущено для краткости--
```

Вспомним из *проекта 6*, что для работы с сервомашинкой нам нужно включить в состав скетча библиотеку Servo ❶, а также создать объект типа Servo, которому мы здесь присвоили имя myServo ❷. Далее создаем две переменные ❸, определяющие контрольные точки для системы управления. Контрольные точки определяют температуры, при которых следует открывать и закрывать окно.

Обратите внимание, что значение setPoint, которое задает температуру для открытия окна, на 2 градуса выше, чем значение returnPoint, задающее температуру для закрытия окна. Это дает нам диапазон в 2 градуса, в котором окно не будет ни закрываться, ни открываться. Этот метод управления называется *гистерезисом*. Он применяется в системах с небольшими температурными колебаниями для того, чтобы предотвратить постоянное открытие и закрытие окна при малейшем изменении температуры.

Затем нам нужно инициализировать сервомашинку, сообщив Arduino, что она подключена к его выводу 9 ❹. Обратите внимание, что эта команда: myServo.attach(9, 1000, 2000); — содержит два дополнительных параметра, а не один, как в *проекте 6*. Этот момент разъясняется далее, во *врезке «Принцип работы сервоприводов в подробностях»*.

Наконец, реализуем логику управления, используя оператор if...else if. Это восемь строк кода ❺ над командой delay(). Этот код поворачивает вал сервомашинки в положение 180 градусов, если температура поднимается выше 85 °F (28,44 °C) (точка setPoint), и возвращает его обратно в положение 0 градусов, если температура опускается ниже 83 °F (точка returnPoint). Вал сервомашинки, в свою очередь, через шарнирную систему открывает и закрывает окно теплички.

Загрузите модифицированный код из листинга 7.5 в Arduino и откройте окно монитора порта (Serial Monitor). Проверьте работу схему, слегка сжав термодатчик пальцами или прикрыв его ладонями и подышав на него, чтобы прогреть его выше 85 °F. Наблюдайте при этом на мониторе порта, как поднимается температура. Как только она достигнет 85 °F, сервомашинка должна включиться и провернуть свой вал в позицию 180 градусов. Теперь дайте датчику остыть. Как только температура опустится ниже 83 °F (значение returnPoint), сервомашинка должна включиться снова и провернуть вал в исходную позицию 0 градусов. Если все работает должным образом, можно приступать к работе над последней составляющей системы — вентилятором.

ПРИНЦИП РАБОТЫ СЕРВОПРИВОДОВ В ПОДРОБНОСТЯХ

Сервопривод представляет собой электродвигатель, вал которого проворачивается в определенное положение под управлением сигнала от микроконтроллера. Рабочий диапазон перемещения вала сервоприводов составляет порядка 180 градусов.

Все сервоприводы имеют три провода: белый (иногда желтый или оранжевый) — для приема управляющего сигнала, красный — для положительного питания и черный — для отрицательного («земли»). Сигнал управления сервоприводом представляет собой последовательность импульсов с периодом 20 мс, или частотой 50 Гц. Вращение вала сервопривода в определенное положение достигается варьированием длительности импульса. (Дополнительную информацию о широтно-импульсной модуляции см. в *проекте 5*.) Для большинства стандартных сервоприводов сигнал импульсов длительностью 1000 микросекунд (1000 мкс) выставляет вал сервопривода в положение 0 градусов (исходное положение), а длительностью 2000 мкс — в положение полного хода 180 градусов. Таким образом, сигнал импульсов длительностью 1500 мкс выставит вал сервопривода в среднее положение — 90 градусов.

Библиотека Arduino Servo сопоставляет длительность импульсов сигнала положению вала сервопривода, но в ней используются слегка другие определения для положения вала и длительности импульсов. В частности, сигнал импульсов длительностью 544 мкс соответствует положению вала 0 градусов, а длительностью 2400 мкс — положению 180 градусов. Такой диапазон длительности импульсов управляющего сигнала позволяет использовать библиотеку для работы с сервоприводами с расширенным рабочим диапазоном хода вала, но он выходит за пределы возможностей большинства стандартных сервоприводов. При этом, команда `myServo.write(0);` будет подавать на сервопривод сигнал с импульсами длительностью 544 мкс, который станет заставлять сервопривод провернуть свой вал за пределы своих физических возможностей. Когда это происходит, сервопривод начинает трястись, жужжать и греться, поскольку он просто физически не в силах провернуть вал в положение, соответствующее импульсам длительностью 544 мкс.

Чтобы избежать такого развития событий, в коде можно установить пределы длительности управляющих импульсов. Для этого в параметры команды инициализации вывода Arduino для управления сервоприводом, кроме номера вывода, добавляются параметры пределов длительности импульсов управляющего сигнала: `myServo.attach(9, 1000, 2000);`. После такой инициализации команда `myServo.write(0);` провернет вал сервопривода в положение 0 градусов без побочных эффектов тряски, жужжания и нагрева.

Дополнительную информацию о принципах работы сервоприводов можно посмотреть в соответствующем учебном пособии на веб-сайте компании SparkFun по адресу: **https://www.sparkfun.com/tutorials/283/**.

Собираем схему для управления электродвигателем вентилятора

Вентилятор для проветривания теплички должен приводиться в действие небольшим электродвигателем постоянного тока — стандартным для любительских проектов устройством с двумя проводами для подключения питания и валом, который при подключении питания вращается. Сервоприводы, которые мы использовали до сих пор, содержат внутри корпуса шестеренчатый редуктор, что позволяет им выполнять точное позиционирование вала в определенном диапазоне перемещений. Вспомните, что вал сервомашинки может поворачиваться только в пределах 180 градусов. Для вентилятора же нам нужен электродвигатель, который будет вращаться все время, пока подается питание.

Наш электродвигатель рассчитан на работу от напряжения питания величиной 3–6 В и при работе он потребляет около 200–300 мА тока. Однако выводы платы Arduino могут выдавать ток силой только порядка 40 мА. Поэтому, чтобы обеспечить для питания электродвигателя ток достаточной силы, мы соберем дополнительную схему, называющуюся *транзисторным усилителем*, которая в технических кругах более известна под названием *усилитель с общим эмиттером*.

Транзистор имеет три вывода: *коллектор* (К), *базу* (Б) и *эмиттер* (Э). Ток, входящий в базу, усиливается на коллекторе. Базу транзистора можно сравнить с управлением заслонкой, которая позволяет току протекать от коллектора к эмиттеру. Чем сильнее мы нажимаем на базу, тем шире открывается заслонка и тем больше тока может протечь от коллектора к эмиттеру. Транзисторы замечательны тем, что небольшой ток на базе вызывает протекание большого тока от коллектора к эмиттеру. Но подача слишком большого тока на базу может сжечь транзистор, поэтому, подобно использованию токоограничивающих резисторов со светодиодами, мы также применим здесь резистор сопротивлением 330 Ом, чтобы ограничить ток на базе транзистора.

Транзисторы включаются в схему многими разными способами, и одним из самых распространенных является использование их в качестве усилителя напряжения. Но в данном случае мы задействуем транзистор в качестве управляемого выключателя (ключа). Для этого мы воспользуемся свойством транзистора, которое заключается в том, что подача на его базу сигнала высокого уровня как бы «включает» транзистор, в результате чего между коллектором и эмиттером сразу начинает протекать ток. Это, по сути, равнозначно соединению этих выводов контактами выключателя.

Вывод эмиттера транзистора подключается к шине отрицательного питания («земле») макетной платы. При этом один вывод питания электродвигателя подключается к шине положительного питания (+5 В) макетной платы, а другой — к выводу коллектора транзистора. Когда транзистор «включен», это соединяет коллектор транзистора с его эмиттером, замыкая цепь питания электродвигателя и вызывая его вращение. Такой режим работы транзистора инженеры называют переключением транзистора между режимами запирания и насыщения. При этом электропитание на электродвигатель поступает непосредственно из шин питания макетной платы. Это означает, что мы можем использовать малый ток на выводах Arduino для управления устройствами, требующими ток намного большей силы.

Транзисторный выключатель для электродвигателя собирается в верхней части беспаечной макетной платы (рис. 7.14).

Найдите требуемый транзистор в своем наборе электронных компонентов. Как можно видеть на рис. 7.15, транзистор выглядит подобно датчику температуры. Различие состоит в том, что транзистор обозначен на плоской грани корпуса надписью 2N2222 или BC337. Мы воспользуемся этим NPN-транзистором как выключателем, управляемым посредством Arduino.

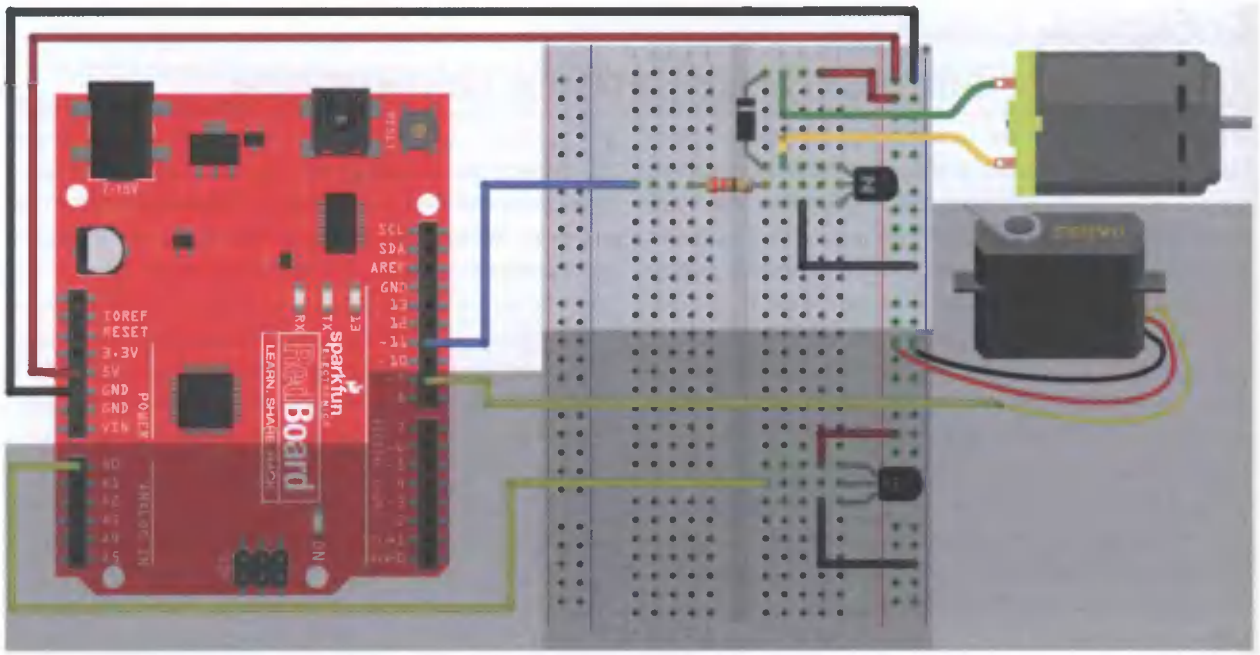


Рис. 7.14. Сборка транзисторного выключателя для электродвигателя вентилятора

Расположите транзистор выводами от себя так, чтобы его плоская сторона смотрела влево, и вставьте его в гнезда верхней части макетной платы так, чтобы его верхний вывод попал приблизительно в шестой ряд гнезд от верхнего края платы. В этом положении верхний вывод будет коллектором, средний — базой, а нижний — эмиттером (см. рис. 7.15).

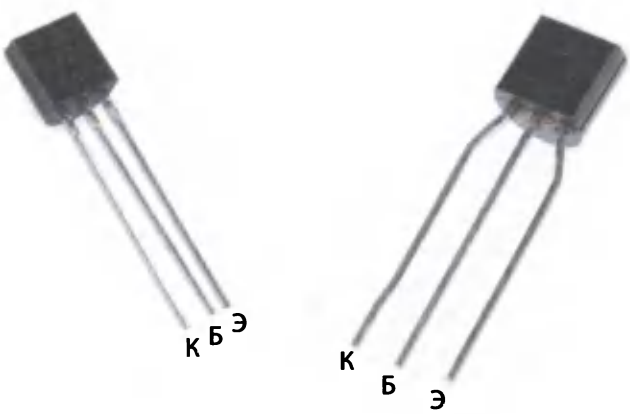


Рис. 7.15. NPN-транзистор 2N2222, используемый в этом проекте (*слева*) и эквивалентный ему при замене транзистор BC337 (*справа*)

Подсоедините резистор сопротивлением 330 Ом одним выводом к выводу базы, а второй его вывод вставьте в гнездо в противоположном ряду на другой стороне платы — через углубление в ее середине — как показано на рис. 7.14. Затем подключите этот вывод резистора к гнезду вывода 11 платы Arduino. Таким образом, вывод 11 платы Arduino будет подключен к базе транзистора через токоограничивающий резистор сопротивлением 330 Ом. Этот вывод послужит источником маломощного сигнала от Arduino, который станет включать и выключать транзистор. Включенный транзистор будет замыкать цепь питания электродвигателя, вызывая его вращение.

С помощью короткой перемычки подключите вывод эмиттера транзистора (нижний вывод) к шине отрицательного питания («земле») макетной платы. Наконец, подключите один из проводов питания электродвигателя к выводу коллектора (верхний вывод) транзистора. В зависимости от того, какой провод питания электродвигателя мы задействуем, двигатель будет вращаться по часовой стрелке или против нее. Но в данном случае направление вращения электродвигателя не имеет

для нас важности, поэтому к коллектору можно подключить любой провод питания электродвигателя. Другой провод питания электродвигателя подключите к шине положительного питания макетной платы, которую затем с помощью проволочной перемычки подключите к гнезду 5 В (5V) на плате Arduino. Подача на базу слаботочного сигнала замкнет цепь коллектор-эмиттер. Это, в свою очередь, замкнет цепь от 5 В через электродвигатель на «землю», в результате чего через обмотку электродвигателя начнет протекать ток, и он начнет вращаться. Принципиальная схема подключения электродвигателя и транзистора для его управления показана на рис. 7.16.

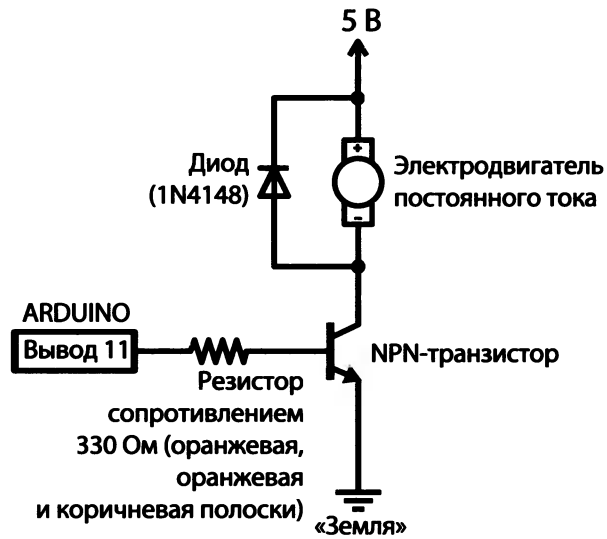


Рис. 7.16. Принципиальная схема подключения электродвигателя и транзистора для управления им

Последним компонентом, который нам нужно подключить в схему, является диод, который в данной ситуации называется *диодом обратного хода тока*. Назначением этого диода является защита транзистора от возможного повреждения током электромотора. Дело в том, что внутри электродвигателя находятся проволочные катушки, и при протекании тока через эти катушки они ведут себя как электромагниты, которые воздействуют на постоянные магниты ротора двигателя, заставляя его вал вращаться. Катушки являются очень интересным электронным компонентом. Создаваемое ими магнитное поле в действительности представляет собой форму накопленной энергии, которая при разрыве электрической цепи кратковременно возвращается в цепь в виде всплеска высокого напряжения. Этот всплеск может повредить транзистор, весьма чувствительный к такого рода явлениям. Диод обратного хода тока создает для этого напряжения путь рассеивания, проводя его мимо транзистора. Эта схема подключения диода иногда называется *демпферной схемой*, а сам диод *демпферным*.

Нам нужно иметь в виду, что выводы диода имеют полярность, и ориентация выводов при подключении диода весьма важна. Полярность выводов диода обозначается полоской на одной из сторон его корпуса (рис. 7.17). Этим выводом диод следует подключить к выводу электродвигателя, подключенному к положительному питанию (5 В).

Подключите диод в схему, обращая при этом внимание, чтобы обозначенный полоской вывод был подсоединен к проводу положительного питания электродвигателя, как показано на рис. 7.16. Другой вывод диода подключается ко второму, отрицательному, выводу питания электродвигателя — это вывод, подключенный к коллектору транзистора. Таким образом, выводы диода оказываются подключенными к выводам электродвигателя. Подключение устройств подобным способом называется *параллельным подключением*.

Итак, у нас есть завершенная система, состоящая из трех отдельных схем, показанных на рис. 7.6. Нам только осталось добавить в скетч несколько строк кода для управления электродвигателем.

Рис. 7.17. Диод обратного хода тока для защиты транзистора, показанный с четвертаком[1] для сравнения размеров

[1] Монета США достоинством 25 центов (четверть доллара).

Разрабатываем код для управления электродвигателем вентилятора

Код для управления электродвигателем не содержит ничего сложного, будучи всего лишь версией кода для включения и выключения светодиода. Добавьте в свой скетч код, выделенный жирным шрифтом в листинге 7.6.

Листинг 7.6. Полный код для управления тепличкой

```
#include<Servo.h>
Servo myServo;
--пропущено для краткости--

void setup()
{
❶   pinMode(11, OUTPUT);
    myServo.attach(9, 1000, 2000);
    Serial.begin(9600);     //Инициализируем
                            //последовательную связь
    --пропущено для краткости--
}

void loop()
{
    --пропущено для краткости--
    Serial.println();       //Символ новой строки
    if(tempF > setPoint)
    {
        myServo.write(180);
❷       digitalWrite(11, HIGH);  //включаем вентилятор
    }
    else if(tempF < returnPoint)
    {
        myServo.write(0);
❸       digitalWrite(11, LOW);   //выключаем вентилятор
    }
    delay(1000);
}
/*********************************************/
float volts(int rawCount)
{
    const float AREF = 5.0;
    float calculatedVolts;
    calculatedVolts = rawCount * AREF / 1023;
    return calculatedVolts;
}
```

Как можно видеть, добавлено всего три новые строки кода. Первая из них — в функции setup(). Эта команда конфигурирует вывод 11, к которому подключена база транзистора управления электродвигателем, на работу в режиме вывода OUTPUT ❶. Следующие две команды добавлены в блок if...else if. Это команды на включение ❷ и выключение ❸ электродвигателя. Не забывайте, что электродвигатель будет выполнять в тепличке роль привода вентилятора. И эти две добавленные строки кода обеспечивают включение вентилятора одновременно с открытием окна и выключение с его закрытием.

Добавив новый код в скетч, загрузите эту последнюю версию скетча в плату Arduino. Откройте окно монитора порта и снова протестируйте работу кода и устройств. Нагрейте термодатчик, сжав его пальцами или подышав на него, и наблюдайте, что происходит. Как только температура поднимется чуть выше 85 °F, должен провернуться вал сервомашинки и включиться электродвигатель. При этом вы, скорее всего, заметите, что как только включится электродвигатель, показатели температуры резко исказятся. Причина такого поведения системы довольно-таки сложная, но, к счастью, для этой проблемы есть одно быстрое решение.

Изолируем влияние электродвигателя

При включении электродвигателя напряжение питания на плате Arduino резко падает до приблизительно 4,1–4,5 В вследствие возникновения дополнительной нагрузки на питание, создаваемой электродвигателем. При этом может наблюдаться следующий эффект: по мере того как электродвигатель набирает обороты, показания температуры меняются беспорядочно, и электродвигатель может выключиться и включиться несколько раз, прежде чем показания температуры стабилизируются. Ранее упоминалось, что функция analogRead() возвращает значение 1023 для значения напряжения 5 В. Это верно только частично. А вся действительность такова, что 1023 соответствует напряжению источника питания, поэтому если напряжение источника понизится до 4,1 В,

то 1023 и будет представлять эти 4,1 В. Это обстоятельство оказывает значительное отрицательное воздействие на способность Arduino выполнять точные измерения.

Чтобы исправить этот недостаток, добавим две строки кода в самом начале цикла loop(), сразу же после первой фигурной скобки. Этот код дает указание Arduino выключить электродвигатель перед тем, как считывать показания датчика температуры.

```
digitalWrite(11, LOW);  //выключаем электродвигатель
                        //перед тем, как считывать
                        //термодатчик
delay(1);               //выдерживаем паузу в 1 мс
                        //прежде чем считывать датчик
```

Теперь Arduino будет выключать электродвигатель за 1 мс до считывания напряжения с датчика температуры. Это изолирует падение напряжения на электродвигателе от функции analogRead(), не требуя добавления слишком большого объема кода.

Загрузите модифицированный таким образом скетч в Arduino и снова нагрейте термодатчик. Теперь система должна работать намного стабильнее. Что ж, мы собрали схему и выполнили отладку кода для нее, — настало время приступать к сборке собственно теплички.

Собираем корпус теплички

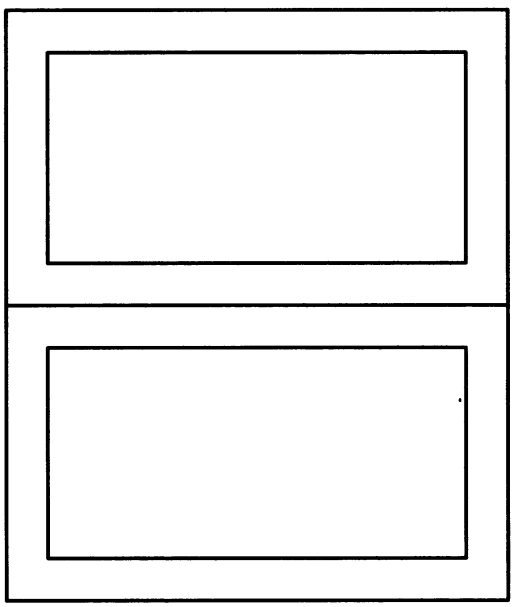
Размеры нашей настольной теплички составляют приблизительно 11,5×11,5 см в основании и 15 см в самой высокой точке. В архиве ресурсов для книги, доступных по адресу **https://www.nostarch.com/arduinoinventor**, предлагается шаблон для корпуса теплички (рис. 7.18), который авторы сделали из картона, но вы можете использовать любой удобный для вас материал. Шаблон можно разнести на три листа картона размером 21,5×28 см каждый или разместить его на одном листе картона размером 28×43 см.

Между прочим, в магазинах IKEA имеется в продаже великолепная небольшая тепличка COKKEP, которую можно легко модифицировать для использования с этим проектом.

Итак, перенесите шаблон на листы картона и аккуратно вырежьте все его части. У вас должны получиться четыре уникальные детали: пятиугольные боковые стенки, передняя и задняя квадратные стенки, окно для крыши и контейнер для электродвигателя.

Возьмите одну квадратную боковую стенку и одну пятиугольную переднюю (или заднюю) стенку и положите их рядом друг с другом, внутренней стороной вверх. С помощью узкой полоски клейкой ленты скрепите эти две стороны, как показано на рис. 7.19. Скрепите таким же образом все четыре стенки теплички за исключением последнего стыка — пусть все четыре стенки пока лежат плоско, чтобы мы могли вставить окна.

Теперь нам потребуется вырезать шесть листов прозрачной пленки слегка большего размера, чем отверстия в стенах и в крыше. В соответствии с шаблоном, нам потребуются четыре квадратных листа размером 11×11 см и два квадратных прямоугольных размером 11×6,5 см. Все шесть панелей можно вырезать из одного листа прозрачной пленки — рекомендуется сначала начертить их на пленке, чтобы экономнее ее использовать.


Крыша

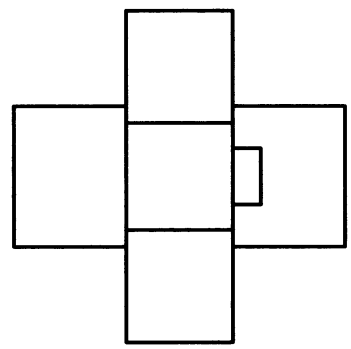

Контейнер для электродвигателя вентилятора

Рис. 7.18. Шаблон для корпуса настольной теплички (в уменьшенном виде)

Сейчас мы вставим окна только в стены, а в крышу вставим их в самом конце. Нанесите тонкий слой клея по бокам стен с внутренней стороны теплички и приклейте оконные панели (рис. 7.20). Вместо клея оконные панели можно также прикрепить с помощью клейкой ленты.

Прикрепив оконные панели, скрепите клейкой лентой оставшийся угол теплички. У вас должна получиться конструкция с квадратным основанием (рис. 7.21). Низ и верх теплички пока остаются открытыми, и она может показаться несколько шаткой, но когда мы поставим на нее крышу, она станет намного жестче.

Рис. 7.19. С помощью узкой полоски клейкой ленты скрепите две стороны

Рис. 7.20. Приклеиваем оконные панели на стены с внутренней стороны теплички

Крепим сервомашинку для управления окном

Наш шаблон предусматривает небольшое отверстие под сервомашинку как можно ближе к оси поворота окна, чтобы при движении качалки сервомашинки окно открывалось на максимальный угол.

Отсоедините сервомашинку от схемы и наденьте качалку на ее вал, если она еще не надета. Рекомендуется использовать одностороннюю качалку, так как с ней легче определить положение вала сервомашинки. Аккуратно вращайте вал по часовой стрелке, пока он не остановится, — так вы установите его в положение 180 градусов (это положение качалки при полностью открытом окне). Снимите качалку с вала и наденьте ее опять, но так, чтобы качалка была направлена в сторону, противоположную стороне сервомашинки, из которой выходят провода (рис. 7.22).

Затем вставьте сервомашинку в отверстие в стенке с внутренней стороны таким образом, чтобы провода были направлены вниз, а качалка смотрела вверх (рис. 7.23). Крепежные выступы корпуса сервомашинки должны плотно прилегать к стенке. Закрепите сервомашинку в стенке, используя винты, поставляемые в комплекте с ней, или с помощью клея.

Рис. 7.21. Собранные стены настольной теплички

Рис. 7.22. Установка качалки на вал в позиции 180 градусов

Рис. 7.23. Установка сервомашинки в стенку теплички

Изготавливаем тягу

Как и в *проекте 6*, нам понадобится тяга, чтобы соединить качалку сервомашинки с окном. Возьмите среднего размера канцелярскую скрепку и распрямите ее, оставив внутренний конец не разогнутым. Теперь возьмите линейку и согните по ней скрепку под прямым углом на расстоянии около 30 мм от неразогнутого конца в направлении, противоположном его изгибу (рис. 7.24).

Устанавливаем крышу

Крыша представляет собой прямоугольный лист картона. Вырежьте в нем отверстия для окон и сделайте неглубокий надрез по центру листа, как показано на рис. 7.25. Этот надрез будет служить шарниром, на котором окно будет вращаться при открытии и закрытии.

Поставьте тепличку перед собой таким образом, чтобы сервомашинка была слева. Одну половину крыши мы приклеим к стенам, а другую оставим свободной, и она будет служить в качестве управляемого окна. С помощью клеевого пистолета приклейте одну сторону крыши к стенам теплички. Наносите клей только на три края одной стороны крыши (то есть на половину из шести краев), чтобы другая сторона оставалась свободной. При этом следите за тем, чтобы свободная сторона крыши была на той же стороне, вдоль которой будет двигаться качалка сервомашинки (рис. 7.26).

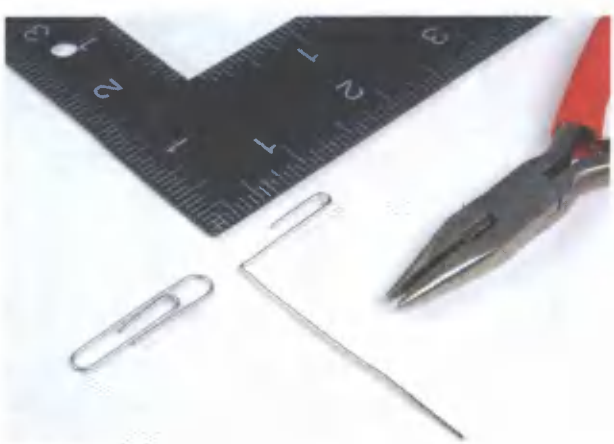


Рис. 7.24. Изготавливаем тягу из канцелярской скрепки

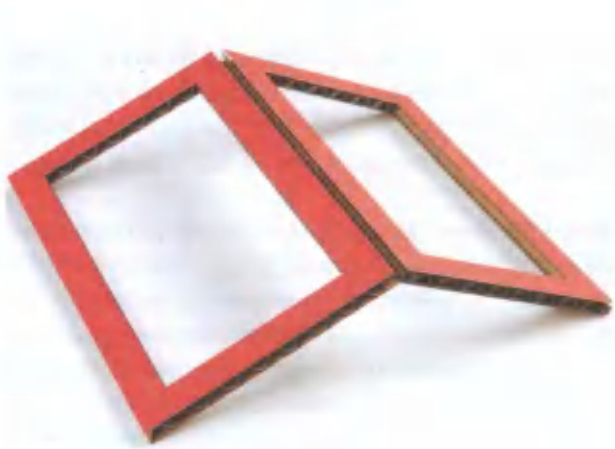


Рис. 7.25. Крыша теплички

Рис. 7.26. Приклеиваем одну сторону крыши, оставляя другую свободной, чтобы она могла открываться и закрываться

Рис. 7.27. Взаимное расположение окна, качалки сервомашинки и тяги

Возьмите тягу, которую мы сделали из скрепки, и вставьте ее согнутым концом, который оставался на ней, в последнее отверстие в качалке сервомашинки, как показано на рис. 7.27. При этом следите, чтобы противоположный согнутый конец был направлен в сторону сервомашинки, — этот конец будет вставлен в боковую «раму» окна.

Удерживая окно открытым, проверните тягу в качалке, чтобы ее можно было вставить во внутреннюю сторону рамы окна прямо через картон. Если тяга недостаточно длинная, можно или выпрямить ее и согнуть снова до требуемой длины, или же переустановить качалку сервомашинки под немного бо□льшим углом, чтобы увеличить диапазон хода тяги. Может быть, вам покажется более удобным поднять корпус теплички и выполнить переустановку качалки снизу. Добившись, чтобы тяга доставала до рамы окна, проткните раму насквозь прямым концом тяги, как показано на рис. 7.28.

Рис. 7.28. Тяга качалки сервомашинки, подсоединенная к окну

Согните выступающий конец тяги, чтобы она не выпала из рамы, и откусите лишнюю длину (рис. 7.29), после чего осторожно проверните вал сервомашинки вперед и назад — окно теплички должно открываться и закрываться.

Рис. 7.29. Закрепляем тягу и откусываем лишнюю длину

Теперь можно приклеить прозрачные панели крыши теплички — по небольшой капле клея по углам крыши будет достаточно, чтобы их закрепить. Панели нужно устанавливать с наружной

стороны, чтобы не мешать тяге открывать и закрывать окно. Осталось выполнить еще одну сборку — контейнера для электродвигателя вентилятора.

Собираем контейнер для электродвигателя

Электродвигатель будет выполнять у нас роль привода вентилятора для проветривания теплички. Чтобы он прочно удерживался в месте крепления, мы поместим его в небольшой контейнер, опять же из картона. Шаблон контейнера представляет собой развертку для открытой с верхней стороны коробочки, в одной из сторон которой вырезано небольшое отверстие для пропуска проводов питания электродвигателя (рис. 7.30).

Рис. 7.30. Развертка контейнера для электродвигателя

Рис. 7.31. Собранный контейнер для электродвигателя с установленным в него электродвигателем

Вырежьте шаблон из картона и аккуратно сделайте неглубокие надрезы по пунктирным линиям, чтобы его можно было сложить в коробочку. Соедините стороны контейнера с помощью клея или клейкой ленты, чтобы он плотно удерживал электродвигатель (рис. 7.31).

Лопастями вентилятора будет служить небольшой отрезок открыточного картона, наклеенный на вал электродвигателя. Ширина лопасти не должна быть более 3 см — чтобы вентилятор случайно не задел растения или что-либо другое внутри крохотной теплички. Для улучшения перемещения воздуха согните края лопасти, как показано на рис. 7.32.

Рис. 7.32. Форма лопасти вентилятора

Рис. 7.33. Наносим клей на вал электродвигателя

Нанесите небольшую каплю клея на вал электродвигателя (рис. 7.33) и прикрепите к ней лопасть вентилятора (рис. 7.34).

Рис. 7.34. Готовый вентилятор

Подключаем электронику

Итак, корпус теплички мы собрали, настало время подключать электронику. Извлеките датчик температуры из макетной платы, возьмите три проволочные перемычки со штыревым разъемом на одном конце и гнездовым на другом и наденьте гнездовые разъемы на выводы термодатчика (рис. 7.35). Как вы можете помнить, мы использовали провода красного цвета для вывода положительного питания, желтого — для сигнального вывода и черного — для вывода отрицательного питания («земли»).

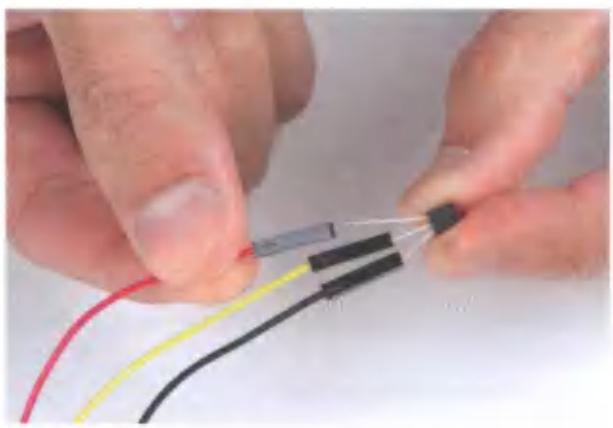


Рис. 7.35. Удлиняем выводы термодатчика с помощью проволочных перемычек

Запомните расположение выводов термодатчика на макетной плате, поскольку и при использовании удлинителей выводов их нужно будет подключить точно так же. Удерживайте термодатчик плоским срезом к себе и выводами, направленными влево, при этом верхним окажется вывод для положительного питания, средним — сигнальный и нижним — для отрицательного питания («земли»).

Датчик температуры должен находиться в тепличке. Лучше всего прикрепить его клейкой лентой к горшочку с растением, которое будет помещено в тепличку (рис. 7.36). Провода термодатчика можно вывести просто под низом теплички или же сделать небольшое отверстие в стенке и пропустить их через него.

Рис. 7.36. Прикрепляем термодатчик непосредственно к горшочку с растением

Поместите контейнер с вентилятором в углу теплички. Провода питания электродвигателя должны иметь достаточную длину, чтобы их можно было напрямую подключить к макетной плате, но, если необходимо, их можно удлинить с помощью проволочных перемычек со штекерным разъемом на одном конце и гнездовым на другом.

Авторы протестировали работу автоматической вентиляции созданного ими образца теплички с помощью софитов, имитирующих жаркие лучи солнца (рис. 7.37).

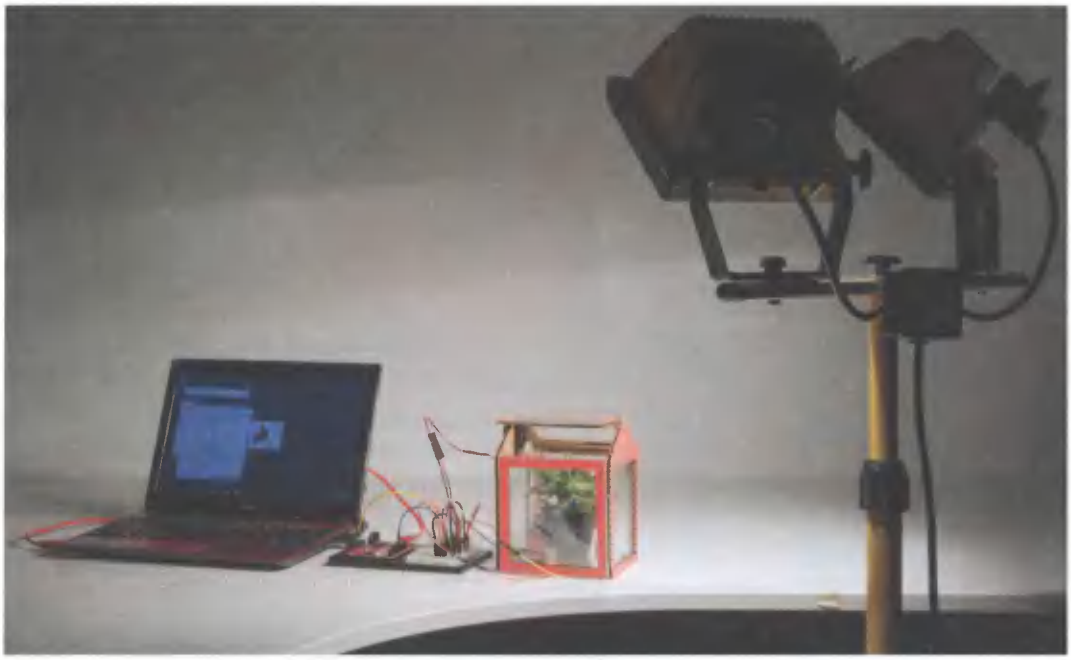


Рис. 7.37. Проверка работы системы автоматики настольной теплички

Идем дальше...

Существует масса возможностей повысить уровень этого проекта.

Экспериментируем с размерами теплицы

Размеры нашей теплички, как она есть сейчас, не очень практичные. Чтобы увеличить ее объем и поместить в нее большее количество растений, найдите большую картонную коробку — типа коробки из-под бумаги для ксерокса. Вырежьте в ней отверстия для окон, заклейте эти отверстия прозрачной пленкой и перенесите свою электронику и механику в эту новую и улучшенную теплицу. Или же рассмотрите возможность приобрести одну из небольших тепличек, предлагаемых компанией IKEA. В этом отношении вам нужно будет продумать, где прикрепить сервомашинку, чтобы открывать и закрывать окно в тепличке большего размера.

Модифицируем код

В реализованном примере контрольная температура установлена на 85 °F (29,5 °C). Эта температура была выбрана по той причине, что ее легко повысить, нагревая термодатчик пальцами или дыханием. Но хотя эта температура комфортна для людей, она довольно-таки низкая для большинства теплолюбивых растений. Узнайте, какую температуру предпочитают ваши растения, и откорректируйте код под эти новые контрольные точки.

Можно также откорректировать частоту температурных проб. Пауза длиной в 1 секунду между замерами слишком короткая. Если температура в вашем помещении сколь-либо колеблется, окно теплички станет открываться и закрываться каждые несколько секунд, что может быстро начать действовать на нервы. Чтобы исправить это, увеличьте задержку до, например, пяти минут, что будет равно 30 000 мс.

РОБОТ-РИСОВАЛЬЩИК

Этим проектом мы воздадим дань уважения проекту «Черепашки Лого», создав робота-рисовальщика, которого можно запрограммировать на движение в разных направлениях, чтобы он рисовал при этом линии в направлении своего движения. Logo — это язык программирования, который был создан в конце 60-х годов прошлого столетия Даниэлем Г. Боброу (Daniel G. Bobrow), Волли Фюрцайг (Wally Feurzeig), Сеймуром Папертом (Seymour Papert) и Синтией Соломон (Cynthia Solomon). Спустя некоторое время этот язык был адаптирован для управления оснащенным фломастером роботом, получившим название «черепашка» (рис. 8.1).

Рис. 8.1. Ранняя версия черепашки Лого

Черепашка управлялась с компьютера, посылающего ей команды на языке Лого, — например fd 10, чтобы проехать вперед 10 метров[1]. Перемещаясь, черепашка рисовала по пути линию прикрепленным к ней фломастером. Эти ранние черепашки Лого служили основой системы для визуального обучения программированию.

[1] fd — сокращение от англ. forward — вперед.

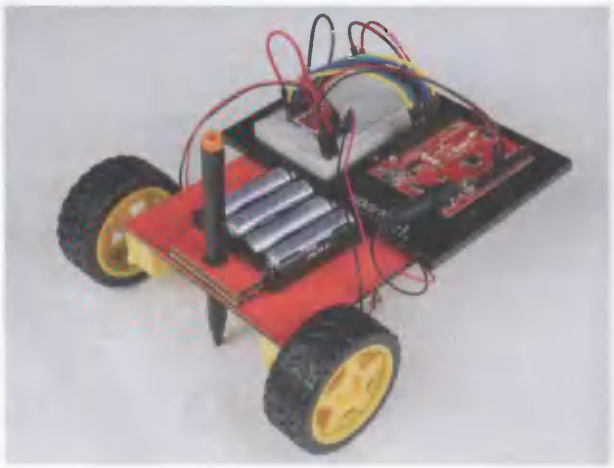


Рис. 8.2. Робот-рисовальщик Рисобот во всем своем великолепии

В этом проекте мы сконструируем нашу версию черепашки под управлением Arduino — робота-рисовальщика Рисобот (рис. 8.2), идея создания которого была вдохновлена работой Сеймура Паперта и его команды.

Необходимые компоненты, инструменты и материалы

Нашего робота мы оснастим двумя колесами, каждое из которых будет приводиться в действие электродвигателем под управлением Arduino с помощью нового электронного компонента, называемого *H-мост*. Это небольшая модульная печатная плата, функционирующая наподобие транзисторной схемы, которую мы использовали в *проекте 7*, с той разницей, что она позволяет управлять как скоростью, так и направлением вращения двигателя. Эти возможности обеспечат нам большую гибкость в управлении роботом.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 8.3):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт.;
- беспаечная макетная плата (PRT-12002), 1 шт.;
- электродвигатель с редуктором (ROB-13302)*, 2 шт.;
- драйвер электродвигателя типа H-мост TB6612FNG (без разъемов ROB-09457 или со штыревыми разъемами ROB-13845)*, 1 шт.;
- резиновые колесики для использования с редукторным электродвигателем (ROB-13259)*, 2 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*;
- держатель для четырех батареек типа AA (PRT-09835)*, 1 шт.

Примечание

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты и материалы (рис. 8.4):

- карандаш;
- макетный нож;
- металлическая линейка;
- клей (клеевой пистолет или клей для моделирования);
- могут пригодиться: дрель и сверло диаметром 4,75 мм;
- может пригодиться: паяльник;
- гофрированный картон (приблизительно 30×30 см) или картонная коробка;
- мячик для настольного тенниса;
- шаблон корпуса (см. рис. 8.12 далее в этом проекте).

Рис. 8.3. Электронные компоненты для проекта робота-рисовальщика

Рис. 8.4. Инструменты и материалы, рекомендуемые для проекта робота-рисовальщика

Два новых компонента

В этом проекте мы применим два новых электронных компонента: драйвер электродвигателя типа H-мост и электродвигатели с редукторами. Давайте познакомимся с этими устройствами поближе.

Интегральная схема H-мостового драйвера электродвигателя

В *проекте 7* мы использовали схему на транзисторе для управления с помощью Arduino одним электродвигателем. Возможности этой схемы были весьма скромными: мы могли только включать или выключать электродвигатель, но не могли управлять ни скоростью, ни направлением его вращения. Используемый в этом проекте новый компонент — H-мостовой драйвер электродвигателя — позволит нам не только включать и выключать электродвигатель, но также управлять как скоростью, так и направлением его вращения.

H-мостовой драйвер электродвигателя представляет собой интегральную схему, состоящую из около дюжины транзисторов, заключенных в пластмассовый корпус небольшого размера. Интегральные схемы, или микросхемы, реализуют различные функциональные электронные компоненты в одном небольшом корпусе, что значительно облегчает сборку сложных проектов. Существует множество различных микросхем, предназначенных для разных целей. Одним из примеров микросхемы является «мозг» Arduino — микросхема ATmega328.

Итак, как уже было сказано, микросхема Н-мостового драйвера электродвигателя позволяет управлять скоростью и направлением вращения электродвигателя с помощью сигналов управления, получаемых от Arduino.

Вспомним из материала *проекта 7*, что транзистор — это просто выключатель, которым можно управлять электронными сигналами. Стандартный Н-мостовой драйвер электродвигателя состоит из четырех или пяти транзисторов (или выключателей), собранных в схему, по конфигурации похожую на букву Н (рис. 8.5).

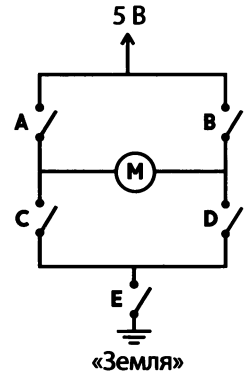


Рис. 8.5. Н-мостовая схема для управления направлением и скоростью вращения электродвигателя (показанный здесь двигатель не является частью Н-моста — он подключается к драйверу пользователем)

Замыкая и размыкая четыре основных переключателя (A–D), можно управлять направлением протекания тока через обмотки электродвигателя, что, в свою очередь, определяет направление его вращения. Пятый переключатель (E) служит для управления скоростью вращения электродвигателя.

Вспомним, что ток протекает от положительного полюса источника питания к отрицательному. При замкнутых выключателях A и D ток будет протекать через электродвигатель слева направо, вращая вал в одном направлении. А при замкнутых выключателях B и C ток потечет через электродвигатель справа налево, вращая вал в противоположном направлении.

Переключатель E включается и выключается сигналом ШИМ (см. *разд. «Создание аналоговых сигналов посредством ШИМ»* в *проекте 5*). Заполнение сигнала ШИМ и определяет скорость вращения электродвигателя. Наш робот будет оснащен двумя электродвигателями — каждый под управлением своего Н-мостового драйвера. Двигатели мы установим на платформе, а на валы двигателей наденем колеса. Таким образом, управляя скоростью и направлением вращения электродвигателей, мы будем управлять скоростью и направлением движения платформы робота.

В этом проекте используется микросхема Н-мостового драйвера электродвигателя TB6612FNG компании Toshiba (рис. 8.6). Микросхема установлена на адаптерную плату с двумя рядами отверстий по краям — расстояние между отверстиями составляет 2,54 мм, что идеально подходит для установки адаптерной платы в макетную плату.

Рис. 8.6. Адаптерная плата с микросхемой TB6612FNG Н-мостового драйвера электродвигателя (без впаянных штыревых разъемов)

Примечание

Вывод STBY микросхемы можно использовать для перевода двигателя в ждущий режим для снижения энергопотребления. Но мы не станем использовать этот режим, поэтому нейтрализуем вывод, подключая его к выводу питания микросхемы VCC.

Как можно понять, микросхема TB6612FNG в действительности содержит два H-моста, что позволяет использовать ее для управления двумя электродвигателями. Выводы управления для каждого двигателя различаются: один двигатель в маркировке на нижней стороне платы обозначен как A, а другой — как B. Сигналы управления на H-мостовую схему подаются с использованием трех проводов: двух — для управления направлением вращения электродвигателя и одного — для управления скоростью вращения.

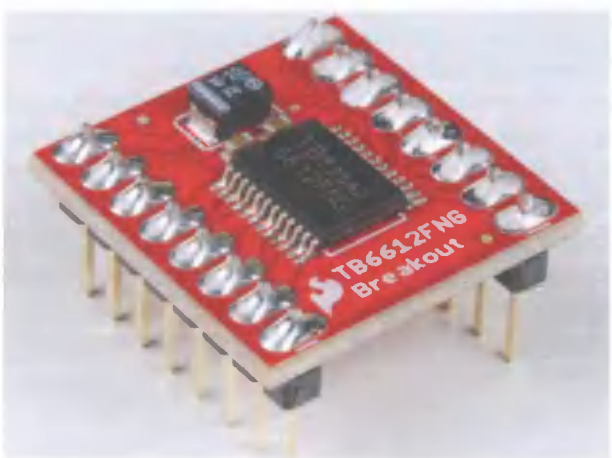


Рис. 8.7. Адаптерная плата ROB-13845 для микросхемы TB6612FNG (с впаянными штыревыми разъемами)

Адаптерная плата с микросхемой TB6612FNG поставляется в двух вариантах: с впаянными штыревыми разъемами и без оных. Поэтому, если вы не хотите морочиться с впаиванием 16 штырьков, приобретайте плату ROB-13845, в которую эти разъемы уже впаяны. Если у вас плата без впаянных штыревых разъемов (ROB-09457), то это тоже не проблема, но вам тогда придется впаять их самому. Инструкции по процессу пайки приведены в *разд. «Работа с паяльником» приложения*. В любом случае, прежде чем приступать к сборке этого проекта, нужна плата, которая выглядит, как показано на рис. 8.7.

Электрический двигатель с редуктором

Простой двигатель постоянного тока, который мы использовали в *проекте 7*, хорошо годится для простых задач — таких как вращение вентилятора, например, но он не создает достаточно большого *крутящего момента*, то есть вращающей силы. Поскольку в этом проекте электродвигатели должны перемещать платформу робота, мы воспользуемся электродвигателями с редуктором (рис. 8.8).

Редуктор электродвигателя, по сути, преобразовывает высокие обороты с малым вращающим моментом в низкие обороты с высоким вращающим моментом. Передаточное отношение редуктора этого электродвигателя составляет 48:1. Это означает, что на 48 оборотов вала электродвигателя выходной вал редуктора совершает один оборот. Замедление скорости вращения в 48 раз повышает вращающий момент в эти же 48 раз. В общем, выходная скорость вращения замедляется, но вращающий момент значительно повышается.

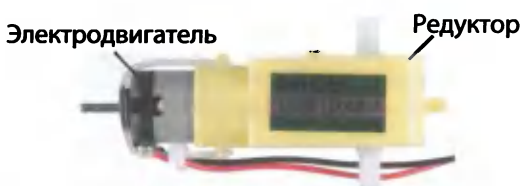


Рис. 8.8. Электродвигатель со встроенным редуктором

Создаем прототип схемы управления Рисоботом

Теперь мы, подкованные достаточными теоретическими знаниями, можем приступить к практической части. Для начала, чтобы разобраться с работой H-мостового драйвера, мы соберем схему управления только для одного электродвигателя. На рис. 8.9 приводится монтажная схема подключения адаптерной платы микросхемы H-мостового драйвера к Arduino и электродвигателя — к адаптерной плате.

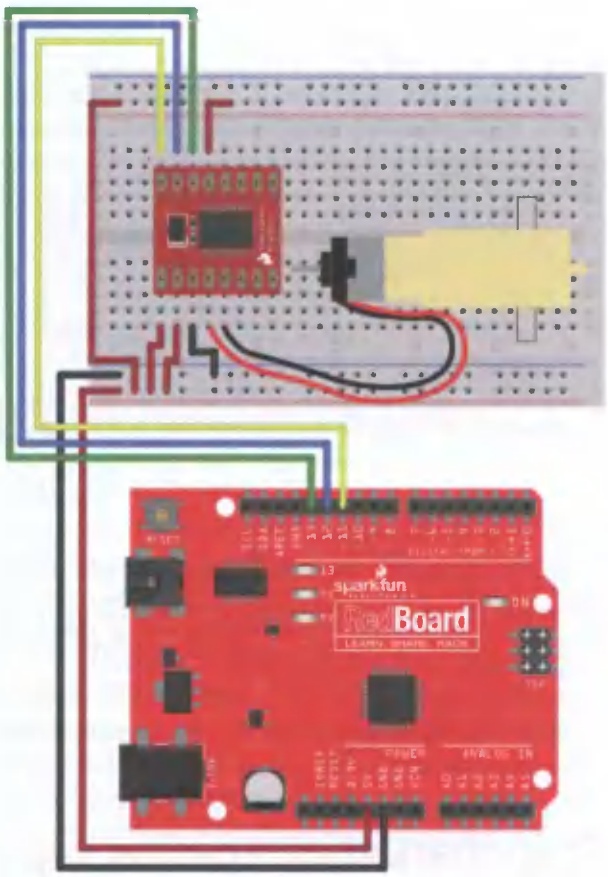


Рис. 8.9. Монтажная схема подключения адаптерной платы микросхемы H-мостового драйвера к Arduino и электродвигателя — к адаптерной плате

Выводы управления платы H-мостового драйвера разделены горизонтально на две группы: верхняя группа управляет двигателем A, а нижняя — двигателем B. Подключите шины питания макетной платы к выводам питания платы Arduino. Соедините также на макетной плате перемычкой обе шины положительного питания (5 В) — это позволит вам использовать любую из этих шин, не загромождая плату переплетением проводов питания.

Начиная с верхних левых выводов адаптерной платы H-мостового драйвера, подключите два верхних вывода: VM и VCC — к положительной шине питания макетной платы (через вывод VM подается питание для электродвигателей, а через вывод VCC — питание для микросхемы H-мостов). Затем подключите перемычкой один из выводов GND адаптерной платы к шине отрицательного питания макетной платы. Как можно видеть на рис. 8.7, на адаптерной плате есть три таких вывода, и подключать можно любой из них.

Затем подключаем электродвигатель. Электродвигатель имеет два провода питания: красный и черный. Ориентация проводов не имеет значения, но чтобы каждый раз не задумываться, мы будем всегда подключать красный провод к выводу A01 адаптерной платы, а черный — к выводу A02.

Остальные контактные выводы с левой стороны платы предназначены для управления вторым электродвигателем, есть там еще один вывод «земли» GND, но на данном этапе мы их задействовать не станем. Верхние выводы с правой стороны адаптерной платы H-мостового драйвера предназначены для подключения сигнальных (управляющих) проводов электродвигателя A. Самый верхний вывод, обозначенный PWMA, управляет скоростью вращения электродвигателя. Подключите этот вывод к гнезду вывода 11 платы Arduino. (Вспомним, что выводы 3, 5, 6, 9, 10 и 11 платы Arduino могут выдавать ШИМ-сигнал, создаваемый функцией analogWrite().)

Следующие два вывода адаптерной платы H-мостового драйвера, обозначенные AIN2 и AIN1, предназначены для управления направлением вращения электродвигателя A. Для этого на них подаются различные комбинации сигналов

высокого и низкого уровня. Список этих комбинаций приведен в табл. 8.1. Подключите вывод AIN2 адаптерной платы к выводу 12 платы Arduino, а вывод AIN1 — к выводу 11.

***Таблица 8.1.** Комбинации сигналов управления направления вращения электродвигателя*

AIN1	AIN2	Функция
Высокий уровень	Низкий уровень	Вращение по часовой стрелке
Низкий уровень	Высокий уровень	Вращение против часовой стрелки
Высокий уровень	Высокий уровень	Электронный тормоз (см. *примечание*)
Низкий уровень	Низкий уровень	Остановка/инерционное вращение

Последней нашей задачей станет дезактивация вывода STBY. Этот вывод служит для управления работой микросхемы H-моста в режиме ожидания, что может понадобиться в ситуациях, когда необходимо следить за энергопотреблением. В этом проекте задача энергосбережения не ставится, поэтому мы дезактивируем этот вывод. Микросхема переходит в режим ожидания, когда на вывод STBY подается сигнал низкого уровня. Чтобы отключить этот режим, мы просто подключим этот вывод напрямую к шине положительного питания макетной платы.

Примечание

Подача сигнала высокого уровня на оба вывода осуществляет электронное торможение электродвигателя. Оба провода питания электродвигателя, по сути, при этом замыкаются друг на друга, в результате чего электродвигатель резко останавливается. И наоборот, подача сигнала низкого уровня на оба вывода просто прекращает активное вращение двигателя, в результате чего он будет вращаться по инерции до полной остановки.

Разрабатываем код для управления Рисоботом

Разработку скетча для управления Рисоботом начнем с небольшой проверки работы электродвигателя. В частности, сначала будем медленно вращать электродвигатель по часовой стрелке в течение 1 секунды, затем поменяем направление вращения и станем вращать его с высокой скоростью против часовой стрелки, после чего остановим электродвигатель на 1 секунду и повторим цикл.

Откройте среду разработки Arduino и скопируйте код из листинга 8.1 в окно редактора скетчей. Затем загрузите скетч в Arduino и наблюдайте за электродвигателем.

Листинг 8.1. Скетч для тестирования H-мостового драйвера

```
❶ const byte AIN1 = 13;
  const byte AIN2 = 12;
  const byte PWMA = 11;

  void setup()
  {
    pinMode(AIN1, OUTPUT);
    pinMode(AIN2, OUTPUT);
    pinMode(PWMA, OUTPUT);
  }

  void loop()
  {
```

```
    //Задаем вращение по часовой стрелке
❷   digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, LOW);
❸   analogWrite(PWMA, 50);
    delay(1000);

    //Задаем вращение против часовой стрелки
❹   digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, HIGH);
    analogWrite(PWMA, 255);
    delay(1000);

    //Тормозим
❺   digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, HIGH);
    delay(1000);
}
```

Скетч начинается с объявления констант с новым типом данных: byte ❶. Константа представляет собой конструкцию наподобие переменной, но с тем отличием, что ее значение остается постоянным на протяжении всего исполнения программы. Константа объявляется с помощью ключевого слова const. Константы полезны для обозначения величин, которые не будут изменяться в ходе исполнения программы, — например, это может быть номер вывода или режим его работы. В данном случае константы определяют номера выводов Arduino, используемые для управления H-мостовым драйвером. Поскольку номера выводов Arduino находятся в диапазоне значений от 0 до 13, для этих констант можно указать тип данных byte.

> **Примечание**
>
> В большинстве случаев не играет большой роли, что использовать: константу или переменную, но константы занимают в Arduino меньший объем памяти. Поэтому рекомендуется использовать константы во всех возможных случаях. Кстати, хотя это и не твердо установленное правило, имена констант обычно пишутся заглавными буквами.

Далее выполняется конфигурирование выводов для работы в режиме вывода данных (OUTPUT), а затем указывается направление вращения электродвигателя. Для указания направления вращения используются два вызова функции digitalWrite(): одна — для вывода AIN1, а другая — для AIN2 ❷. В первом блоке цикла loop() на вывод AIN1 с платы Arduino подается сигнал высокого уровня, а на вывод AIN2 — низкого. Таким образом задается вращение электродвигателя по часовой стрелке. Скорость вращения электродвигателя задается вызовом функции analogWrite() для вывода PWMA ❸. Вспомним из *проекта 5*, что с помощью функции analogWrite() на аналоговый вывод платы Arduino можно подать ШИМ-сигнал со значением заполнения от 0 до 255. В данном случае используется довольно низкое значение заполнения: 50. Блок завершается инструкцией delay(1000), в результате чего двигатель будет вращаться в заданном направлении и с заданной скоростью в течение 1 секунды. Направление и скорость вращения электродвигателя меняются в следующем блоке инструкций, начиная с двух инструкций digitalWrite() ❹. Эти инструкции меняют уровни сигналов на выводах управления направлением вращения двигателя на обратные, в результате чего двигатель станет вращаться в противоположном направлении. Далее инструкция analogWrite(255) задает более высокую скорость вращения электродвигателя. В этой конфигурации электродвигатель будет работать также 1 секунду, что задается последней инструкцией этого блока — delay(1000).

В последнем блоке инструкций цикла loop() на оба вывода: AIN1 и AIN2 — подается сигнал высокого уровня ❺, что вызывает электронное торможение двигателя. Двигатель находится в остановленном состоянии 1 секунду, как указывается инструкцией delay(1000), после чего цикл повторяется.

Используя этот код в качестве образца, мы можем управлять скоростью и направлением вращения электродвигателя с помощью всего трех строк кода. Но код можно упростить еще больше, создав пользовательские функции.

Создаем пользовательскую функцию

На данный момент всякий раз, когда мы хотим изменить режим работы электродвигателя, нам требуется три строки кода: две — для управления направлением вращения и одна — для управления скоростью вращения. Здесь мы создадим пользовательскую функцию, изменяющую как направление, так и скорость вращения электродвигателя, при передаче ей всего лишь одного параметра. Этот параметр может быть любым числом в диапазоне от –255 до 255: абсолютная величина числа будет определять скорость вращения электродвигателя, а его знак — направление.

Итак, добавьте в самый конец своего скетча код из листинга 8.2.

Листинг 8.2. Пользовательская функция для управления направлением и скоростью вращения двигателя A

```
  void ❶ setMotorA( ❷ int motorSpeed)
  {
❸   if (motorSpeed > 0)
    {
      digitalWrite(AIN1, HIGH);
      digitalWrite(AIN2, LOW);
    }
❹   else if (motorSpeed < 0)
    {
      digitalWrite(AIN1, LOW);
      digitalWrite(AIN2, HIGH);
    }
❺   else
    {
      digitalWrite(AIN1, HIGH);
      digitalWrite(AIN2, HIGH);
    }
❻   analogWrite(PWMA, abs(motorSpeed));
  }
```

Присвоим нашей функции имя setMotorA() ❶. Функция принимает один числовой аргумент: motorSpeed ❷, задающий скорость вращения электродвигателя. Далее оператор if...else if определяет знак передаваемого в аргументе числа, проверяя, больше оно или меньше нуля. Если число motorSpeed положительное ❸, выполняется блок инструкций части if, которые задают вращение электродвигателя по часовой стрелке. Если число отрицательное ❹, выполняется блок инструкций части else if, которые задают вращение электродвигателя против часовой стрелки. Если же число ни положительное, ни отрицательное (то есть равно 0), выполняется блок инструкций части else ❺, которые задают электронное торможение электродвигателя.

Далее, с помощью математической функции abs(), в точке ❻ определяется абсолютное значение числа motorSpeed. Скорость вращения электродвигателя задается функцией analogWrite(), но она работает только со значениями в диапазоне от 0 до 255. Функция abs() обеспечивает, что только положительная часть (абсолютная величина) числа motorSpeed используется для задания скорости вращения электродвигателя.

Расчищаем код

Теперь давайте воспользуемся нашей пользовательской функцией, чтобы расчистить код в цикле loop(). Результат очистки показан в листинге 8.3. Как можно видеть, код в цикле loop() стал намного короче и более удобочитаемым.

Выполните эти изменения в своем скетче и загрузите его в Arduino. Под управлением этого скетча электродвигатель будет работать так же, как и под его предыдущей версией. Но теперь мы можем изменять скорость и направление вращения электродвигателя, откорректировав всего лишь одну строку кода!

Листинг 8.3. Упрощенная версия цикла loop() с применением пользовательской функции setMotor()

```
void loop()
{
  //Задаем вращение по часовой стрелке
  setMotorA(100);
  delay(1000);
```

```
  //Задаем вращение против часовой стрелки
  setMotorA(-255);
  delay(1000);

  //Тормозим
  setMotorA(0);
  delay(1000);
}
```

В этом коде задается значение параметра функции setMotorA() и значение задержки для каждого изменения направления и скорости вращения. Теперь у нас есть начальная часть скетча для управления Рисоботом. Продолжим работу над ним, подключив второй электродвигатель.

Подключаем второй электродвигатель

Имея одно рабочее колесо, наш Рисобот будет хромать. И чтобы приводить в действие второе колесо, нам нужно включить в работу второй электродвигатель. На рис. 8.10 приводится монтажная схема подключения второго электродвигателя к адаптерной плате H-моста.

Подключите провода питания второго электродвигателя в гнезда макетной платы сразу же под гнездами подключения первого электродвигателя. При этом красный провод должен быть подсоединен к выводу B02 адаптерной платы H-мостового драйвера, а черный — к выводу B01. Затем подключите сигнальные выводы Arduino к адаптерной плате H-мостового драйвера (на рис. 8.10 не показано) — сразу же под выводом STBY адаптерной платы с ее правой стороны: ввод 10 Arduino подключается к выводу PWMB адаптерной платы H-моста для управления скоростью вращения, а выводы 8 и 9 Arduino — к выводам BIN1 и BIN2 адаптерной платы, соответственно, для управления направлением вращения.

Теперь нам нужно добавить в наш скетч выделенный полужирным код из листинга 8.4 для управления вторым электродвигателем.

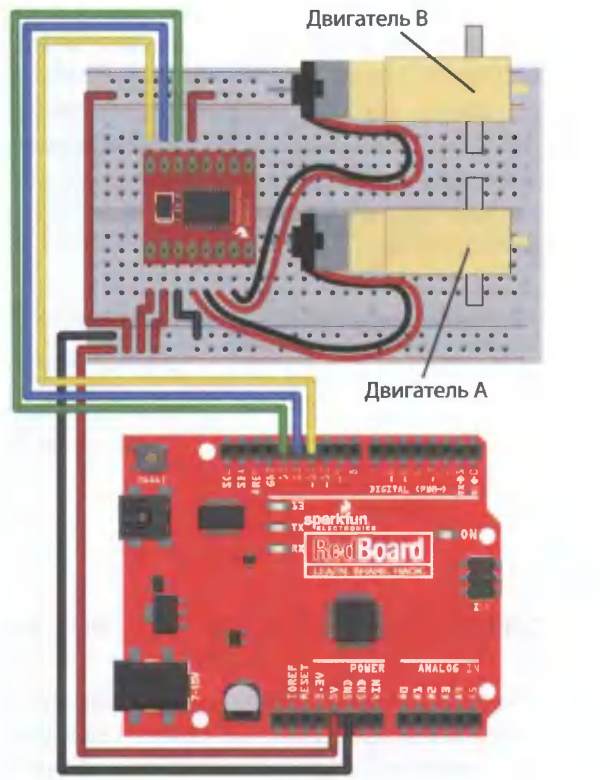


Рис. 8.10. Монтажная схема подключения H-мостового драйвера и двух электродвигателей (подключение сигнальных проводов двигателя B здесь не показано)

Листинг 8.4. Добавляем константы и функции pinMode() для двигателя B

```
const byte AIN1 = 13;
const byte AIN2 = 12;
const byte PWMA = 11;

❶ const byte BIN1 = 8;
  const byte BIN2 = 9;
  const byte PWMB = 10;

void setup()
{
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

❷ pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(PWMB, OUTPUT);
}
```

В этом коде добавляются три дополнительные константы ❶ для выводов сигналов управления двигателем B, а также выполняется конфигурация этих выводов для работы в режиме вывода данных ❷.

Далее мы снова создадим пользовательскую функцию, на этот раз для управления двигателем B. Код этой функции настолько идентичен коду функции setMotorA() для управления двигателем A, что его можно просто скопировать и вставить в скетч под кодом функции setMotorA(), а затем лишь заменить букву A в обозначении двигателя на букву B[2]. Программисты часто прибегают к такой уловке, которая позволяет сэкономить много времени. Только необходимо быть внимательным и заменить все A на B во второй функции (листинг 8.5), иначе код не будет работать должным образом.

[2] При этом необходимо обратить внимание, что вставляется латинская буква B, а не русская В.

Листинг 8.5. Пользовательская функция для управления двигателем B

```
void setMotorB(int motorSpeed)
{
  if (motorSpeed > 0)
  {
    digitalWrite(BIN1, HIGH);
    digitalWrite(BIN2, LOW);
  }
  else if (motorSpeed < 0)
  {
    digitalWrite(BIN1, LOW);
    digitalWrite(BIN2, HIGH);
  }
  else
  {
    digitalWrite(BIN1, LOW);
    digitalWrite(BIN2, LOW);
  }
  analogWrite(PWMB, abs(motorSpeed));
}
```

Теперь для скетча требуется значения motorSpeed как для функции setMotorA(), так и для функции setMotorB(). Давайте добавим эти значения, чтобы испытать совместную работу электродвигателей.

Проверяем работу обоих электродвигателей

Чтобы Рисобот двигался вперед, правый электродвигатель должен вращаться по часовой стрелке, а левый — против. С первого взгляда это кажется нелогичным, но давайте взглянем на платформу робота с обеих сторон. На рис. 8.11 показаны обе стороны платформы робота, где прямыми стрелками обозначено движение вперед, а круглыми — вращение колес.

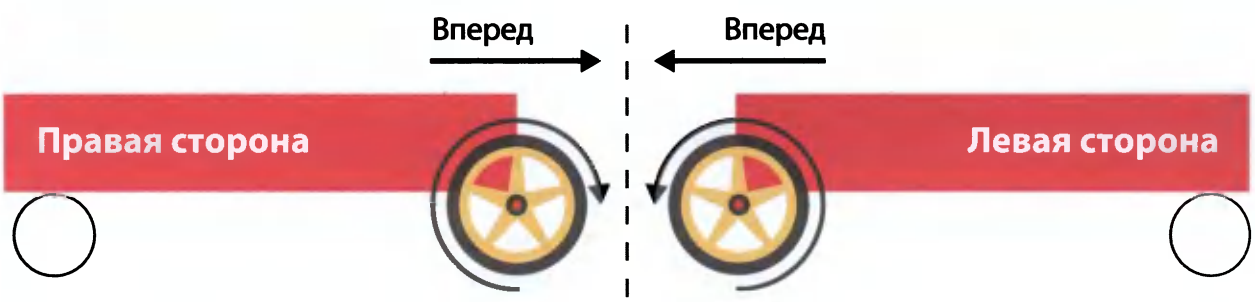


Рис. 8.11. Вид сбоку на обе стороны платформы робота: для поступательного движения платформы правое колесо должно вращаться по часовой стрелке, а левое — против

Итак, для движения робота вперед колесо правой стороны платформы робота должно вращаться по часовой стрелке, а колесо левой стороны — против нее. Чтобы было более удобно различать направление вращения вала каждого электродвигателя, прикрепите на валы по кусочку клейкой ленты. Ну, а чтобы робот двигался назад, нужно просто поменять направление вращения каждого электродвигателя на противоположное.

Вставив код функции setMotorB() в скетч, откорректируйте цикл функции loop(), как показано в листинге 8.6, а затем загрузите скетч в Arduino и любуйтесь вращением электродвигателей!

В результате исполнения последней версии скетча двигатель A (правая сторона) должен вращаться по часовой стрелке, а двигатель B (левая сторона) — против нее. Если окажется, что оба двигателя вращаются в одном направлении, поменяйте местами провода питания одного из них.

Теперь, с помощью нескольких линий строк мы можем заставить нашего Рисобота двигаться вперед, назад, поворачивать направо и налево — короче, двигаться, куда нам нужно.

На следующем этапе мы создадим для робота платформу. Пока мы занимаемся этой задачей, отключите Arduino от компьютера, чтобы остановить вращение электродвигателей.

Листинг 8.6. Новый цикл loop() для тестирования обоих электродвигателей

```
void loop()
{
  //Двигаемся вперед на средней скорости в течение
  //одной секунды
  setMotorA(100);
  setMotorB(-100);
  delay(1000)

  //Двигаемся назад на высокой скорости в течение
  //одной секунды
  setMotorA(-255);
  setMotorB(255);
  delay(1000);

  //Тормозим и стоим одну секунду
  setMotorA(0);
  setMotorB(0);
  delay(1000);
}
```

Создаем платформу для Рисобота

Если вы используете держатель макетной платы и платы Arduino из комплекта изобретателя SparkFun Inventor's Kit, платформа для Рисобота должна быть, по крайней мере, такого же размера, как и держатель (размеры держателя составляют 15×11 см).

Платформу можно выполнить из картона или тонкой фанеры. Авторы сделали платформу для реализованного ими проекта из гофрированного картона размером 15×20 см (рис. 8.12). Файл ***P8_DrawbotTemplate.pdf*** с шаблоном такой платформы находится в архиве ресурсов для книги, которые можно загрузить по адресу: **https://www.nostarch.com/arduinoinventor/**.

С помощью клейкой ленты или клеевого пистолета (рис. 8.13, *слева*) прикрепите электродвигатели к нижней части платформы. Сориентируйте их, как показано на рис. 8.13, *справа* — узлом электродвигателя в сторону задней части платформы, а редуктором — в сторону ее передней части. (Клеевой пистолет — это вообще прекрасное средство для крепления деталей, поскольку

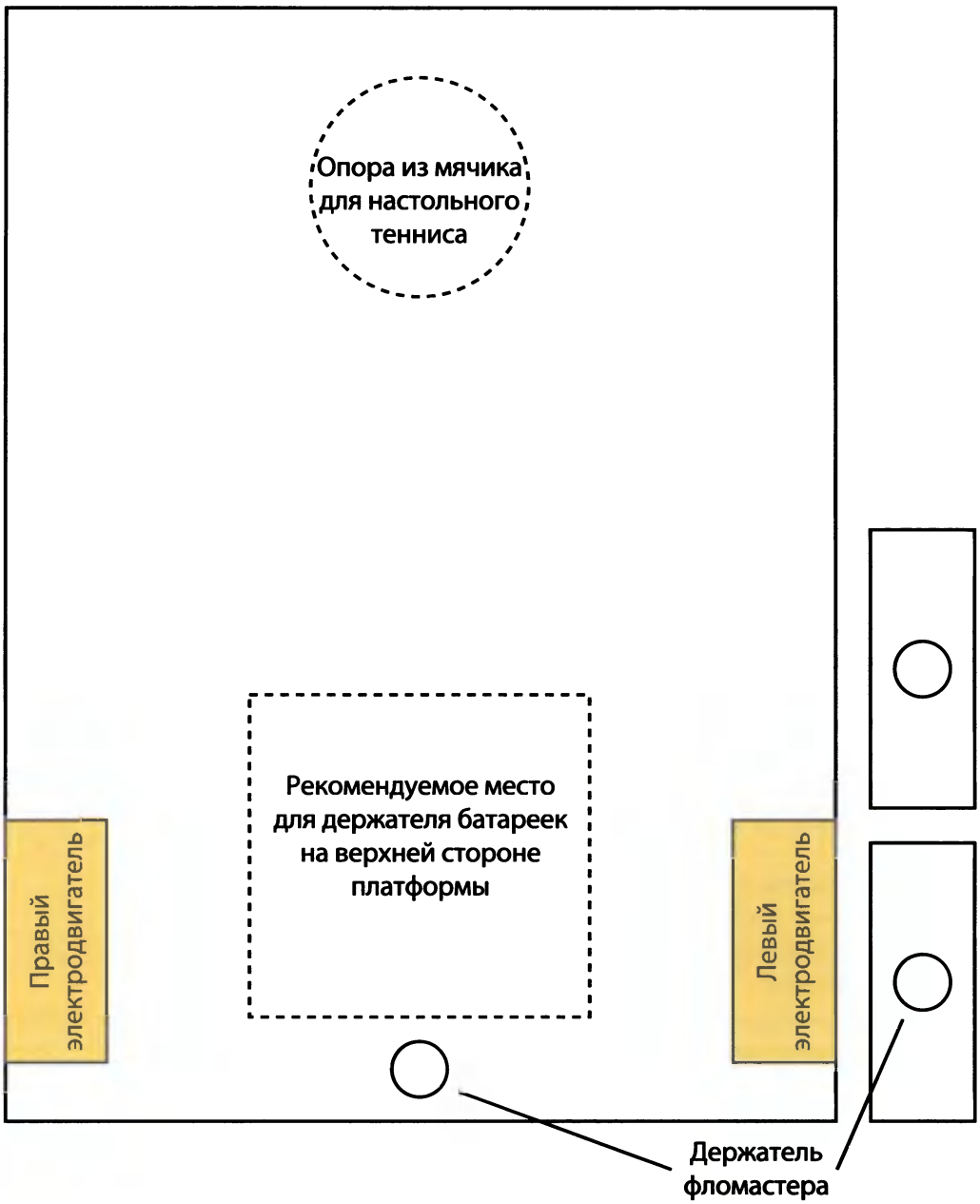


Рис. 8.12. Шаблон платформы Рисобота: вид снизу (в уменьшенном виде)

деталь можно просто снять, соскрести с нее клей и использовать в другом проекте.) Прикрепляя электродвигатели к платформе, отсоедините их от макетной платы, а когда клей схватится, подключите их обратно. Если вы забыли, как их подключать, взгляните на монтажную схему, показанную на рис. 8.10, и описание к ней.

На краю платформы имеется отверстие для фломастера или маркера — ведь наш робот должен оправдывать свое название! Для придания креплению фломастера большей жесткости, склейте два небольших кусочка картона (рис. 8.14, *слева*) с отверстиями под фломастер в них — чтобы держатель стал более высоким (рис. 8.14, *посередине*).

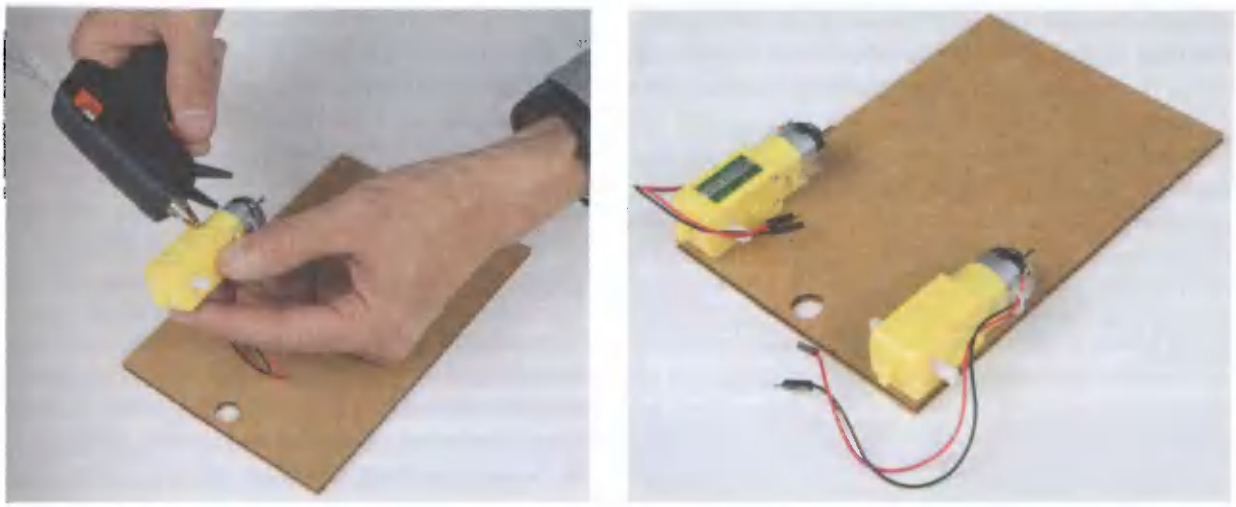


Рис. 8.13. Прикрепляем электродвигатели к платформе: нанесение клея (*слева*) и ориентация относительно платформы (*справа*)

Рис. 8.14. Усиление жесткости держателя фломастера: нанесение клея на фрагменты держателя (*слева*), склеивание фрагментов (*посередине*) и приклеивание держателя к платформе (*справа*)

Склеенный держатель наклейте на отверстие в платформе, как показано на рис. 8.14, *справа*.

Теперь наденем колеса на валы электродвигателей. Обратите внимание, что на валах электродвигателей имеются две противоположные плоские грани (рис. 8.15). Эти грани на валах необходимо совместить с плоскими гранями в осевых отверстиях обеих колес. Колеса могут насаживаться на валы электродвигателей с некоторым натягом, поэтому при насадке колеса удерживайте электродвигатель, чтобы не сорвать его с платформы.

Рисобот будет перемещаться, используя оба своих колеса для движения и рулежки, а опора из

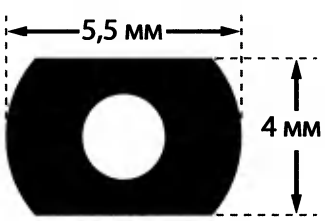


Рис. 8.15. Вид с торца на вал электродвигателя: плоские грани на валу необходимо совместить с плоскими гранями в осевых отверстиях колес

Примечание

Если вы используете свои колеса, их осевые отверстия должны иметь профиль и размеры, показанные на рис. 8.15.

мячика для настольного тенниса на противоположном конце платформы обеспечит его устойчивость. Такой метод управления называется *дифференциальным*. У нашего Рисобота оба передних колеса — ведущие, а вместо заднего колеса используется опора из теннисного мячика, которая будет скользить при движении Рисобота по поверхности. Приклейте теннисный мячик к платформе, как показано на рис. 8.16, стараясь разместить его как можно точнее по центру платформы.

Рис. 8.16. Приклеиваем опору из теннисного мячика

Рис. 8.17. Держатель платы Arduino и макетной платы и держатель батареек на платформе Рисобота

Наконец, разместите держатель с платой Arduino и макетной платой и держатель батареек на платформе Рисобота, как показано на рис. 8.17. Прикрепите их к платформе с помощью клейкой ленты, чтобы они не двигались и не сползли с платформы.

Тестирование и отладка

Подключите Arduino к USB-порту компьютера или к внешнему источнику питания из батареек в держателе и понаблюдайте, что происходит. Если питание подается от компьютера через USB-кабель, не помешает придерживать кабель, чтобы он не запутался. Робот должен медленно двигаться вперед в течение одной секунды, затем быстро двигаться назад тоже в течение одной секунды, а затем затормозить и стоять на месте опять же одну секунду. Поскольку весь код для управления Рисоботом содержится в цикле loop(), то эта последовательность движений будет продолжаться бесконечно, пока на Arduino подается питание.

> **Совет**
>
> Если вам понадобится выполнить отладку кода, поставьте Рисобота на какую-либо подставку — например, на стопку книг, чтобы колеса не касались поверхности.

Если что-то пойдет не так, и робот не станет двигаться, как только что описано, необходимо выполнить диагностирование, чтобы найти причину неполадки и устранить ее. Сначала необходимо определить проблему. Авторы сталкивались с двумя типами проблем с этим роботом: движение не в том направлении и движение по кругу. Если робот движется не вперед, а назад, поменяйте местами провода питания обоих электродвигателей, подключенные к H-мостовому драйверу. Если же робот движется по кругу, а не по прямой, попробуйте поменять местами провода питания на одном из электродвигателей: или A, или B. Но не меняйте в этом случае местами провода питания на обоих двигателях — так вы

снова получите ту же проблему. После устранения проблем Рисобот должен двигаться сначала вперед, потом назад, затем затормозить, после чего снова повторять весь цикл.

Прежде чем продолжать работу над проектом, замерьте, какое расстояние Рисобот проходит за 1 секунду, и запишите эту информацию. Когда мы оснастим робота фломастером для рисования линий, возможно, нам придется откорректировать скорость и длительность движения, чтобы Рисоботом было легче управлять. Если вашей испытательной площадкой служит стол небольшого размера, будет лучше использовать медленную скорость и короткие периоды движения.

Танец робота — делаем повороты и рисуем узоры

Теперь, когда мы научились управлять движением робота вперед и назад, попробуем рисовать им различные узоры. Но прежде чем давать Рисоботу фломастер, нам нужно научить его делать повороты.

Чтобы сделать правый поворот, оба электродвигателя должны вращаться против часовой стрелки, а чтобы сделать левый — по часовой стрелке. Давайте попробуем нарисовать Рисоботом квадрат. Для этого он должен выполнить такую последовательность движений (рис. 8.18):

1. Движение вперед.
2. Поворот на 90 градусов.
3. Движение вперед.
4. Поворот на 90 градусов.
5. Движение вперед.
6. Поворот на 90 градусов.
7. Движение вперед.
8. Поворот на 90 градусов.

Вы, наверное, уже заметили, что вся эта последовательность состоит из повторения двух операций четыре раза. Мы уже знаем, как выполнять повторяющиеся действия с помощью цикла loop(), но этот цикл выполняется бесконечно, а нам нужно остановить робота после четырех поворотов. Эта проблема решается с помощью цикла другого типа, позволяющего выполнять набор операций заданное количество раз — цикла for().

Структура цикла for() приводится в листинге 8.7.

Листинг 8.7. Структура цикла for()

```
for(❶ int count = 0; ❷ count < 4; ❸ count++)
{
❹ //Здесь идет блок кода, исполнение которого нужно
  //повторять.
}
```

Аргумент ❶ int count = 0; объявляет и инициализирует переменную счетчика, которая подсчитывает количество исполнений цикла. В данном случае для переменной объявляется целочисленный тип данных int, присваивается имя count и устанавливается исходное значение 0. Переменной счетчика можно присвоить любое имя, при условии, что это же имя используется в следующих двух

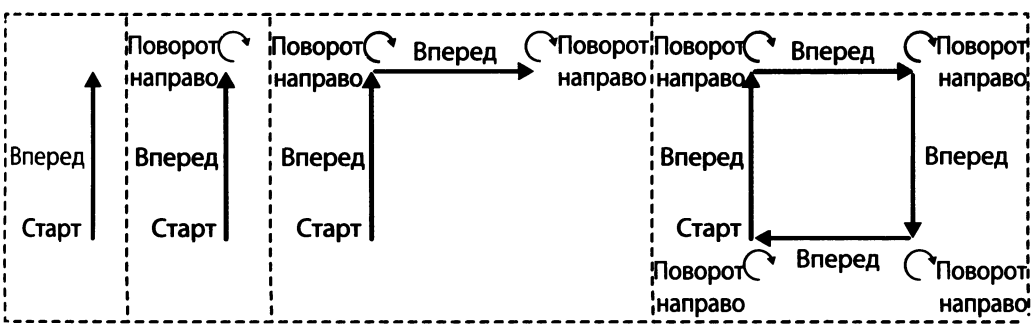


Рис. 8.18. Последовательность движений робота при рисовании квадрата

аргументах. Далее следует аргумент условия ❷ count < 4;, которое управляет повторением исполнения цикла. В данном случае цикл исполняется до тех пор, пока удовлетворяется условие count < 4. Поскольку переменной count при инициализации было присвоено значение 0, то условие удовлетворяется при входе в цикл, и цикл исполняется. Третий аргумент — ❸ count++ — представляет собой оператор *инкремента*, который задает операцию над переменной счетчика после каждого исполнения цикла. В данном случае count++ означает приращение инструкции: count = count + 1. Эта инструкция увеличивает значение переменной счетчика count на 1 после каждого исполнения цикла. Код, который нужно исполнять в цикле, помещается в часть ❹ в фигурных скобках.

Таким образом, аргументы цикла for() дают указание исполнить цикл четыре раза. После этого значение переменной count будет равно 4, вследствие чего условие цикла больше не будет удовлетворяться, и исполнение цикла прекратится. Цикл for() предоставляет очень удобный способ подчистить код, повторяя инструкции определенное количество раз.

Теперь у нас есть все знания для того, чтобы заставить Рисобота нарисовать квадрат. Замените цикл loop() в скетче циклом loop(), приведенным в листинге 8.8. Все остальные части скетча не меняются.

Листинг 8.8. Код для рисования квадрата

```
void loop()
{
❶  for(int count = 0; count < 4; count++)
   {
      //Едем вперед
   ❷ setMotorA(100);
      setMotorB(-100);
   ❸ delay(500);

      //Поворачиваем направо
   ❹ setMotorA(-100);
      setMotorB(-100);
      delay(250);
   }
❺  delay(1000);
}
```

БЫСТРЫЙ СПОСОБ МАНИПУЛИРОВАНИЯ ПЕРЕМЕННЫМИ

Часто нам требуется инкрементировать, декрементировать или просто модифицировать значение переменной в коде. Наиболее часто приходится увеличивать значение переменной на 1 для каждого исполнения цикла — это можно осуществить с помощью, например, следующего кода: имяПеременной = имяПеременной +1. Но эту операцию, а также другие подобные операции модифицирования значения переменной можно выполнить более коротким способом, как показано в табл. 8.2.

Таблица 8.2. *Операции модифицирования значения переменной*

Сокращенный код	Обычный код	Описание
имяПеременной++;	имяПеременной = имяПеременной + 1;	Инкремент на 1
имяПеременной+=2;	имяПеременной = имяПеременной + 2;	Инкремент на 2
имяПеременной+=n;	имяПеременной = имяПеременной + n;	Инкремент на n
имяПеременной--;	имяПеременной = имяПеременной - 1;	Декремент на 1
имяПеременной-=2;	имяПеременной = имяПеременной - 2;	Декремент на 2
имяПеременной -= n;	имяПеременной = имяПеременной - n;	Декремент на n
имяПеременной *= n;	имяПеременной = имяПеременной * n;	Умножить на n
имяПеременной /= n;	имяПеременной = имяПеременной / n;	Разделить на n

Для рисования квадрата в скетче используется цикл for()❶, инструкции в котором исполняются четыре раза. Сначала робот движется вперед ❷ в течение короткого времени (всего лишь полсекунды ❸) — мы не хотим, чтобы Рисобот слишком разогнался и разрисовал нам весь стол. Далее, чтобы выполнить поворот, скетч задает вращение против часовой стрелки для обоих электродвигателей ❹. В завершение рисования квадрата добавляем короткую паузу длиной в одну секунду ❺. Обратите внимание, что эта пауза вставляется уже после закрывающей фигурной скобки цикла for().

Теперь Рисобот выполняет движения для рисования квадрата: проехать вперед и повернуть четыре раза, а затем ничего не делает в течение одной секунды, прежде чем повторять весь цикл loop(). Это даст нам возможность вручную передвинуть робота на новое место.

Значения в этом коде работали должным образом в роботе, реализованном авторами, но, возможно, вам придется выполнить тонкую настройку и поиграть немного с настройками скорости и пауз для вашей реализации Рисобота. Вносите поправки в эти параметры до тех пор, пока Рисобот не станет двигаться по квадратоподобному маршруту. Не переживайте, если сначала квадрат получается не совсем квадратным, — просто продолжайте экспериментировать со скоростью поворота. Все полученные узоры будут частью создаваемого вами образца искусства!

В листинге 8.9 приводится полный код для рисования квадрата Рисоботом.

Листинг 8.9. Полный код для рисования квадрата Рисоботом

```
const byte AIN1 = 13;
const byte AIN2 = 12;
const byte PWMA = 11;
const byte BIN1 = 8;
const byte BIN2 = 9;
const byte PWMB = 10;

void setup()
{
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(PWMB, OUTPUT);
}

void loop()
{
  for(int count = 0; count < 4; count++)
  {
    //Едем вперед
    setMotorA(100);
    setMotorB(-100);
    delay(500);

    //Поворачиваем направо
    setMotorA(-100);
    setMotorB(-100);
    delay(250);
  }
  delay(1000);
}

void setMotorA(int motorSpeed)
{
  if (motorSpeed > 0)
  {
    digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, LOW);
  }
  else if (motorSpeed < 0)
  {
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, HIGH);
  }
  else
  {
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, LOW);
```

```
    }
    analogWrite(PWMA, abs(motorSpeed));
}

void setMotorB(int motorSpeed)
{
    if (motorSpeed > 0)
    {
        digitalWrite(BIN1, HIGH);
        digitalWrite(BIN2, LOW);
    }
    else if (motorSpeed < 0)
    {
        digitalWrite(BIN1, LOW);
        digitalWrite(BIN2, HIGH);
    }
    else
    {
        digitalWrite(BIN1, LOW);
        digitalWrite(BIN2, LOW);
    }
    analogWrite(PWMB, abs(motorSpeed));
}
```

Рис. 8.19. Рисобот в действии. Позаботьтесь, чтобы поверхность для рисования была достаточно большого размера, чтобы не разрисовать весь пол

Рис. 8.20. Крендель-Бот: плата Arduino и макетная плата скрыты внутри красной коробки

Как упоминалось ранее, в передней части платформы Рисобота имеется отверстие для крепления фломастера или маркера. Для рисования рекомендуется использовать смываемый или стираемый маркер. Раздобудьте большой кусок плакатной бумаги или рисовальную доску, которую можно положить на пол. Будьте очень осторожным, чтобы не разрисовать весь пол! Это может создать для вас проблемы. (Поверьте авторам, которые допустили эту ошибку в прошлом и весьма сожалели об этом впоследствии.)

Поставьте своего Рисобота на поверхность для рисования. Установите маркер в держателе таким образом, чтобы он надежно контактировал с поверхностью для рисования, и закрепите его клейкой лентой. Подвигайте Рисобота по поверхности для рисования вручную, чтобы проверить правильность и надежность установки маркера. Теперь подключите Рисобота к компьютеру (или подсоедините внешний источник питания на батарейках) и наблюдайте за происходящим. Будьте наготове схватить Рисобота, если окажется, что он норовит сбежать с поверхности для рисования на пол.

Для разнообразия вставьте маркер другого цвета или модифицируйте код для рисования квадратов разных размеров. Попробуйте заставить Рисобота рисовать спирали или звезды. На рис. 8.19 показан пример узоров, которые Рисобот авторов рисовал в их офисе.

Вы можете художественно оформить своего Рисобота, украсив его какими-либо своими поделками. Авторы поставили на свой Рисобот старую банку из-под кренделей — теперь он называется Крендель-Бот и разъезжает по офису, предлагая бесплатные кренделки (рис. 8.20).

Идем дальше...

Проект робота-рисовальщика — это всего лишь введение в основы роботехники. Самые простые роботы содержат всего лишь контроллер и два электродвигателя, и это как раз то, из чего состоит наш проект. Далее приводится несколько идей, показывающих, как повысить его уровень.

Экспериментируем с кодом

Запрограммированное движение поначалу занимательно, но необходимость перепрограммировать Arduino всякий раз, когда нужно изменить маршрут/узор, быстро становится головной болью. Однако слегка модифицировав код, движением Рисобота можно будет управлять из окна монитора порта.

До этого момента мы использовали монитор порта только для чтения данных, отправляемых Arduino на компьютер в процессе работы скетча проекта, — например, данных различных датчиков. Но с помощью монитора порта данные также можно отправлять и на Arduino. Откройте окно монитора порта. В верхней части его окна находится текстовая строка, в правом конце которой имеется кнопка **Send** (Отправить), как показано на рис. 8.21. Посредством этой текстовой строки и кнопки можно отправлять данные на плату Arduino с целью управления ею.

В архиве ресурсов для книги, которые можно загрузить по адресу: **https://www.nostarch.com/arduinoinventor/**, вы найдете скетч P8_DrawbotSerial.ino. С помощью этого скетча из монитора порта на Arduino можно отправлять три числа для управления двигателями A и B и длительностью времени движения.

Код скетча не приводится в книге для краткости, так что откройте файл скетча в каком-либо текстовом редакторе и просмотрите его. В скетче объявляются три переменные для хранения значений скорости двигателей A и B и времени задержки. Это те три числа, которые мы отправляем из окна монитора порта на Arduino. Связь с Arduino инициализируется инструкцией Serial.begin(9600);, позволяющей нам отправлять и получать данные посредством монитора порта. Arduino считывает данные, введенные в монитор порта, и присваивает их значения переменным speedA, speedB и delayTime, которые используются в уже знакомых нам функциях setMotorA(), setMotorB() и delay(). Более подробное объяснение работы кода вы найдете в комментариях к нему.

Загрузите скетч в Arduino, откройте окно монитора порта, введите в текстовую строку числа: 100, –100 и 500 и нажмите клавишу <Enter> или кнопку **Send** (рис. 8.22). Выполняя эти команды, Рисобот должен будет в течение полсекунды двигаться вперед, а затем остановиться.

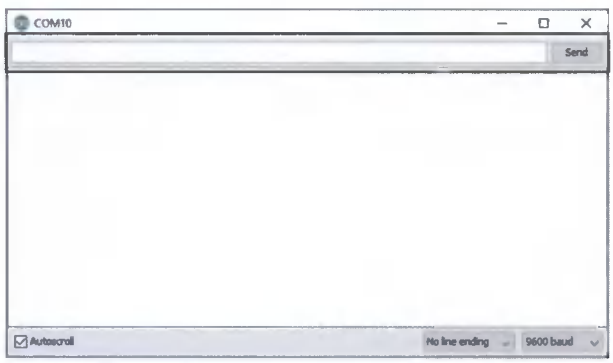


Рис. 8.21. Окно монитора порта с текстовой строкой и кнопкой **Send**

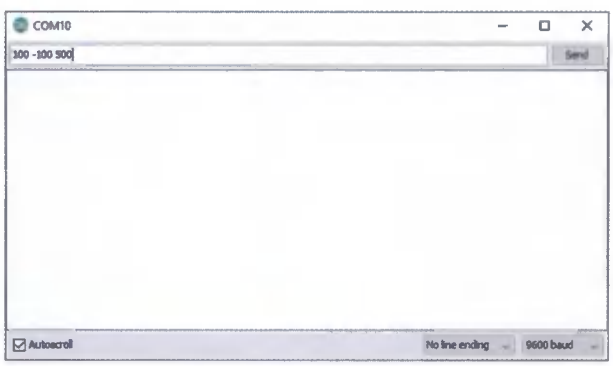


Рис. 8.22. Интерактивное управление Рисоботом

Что ж, теперь вы можете рисовать с помощью Рисобота различные узоры без необходимости загружать код для каждого изменения маршрута. Код с заданными в окне монитора порта параметрами выполняется роботом всего лишь один раз, а затем он ожидает нового набора параметров. Но что будет, если ввести шесть чисел, — например: 100 -100 500 -100 -100 250? Попробуйте задать Рисоботу узор, представленный последовательностью чисел.

Модифицируем код

Какие другие фигуры можно запрограммировать для рисования Рисоботом? Примените свои знания цикла for() и попробуйте модифицировать код для рисования треугольника или звезды. Скорее всего, вам придется немного поэкспериментировать, чтобы определить подходящие значения для скорости и пауз. А что будет, если одновременно вращать только одно колесо робота?

Бонус

Архив ресурсов для книги, доступный по адресу: **https://www.nostarch.com/arduinoinventor/**, содержит бонусный скетч, с помощью которого для управления Рисоботом можно применять даже еще более простые команды, — например: fd 10 или bk 10, чтобы проехать 10 шагов вперед или назад соответственно. Скетч называется P8_BonusTurtle.ino. Откройте среду разработки Arduino и загрузите этот скетч в окно редактора скетчей. Затем откройте окно монитора порта и введите несколько следующих команд: fd 10 — вперед на 10 шагов, bk 10 — назад на 10 шагов, rt 90 — поворот направо на 90 градусов и lt 90 — поворот налево на 90 градусов. Попробуйте нарисовать Рисоботом квадрат, управляя им с помощью этих команд.

ХРОНОМЕТРИСТ АВТОГОНОК

В *проекте 4* мы собрали измеритель скорости реакции на зажигающийся светодиод. А здесь мы используем приобретенные тогда знания, чтобы создать хронометриста для гоночной трассы (рис. 9.1). Мы сделаем наш проект независимым от компьютера — финишное время будет отображаться на небольшом жидкокристаллическом дисплее (ЖКД). Мы рассмотрим также, как модифицировать базовый проект, добавив вторую трассу и индикатор, называющий победителя гонок. Ну что ж, приступим?

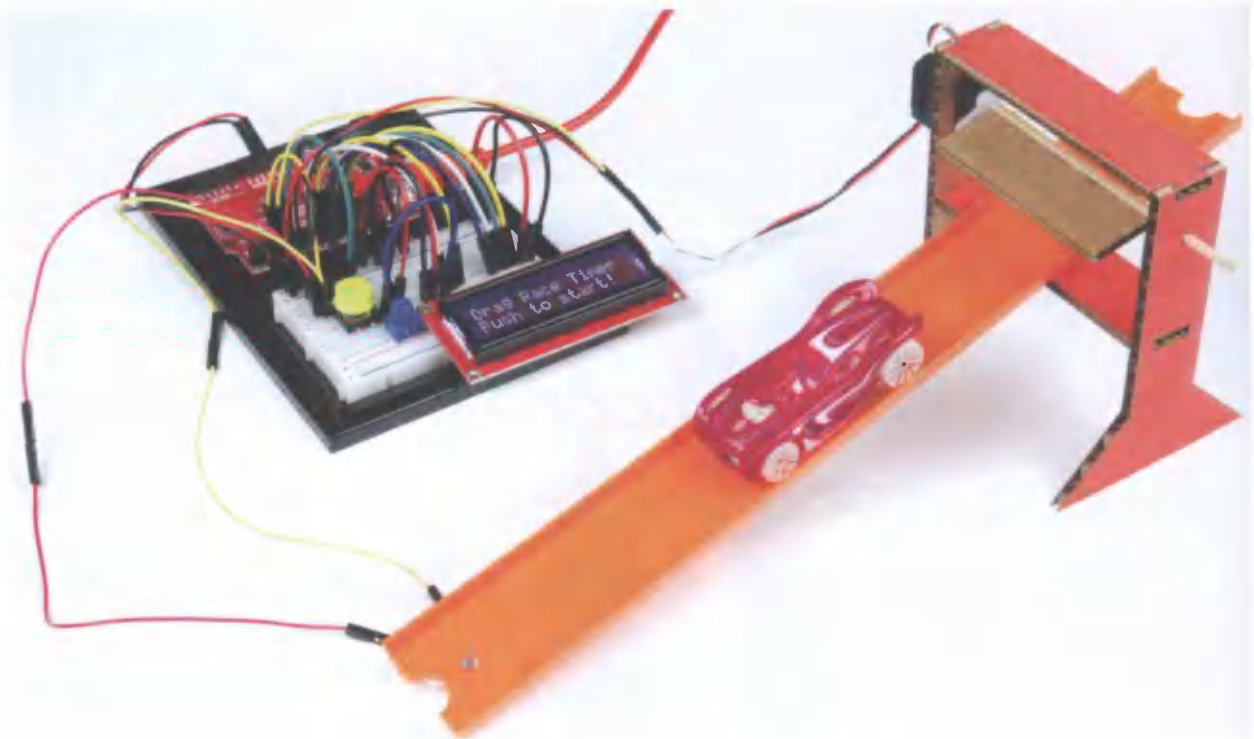


Рис. 9.1. Завершенный проект хронометриста автогонок

Необходимые компоненты, инструменты и материалы

Мы уже знакомы со многими из компонентов, используемых в этом проекте. Новый здесь только один компонент — жидкокристаллический дисплей (ЖКД), позволяющий отображать две строки по 16 символов. Финишное время будет отображаться на этом дисплее, а не выводиться в окне монитора порта.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 9.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт. ;
- беспаечная макетная плата (PRT-12002), 1 шт.;
- резистор 10 кОм (COM-08374 или COM-11508 для пакета, содержащего 20 шт.), 1 шт. Если вы захотите расширить проект до двух трасс, то потребуется два таких резистора;
- фоторезистор (SEN-09088), 1 шт. Если вы захотите расширить проект до двух трасс, то потребуется два таких фоторезистора;
- кнопка (COM-10302), 1 шт.;
- потенциометр 10 кОм (COM-09806), 1 шт.;
- жидкокристаллический дисплей, 2 строки по 16 символов (16×2) (LCD-00255), 1 шт.;
- миниатюрный серводвигатель (ROB-09065), 1 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026);
- проволочные перемычки со штекером на одном конце и гнездом на другом (PRT-09140)*.

Примечание

Компоненты, обозначенные звездочкой «*», не входят в состав стандартного комплекта изобретателя SparkFun Inventor's Kit, но предлагаются в отдельном дополнительном комплекте или могут быть приобретены вами по отдельности.

Примечание

В *разд. «Идем дальше...»* в конце главы рассматривается, как расширить проект до двух трасс и отображать финишное время победителя. Стандартный набор изобретателя компании SparkFun содержит всего лишь один фоторезистор (SEN-09088), но для расширенного проекта их потребуется два. Впрочем, фоторезисторы не являются дорогостоящими компонентами, поэтому можно или приобрести другой фоторезистор в радиомагазине, или одолжить его у приятеля.

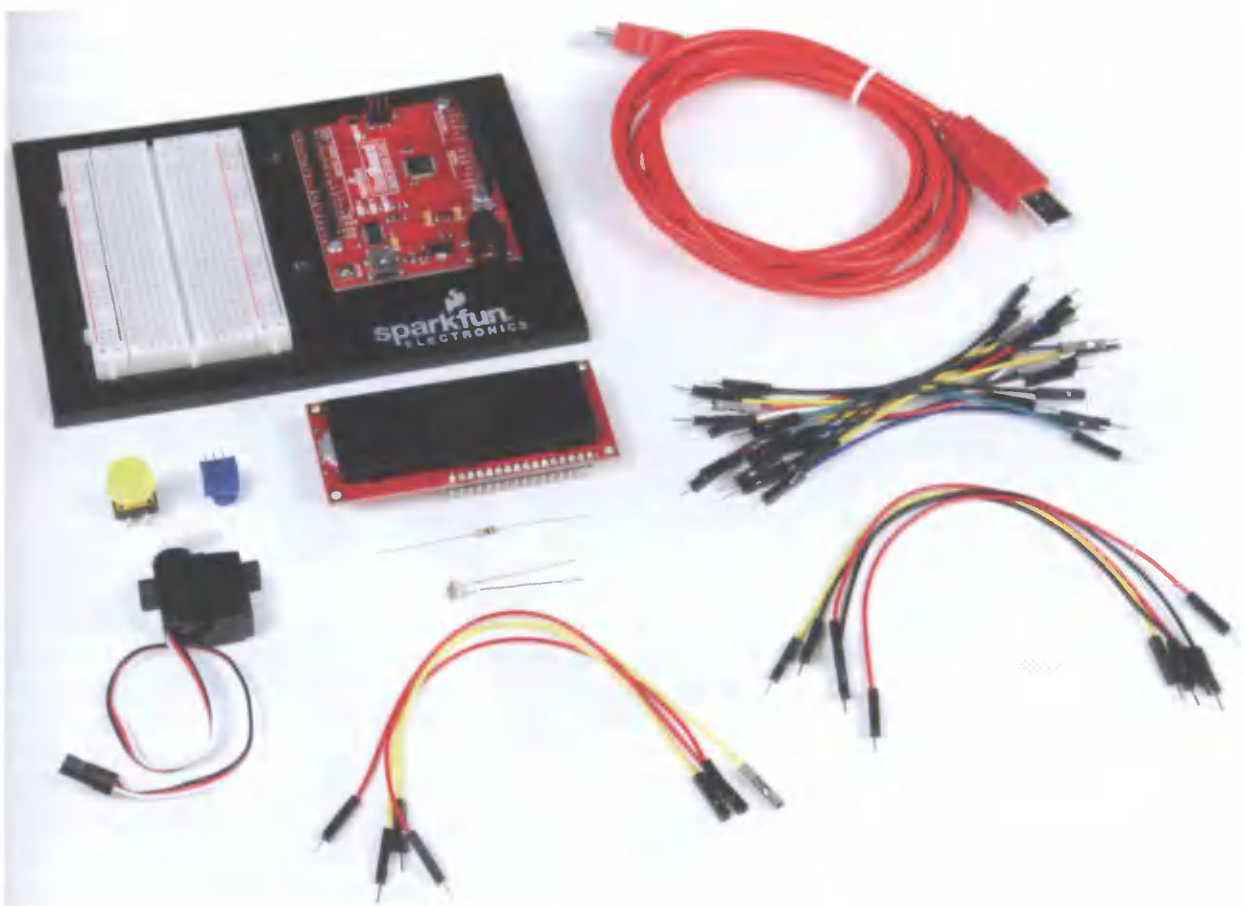


Рис. 9.2. Электронные компоненты для проекта хронометриста автогонок

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты и материалы (рис. 9.3):

- макетный нож;
- металлическая линейка;
- длинногубцы (игольчатые плоскогубцы);
- кусачки;
- клей (клеевой пистолет или клей для моделирования);
- клейкая лента типа «скотч»;
- гофрированный картон (приблизительно 21,5×30 см), небольшая картонная коробка или толстый открыточный картон;
- бамбуковая палочка;
- шаблон корпуса (см. рис. 9.15 далее в этом проекте);
- игрушечная гоночная машинка, например, из серии Hot Wheels[1] (на рис. 9.3 не показана);
- может пригодиться: игрушечная гоночная трасса (на рис. 9.3 не показана).

В предыдущих проектах информация, передаваемая от Arduino компьютеру, выводилась в окне монитора порта. В этом проекте мы добавим к проекту ЖКД, и полученные в процессе этого знания окажутся весьма полезными вам для дальнейшей работы с Arduino.

Для подключения ЖКД используется много проводов, но вам не стоит этим озабочиваться — мы спокойно подключим их один за другим. Научившись работать с ЖКД, вы сможете оснастить им свои предыдущие проекты, чтобы сделать их по-настоящему мобильными.

[1] См. **https://ru.wikipedia.org/wiki/Hot_Wheels**.

Рис. 9.3. Инструменты и материалы, рекомендуемые для проекта хронометриста автогонок

Новый компонент: жидкокристаллический дисплей

Для краткости в дальнейшем мы будем употреблять сокращенное название жидкокристаллического дисплея — ЖКД. Технология использования жидких кристаллов была разработана около 40 лет тому назад, и в настоящее время получила очень широкое применение: в цифровых часах, будильниках, проекторах, телевизорах, компьютерных мониторах и во многих других устройствах.

В этом проекте мы задействуем монохромный (одноцветный) ЖКД. Внутри ЖКД находится слой жидкого кристалла — особого химического вещества, способного под воздействием слабого электрического тока переходить из прозрачного состояния в непрозрачное. Разместив слой жидких кристаллов над освещаемой подложкой или зеркалом, можно создать очень простой дисплей. В зависимости от того, подается или нет электричество на определенные области жидкого кристалла, свет или проходит через него, или не проходит. Это означает, что управляя подачей тока на различные области жидкого кристалла, можно создавать на нем требуемые изображения.

Наш дисплей способен отображать 32 символа[2] — по 16 в каждой из двух строк. Каждый символ собирается из матрицы размером 5×8 пикселов. Под воздействием электрического тока, управляемого Arduino, пиксел матрицы может становиться прозрачным или непрозрачным. Например, чтобы отобразить букву A, пикселы матрицы, обозначенные на рис. 9.4 желтым, делаются непрозрачными.

Если каждая матрица содержит 40 отдельных пикселов, то весь дисплей содержит 32 × 40 = 1280 пикселов. Неужели для управления этими пикселами нам потребуется 1280 отдельных линий управления?! Не беспокойтесь — наш ЖКД оснащен встроенной микросхемой HD44780 драйвера ЖКД производства компании Hitachi. Эта микросхема позволяет отображать символы на дисплее, используя всего шесть сигнальных линий от Arduino. На рис. 9.5 показано размещение выводов микросхемы и их обозначения.

[2] Отображаются только буквы английского алфавита.

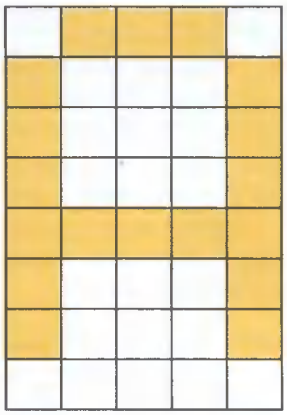


Рис. 9.4. Отображение заглавной буквы A в матрице размером 5×8 пикселов

Рис. 9.5. Простой ЖКД на 16×2 символа и обозначение его выводов

> **Примечание**
>
> Этот ЖКД имеет восемь выводов для передачи данных (D0–D7), но мы задействуем только четыре из них: D4–D7.

Всего ЖКД имеет 16 выводов, но для этого проекта мы задействуем только выводы 1–6 и 11–16. Если разместить ЖКД выводами кверху, то нумерация

их пойдет с 1 по 16 слева направо. В табл. 9.1 приводится описание назначения выводов ЖКД. В некоторых справочных листках определенные обозначения могут быть надчеркнуты, как обозначение Ч/$\overline{З}$ для вывода 5. Надчеркивание означает, что этот режим работы активируется при подаче на вывод сигнала низкого уровня. Таким образом, в таком случае, чтобы активировать режим записи данных на ЖКД, на его вывод 5 необходимо подать сигнал низкого уровня. Этот вопрос рассматривается более подробно в *разд. «Подключаем линии данных и управления»* далее в этом проекте.

Вместо того, чтобы управлять каждым из 40 пикселов матрицы по отдельности, микросхема драйвера ЖКД HD44780 интерпретирует данные, получаемые от Arduino по четырем линиям данных и двум линиям управления, преобразовывая эту информацию в символ для отображения. Интерфейс упрощается еще больше, благодаря наличию библиотеки функций для работы с ЖКД, собранной сообществом Arduino. Все это будет рассмотрено далее, при описании кода проекта.

Таблица 9.1. *Назначение выводов ЖКД на 16×2 символа*

Вывод	Описание
1	«Земля» (–)
2	Питание для ЖКД (5 В)
3	Настройка контраста (0–5 В)
4	Выбор регистра (ВР)
5	Выбор режима Ч/$\overline{З}$ (Чтение/$\overline{Запись}$)
6	Разрешающий сигнал
7–10	Линии данных D0–D3 (не используются)
11–14	Линии данных D4–D7 (данные передаются по 4 бита за раз)
15	Питание подсветки (5 В)
16	Питание подсветки (–/«Земля»)

Принцип работы хронометриста автогонок

Прежде чем приступать к монтажу электронных компонентов, давайте рассмотрим, как наш хронометрист функционирует: нажатие кнопки приводит в действие сервомашинку, которая открывает стартовые ворота, что позволяет игрушечной машинке катиться вниз по трассе. Одновременно Arduino записывает время старта и ожидает, пока машинка не затенит фоторезистор, вмонтированный в трассу в самом ее конце. Фоторезистор этот встроен в такую же схему светодатчика, что и в *проекте 5*. Когда фоторезистор перекрывается проезжающей над ним машинкой, Arduino определяет событие затенения, записывает финишное время и вычисляет время прохождения трассы, отнимая финишное время от стартового. Если эта схема выглядит похожей на измеритель реакции из *проекта 4* — ничего удивительного, поскольку это так и есть!

Собираем схему с ЖКД

Работу над проектом мы начнем со сборки схемы подключения ЖКД. Хотя ЖКД имеет 16 выводов, мы, как уже отмечалось ранее, задействуем только 12 из них. На рис. 9.6 показана принципиальная схема подключения ЖКД.

Со своими 16-ю выводами ЖКД займет 16 рядов макетной платы, поэтому нам нужно быть аккуратными с размещением компонентов этого проекта. Установим ЖКД в первые 16 рядов с правой стороны макетной платы. При этом не забудьте соединить перемычками разъемы +5 В и GND платы Arduino с шинами положительного и отрицательного питания с левой стороны макетной платы.

Обратите внимание на то обстоятельство, что выводы ЖКД не обозначены маркировкой. В процессе рассмотрения его подключения мы будем ссылаться на выводы ЖКД по их номерам, начиная с самого нижнего вывода 1.

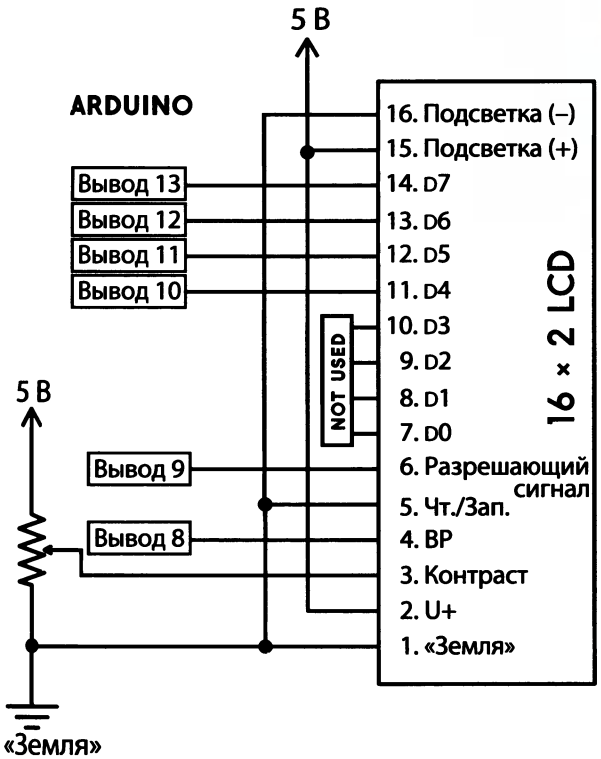


Рис. 9.6. Принципиальная схема подключения ЖКД

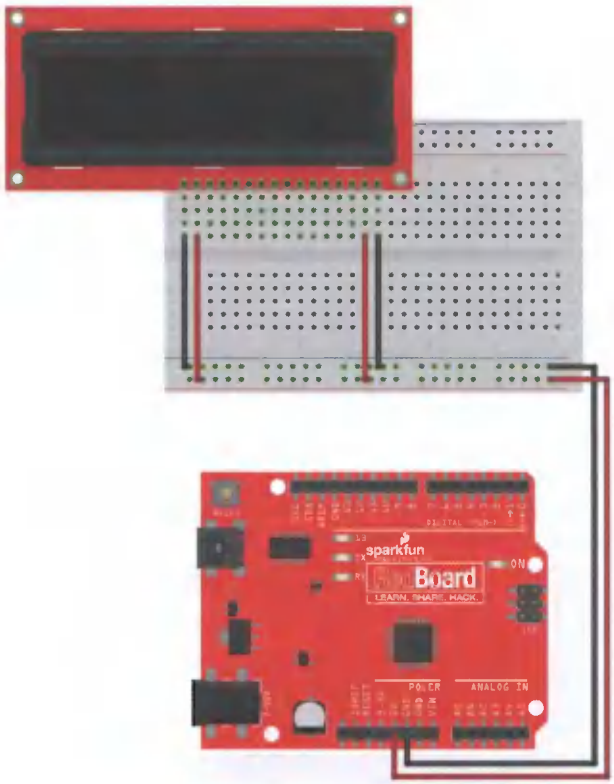


Рис. 9.7. Подключение питания для схемы управления и для схемы подсветки ЖКД

Подключаем питание ЖКД

Модуль ЖКД состоит, по сути, из двух отдельных устройств: собственно ЖКД и микросхемы для управления им. Для каждого из этих устройств требуется отдельный источник питания (в случае ЖКД питание требуется для его подсветки).

Подключите вывод 1 ЖКД к шине отрицательного питания макетной платы, а вывод 2 — к шине положительного питания. Это обеспечит питание для схемы управления ЖКД и микросхемы драйвера ЖКД HD44780. Затем подключите вывод 16 ЖКД к шине отрицательного питания макетной платы, а вывод 15 — к шине положительного питания. Это обеспечит питание для схемы подсветки ЖКД. Монтажная схема подключения питания показана на рис. 9.7.

Настройка контраста ЖКД

Контраст ЖКД регулируется изменением напряжения на его выводе 3. Это можно осуществлять с помощью простого делителя напряжения на потенциометре, подобно тому, как это делалось в *проекте 6*. Вспомним, что потенциометр является переменным резистором с тремя выводами, и при вращении вала потенциометра сопротивление между центральным выводом и обоими концевыми выводами меняется. Если подключить верхний и нижний выводы потенциометра к положительному и отрицательному («земле») питанию соответственно, мы получим регулируемый делитель напряжения. В частности, в зависимости от величины поворота вала, напряжение на центральном выводе потенциометра будет варьироваться в диапазоне от 5 до 0 В (рис. 9.8).

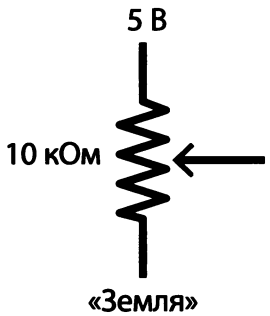


Рис. 9.8. Принципиальная схема подключения потенциометра в качестве регулируемого делителя напряжения

Вставьте потенциометр в гнезда макетной платы чуть ниже ЖКД. Подключите крайние выводы потенциометра к шинам положительного и отрицательного питания макетной платы, а центральный вывод — к выводу 3 ЖКД для управления контрастом.

Теперь нам осталось выполнить монтаж линий данных и управления для ЖКД.

Подключаем линии данных и управления

Чтобы завершить подключение ЖКД, нам потребуется еще семь перемычек: четыре для линий данных и три для линий управления. Вывод 5 ЖКД управляет режимом чтения/записи, позволяя Arduino считывать данные с дисплея и записывать данные в него. Мы будем использовать ЖКД только в режиме записи, то есть только передавать на него данные. Поэтому подключим этот вывод к шине отрицательного питания («земле»). Обращаясь к табл. 9.1, можно заметить, что часть «Запись» маркировки Чтение/$\overline{\text{Запись}}$ надчеркнута. Как упоминалось ранее, в документации на ЖКД надчеркивание часто обозначает, что данный режим работы активируется сигналом низкого уровня. Сигнал низкого уровня эквивалентен 0 В («земле»), поэтому мы подключаем вывод 5 ЖКД к шине отрицательного питания макетной платы, как показано на рис. 9.9.

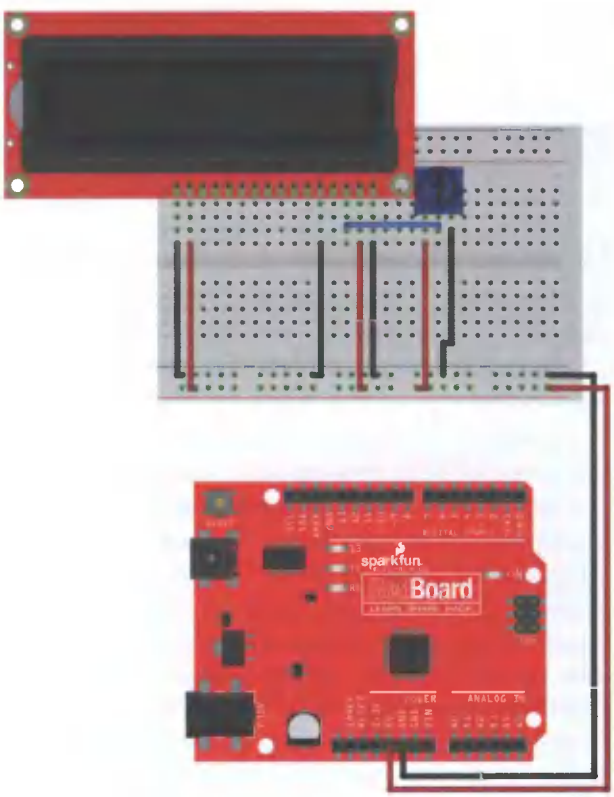


Рис. 9.9. Конфигурация ЖКД на запись: подключаем вывод 5 к шине отрицательного питания

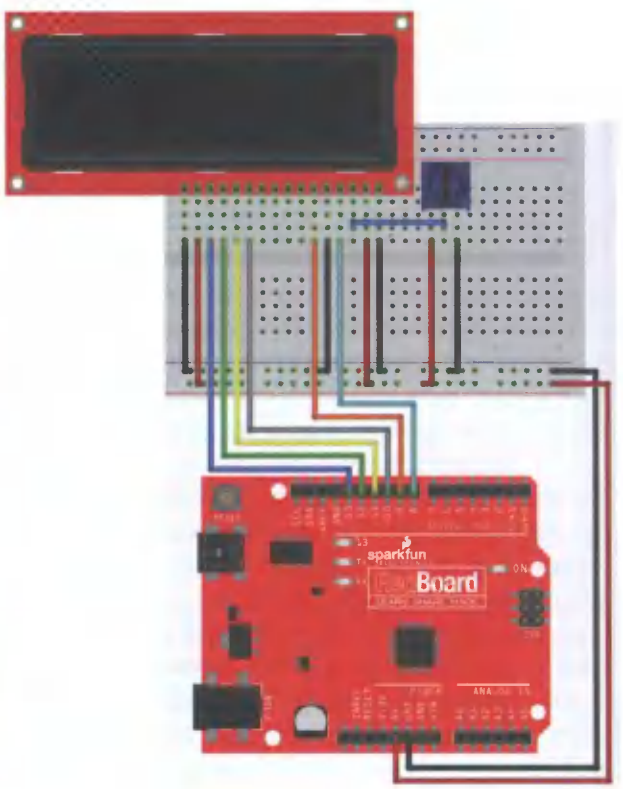


Рис. 9.10. Полностью собранная схема подключения ЖКД

Оставшимися шестью проводами мы подключаем ЖКД к Arduino. Для приема данных с Arduino будут использоваться выводы 11–14 ЖКД. Подключите эти выводы ЖКД к выводам 10, 11, 12 и 13 платы Arduino, как показано на рис. 9.10. Провода этих подключений должны идти напрямую с платы Arduino к ЖКД, не допуская перекрещиваний.

Напоследок нам нужно подключить вывод 6 для сигнала разрешения ЖКД и вывод 4 для управления выбором регистра. На вывод 6 подается на ЖКД сигнал разрешения передачи данных, а сигнал на выводе выбора регистра определяет, каким образом интерпретировать принимаемые данные: или как символ для отображения, или как инструкцию, например, очистки дисплея или перемещения курсора. Это позволяет более гибко управлять отображаемым на дисплее содержимым. Итак, подключите вывод 9 платы Arduino к выводу 6 ЖКД, а вывод 8 Arduino — к выводу 4 ЖКД, как показано на рис. 9.10.

В табл. 9.2 приводится полный список подключений выводов ЖКД к питанию и выводам Arduino — это поможет вам проверить правильность монтажа ЖКД.

Таблица 9.2. *Подключения выводов ЖКД к питанию и выводам Arduino*

Вывод ЖКД	Подключение
16	«Земля» (0 В)
15	5 В
14	Вывод 13 Arduino
13	Вывод 12 Arduino
12	Вывод 11 Arduino
11	Вывод 10 Arduino
10	Не используется
9	Не используется
8	Не используется
7	Не используется
6	Вывод 9 Arduino
5	«Земля» (0 В)
4	Вывод 8 Arduino
3	Средний вывод потенциометра
2	5 В
1	«Земля» (0 В)

Проверяем работу ЖКД

Прежде чем продолжать сборку проекта, проверим работу уже подключенного ЖКД.

Итак, подключите Arduino к компьютеру. Сразу же при подключении должна загореться подсветка ЖКД. Попробуйте вращать ручку потенциометра. Даже когда на дисплее ничего не отображается, должно наблюдаться изменение контраста: от полностью темного экрана при наименьшем контрасте до ярких 32 прямоугольников при наивысшем. Если такое изменение контраста не наблюдается, проверьте правильность подключения ЖКД. Проверьте, что все подключения соответствуют указанным в табл. 9.2.

Убедившись в правильной работе ЖКД, создайте новый скетч в среде разработки Arduino, скопируйте в него код из листинга 9.1 и загрузите скетч в Arduino. В результате исполнения этого скетча на ЖКД должен отображаться текст **SparkFun Arduino** на первой строке и счетчик истекших секунд на второй.

Листинг 9.1. Проверочный код для отображения на ЖКД текста и счетчика секунд

```
❶ #include<LiquidCrystal.h>
❷ LiquidCrystal lcd(8, 9, 10, 11, 12, 13);

  void setup()
  {
❸   lcd.begin(16, 2);      //Инициализация интерфейса
                           //с ЖКД
❹   lcd.clear();
❺   lcd.print("SparkFun Arduino");
  }

  void loop()
  {
❻   lcd.setCursor(0, 1); //Перемещаем курсор на 2-ю
                         //строку (столбец 0, ряд 1)
❼   lcd.print(millis()/1000); // Выводим число истекших
                              //секунд
  }
```

Давайте разберемся, что происходит в этом коде. Сначала в скетч включается библиотека LiquidCrystal.h ❶, созданная сообществом разработчиков Arduino для упрощения работы с шестью разными линиями данных и управления. С помощью этой библиотеки будет легче отправлять инструкции для ЖКД.

> **Примечание**
>
> Библиотека LiquidCrystal поддерживает только символьные ЖКД. Для графических ЖКД используется другая библиотека — OpenGLCD, которая поддерживает отображение графических элементов: линий, прямоугольников, кругов, а также текста.

Далее в коде создается объект с именем lcd, который использует библиотеку *LiquidCrystal* ❷. Обратите внимание, что при создании объекта в качестве параметров ему передается набор выводов Arduino, соответствующих выводам ЖКД: Выбор Регистра, Разрешающий сигнал и четыре вывода линий данных. Таким образом выполняется конфигурирование выводов Arduino для управления функциями ЖКД. В некоторых справочных материалах эта команда дается в виде: LiquidCrystal lcd(RS, Enable, d4, d5, d6, d7).

Возможно, вы задаетесь вопросом, почему мы используем только четыре вывода данных ЖКД, а не все восемь. Конкретный тип ЖКД может выполнять обмен данными, используя как восемь, так и четыре линии данных. Согласно сопроводительной документации на ЖКД, при использовании четырех линий данных задействуются верхние четыре вывода ЖКД, обозначенные D4, D5, D6 и D7. Хотя при работе в таком режиме передача данных на ЖКД занимает вдвое больше времени, это значительно упрощает схему. Кроме того, не забывайте, что Arduino работает на частоте 16 МГц. А это очень высокая скорость работы!

Библиотека LiquidCrystal содержит около 20 разных команд, упрощающих управление ЖКД. В этом примере мы рассмотрим несколько базовых команд для настройки размера экрана, очистки экрана, отображения информации и перемещения курсора.

Блок setup() скетча содержит несколько инструкций, которые исполняются только один раз при включении или сбросе Arduino. Первая из этих команд: lcd.begin(16,2); ❸ — задает работу ЖКД в режиме 16×2 (2 строки по 16 символов каждая), позволяя библиотеке правильно заверстывать текст и переходить с одной строки на другую.

Следующая инструкция: lcd.clear(); ❹ — очищает экран для отображения нового текста. При этом курсор устанавливается на первой позиции первой строки экрана. Эта команда позволяет убрать с дисплея все, что было на нем отображено ранее.

Затем команда lcd.print("SparkFun Arduino"); ❺ выводит на экран текст: SparkFun Arduino. Так как перед этим мы очистили экран с помощью команды lcd.clear(), этот текст выводится на первой строке дисплея. Поскольку он содержит ровно 16 символов, то и должен заполнить всю первую строку ЖКД. Команда lcd.print() похожа на Serial.print(), но не требует подключения Arduino к компьютеру и открытия окна монитора порта для отображения информации с Arduino.

Цикл loop() обновляет экран, записывая на него новую информацию при каждом исполнении. Сначала курсор переводится на вторую строку ЖКД с помощью команды lcd.setCursor(0, 1); ❻ — чтобы не затирать текст SparkFun Arduino в первой строке. Параметры, передаваемые методу setCursor(), указывают позицию символа в строке (0) и номер строки (1). Как и в случае с большинством сред программирования, Arduino начинает счет с 0, а не с 1.

Наконец, с помощью другой инструкции lcd.print() на экран выводится значение счетчика секунд ❼. Число секунд вычисляется делением на 1000 значения миллисекунд, прошедших с момента подачи питания на Arduino, число которых определяется

с помощью функции millis(). Подобный метод будет использоваться в работе нашего хронометриста.

В качестве небольшого теста на усвоение материала попробуйте откорректировать скетч, чтобы выводить свое имя в первой строке ЖКД. Также попробуйте выводить истекшее время в минутах, а не в секундах. Поиграйте немного с кодом этого примера, пока не почувствуете уверенность в своих способностях выводить данные на ЖКД. Задействовав шесть выводов общего назначения Arduino, можно добавить табло для вывода данных в любой проект!

В нашем примере рассмотрены только наиболее часто используемые инструкции библиотеки LiquidCrystal для Arduino. Информацию о других доступных командах можно найти в руководстве по этой библиотеке по адресу: **https://www.arduino.cc/en/Reference/LiquidCrystal/**.

Теперь, когда у нас есть работающий ЖКД, можно добавить в схему кнопку, сервомашинку и светодатчик.

Добавляем остальные компоненты

В этом проекте используются некоторые компоненты, с которыми мы уже познакомились в предыдущих проектах: кнопка для старта гонок, сервомашинка для управления стартовыми воротами и фоторезистор для определения момента финиша. На рис. 9.11 показаны принципиальные схемы подключения этих трех компонентов к Arduino.

Установите кнопку на макетную плату над ее центральным углублением таким образом, чтобы разомкнутые пары выводов оказались по разным сторонам углубления. Подключите один из разомкнутых выводов на левой стороне макетной платы к выводу 5 платы Arduino, а другой — к шине отрицательного питания («земле»). Компоненты для этого проекта займут бо□льшую часть макетной платы, поэтому внимательно следите за использованием рядов макетной платы и соединением компонентов. Чтобы сэкономить место на макетной плате, мы не используем совместно с кнопкой

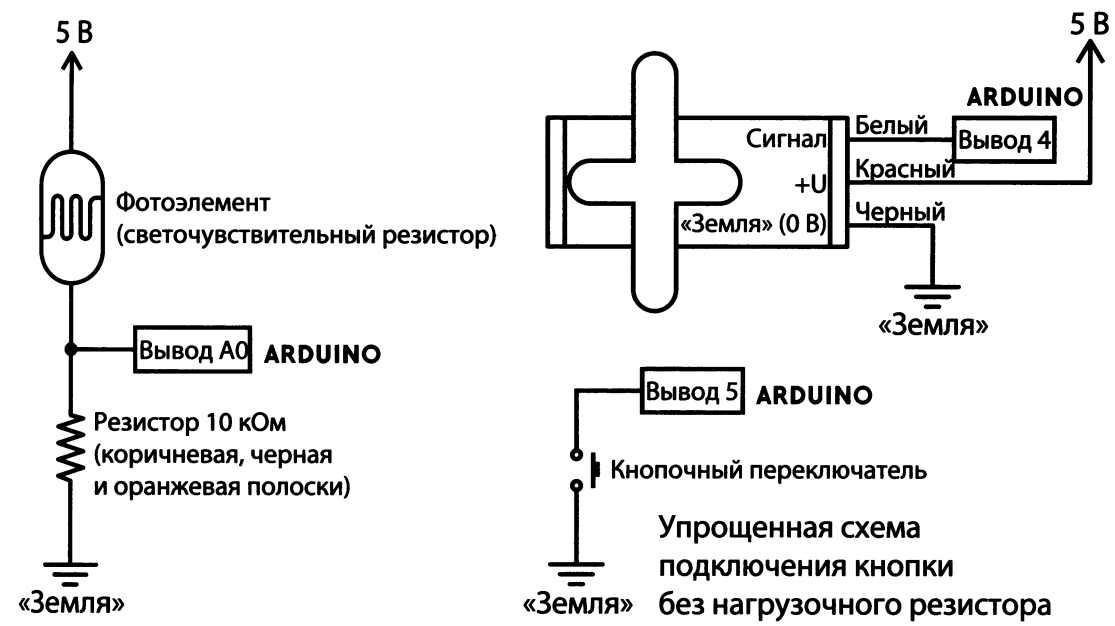


Рис. 9.11. Принципиальные схемы подключения остальных компонентов проекта хронометриста автогонок: фоторезистора (*слева*), сервомашинки (*справа вверху*) и кнопки (*справа внизу*)

внешний нагрузочный резистор, как в *проекте 4*, а задействуем нагрузочный резистор, встроенный в Arduino.

Далее, подключим сервомашинку, которая будет открывать стартовые ворота. Для этого берем три перемычки со штыревыми разъемами на обоих концах и подключаем сигнальный вывод сервомашинки (белый провод) к выводу 4 платы Arduino, красный провод — к шине положительного питания макетной платы, а черный провод — к шине отрицательного питания («земле»).

Наконец, подключим в схему фотодатчик с делителем напряжения: один вывод фоторезистора подключаем к шине положительного питания, а другой — через последовательный резистор сопротивлением 10 кОм — к шине отрицательного питания («земле»). Кроме того, вывод фоторезистора, к которому подключен резистор сопротивлением 10 кОм, подключаем к выводу A0 платы Arduino. Эта часть схемы должна выглядеть похожей на схему ночника из *проекта 5*.

Схема этого проекта содержит много компонентов, поэтому уделите достаточное время проверке правильности ее монтажа, сверяя его с монтажной схемой на рис. 9.12.

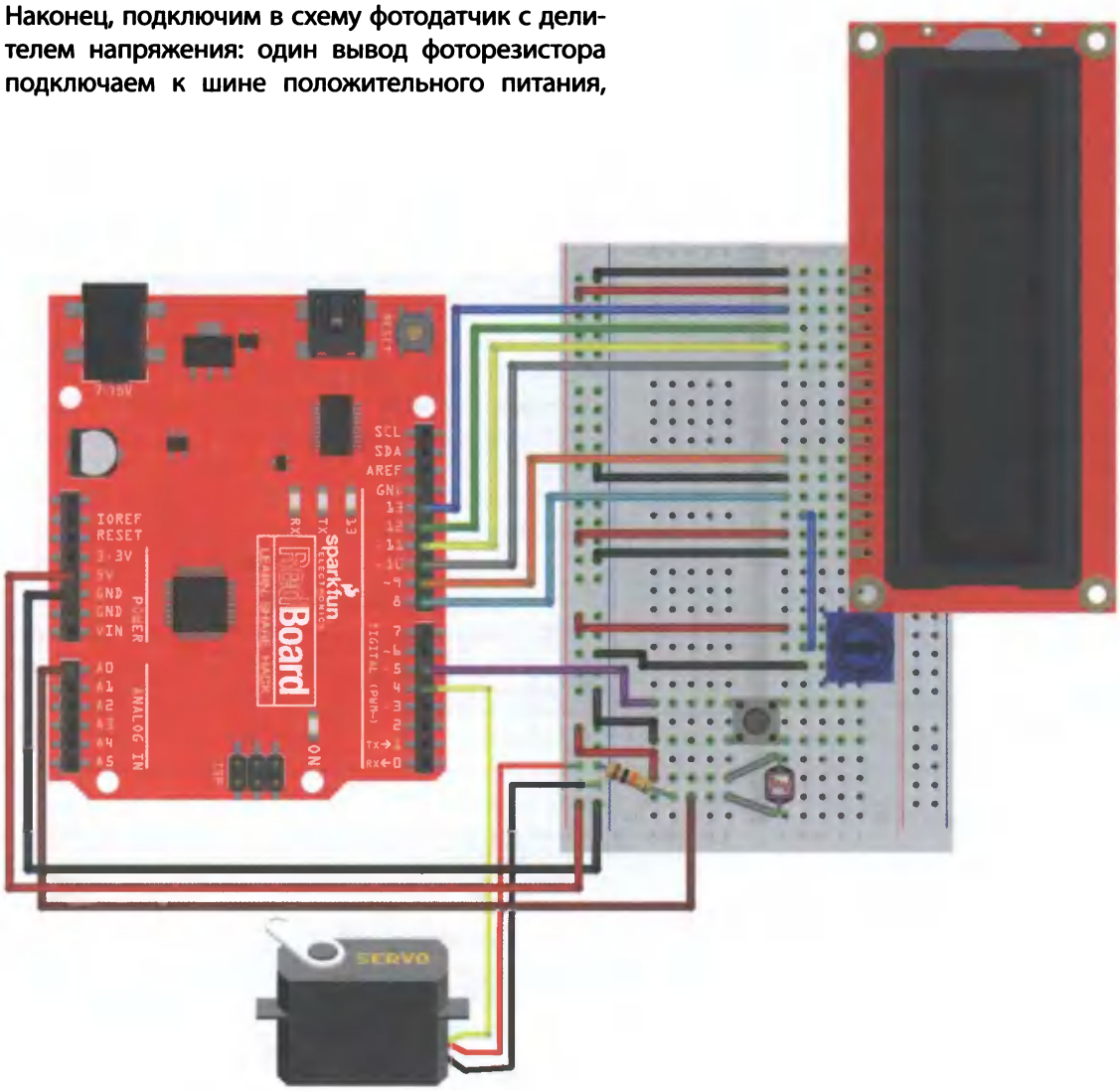


Рис. 9.12. Полностью собранная схема хронометриста автогонок, включающая кнопку старта и сервомашинку для управления стартовыми воротами

Программа для хронометриста автогонок

Теперь нам нужна программа для управления совместной работой всех этих компонентов. Создайте новый скетч и введите в него код из листинга 9.2. Код можно также скопировать из файла ***P9_Listing9_2_SingleCarTimer***, содержащегося в архиве ресурсов для книги, который можно загрузить по адресу: **https://www.nostarch.com/arduinoinventor/**. В этом коде сведены воедино несколько концепций и идей, реализованных в предыдущих проектах.

Листинг 9.2. Скетч для управления хронометристом автогонок

```
❶ #include<LiquidCrystal.h>
  #include<Servo.h>

  LiquidCrystal lcd(8, 9, 10, 11, 12, 13);
❷ Servo startingGate;

❸ const byte buttonPin = 5;
  const byte servoPin = 4;
  const byte finishSensor1Pin = A0;
  const int darkThreshold = 500;

❹ int finishSensor1;
  boolean finishFlag = false;
  long startTime;
  long stopTime;
  float raceTime;

  void setup()
  {
❺   pinMode(buttonPin, INPUT_PULLUP);

    startingGate.attach(servoPin, 1000, 2000);
    startingGate.write(0);

❻   lcd.begin(16, 2);
    lcd.clear();
    lcd.print("Drag Race Timer");     // Хронометрист
                                      //автогонок
    lcd.setCursor(0, 1);
    lcd.print("Push to start!");      // Нажмите кнопку
                                      //для старта

❼   while (digitalRead(buttonPin) == HIGH)
    {
    }

    lcd.clear();
    lcd.print("Go!");       // Марш!

    startingGate.write(180);
    startTime = millis();
  }

  void loop()
  {
❽   finishSensor1 = analogRead(finishSensor1Pin);
❾   if ((finishFlag == false) && (finishSensor1 <
      darkThreshold))
    {
      finishFlag = true;
      stopTime = millis();
      raceTime = stopTime - startTime;

      lcd.clear();
      lcd.print("Finish Time:");       //Финишное время
      lcd.setCursor(0, 1);
❿     lcd.print(raceTime / 1000, 3);
    }
  }
```

Давайте разберемся, как все это работает. Первым делом в скетч включаются две библиотеки: LiquidCrystal.h и Servo.h ❶. Затем инициализируются объект LiquidCrystal с именем lcd и объект Servo с именем startingGate ❷.

Далее объявляется набор констант для номеров выводов Arduino, используемых со схемами кнопки, сервомашинки и фоторезистора ❸. Использование констант вместо соответствующих номеров выводов удобно тем, что если в будущем мы поменяем вывод Arduino, служащий для управления каким-либо компонентом, нам нужно будет сделать соответствующую корректировку только в объявлении константы для этого вывода, а не искать номер вывода по всему коду, что чревато ошибками. Последняя константа, darkThreshold, служит для задания порогового уровня освещения, которое нужно определить, когда автомобиль блокирует светодатчик. В данном случае этот уровень установлен равным 500 — грубо посередине диапазона 0–1023, но вам может потребоваться изменить это значение в соответствии с условиями освещенности вашей комнаты.

Затем в скетче объявляются несколько переменных ❹. Переменная finishSensor1 используется для хранения значения светодатчика. Следующая переменная, finishFlag, является переменной состояния и служит для хранения значения состояния скетча. Этой переменной присваивается исходное значение false (ложь), которое меняется на true (истина), когда автомобиль пересекает финишную черту, фиксируя таким образом окончание гонки. Смена значения этой переменной осуществляется кодом на основе значения светодатчика. Следующие три переменные служат для вычисления времени гонки, используя встроенный в Arduino таймер millis().

Далее следует блок setup(). В первой инструкции блока вывод Arduino, служащий для считывания состояния кнопки, конфигурируется на работу в режиме приема данных, а также на использование внутреннего повышающего резистора (INPUT_PULLUP) ❺. Так устраняется необходимость использования внешнего нагрузочного резистора, как это делалось в *проекте 4*.

Затем скетч инициализирует сервомашинку и устанавливает ее в положение по умолчанию — 0 градусов. В таком положении будут находиться стартовые ворота в закрытом состоянии.

Далее скетч отображает на ЖКД краткую информацию ❻ для пользователя о том, как начинать процесс гонок. Эти несколько строк кода инициализируют ЖКД, очищают экран и отображают две строки текста. Следите за тем, чтобы текст в каждой строке не превышал 16 символов, — избыточные символы будут «убегать» с экрана с правой стороны. Затем исполняется блокирующий цикл while() ❼, который приостанавливает исполнение скетча, пока не будет нажата кнопка старта. При нажатии кнопки старта функция digitalRead(buttonPin) считывает сигнал низкого уровня на выводе 5 (константа buttonPin) Arduino, код перемещает вал сервомашинки в верхнее положение и присваивает значение переменной startTime.

В цикле loop() скетч считывает значение светодатчика и сохраняет текущее показание в переменной finishSensor1 ❽. При пересечении финишной линии автомобиль проезжает над светодатчиком, встроенным в поверхность трассы в ее конце, перекрывая его освещение. Подобно тому, как это делалось в *проекте 5*, скетч сравнивает значение датчика с пороговым значением затенения darkTreshold.

Примечание

Необходимо разместить трассу проекта в хорошо освещенном месте — чтобы разница между освещенным и затененным датчиком была способна вызвать его срабатывание. Имейте в виду, что верхнее освещение может вызывать ложные срабатывания при падении на датчик, например, тени от вашего тела. Для надежной работы датчика лучше всего освещать его с помощью размещенной над ним небольшой настольной лампы.

Не забывайте также, что за время, в течение которого автомобиль проезжает над датчиком, цикл loop() может выполниться несколько раз.

Поскольку нам нужно уловить только тот первый момент, когда автомобиль пересекает финишную черту, скетч использует оператор if() с составным условием ❾ — чтобы определить момент, когда переменная finishFlag имеет значение false (ложь) и одновременно затенен фоторезистор (то есть когда значение напряжения фоторезистора меньше, чем пороговое значение darkThreshold). Двойной символ && обозначает логическую операцию И (см. *врезку «Составные логические операторы»* далее). Очень внимательно следите за количеством скобок оператора if(), поскольку скобки определяют порядок выполнения операции и применения логики. Такое использование составного оператора if() позволяет уловить — когда значение переменной состояния finishFlag меняется на true (истина) — только первое событие пересечения автомобилем датчика.

Затем скетч записывает финишное время и вычисляет время прохождения автомобилем трассы. Наконец, скетч отображает время прохождения трассы на ЖКД.

Переменная raceTime объявляется с типом данных float (с плавающей запятой), чтобы она могла сохранять числа с десятичными дробями. По умолчанию метод lcd.print() отображает десятичные дроби с точностью до двух цифр после запятой, но количество отображаемых цифр можно уменьшить или увеличить, передавая этому метода второй параметр. В точке ❿ скетч преобразовывает время прохождения трассы в секунды, разделяя на 1000 время прохождения в миллисекундах. Второй параметр в инструкции lcd.print(raceTime / 1000, 3); дает указание Arduino отображать время прохождения с тремя цифрами после запятой, так что это время будет отображаться с точностью до миллисекунд.

Не забудьте и о двух фигурных скобках в конце кода. Тщательно проверьте, что ваш код в точности совпадает с кодом в листинге 9.2, а затем загрузите скетч в свою плату Arduino.

СОСТАВНЫЕ ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Реализуя *проект 4*, мы разобрались, как сравнивать два значения с помощью простых логических операторов сравнения. Вспомним, что результат логического сравнения может быть только true (истина) или false (ложь). Но иногда в программе необходимо оценить одновременно несколько условий. Например, когда определенная переменная имеет значение false, а значение датчика ниже порогового значения: ((finishFlag == false) && (finishSensor1 < darkThreshold)). В этом примере обратите внимание на то, что операции логического сравнения сгруппированы в скобках по обе стороны оператора И (AND), представленного двойным символом &&.

Комбинация двух или более логических сравнения называется *составной логической операцией* или *составным логическим выражением*. Логические выражения вычисляются слева направо. Для упорядочивания составных частей логических выражений и обеспечения правильного порядка их вычисления рекомендуется заключать отдельные выражения в скобки. Основными двумя логическими операторами для создания составных логических выражений являются операторы И и ИЛИ, которые представлены в табл. 9.3.

Таблица 9.3. *Использование операторов И и ИЛИ в составных логических выражениях*

Обозначение	Составной оператор	Описание
(выражение A) && (выражение B)	И	Оба выражения: A и B — должны иметь значение ИСТИНА
(выражение A) II (выражение B)	ИЛИ	Или выражение A, или выражение B должно иметь значение ИСТИНА

Быстрая проверка

Если вы выполнили монтаж без ошибок, и код скетча также не содержит ошибок, по загрузке скетча в Arduino вал сервомашинки должен провернуться в положение 0, а на экране ЖКД должно отобразиться сообщение, как показано на рис. 9.13. Если текст выводится с искажениями или вообще выводится бессмыслица, проверьте правильность и надежность монтажа ЖКД, кнопки и фотодатчика.

Нажмите кнопку и наблюдайте за происходящим — вал сервомашинки должен провернуться, а на ЖКД должно отобразиться сообщение **Go!** (Марш!). Затените фоторезистор пальцем — на ЖКД должно отобразиться сообщение **Finish Time:** (Финишное время:) и время, прошедшее с момента нажатия кнопки до момента затенения фоторезистора (рис. 9.14).

Если фотодатчик не срабатывает так, как вы этого от него ожидаете, попробуйте изменить пороговое значение darkThreshold: если фотодатчик слишком чувствительный или активируется преждевременно, уменьшите пороговое значение darkThreshold, а если затенение фотодатчика не вызывает никакой реакции, попробуйте это значение увеличить. Откорректировав скетч, снова загрузите его в Arduino и повторите проверку.

Убедившись, что электронная часть функционирует должным образом, приступайте к работе над гоночным комплексом в составе стартовой башни и собственно гоночной трассы.

Рис. 9.13. Содержимое ЖКД перед началом гонок

Рис. 9.14. Отображение на ЖКД финишного времени

Собираем гоночный комплекс

Гоночный комплекс содержит стартовую башню с вращающимися стартовыми воротами, которые управляют выпуском автомобиля на трассу, и собственно трассу.

Для трассы можно использовать или часть игрушечной гоночной трассы из набора Hot Wheels или смастерить ее самому из картона. PDF-файлы с шаблонами стартовой башни и гоночной трассы находятся в папке ***P9_Templates*** архива ресурсов для книги, которые можно загрузить по адресу **https://www.nostarch.com/arduinoinventor/**. Шаблон для самодельной стартовой башни показан на рис. 9.15.

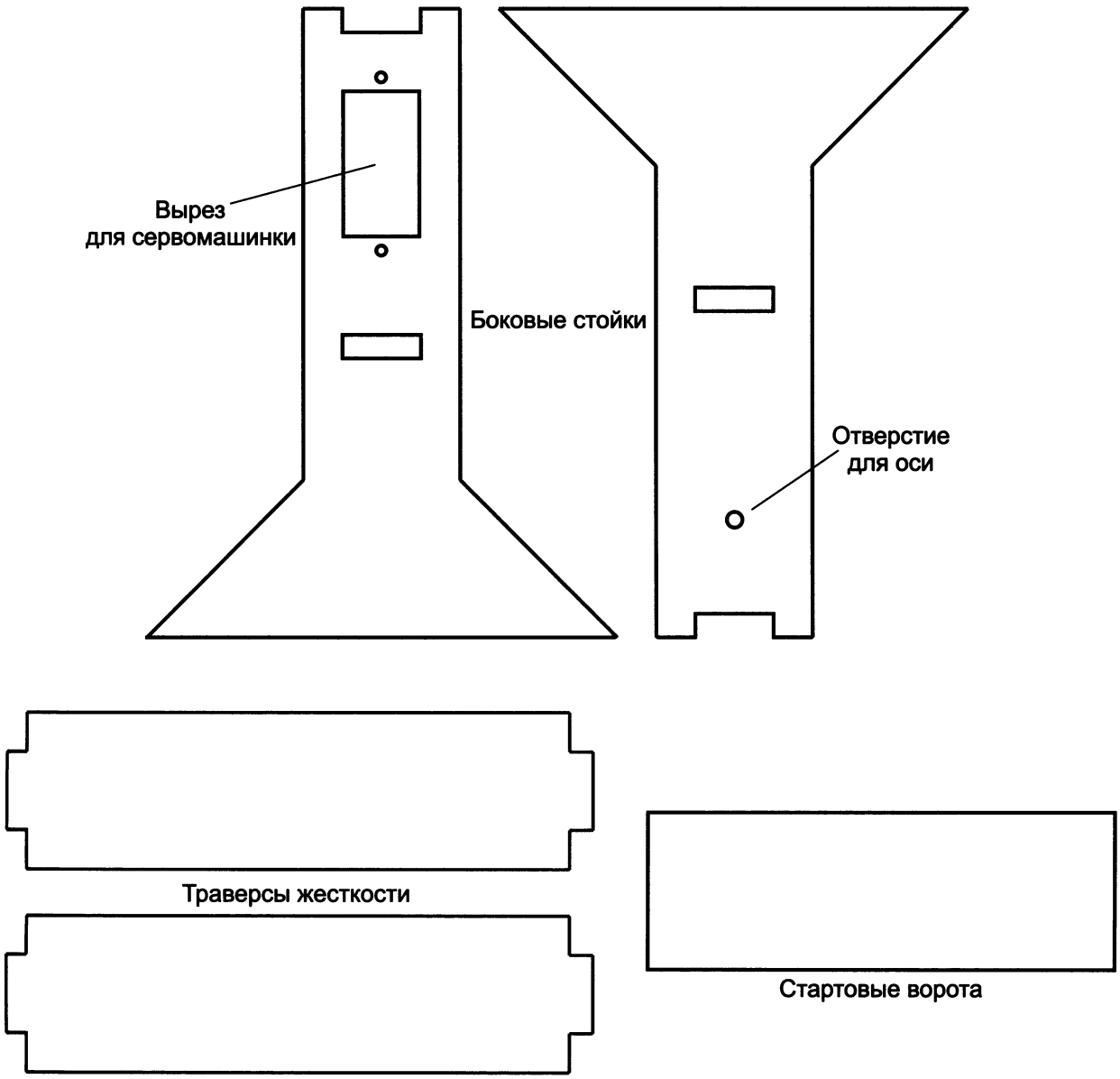


Рис. 9.15. Шаблон для деталей стартовой башни (в уменьшенном виде)

Собираем стартовую башню

Перенесите контуры шаблона деталей стартовой башни на картон, а затем вырежьте их по намеченным линиям. В одной из боковых стоек башни нужно вырезать отверстие для установки сервомашинки, а во второй — сделать отверстие для установки бамбуковой палочки, которая будет служить осью для стартовых ворот. Траверсы жесткости и стартовые ворота просто вырезаются по контуру.

Установите сервомашинку в соответствующее отверстие для нее в боковой стойке (см. рис. 9.15). Сервомашинка устанавливается с наружной стороны башни валом вовнутрь. Закрепить

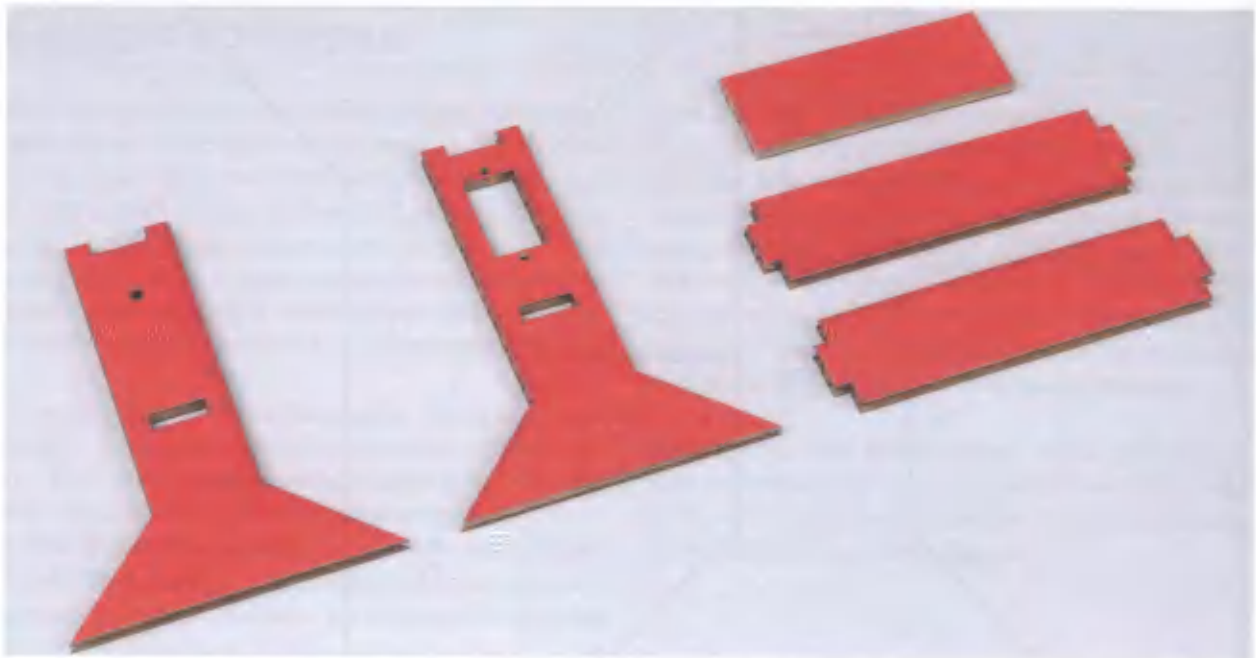


Рис. 9.16. Составные части стартовой башни

сервомашинку в отверстии можно или с помощью винтов, входящих в комплект сервомашинки, или с помощью клеевого пистолета, как показано на рис. 9.17. Качалку на сервомашинку покуда не устанавливайте, так как нам нужно будет прикрепить ее к стартовым воротам, что мы сделаем чуть позже.

Теперь вставьте и приклейте траверсы жесткости. Нижняя траверса вставляется в сквозные прорези в боковых стойках, а верхняя — в вырезы в верхней кромке каждой боковой стойки. Закрепите траверсы жесткости небольшой каплей клея. Собранная стартовая башня должна выглядеть, как показано на рис. 9.18.

Рис. 9.17. Закрепляем сервомашинку с помощью клеевого пистолета

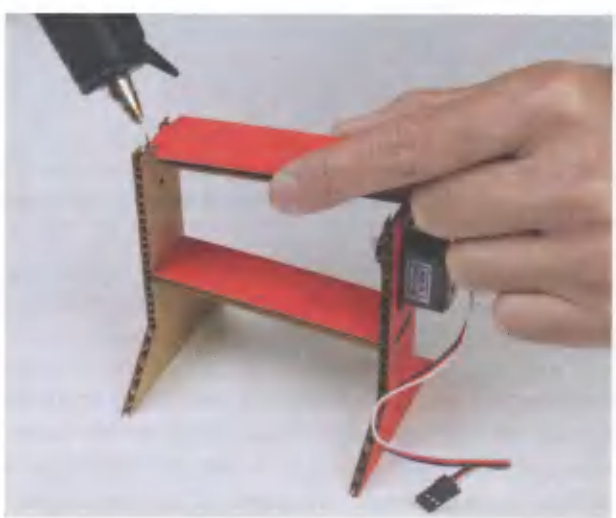


Рис. 9.18. Готовая (почти) стартовая башня

Собираем и вставляем стартовые ворота

Для стартовых ворот нам потребуется кусок картона размером 6,5×2,5 см и короткий отрезок бамбуковой палочки или тонкой трубочки для размешивания кофе, сыграющий роль оси для стартовых ворот.

Нанесите тонкую полоску клея на кромку стартовых ворот (9.19, *слева*) и приклейте качалку сервомашинки таким образом, чтобы ее ступица слегка выступала за край ворот, как показано на рис. 9.19, *справа*.

Отрежьте от бамбуковой палочки отрезок длиной около 9 см, который будет служить осью стартовых ворот. Нанесите полоску клея вдоль длинного края ворот со стороны ступицы качалки сервомашинки и приклейте ось к воротам (рис. 9.20, *слева*), совместив ее со ступицей, как показано на рис. 9.20, *справа*.

Подключите Arduino к компьютеру и нажмите кнопку сброса платы, чтобы установить сервомашинку в исходное положение. Вспомним, что исполнение кода начинается, когда вал сервомашинки находится в положении 0 градусов. Это и будет нижним положением стартовых ворот, когда они удерживают автомобиль на старте.

Рис. 9.19. Наносим полоску клея на кромку стартовых ворот (*слева*) и приклеиваем ступицу сервомашинки к стартовым воротам (*справа*)

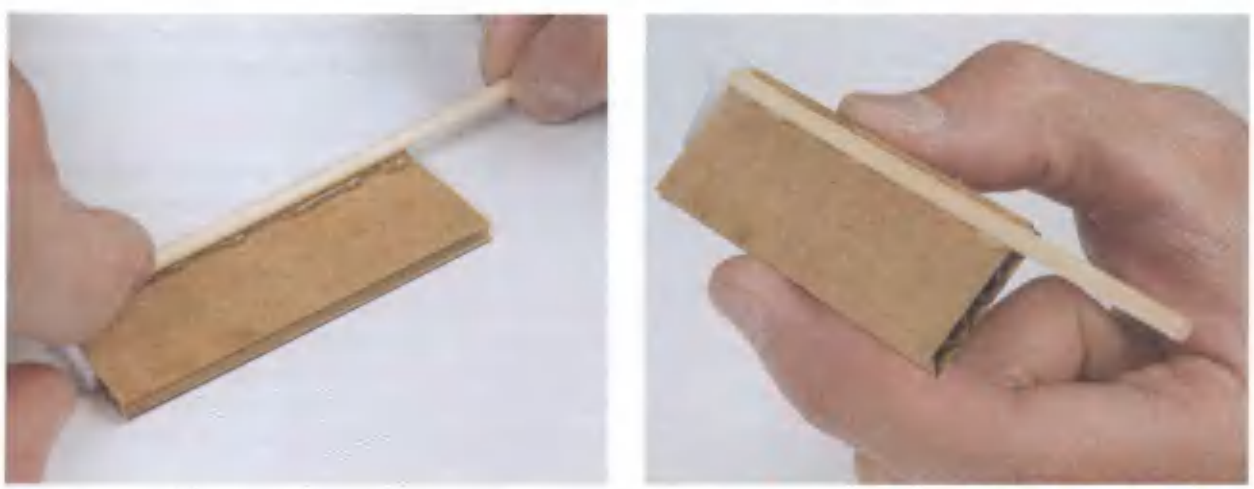


Рис. 9.20. Приклеиваем ось к створке стартовых ворот (*слева*), совмещая ее со ступицей (*справа*)

Теперь вставим ворота в башню. Для этого сначала вставьте ось в отверстие в стойке ворот, противоположной стойке с установленной сервомашинкой (рис. 9.21). Имейте в виду, что при открытии створка ворот будет вращаться против часовой стрелки.

Затем наденьте ступицу качалки на вал сервомашинки. Стартовая башня с установленными стартовыми воротами должна выглядеть так, как показано на рис. 9.22.

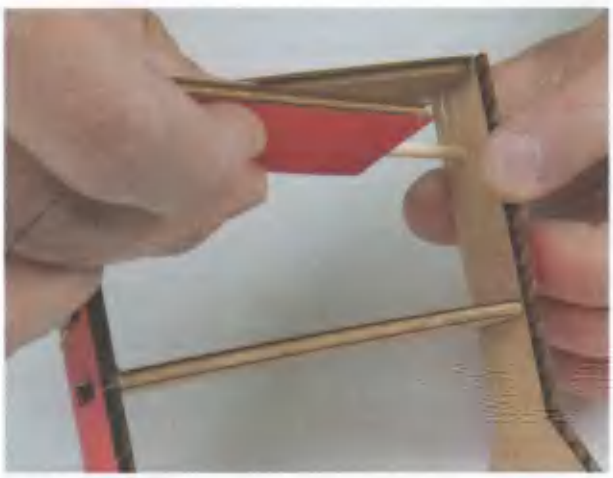


Рис. 9.21. Вставляем ось ворот в боковую стойку башни

Рис. 9.22. Полностью собранная стартовая башня

Теперь нам нужна гоночная трасса. Для нее можно использовать готовую трассу от игрушечных автомобильных гонок Hot Wheels или же собрать ее самим. В первом случае можно пропустить следующий раздел и перейти к *разд. «Монтируем фоторезистор»*, а во втором — читаем следующий раздел, в котором рассказано, как изготовить свою гоночную трассу.

Изготавливаем гоночную трассу

Для изготовления гоночной трассы нам потребуется кусок картона размером приблизительно 9×28 см. Можно сделать несколько отрезков трассы и скрепить их клейкой лентой, чтобы получить более длинную трассу, но для нашего примера мы ограничимся одним отрезком.

> **Примечание**
>
> Напомним, что шаблон гоночной трассы приведен в PDF-файле, который, вместе с шаблоном стартовой башни, находится в папке ***P9_Templates*** архива ресурсов для книги.

Согните длинные края картонной заготовки трассы под прямым углом на расстоянии около 5 мм от краев — мы получим два невысоких буртика по сторонам трассы. (Удобно загибать буртики, приложив металлическую линейку вдоль линии изгиба — так получатся прямые и аккуратные сгибы.) Буртики будут удерживать автомобиль на трассе, а также добавят ей немного жесткости. Готовая трасса должна выглядеть так, как показано на рис. 9.23.

Теперь возьмите ручной дырокол для бумаги и сделайте отверстие для фоторезистора на расстоянии приблизительно 2,5 см от конца трассы. Если у вас нет дырокола, отверстие можно сделать с помощью макетного ножа или просто остро заточенного карандаша. Только следите за тем, чтобы не подставлять пальцы с обратной стороны прокалываемого отверстия, и положите трассу на коврик для резки, когда используете макетный нож. Размер отверстия должен быть таким, чтобы через него проходил фоторезистор.

Рис. 9.23. Готовая трасса с буртиками

Монтируем фоторезистор

Как готовую игрушечную трассу, так и самодельную, необходимо оснастить фоторезистором, который будет играть роль финишного датчика. (В конце стандартной трассы из Hot Wheels имеется небольшое отверстие, чуть меньшего размера, чем диаметр фоторезистора. К счастью, пластмасса трассы достаточно пластичная, и корпус фоторезистора можно просто продавить через это отверстие.)

Извлеките фоторезистор из макетной платы и согните его выводы под прямым углом вдоль корпуса, как показано на рис. 9.24, после чего вставьте его в отверстие в трассе таким образом, чтобы он немного выступал над поверхностью трассы. При этом следите за тем, чтобы выступал он не слишком высоко, иначе автомобиль будет цепляться за него, а он должен проезжать над фоторезистором беспрепятственно. Отрежьте кусочек клейкой ленты и прикрепите выводы фоторезистора к нижней стороне трассы, как показано на рис. 9.25.

Возьмите теперь пару достаточно длинных перемычек со штыревым разъемом на одном конце и гнездовым на другом и снова подключите фоторезистор в схему на макетной плате. Если одной перемычки окажется недостаточно, используйте дополнительные.

Рис. 9.24. Изгибаем выводы фоторезистора под прямым углом

Рис. 9.25. Прикрепляем фоторезистор к гоночной трассе

ВЫЧИСЛЯЕМ СРЕДНЮЮ СКОРОСТЬ

Пока что мы разобрались, как точно измерить время прохождения автомобилем трассы, но не скорость движения автомобиля.

Но это совсем простая задача. У нас есть время прохождения трассы автомобилем, и мы знаем длину трассы. Имея эти данные, мы без труда можем *приблизительно* вычислить скорость автомобиля. Приблизительно — потому, что в действительности это будет *средняя* скорость автомобиля на протяжении всей трассы, а не точная скорость в момент пересечения им финишной черты с фоторезистором в нижней части трассы. Наблюдая процесс прохождения трассы автомобилем, мы можем заметить, что сначала он неподвижен, а затем начинает медленно двигаться, постепенно набирая скорость по мере движения вниз по трассе.

Средняя скорость определяется как расстояние, пройденное за единицу времени. Таким образом, чтобы определить среднюю скорость, необходимо измерить длину гоночной трассы и разделить ее на время прохождения трассы автомобилем.

$$\text{средняя скорость} = \frac{\text{общее расстояние}}{\text{время прохождения}}$$

Длина трассы в нашем примере составляет около 21,5 см от стартовых ворот до датчика финишной черты, а в последней проверке работы схемы авторы получили время 0,581 секунды. Подставляя эти значения в уравнение средней скорости, получим значение средней скорости, равное 37 см в секунду.

$$\text{средняя скорость} = \frac{21{,}5\ \text{см}}{0{,}581\ \text{сек}} = 37\ \text{см/сек}$$

И не забывайте, что это *средняя* скорость автомобиля. В нашей простой конфигурации с прямой гоночной трассой это и есть *приблизительная* скорость движения автомобиля посередине трассы. Поскольку в начале трассы он не двигался, будем считать, что в конце трассы он двигался со скоростью, вдвое превышающей среднюю. А какая скорость у автомобиля в вашей реализации проекта?

Тестирование и отладка

Наконец, все готово для полевых испытаний. Установите начало гоночной трассы (тот ее конец, который без фоторезистора) на нижнюю траверсу жесткости стартовой башни, чтобы она выходила за пределы стартовых ворот приблизительно на длину вашего автомобильчика (рис. 9.26), — это будет его стартовая позиция.

Подключите Arduino к компьютеру или к внешнему источнику питания. Нажмите кнопку сброса платы Arduino, чтобы начать исполнение скетча с самого начала. Возьмите свой любимый игрушечный автомобиль и поставьте его на стартовый отрезок гоночной трассы за стартовыми воротами. Нажмите кнопку запуска и только смотрите, как ваш автомобиль несется по трассе!

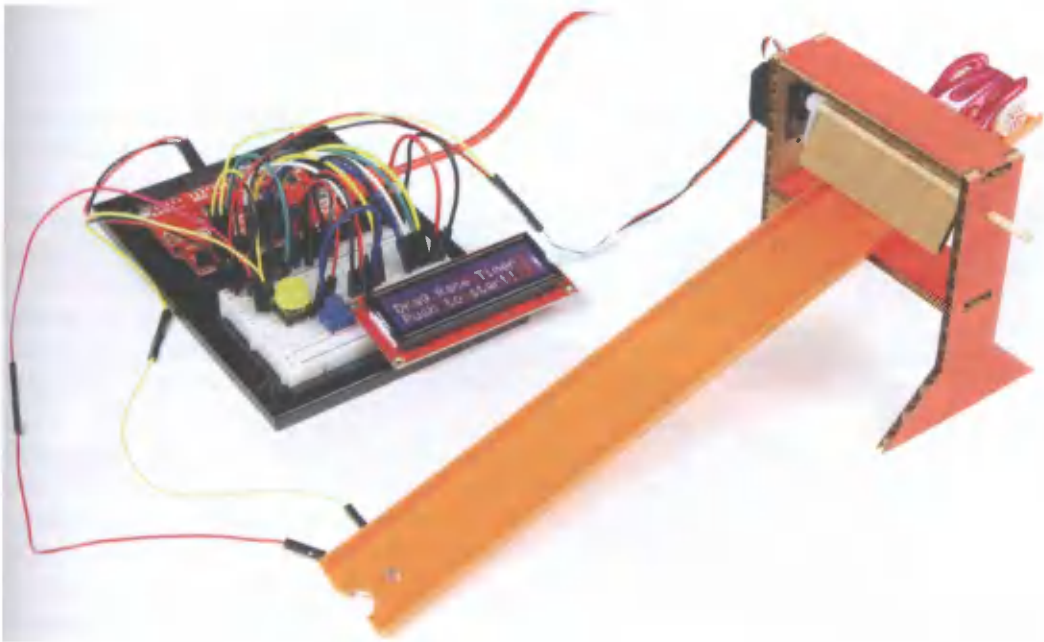


Рис. 9.26. Трасса готова к гонкам

Сколько времени заняло ему доехать до финиша? В проекте, реализованном авторами, автомобильчики достигали конца трассы всего за чуть более 0,5 секунды. Поэкспериментируйте с гонками разных автомобильчиков или пригласите своих друзей и посоревнуйтесь, чей автомобиль быстрее. Попробуйте прикрепить клейкой лентой к автомобильчику несколько мелких монет. Как это отразилось на его скорости? Поэкспериментируйте, как установка разных предметов на автомобильчик влияет на время прохождения им гоночной трассы.

Идем дальше...

В этом проекте мы узнали, как использовать ЖКД для отображения информации непосредственно из скетча Arduino. Далее приводится несколько предложений по расширению области применения полученных знаний.

Экспериментируем с проектом

Катание по трассе одного автомобильчика — это, в общем-то, невеликое удовольствие. Давайте посмотрим, как добавить в наш проект вторую гоночную трассу, чтобы в гонках могли участвовать два автомобиля (рис. 9.27).

Для этого нам потребуется дополнительный, второй фоторезистор, которого нет в стандартном наборе изобретателя SparkFun. Поэтому вам нужно будет или приобрести его в магазине радиодеталей, или попросить взаймы у приятеля, обладающего таким же стандартным набором изобретателя, или взять его из дополнительного набора деталей для этой книги, если он у вас есть.

Затем надо собрать отдельную схему с датчиком финишной линии. Авторы смогли втиснуть еще один фоторезистор и понижающий резистор для него в нижнюю часть макетной платы, как показано на рис. 9.28. Один вывод второго фоторезистора подключается к шине положительного питания, а второй — к гнезду вывода 2 платы Arduino через последовательный понижающий резистор сопротивлением 10 кОм.

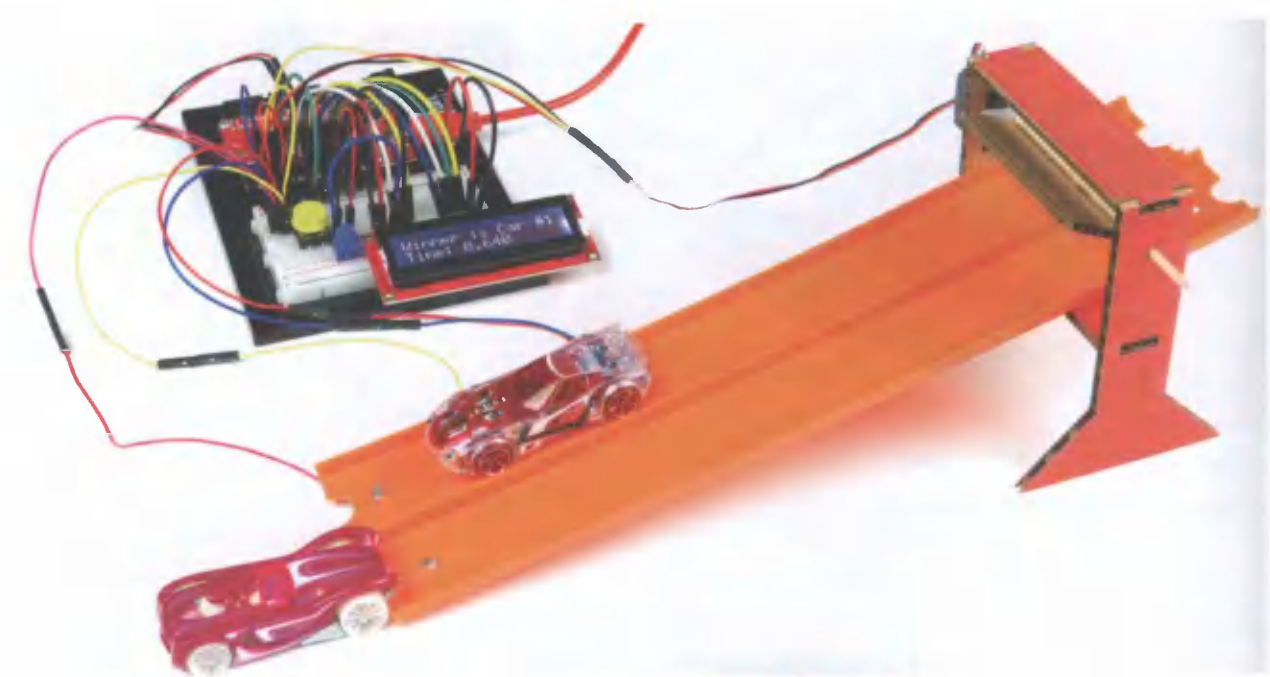


Рис. 9.27. Гоночная трасса в две полосы

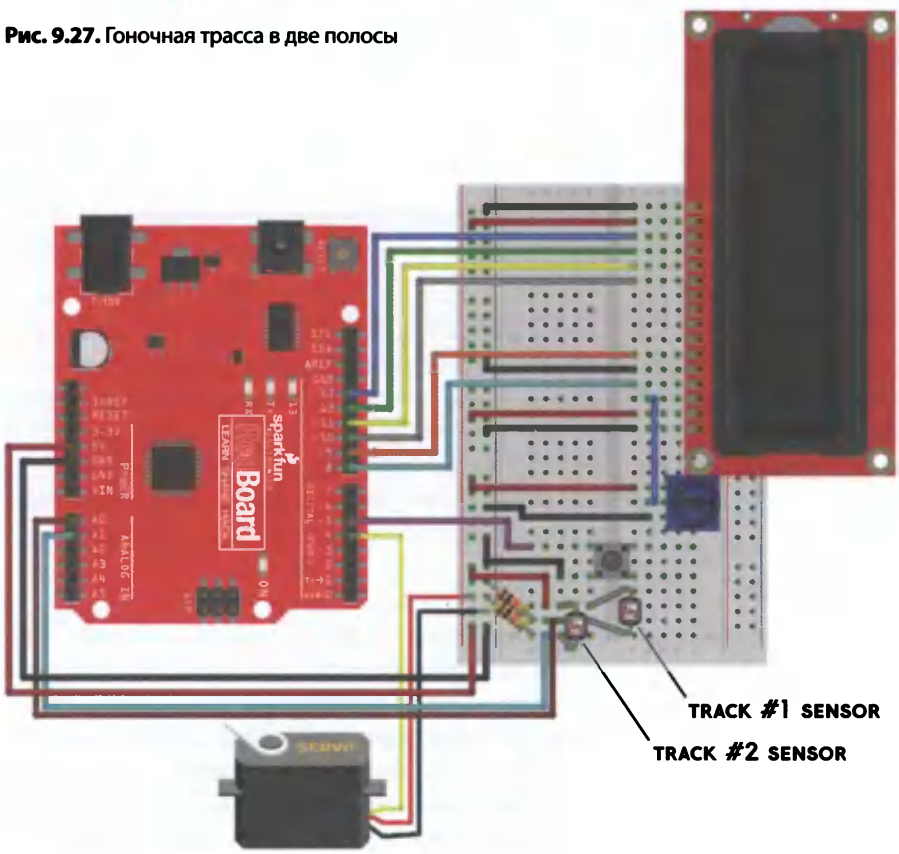


Рис. 9.28. Добавление второго фоторезистора для одновременных гонок двух автомобилей

Изготовьте по примеру первой вторую полосу гоночной трассы, вмонтируйте в нее второй фоторезистор и подключите его к макетной плате с помощью перемычек. Установите вторую полосу в стартовую башню рядом с первой.

Для одновременных гонок двух автомобилей в исходный скетч нужно добавить всего лишь несколько новых строк кода. Скетч для этой модификации (файл ***P9_TwoCarDragRaceTimer.ino***) находится в архиве ресурсов книги, доступном по адресу: **https://www.nostarch.com/arduinoinventor/**.

Откроем этот скетч в среде разработки Arduino или в любом текстовом редакторе и разберемся с работой сделанных в нем дополнений. Сначала в код добавляются новая константа и переменная для второго фоторезистора финишной черты: finishSensor2Pin и finishSensor2.

Затем с помощью составного оператора if() выполняется проверка, какой датчик был активирован первым. Если первым пришел автомобиль, шедший по первой трассе, значение переменной finishSensor1 будет 0, а значение переменной finishSensor2 так и останется 1. В этом блоке if() информация о победителе выводится на ЖКД, и переменной состояния finishFlag присваивается значение true (истина).

В блоке инструкций оператора else...if() выполняется проверка, не был ли первым автомобиль, шедший по второй трассе. В таком случае переменная finishSensor2 будет иметь значение 0, а значение переменной finishSensor1 так и останется 1. В маловероятном случае одновременного пересечения финишной линии обоими автомобилями код не делает ничего. Посмотрите, не сможете ли вы вычислить, как добавить возможность реагирования на ничью.

Код скетча изобилует комментариями с дополнительными подробными объяснениями его работы. Загрузите его в свой Arduino и организуйте гонки с друзьями, чтоб выяснить, у кого автомобиль самый быстрый.

Подключение ЖКД через модуль IIC/I2C

При разработке больших проектов на Arduino Uno очень часто не хватает выводов для подключения различных устройств. Например, для подключения одного ЖКД у нас задействовано 6 цифровых выводов (D8–D13) и два вывода питания платы Arduino (см. рис. 9.6).

Чтобы сократить количество выводов для подключения ЖКД, можно использовать специальный интерфейсный модуль IIC/I2C (рис. 9.29), с помощью которого обмен данными между ЖКД и Arduino осуществляется по последовательному протоколу I2C (Inter-Integrated Circuit). В этом случае будет задействовано только 4 вывода Arduino: **GND** («земля»), **VCC** (Питание +5V), **A4** (SDA, Serial Data — последовательные данные), **A5** (SCL, Serial Clock — тактовый сигнал). На модуле расположен потенциометр для управления контрастом ЖКД. Удобно приобрести модуль ЖКД, на котором модуль IIC/I2C уже установлен (рис. 9.30).

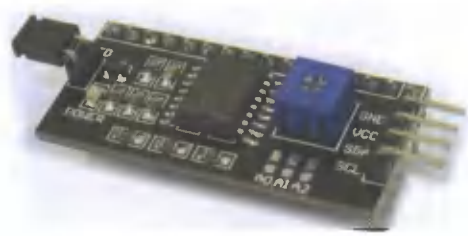


Рис. 9.29. Модуль I2C

Рис. 9.30. Плата ЖКД 1602 с модулем I2C

На рис. 9.31 показана принципиальная схема подключения модуля IIC/I2C к Arduino.

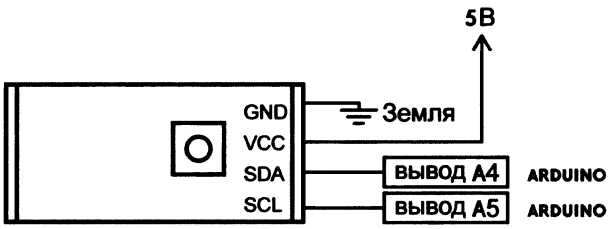


Рис. 9.31. Принципиальная схема подключения модуля IIC/I2C

Полностью собранная схема гоночного хронометриста, ранее показанная на рис. 9.12, примет несколько другой вид (рис. 9.32).

Как видно из схемы, у нас освободилось 6 цифровых выводов Arduino, к которым мы можем подключать другие устройства.

Для управления ЖКД потребуется специальная библиотека, которую можно скачать по ссылке **https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library**. Библиотека представляет собой архив в формате ZIP (файл ***Arduino-LiquidCrystal-I2C-library-master.zip***).

Для подключения библиотеки в среду разработки Arduino IDE выберите команду **Скетч | Подключить библиотеку | Добавить .ZIP библиотеку...** (рис. 9.33).

Замените в листинге 9.2 три строки на строки, выделенные в листинге 9.3 красным полужирным шрифтом. В этих строках вы объявляете новую библиотеку ❶, создаете переменную типа LiquidCrystal_I2C ❷ и инициализируете ЖКД ❸.

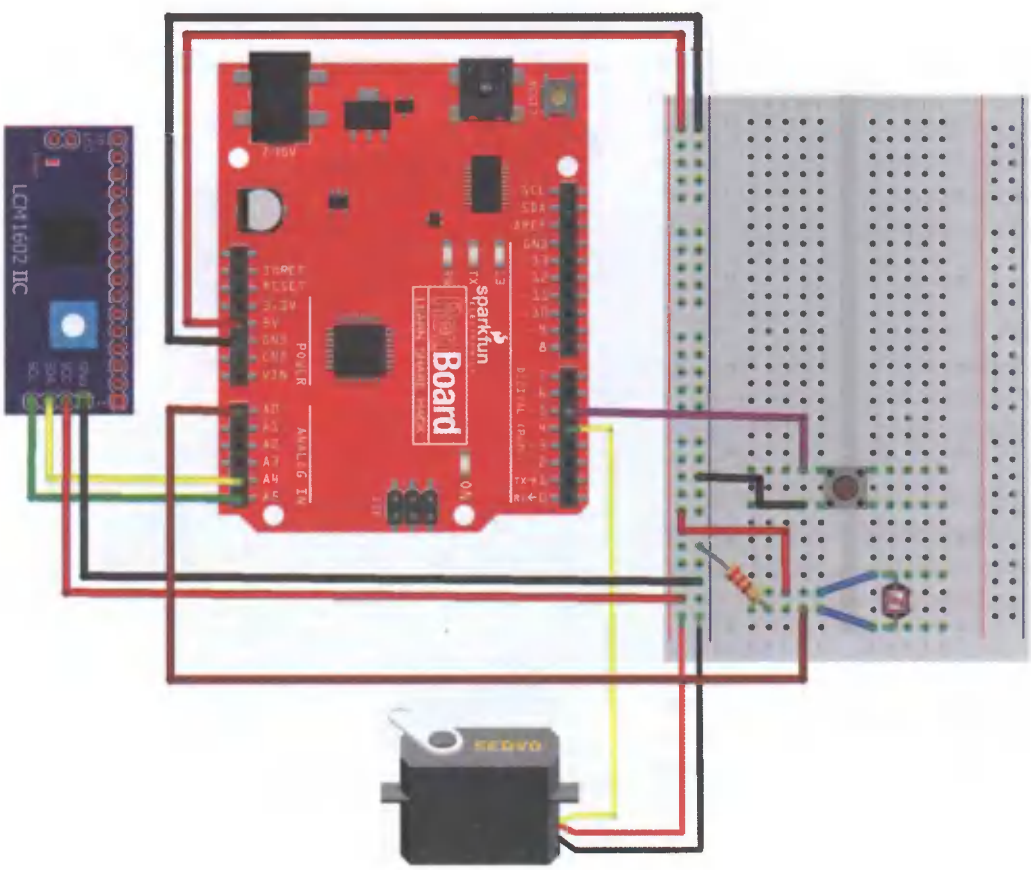


Рис. 9.32. Схема гоночного хронометриста с одной кнопкой старта и ЖКД, подключенным с помощью модуля IIC/I2C

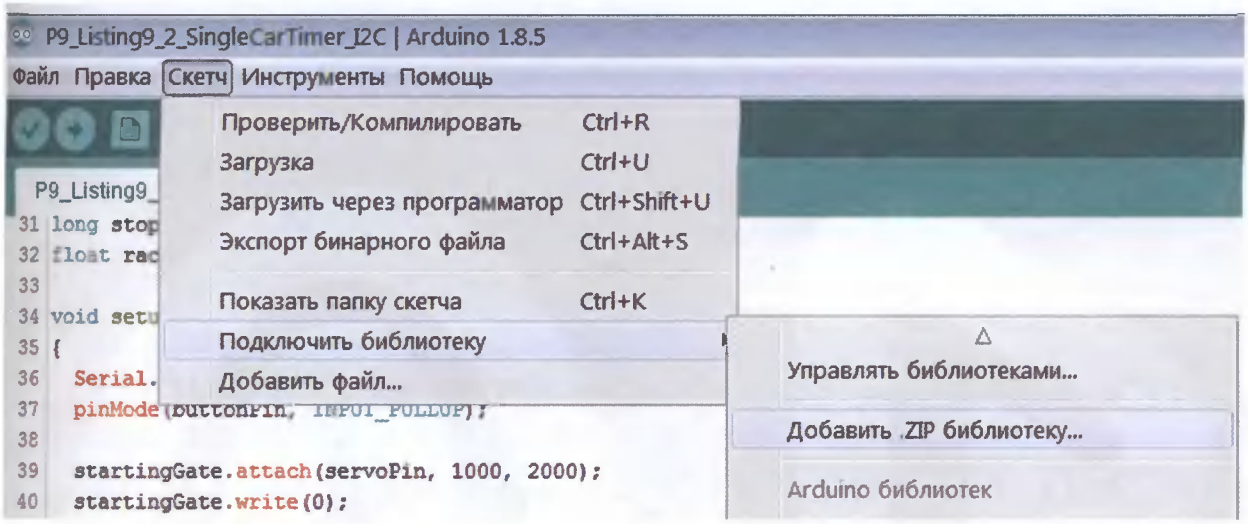


Рис. 9.33. Добавление внешней библиотеки в среду разработки Arduino IDE

Листинг 9.3. Фрагмент скетча для управления гоночным хронометристом

```
❶ #include<LiquidCrystal_I2C.h>
  #include<Servo.h>

❷ LiquidCrystal_I2C lcd(0x27,16,2);
  Servo startingGate;

  const byte buttonPin = 5;
  const byte servoPin = 4;
  const byte finishSensor1Pin = A0;
  const int darkThreshold = 500;
  --пропущено для краткости--
  void setup()
  {
    pinMode(buttonPin, INPUT_PULLUP);

    startingGate.attach(servoPin, 1000, 2000);
    startingGate.write(0);

❸   lcd.begin();
    lcd.clear();
    lcd.print("Drag Race Timer");     //Гоночный
                                      //хронометрист
    lcd.setCursor(0, 1);
    lcd.print("Push to start!");      //Нажмите кнопку
                                      //для старта
  --пропущено для краткости--   }

  void loop()
  {
  --пропущено для краткости--
  }
```

Полный текст программы и библиотеку можно найти в электронном архиве к книге по адресу **ftp://ftp.bhv.ru/9785977539722.zip**.

Модифицируем предыдущие проекты

Теперь, когда вы знаете, как использовать ЖКД, попробуйте оснастить этим устройством один из ранее созданных проектов. В любом из предыдущих проектов, в котором информация выводится в окно монитора порта, — например, в измерителе скорости реакции в *проекте 4* или в настольной тепличке из *проекта 7*, монитор порта можно заменить на ЖКД. Но для этого вам нужно будет разобраться с необходимой разводкой проводов и конфигурацией выводов Arduino.

Пример подключения ЖКД к проекту измерителя скорости реакции приводится в соответствующем руководстве на площадке изобретателя Inventor Space по адресу: **https://invent.sparkfun.com/cwists/preview/1145-sik-icd-reaction-timer/**.


10

ЭЛЕКТРОННОЕ МИНИ-ПИАНИНО

В этом проекте мы используем специальный тактильный датчик и пьезоэлектрический зуммер, чтобы соорудить миниатюрное электронное пианино под управлением Arduino (рис. 10.1). Независимо от того, есть у вас музыкальный слух или нет, вы получите удовольствие, работая над этим проектом!

Рис. 10.1. Завершенный проект электронного мини-пианино

Необходимые компоненты, инструменты и материалы

В этом проекте используются всего лишь два компонента, но оба они новые. Это мембранный потенциометр, играющий роль клавиатуры, и пьезоэлектрический зуммер, выступающий в качестве источника звука.

Электронные компоненты

Для реализации этого проекта вам потребуются следующие компоненты (рис. 10.2):

- плата RedBoard компании SparkFun (DEV-13975), или плата Arduino Uno (DEV-11021), или любая другая совместимая с Arduino плата, 1 шт.;
- кабель Mini-B USB (CAB-1101) или кабель USB, идущий в комплекте с вашей платой, 1 шт.;
- беспаечная макетная плата (PRT-12002), 1 шт.;
- резистор 10 кОм (COM-08374 или COM-11508 для пакета, содержащего 20 шт.), 1 шт.;
- мембранный потенциометр SoftPot длиной 50 мм (SEN-08680), 1 шт.;
- пьезоэлектрический зуммер (COM-07950), 1 шт.;
- проволочные перемычки со штекерами на обоих концах (PRT-11026).

Примечание

Все электронные компоненты, используемые в этом проекте, входят в стандартный состав набора изобретателя Inventor's Kit компании SparkFun.

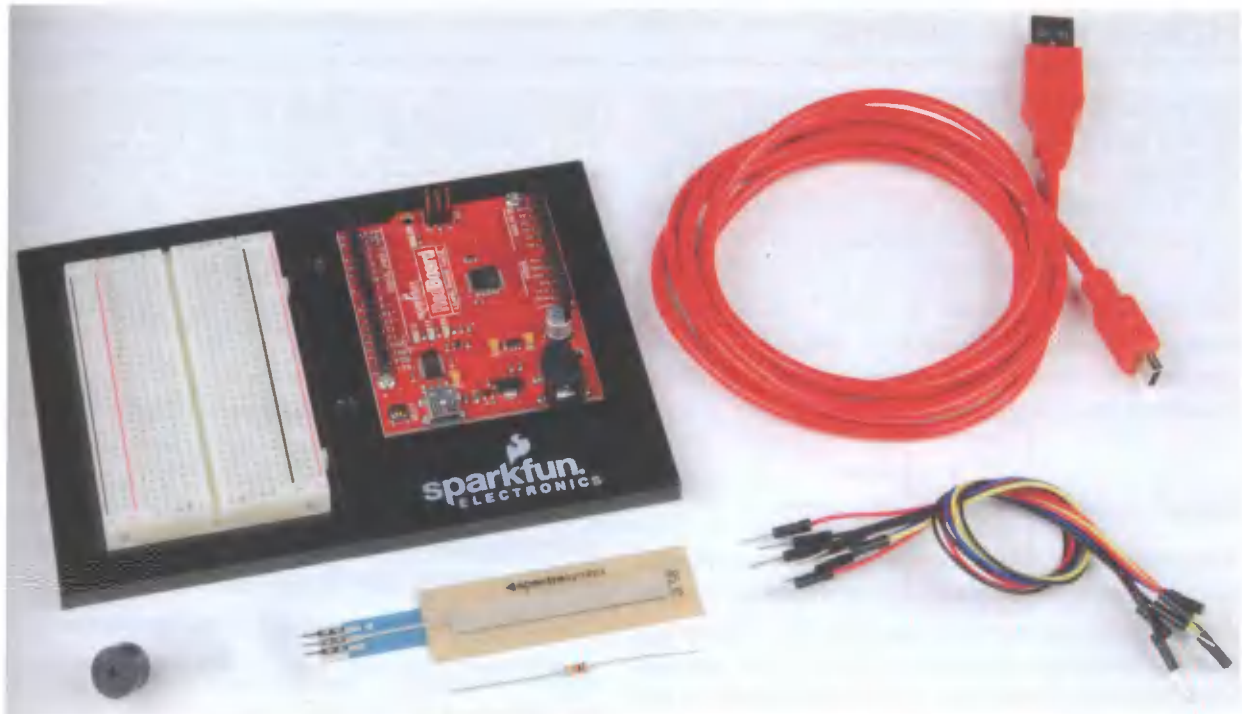


Рис. 10.2. Электронные компоненты для проекта электронного мини-пианино

Прочие инструменты и материалы

Для реализации этого проекта вам потребуются следующие инструменты и материалы (рис. 10.3):

- карандаш;
- макетный нож;
- металлическая линейка;
- может пригодиться: паяльник;
- клейкая лента типа «скотч»;
- гофрированный картон (приблизительно 10×13 см), небольшая картонная коробка или толстый открыточный картон.

Рис. 10.3. Инструменты и материалы, рекомендуемые для проекта электронного мини-пианино

Новые компоненты

Как упоминалось ранее, в этом проекте используются два новых компонента. Один из них — мембранный тактильный потенциометр. Так же, как и у потенциометра, задействованного нами в *проекте 6*, мы можем изменять его сопротивление, но не поворотом ручки, а надавливанием где-либо по его длине. Второй новый компонент — пьезоэлектрический зуммер. Давайте рассмотрим более подробно, что собой представляют эти компоненты.

Мембранный потенциометр

В *проекте 6* мы познакомились с простым потенциометром, вращая ручку которого, можно было изменять его сопротивление, управляя, таким образом, балансирной балкой. Точно так же мы можем изменять сопротивление и мембранного потенциометра (рис. 10.4), но не вращением ручки, а надавливанием где-либо по длине его рабочей области.

Мембранный потенциометр представляет собой тонкую и гибкую полосу полиэфирной пленки, на которую нанесены две дорожки из резистивного материала, над которыми располагается мембрана с проводящей поверхностью. При надавливании на мембрану в любой точке ее рабочей области она замыкает резистивные дорожки в этой точке, изменяя тем самым сопротивление между центральным и ближним выводами компонента в диапазоне от 0 до 10 кОм в зависимости от силы прилагаемого давления.

Рис. 10.4. Мембранный потенциометр длиной 50 мм

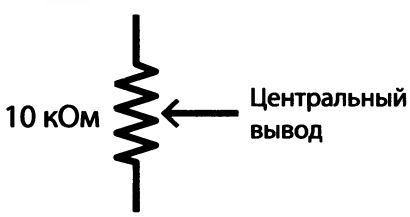


Рис. 10.5. Символическое изображение мембранного потенциометра такое же, как и обычного потенциометра (см. рис. 6.5)

Мембранные потенциометры обладают очень высокой точностью и почти безграничным разрешением. Они часто используются в промышленности для определения положения скользящих частей различных агрегатов, манипуляторов роботов и других компонентов, выполняющих точные движения.

Функционально мембранный потенциометр практически то же самое, что и потенциометр с ручкой, который мы использовали в предыдущих проектах. И на принципиальных схемах он обозначается примерно так же (рис. 10.5).

В этом проекте мы воспользуемся мембранным потенциометром, сделав из него клавиатуру для нашего пианино. Для этого мы разделим рабочую полосу датчика на восемь частей, или «клавиш», каждая из которых будет отвечать за отдельную ноту.

Пьезоэлектрический зуммер

Пьезоэлектрический зуммер (рис. 10.6) похож на динамик и издает щелчки при подаче напряжения на его выводы. Если подавать это напряжение с очень высокой частотой — порядка нескольких

Рис. 10.6. Зуммер

сотен, а то и тысяч раз в секунду, — издаваемые зуммером щелчки будут восприниматься нами как соответствующий тон. Внутри своего цилиндрического корпуса такой зуммер содержит специальный кристалл — *пьезоэлемент*, который деформируется, когда на него подается напряжение. К этому кристаллу присоединен круглый металлический диск, который при деформировании кристалла движется вместе с ним, создавая вибрацию воздуха, которую мы и воспринимаем как звук.

> **Примечание**
>
> Зуммер в комплекте изобретателя SparkFun в действительности содержит не пьезокристалл, а небольшую электромагнитную катушку, но работает так же, как и пьезоэлектрический зуммер. Поэтому мы далее будем называть этот компонент просто зуммером.

Мы уже знаем, как с помощью Arduino заставить с разной частотой мигать светодиод. Таким же образом можно «мигать» и зуммером с частотой сотен раз в секунду, но вместо света эти мигания будут создавать звуки разной тональности. Зуммеры, динамики и подобные им элементы представляются в принципиальных диаграммах посредством различных условных символов, и для представления зуммера в этом проекте мы воспользуемся символом, показанным на рис. 10.7.

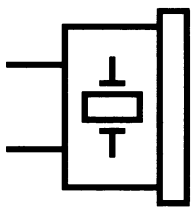


Рис. 10.7. Условное обозначение пьезоэлектрического зуммера

Собираем схему

Схема этого проекта содержит только два электронных компонента: зуммер и мембранный потенциометр. Принципиальные схемы подключения этих элементов к Arduino показаны на рис. 10.8, а на рис. 10.9 приведена соответствующая монтажная схема.

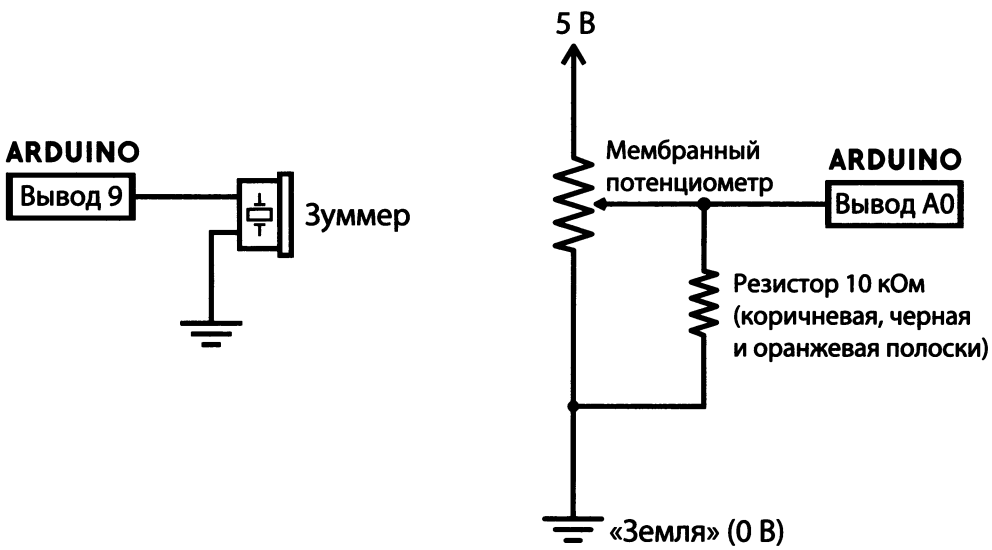


Рис. 10.8. Принципиальные схемы включения компонентов проекта электронного пианино: зуммера (*слева*) и мембранного потенциометра (*справа*)

Вставляя зуммер в макетную плату, можно заметить, что расстояние между его выводами слегка меньше, чем расстояние между рядами гнезд платы. При этом выводы зуммера нужно вставить в плату не в сопредельные ряды, а так, чтобы между ними был один свободный ряд. Так что просто раздвиньте слегка выводы, чтобы они совпадали с гнездами платы должным образом, и вставьте зуммер.

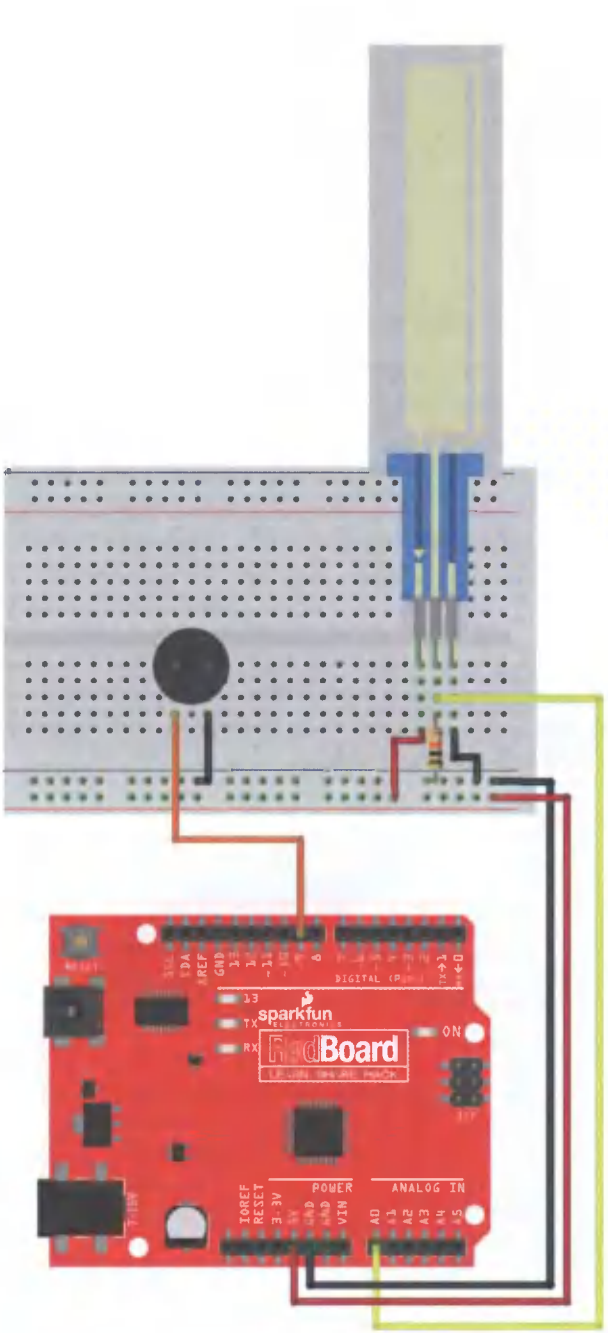


Рис. 10.9. Монтажная схема проекта электронного минипианино

Подобно обычному потенциометру, который мы использовали в *проекте 6*, мембранный потенциометр имеет три вывода. Но, в отличие от обычного потенциометра, мембранный потенциометр необходимо подключать через резистор — чтобы при отсутствии давления схема выдавала номинальное значение по умолчанию, равное 0 В. Так мы предотвращаем звучание зуммера при отсутствии давления на датчик.

Руководствуясь монтажной схемой, представленной на рис. 10.9, соберите схему проекта в соответствии с принципиальными схемами, приведенными на рис. 10.8. Схема проекта, как вы можете видеть, весьма простая. Сначала соедините перемычками разъемы +5 В и GND платы Arduino с шинами положительного и отрицательного питания с левой стороны макетной платы. Затем вставьте в макетную плату зуммер — примерно в 10-й ряд от верхнего края макетной платы. Подключите один вывод зуммера к выводу 9 платы Arduino, а другой — к шине отрицательного питания макетной платы. Затем вставьте мембранный потенциометр в нижнюю часть макетной платы. Подключите верхний вывод мембранного потенциометра к шине положительного питания (5 В), нижний вывод — к шине отрицательного питания («земле»), а центральный — к выводу A0 аналогового входа платы Arduino. Наконец, подключите резистор сопротивлением 10 кОм между центральным выводом мембранного потенциометра и шиной отрицательного питания («землей») макетной платы.

На этом сборка прототипа схемы завершена и можно приступать к ее испытанию, исполнив на ней пару скетчей в качестве примеров.

Программируем электронное пианино

Сначала мы проверим собранный прототип схемы, создавая зуммером звуки и распознаваемые тоны, а затем сопоставим эти звуки и тоны определенным участкам мембранного потенциометра. Таким образом, научившись создавать с помощью кода звуки, мы сможем воспроизводить восемь разных нот на мембранном потенциометре, как на маленьком пианино.

Мы уже видели, что Arduino может генерировать высокочастотные импульсы и реагировать на моментальные нажатия кнопки. Мы используем эти возможности для создания различных звуков и тестирования частотных характеристик зуммера.

Тестируем работу зуммера

В репертуаре Arduino содержится несколько команд, которые упрощают задачу воспроизведения нотных звуков. В частности, для управления воспроизведением звуков Arduino использует две функции: функция tone() дает команду зуммеру воспроизвести указанную частоту, а функция noTone() дает команду прекратить воспроизведение, что предоставляет пользователю возможность управлять длительностью звучания ноты. Формат вызова этих функций таков:

```
//Создает тон заданной частоты на заданном выводе
//в течение мс
tone(вывод, частота, длительность);

//Прекращает воспроизведение тона на заданном выводе
noTone(вывод);
```

При вызове функции tone() ей передаются три параметра: номер вывода платы Arduino, к которому подключен зуммер, частота тона для воспроизведения и время, в течение которого следует воспроизводить заданную ноту. Таким образом функция tone() создает сигнал прямоугольной формы заданной частоты на указанном выводе в течение заданного периода времени. Этот сигнал вызывает вибрацию диска внутри зуммера, подключенного к данному выводу, что создает звук выдаваемой частоты. Звук, запущенный функцией tone(), воспроизводится в течение заданного времени, или до повторного вызова функции tone() с другой частотой, или до вызова функции noTone(), которая прекращает воспроизведение звука.

Создайте новый скетч в среде разработки Arduino и скопируйте в него код из листинга 10.1. Вы также можете открыть готовый скетч из архива ресурсов книги, которые можно загрузить по адресу: **https://www.nostarch.com/arduinoinventor/**. В этом примере мы задаем требуемую частоту звука с помощью монитора порта, а зуммер воспроизводит этот звук в течение полсекунды.

Листинг 10.1. Тестирование зуммера с помощью монитора порта

```
//Тестирование зуммера с помощью монитора порта
//Загрузите этот скетч в Arduino, а затем откройте
//монитор порта

int freq;

void setup()
{
❶  pinMode(9, OUTPUT);
   Serial.begin(9600);
   Serial.println("Type in a frequency to play.");
}  // Введите частоту тона для воспроизведения

void loop()
{
❷  if(Serial.available() > 0)     //Ожидаем ввода строки
                                  //с последовательного
                                  //порта
   {
❸     freq = Serial.parseInt();  //Преобразовываем
                                  //в целое число
      Serial.print("Playing note: "); //Информация для
                                       //пользователя
```

```
        // Воспроизводится нота:
        Serial.println(freq);
❹       tone(9, freq, 500);        //Воспроизводим ноту
                                   //в течение 500 мс
❺       delay(500);   //Пауза для воспроизведения ноты
    }
    else
    {
        noTone(9);
    }
}
```

Теперь давайте разберемся, как этот скетч работает. В блоке setup() ❶ выполняется конфигурация вывода 9 Arduino для вывода данных (функция pinMode(9, OUTPUT)), инициализируется режим последовательной связи Arduino с компьютером и выводится короткое сообщение с инструкцией для пользователя.

Далее, в цикле loop() инструкция if(Serial.available() > 0) ❷ выполняет проверку данных, полученных из монитора порта следующим образом: функция Serial.available() возвращает количество байтов, полученных из монитора порта, а оператор if() сравнивает это значение с нулем. Если количество байтов больше нуля, это означает, что были получены данные, и скетч считывает эти данные ❸. Функция Serial.parseInt() преобразовывает полученную строку в целое число, которое затем сохраняется в переменной freq.

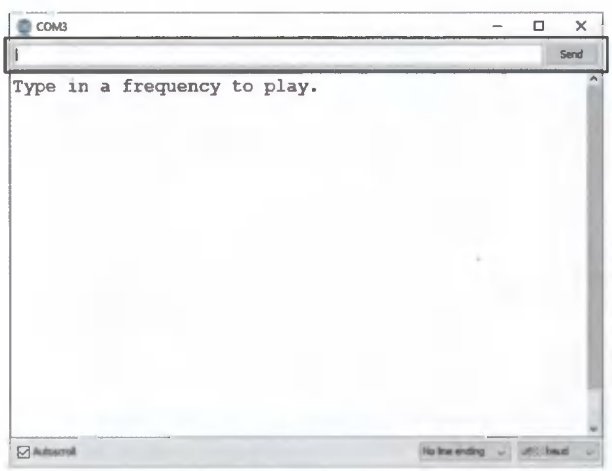


Рис. 10.10. Задаем режим передачи данных без символа окончания строки: **No line ending**

Функция tone() ❹ воспроизводит тон с частотой, сохраненной в переменной freq в течение 500 мс, а затем следует команда delay(500); ❺ для паузы в исполнении кода. Эта команда определяет, что текущая нота будет воспроизводиться в течение 500 мс, прежде чем начинать воспроизведение другой ноты. Команда tone() не является блокирующей, то есть после ее исполнения управление немедленно передается следующей команде.

Загрузите скетч в Arduino, а затем откройте окно монитора порта (нажав комбинацию клавиш <Ctrl>+<Shift>+<M> или выполнив последовательность команд **Tools | Serial Monitor**). В окне монитора порта щелкните на выпадающем списке слева от выпадающего списка скорости передачи в бодах (в правом нижнем углу окна) и выберите опцию **No line ending** (Без символа окончания строки) (рис. 10.10).

Дело в том, что при передаче данных между цифровыми устройствами иногда используется невидимый символ окончания строки, указывающий на конец сообщения. Наиболее часто для этого служат символы новой строки или возврата каретки. Хотя мы этот символ не видим, Arduino считывает его код в виде числового значения. Установка опции **No line ending** предотвращает наличие в сообщении каких-либо других символов — в частности, символов окончания строки — кроме, собственно, символов сообщения.

Теперь, в текстовую строку в верхней части основного окна монитора порта можно ввести любое число, нажать клавишу <Enter>, и зуммер будет воспроизводить тон с этой частотой в течение полсекунды (500 мс). Человеческое ухо может различать звуки в частотном диапазоне от 20 до 20 000 Гц, так что попробуйте поиграть с разными частотами, но, конечно, в пределах этого диапазона.

Для последовательной связи в Arduino используется буфер вместимостью 64 байта. Буфер можно сравнить с залом ожидания для данных на их пути в устройство, и в данном случае он позволяет отправлять из монитора порта несколько нот за раз.

Например, введя в монитор порта приведенную далее последовательность чисел, разделенных запятыми без пробелов, можно сыграть музыку для песни «Twinkle, Twinkle, Little Star» («Мерцай, мерцай, звездочка»)[1].

```
262,262,392,392,440,440,392,392,349,349,330,330,294,
294,262,262,
```

Важно также ввести последнюю запятую. Дело в том, что команда Serial.parseInt() ожидает входные данные в виде цифровых символов, разделенных нецифровыми символами, — такими как, например, запятые. Чтобы плата Arduino могла различить последнюю ноту, за ней должна следовать запятая. Также важно не допускать пробелов нигде в строке.

И еще. Каждый символ занимает 1 байт данных, и если попытаться передать больше чем 64 символа (размер буфера), все данные свыше этого числа будут обрезаны.

Наконец, зуммер оптимизирован под частоту 2047 Гц. Иными словами, зуммер воспроизводит звуки на этой частоте очень и очень громко, что может вызывать раздражение. Поэтому следует иметь в виду, что окружающим могут не понравиться пронзительные звуки, издаваемые зуммером на частоте 2047 Гц.

Создаем конкретные ноты

Давайте попробуем воспроизводить нашим минипианино более привычные мелодии — для этого нам нужно сопоставить звуковым частотам настоящие музыкальные ноты. Для многих музыкальных инструментов в качестве исходной точки используется нота C (До) гаммы До мажор, которая имеет частоту приблизительно 262 Гц. В табл. 10.1 приводится список нот октавы гаммы До мажор и соответствующие им частоты.

Поэкспериментируйте с этими нотами и попробуйте сыграть какую-либо песню. Для примера, музыка колыбельной «Мерцай, мерцай, звездочка» начинается с нот CC, GG, AA, GG, FF, EE, DD, CC.

[1] Популярная английская колыбельная.

Таблица 10.1. *Ноты первой октавы гаммы До мажор и соответствующие им частоты*

Нота	Приблизительная частота
C (До)	262
D (Ре)	294
E (Ми)	330
F (Фа)	349
G (Соль)	392
A (Ля)	440
B (Си)	494
C (До)	524

Возможно, вы заметили, что создаваемые Arduino звуки трудно назвать мелодичными. Это объясняется тем, что Arduino может подавать на вывод, управляющий зуммером, сигнал только или высокого, или низкого уровня. Это создает звуковую волну прямоугольной формы, которая имеет жесткое металлическое звучание. Существуют определенные приемы с использованием конденсаторных фильтров для смягчения этого звука, но это тема уже для другой книги.

Создаем звуки посредством мембранного потенциометра

Воспроизведение звуков и даже коротких музыкальных отрывков введением соответствующих данных с монитора порта может быть интересным и занимательным на первых порах, но быстро теряет свою новизну и становится скучной работой. Однако это можно исправить, воспользовавшись мембранным потенциометром в качестве датчика, разные значения сопротивления которого соответствуют частотам разных нот.

Наш мембранный потенциометр подключен к Arduino как простой делитель напряжения. Вследствие этого давление на какой-либо участок его рабочей области будет преобразовываться в соответствующее напряжение в диапазоне от 0 до 5 В. Вспомним, что с помощью функции analogRead() Arduino может преобразовывать

аналоговые напряжения в диапазоне от 0 до 5 В в соответствующие числовые значения в диапазоне от 0 до 1023.

С помощью простого скетча (листинг 10.2) мы можем заставить зуммер звучать на разных частотах, нажимая на соответствующие участки мембранного потенциометра. Когда давление на мембранный потенциометр отсутствует, понижающий резистор создает на его выходе напряжение величиной 0 В, поэтому не будет создаваться никакого звука.

Создайте новый скетч в среде разработки Arduino, скопируйте в него код из листинга 10.2 и загрузите скетч в Arduino.

Листинг 10.2. Код для музицирования с помощью мембранного потенциометра

```
//Музицирование с помощью мембранного
//потенциометра
//Загрузите этот скетч в Arduino, а затем начните
//нажимать на мембранный потенциометр
❶ int sensorValue;
void setup()
{
  pinMode(9, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
❷  sensorValue = analogRead(A0);
❸  if (sensorValue > 0)  //Если на датчике присутствует
                         //давление, воспроизводим ноту
   {
❹     Serial.print("Raw sensor reading: ");    //Значения
                                               //датчика:
      Serial.println(sensorValue);
❺     tone(9, sensorValue, ❻ 50);
      delay(❼ 50);
   }
❽  else
   {
      noTone(9);
   }
}
```

Давайте разберемся, что в этом коде происходит. Как обычно, начинаем с объявления переменной для хранения целочисленного представления напряжения, снимаемого с датчика ❶. В цикле loop() функция analogRead() считывает напряжение на выводе Arduino A0 ❷ и присваивает соответствующее числовое представление переменной sensorValue. При отсутствии давления на датчик напряжение на этом выводе будет 0 В. Далее полученное значение проверяется оператором if()❸ — если оно больше 0, скетч выводит его в мониторе порта ❹ и с помощью команды tone() воспроизводит соответствующую ноту ❺. Если скетч застревает на воспроизведении какой-либо ноты, попробуйте заменить значение 0 в условии сравнения оператора if() ❸ на чуть большее значение — например, 10 или 20. Это называется *обходом зоны нечувствительности*. Некоторые датчики не всегда определяют теоретическое нулевое значение, и для них задается диапазон значений, которые программа может считать равнозначными нулю. Разные датчики могут иметь в этом плане разные характеристики.

Обратите внимание, что в этом примере длительность воспроизведения ноты tone()❻ и соответствующей приостановки исполнения скетча delay()❼ были изменены на 50 мс. Это позволит нам быстрее изменять ноты. В противном случае все воспроизводимые ноты длились бы полсекунды! Наконец, необходимо, чтобы Arduino воспроизводил тон только при нажатии на мембранный потенциометр. Для этого оператор else() определяет, когда на мембранном потенциометре отсутствует давление ❽ — в этом случае команда noTone() выключает зуммер.

В итоге, вместо управления воспроизведением звуков посредством отправки значений

соответствующих частот из монитора порта, Arduino использует целочисленные представления напряжения на мембранном потенциометре для определения частоты воспроизводимого звука.

Поиграйте немного с новой игрушкой, сжимая мембранный потенциометр пальцами в разных точках датчика. Скользя пальцами вдоль потенциометра и продолжая сжимать его, можно создавать звуки с частотой в диапазоне от 1 до 1023 Гц. На что похожи получаемые звуки? Плохонький синтезатор? Приземление пришельцев из иных миров? Классно! Теперь у нас есть свой собственный генератор спецэффектов. Попробуйте сыграть на нем что-либо похожее на песню. Далее приводится пример, который поможет вам справиться с этой задачей.

Играем по нотам

Теперь, когда у нас есть определенный опыт генерирования звуков с помощью Arduino, можно переходить к следующему этапу — сопоставлению выходных напряжений датчика настоящим нотам, чтобы с его помощью можно было воспроизводить настоящую нотную музыку. Для этого мы разобьем датчик на восемь отдельных секций, которые будут играть роль клавиш, и сопоставим эти секции индексу, который можно использовать для воспроизведения нот.

Откройте среду разработки Arduino, скопируйте код из листинга 10.3 в окно редактора скетчей, а затем загрузите его в свою плату Arduino.

Листинг 10.3. Скетч для электронного пианино

```
//Скетч для электронного пианино
❶ int frequencies[] = {262, 294, 330, 349, 392, 440, 494, 524};
int sensorValue;
❷ byte note;

void setup()
{
  pinMode(9, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  sensorValue = analogRead(A0);
  if (sensorValue > 0)    //Если это нота,
                          //воспроизводим ее
  {
    //Сопоставляем нажатую клавишу ноте
❸   note = map(sensorValue, 0, 1023, 0, 8);
❹   note = constrain(note, 0, 7);
    Serial.print(sensorValue);
    Serial.print("\t");
    Serial.println(❺ frequencies[note]);
    tone(9, ❻ frequencies[note], 50);
    delay(50);
  }
  else
  {
    noTone(9);
  }
}
```

Рассмотрим, что происходит в этом коде. Сначала объявляется специальная структура данных, называющаяся *массивом*. Массив представляет собой тип переменной, способной сохранять несколько значений, а не одно, как обычная переменная. Массив может иметь стандартный тип данных, включая byte, int, long и float, ему также можно присваивать пользовательский тип данных. Тип данных массива объявляется перед его именем.

Массив объявляется подобно переменной, с той разницей, что после имени массива следуют две квадратные скобки: []. При инициализации массива внутри фигурных скобок задаются значения его элементов, разделенные запятыми:

```
типДанныХ имяМассива[] = {знач0, знач1, знач2, ... ... значN};
```

В листинге 10.3 объявляется массив frequencies[] значений типа int, который содержит список из восьми значений для частот музыкальных нот ❶. Доступ к элементам списка массива можно

получить, указывая в квадратных скобках номер требуемого элемента. Нумерация элементов списка массива начинается с 0, таким образом, ссылка на первый элемент будет frequencies[0], на второй — frequencies[1] и так далее. Еще раз обращаем ваше внимание, что нумерация элементов массива начинается с 0. В табл. 10.2 приводится список элементов массива и соответствующее им содержимое.

Таблица 10.2. *Элементы массива frequencies[] и их содержимое*

Массив frequencies[]								
Индекс	0	1	2	3	4	5	6	7
Значение	262	294	330	349	392	440	494	524

Далее в скетче объявляется переменная для индекса с именем note ❷. Поскольку наш массив содержит всего лишь восемь значений, этой переменной можно присвоить тип данных byte. Так мы сэкономим немного памяти Arduino, поскольку байтовые значения занимают меньше памяти, чем целочисленные. Затем функция map() ❸ сопоставляет целочисленные представления напряжения в переменной sensorValue в диапазоне 0–1023 в значения в диапазоне 0–8.

Функция map() представляет собой замечательное средство для преобразования одного диапазона значений в другой. Для этого ей просто передаются в параметрах входное значение, границы входного диапазона и границы требуемого выходного диапазона:

```
map(входЗначение, входМин, входМакс, выхМин,
выхМакс);
```

Преобразованное таким образом значение сохраняется в переменной note и будет служить в качестве указателя на элементы массива. При этом функция map() сводит диапазон значений 0–1023 к диапазону значений 0–8 (то есть диапазону из девяти значений), хотя наш массив frequencies[] имеет только восемь элементов, пронумерованных от 0 до 7.

Это объясняется тем, что при преобразовании из одного диапазона значений в другой функция map() округляет с понижением, поэтому, чтобы получить восемь равномерно распределенных значений, нам нужно ввести в функцию девять значений. Значение 8 выдается только для входного значения 1023. Поскольку значение 8 (девятое значение) не является действительным указателем на элемент массива, это обстоятельство исправляется другой командой: constrain()❹, которая ограничивает значения диапазоном значений от 0 до 7: любое значение ниже нуля ограничивается минимальным значением 0, а любое значение выше 7 ограничивается максимальным значением 7.

Кстати, функция constrain() часто используется совместно с функцией map() для масштабирования и ограничения значений. Она имеет следующий формат:

```
constrain(значение, минЗначение, максЗначение);
```

Наконец, скетч выводит частоту текущей активируемой ноты в окно монитора порта ❺ и с помощью функции tone() воспроизводит ноту ❻.

Теперь проверим работу как самого пианино, так и скетча для управления им, нажимая на разные участки мембранного потенциометра. Если схема выполнена правильно, и скетч не содержит опечаток, зуммер должен воспроизводить разные ноты. Попробуйте теперь сыграть какую-либо мелодию.

Что ж, когда и схема, и скетч работают должным образом, сделаем из нашего прототипа что-то похожее на собственно пианино.

Собираем мини-пианино

Чтобы сделать из нашего прототипа более функциональное пианино, нужно только расположить мембранный потенциометр на плоской поверхности и обозначить на нем клавиши.

К сожалению, выводы мембранного потенциометра недостаточно длинные и недостаточно толстые, чтобы вставить их в гнездовой разъем перемычки. Поэтому, чтобы использовать его вне макетной платы, рекомендуется припаять на его выводы перемычки со штыревыми разъемами на обоих концах, как показано на рис. 10.11.

Рис. 10.11. Припаиваем перемычки к выводам мембранного потенциометра

Впрочем, можно вставить выводы мембранного потенциометра в гнездовые разъемы перемычек, а затем зажать выводы в гнездах, расплющив гнезда плоскогубцами. При этом нужно следить за тем, чтобы между выводами датчика и гнездовыми разъемами перемычек был надежный контакт.

На обратной стороне мембранного потенциометра имеется клейкая подложка, которой его можно приклеить к любой твердой поверхности типа небольшого куска картона (рис. 10.12, *слева*) или даже на сам держатель платы Arduino и макетной платы (рис. 10.12, *справа*).

Длина мембранного потенциометра составляет 50 мм. Эту длину нужно разделить на восемь клавиш, а это означает, что ширина каждой клавиши должна быть около 6,5 мм. Разметьте восемь клавиш на клейкой ленте или на листе бумаги, как показано на рис. 10.13.

Затем закрепите размеченную клавиатуру на верхней стороне мембранного потенциометра. Готовая клавиатура должна выглядеть примерно так, как показано на рис. 10.14. Для удобства клавиши можно обозначить соответствующими нотами.

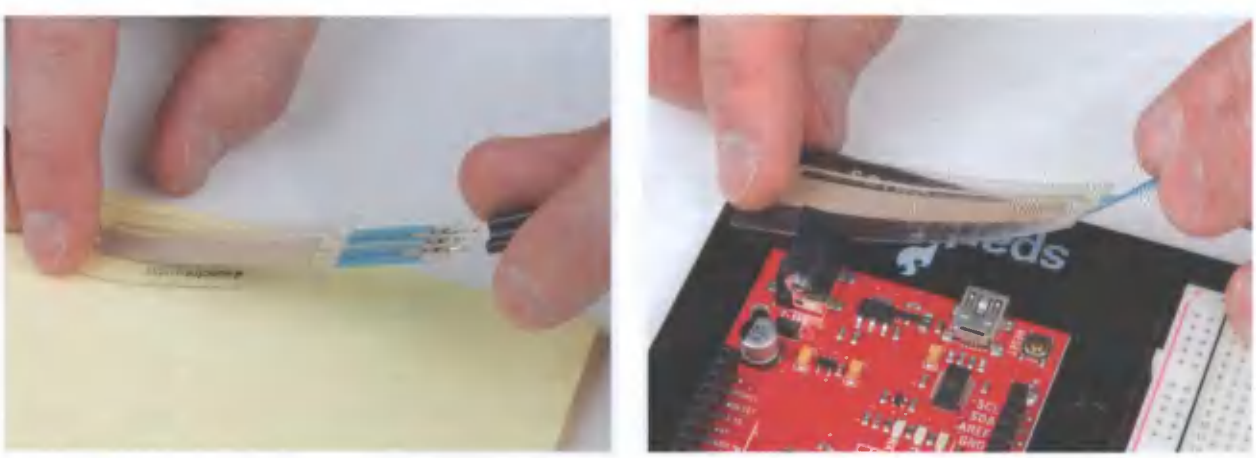


Рис. 10.12. Приклеиваем мембранный потенциометр к держателю платы Arduino (*слева*) или к кусочку картона (*справа*)

Рис. 10.13. Разметка клавиш для мини-пианино

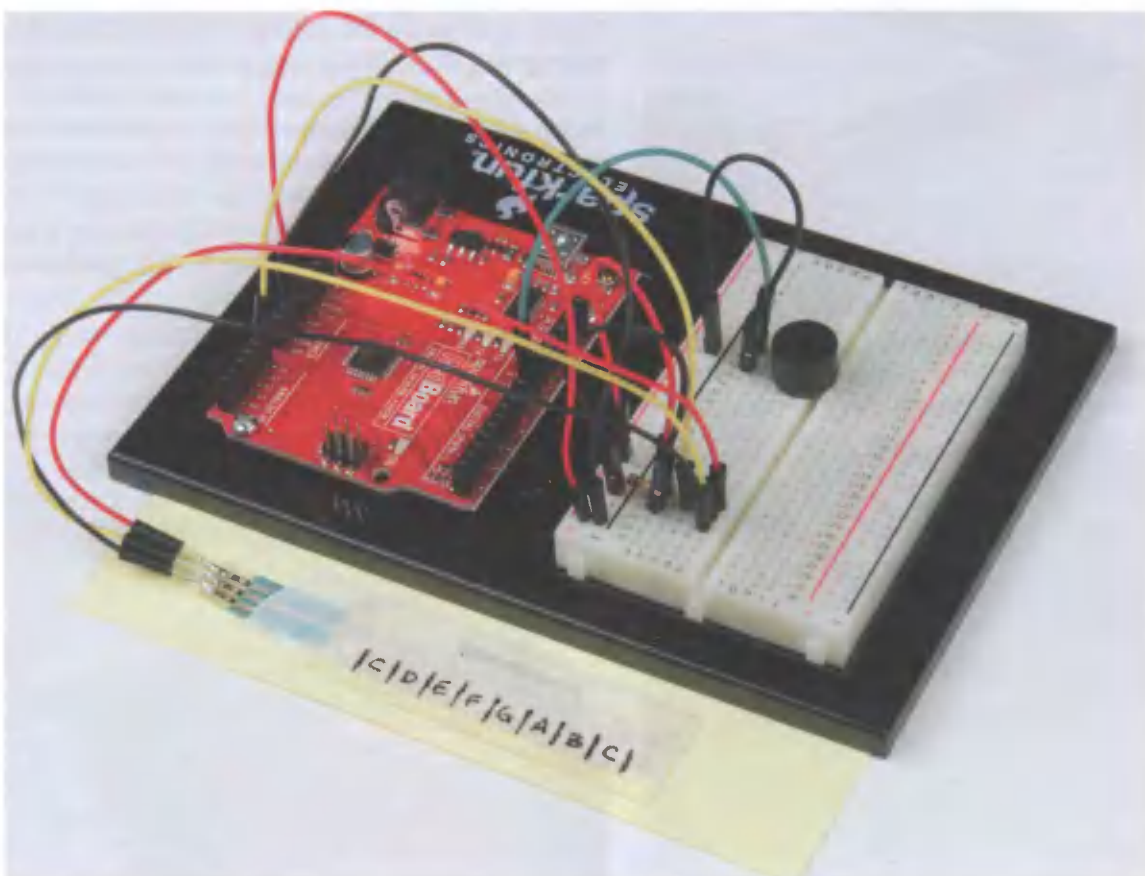


Рис. 10.14. Готовая клавиатура электронного мини-пианино

Идем дальше...

В этом проекте мы рассмотрели использование зуммера и мембранного потенциометра. Далее приводится несколько предложений по расширению области применения полученных знаний.

Экспериментируем с кодом

Поэкспериментируйте с кодом, чтобы посмотреть, какие другие интересные звуки можно получить. Например, для воспроизведения нот в другой тональности или другой гаммы можно заменить в массиве частоты.

В табл. 10.3 приводятся частоты для изменения тональности нашего пианино. Двумя наиболее часто используемыми музыкальными гаммами являются гамма До мажор и гамма Соль мажор. Найдите в Интернете ноты для каких-либо произведений и попробуйте играть по ним, просто сочиняйте свои собственные музыкальные произведения или попробуйте сыграть на своем мини-пианино какую-нибудь популярную мелодию.

Таблица 10.3. *Приблизительные частоты для некоторых нот в трех октавах*

Нота	Прибл. частота	Нота	Прибл. частота	Нота	Прибл. частота
C_3	131	C_4	262	C_5	524
$C^{\#}_3/D^{b}_3$	139	$C^{\#}_4/D^{b}_4$	277	$C^{\#}_5/D^{b}_5$	554
D_3	147	D4	294	D_5	587
$D^{\#}_3/E^{b}_3$	156	$D^{\#}_4/E^{b}_4$	311	$D^{\#}_5/E^{b}_5$	622
E_3	165	E_4	330	E_5	659
F3	175	F4	349	F_5	698
$F^{\#}_3/G^{b}_3$	185	$F^{\#}_4/G^{b}_4$	370	$F^{\#}_5/G^{b}_5$	740
G_3	196	G_4	392	G_5	784
$G^{\#}_3/A^{b}_3$	208	$G^{\#}_4/A^{b}_4$	415	$G^{\#}_5/A^{b}_5$	831
A3	220	A_4	440	A_5	880
$A^{\#}_3/B^{b}_3$	233	$A^{\#}_4/B^{b}_4$	466	$A^{\#}_5/B^{b}_5$	932
B_3	247	B_4	494	B_5	988

Модифицируем схему и код

Чтобы сделать проект более интересным, добавьте к нему кнопку переключения октав — это предоставит в ваше распоряжение 16 нот вместо 8-ми. Подключите один вывод кнопки к гнезду вывода 2 платы Arduino, а другой — к шине отрицательного питания макетной платы (рис. 10.15).

Для управления кнопкой переключения октав в скетч нужно добавить всего лишь несколько новых строк кода, которые будут переключать переменную множителя при каждом нажатии кнопки. Такой способ переключения называется *машиной состояний*, поскольку при каждом нажатии кнопки будет меняться состояние переменной.

Скетч для этой модификации (файл P10_TinyPiano.ino) находится в архиве ресурсов книги, доступном по адресу **https://www.nostarch.com/arduinoinventor/**.

В дополнительном коде объявляется переменная состояния octaveMultiplier, и с помощью команды pinMode() вывод 2 платы Arduino конфигурируется для работы в режиме ввода данных (INPUT). При нажатой кнопке переключения октав осуществляется инкрементирование переменной состояния

Примечание

В табл. 10.3 приведены ноты в латинской системе записи. Чтобы российскому читателю, не знакомому с такой записью, стало понятнее, какие ноты имеются в виду, посмотрите соответствие латинских нот русским, приведенное в табл. 10.1, и учтите, что знаки # и b, записанные у нот в верхнем регистре, обозначают соответственно диез и бемоль, а цифры, записанные у нот в нижнем регистре, обозначают номер октавы. Таким образом, запись $G^{\#}_3$ соответствует ноте Соль бемоль 3-й октавы.

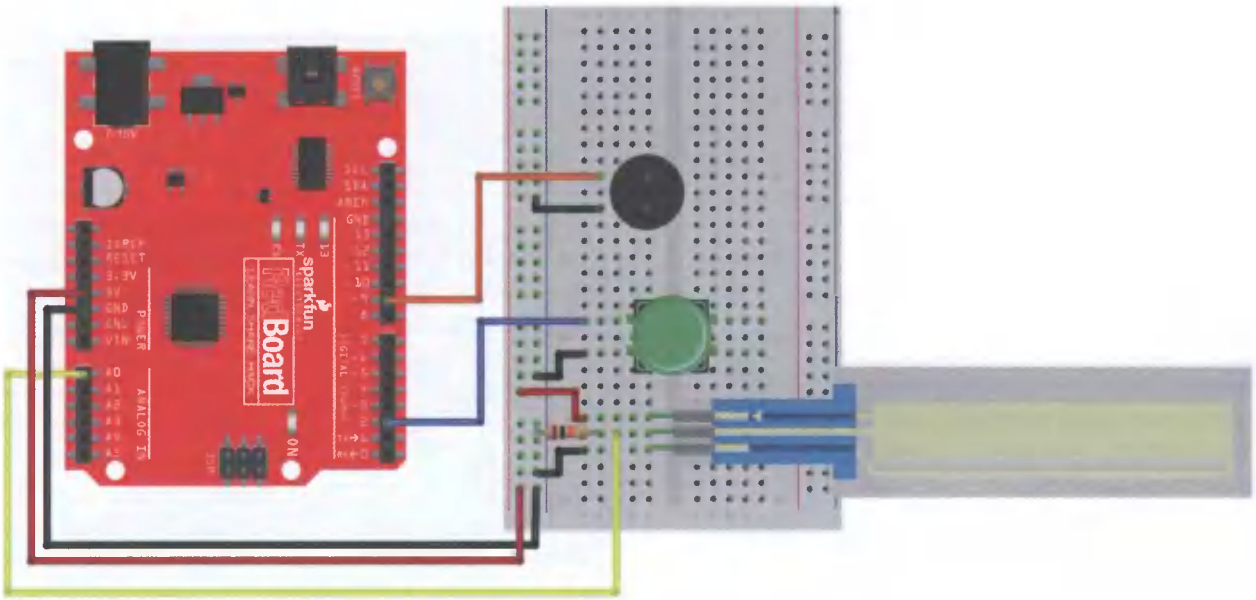


Рис. 10.15. Добавление кнопки на вывод 2 платы Arduino для переключения октав

octaveMulitplier, в результате чего частота ноты повышается.

Попробуйте играть на пианино с этой кнопкой — теперь у вас есть целых 16 нот для игры на таком простом инструменте на основе Arduino.

Бонусный проект: цифровая труба

Использование мембранного потенциометра в качестве клавиатуры для пианино, конечно же, оригинальная идея. Бонусом к этой идее предлагается создание на основе Arduino музыкального инструмента, для управления которым вместо мембранного потенциометра в качестве клавиш используются настоящие кнопки.

Назовем этот последний проект *цифровой трубой*. Причина для такого названия станет ясна чуть позже. В цифровой трубе три кнопки служат для указания воспроизводимой ноты, а четвертая кнопка запускает воспроизведение. Такая организация инструмента похожа на организацию трубы с ее тремя клапанами. С помощью трех кнопок можно указать до восьми разных нот, нажимая их, как показано в табл. 10.4.

Таблица 10.4. *Комбинации кнопок цифровой трубы*

Кнопка 1	Кнопка 2	Кнопка 3	Нота
Отпущена	Отпущена	Отпущена	C (262 Гц)
Отпущена	Отпущена	Нажата	D (294 Гц)
Отпущена	Нажата	Отпущена	E (330 Гц)
Отпущена	Нажата	Нажата	F (349 Гц)
Нажата	Отпущена	Отпущена	G (392 Гц)
Нажата	Отпущена	Нажата	A (440 Гц)
Нажата	Нажата	Отпущена	B (494 Гц)
Нажата	Нажата	Нажата	C (524 Гц)

Возможно, вы уже распознали в этих комбинациях двоичную последовательность, ведущую счет: 000, 001, 010, 011, 100, 101, 110 и 111.

Чтобы высвободить на макетной плате место для кнопок, переместите зуммер к верхнему краю платы, а затем вставьте кнопки, как показано на рис. 10.16.

Скетч для этой модификации (файл ***P10_TinyBinaryTrumpet.ino***) находится в архиве ресурсов книги, доступном по адресу **https://www.nostarch.com/arduinoinventor/**.

Чтобы играть на цифровой трубе, потребуется определенная тренировка, в которой вам поможет список комбинаций нажатия кнопок, приведенный в табл. 10.4. И хотя в таблице показано всего лишь восемь комбинаций, в действительности с четырьмя кнопками можно воспроизводить до 16 разных нот. Подумайте, как модифицировать скетч для решения этой задачи.

Независимо от того, какой инструмент вам нравится больше — электронное пианино или цифровая труба, — авторы надеются, что любой из них поможет вам в вашей музыкальной карьере. Только найдите терпеливых слушателей, которым вы сможете продемонстрировать свои новые навыки. И не обращайте внимания на критику. Моцарта тоже в свое время критиковали.

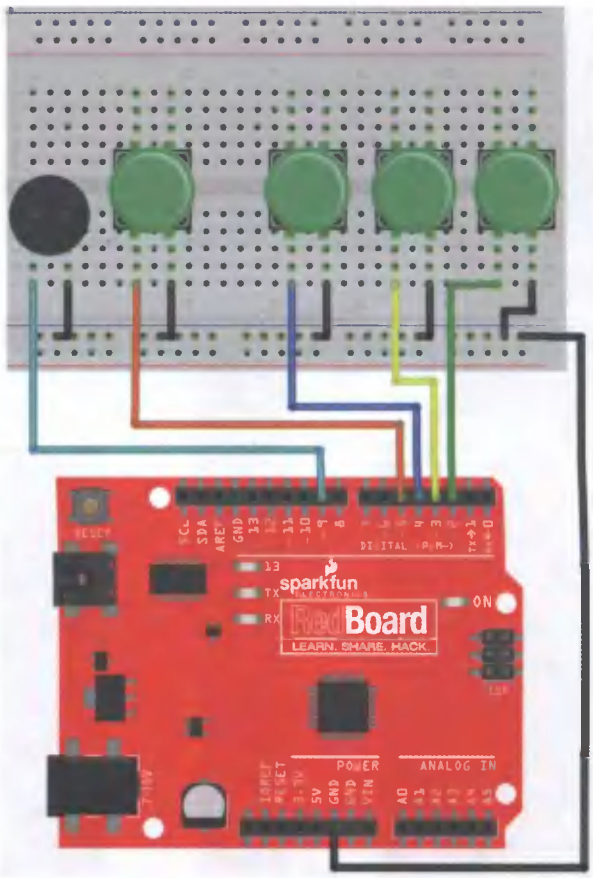
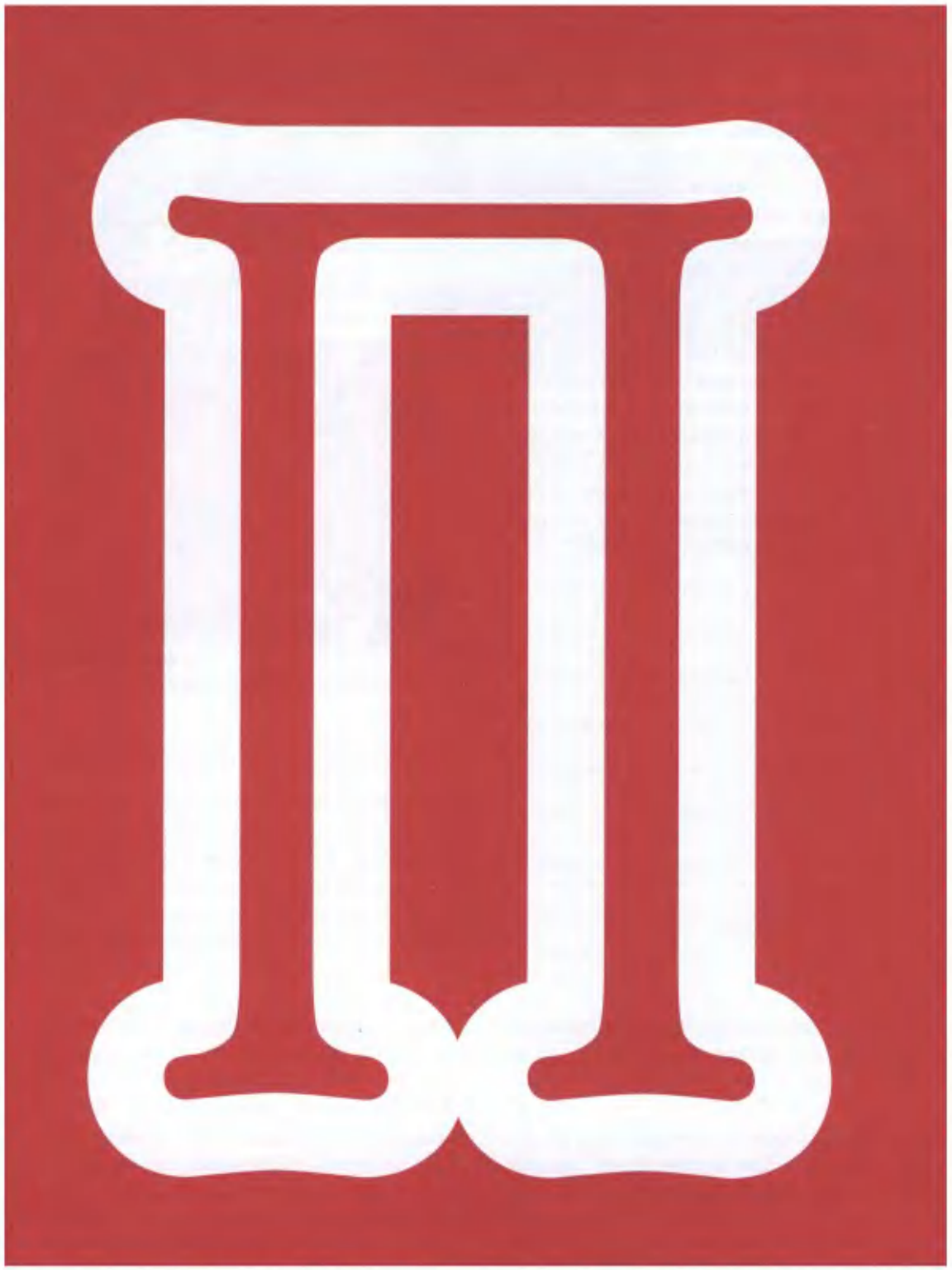


Рис. 10.16. Монтажная схема цифровой трубы



ПРИЛОЖЕНИЕ. Дополнительные практические сведения по электронике

В этом приложении рассматривается использование мультиметра и паяльника, а также предоставляется информация о том, как расшифровывать значение сопротивления резисторов по их цветным полосам.

Электрические измерения с помощью мультиметра

Мультиметр (тестер) является одним из незаменимых инструментов для диагностирования и поиска и устранения неисправностей в электронных схемах. Как можно понять по его названию, это прибор, способный измерять различные параметры электрических цепей, а именно ток, напряжение, сопротивление и неразрывность.

Теперь давайте разберемся, как с мультиметром работать. Для наглядной демонстрации мы взяли мультиметр VC830L компании SparkFun (TOL-12966), показанный на рис. П.1, но рассматриваемые приемы работы относятся к большинству мультиметров этой категории.

Функциональные части мультиметра

Мультиметр содержит три основные части: дисплей, ручку выбора функций и разъемы (рис. П.1).

Дисплей обычно может отображать четыре цифры и знак минус. Ручка выбора функции позволяет выбирать режим измерения определенного параметра электрической цепи: ток (мА), напряжение (В) или сопротивление (Ом). Каждый режим измерений разбит на несколько диапазонов с разными максимальными значениями.

Рис. П.1. Типичный мультиметр

В некоторых мультиметрах единицы измерения не отображаются. В таких случаях предполагается, что отображаемые значения измеряются в таких же единицах, как и выбранный максимальный предел данного режима измерений. Например, если указатель выбора режима установлен на максимальный предел 200 Ω (200 Ом), то отображаемые значения измеряются в омах. А если указатель установлен на 2, 20 или 200 kΩ, то отображаемые значения измеряются в килоомах.

Большинство мультиметров содержат в комплекте два щупа, которые вставляются в два из трех гнездовых разъемов на передней панели мультиметра. Эти разъемы обозначены: COM, mAVΩ и 10A. Маркировка COM означает *общий*[1] *(провод), и вставляемый в это гнездо щуп всегда подключается к «земле» или отрицательному полюсу питания схемы. В гнездо mAVΩ вставляется щуп для измерения тока* (mA — мА) *до 200 мА, напряжения* (V — В) и сопротивления (Ω — Ом). В гнездо 10A вставляется щуп для измерения токов, превышающих 200 мА.

Большинство щупов для мультиметров оснащены стандартным штекером типа «банан» на конце, который вставляется в гнездо мультиметра. Другой конец стандартного щупа оснащается штыревым пробником, но также имеются щупы с зажимом типа «крокодил», клипсой или иным типом наконечника. Для большинства измерений черный щуп вставляется в гнездо COM, а красный — в гнездо mAVΩ.

Определение неразрывности электроцепи

Этот режим измерений обозначается на мультиметре символом, показанным на рис. П.2.

Определение неразрывности электроцепи является, пожалуй, самой важной функцией при

[1] От англ. Common — общий.

поиске и устранении неисправностей схем. Эта функция позволяет выполнить проверку электрической цепи на отсутствие разрывов в ней, а также убедиться в наличии или отсутствии электрических соединений.

Проверка выполняется подключением одного щупа к одному концу цепи, а другого щупа — к другому. При отсутствии разрывов встроенный в мультиметр зуммер будет издавать непрерывный тон. В ранних версиях тестеров вместо тона использовался звонок, и поэтому проверка на неразрывность цепи в профессиональных кругах называется *прозвонкой* электроцепи. С помощью этой функции можно проверить, какие гнезда макетной платы соединены между собой, а какие нет.

Диаметр щупов обычно слишком велик, чтобы вставить их непосредственно в гнезда макетной платы, но эта проблема решается просто: сначала провода вставляются в гнезда платы, а уже к проводам прижимаются пробники. Эту функцию также можно использовать для трассировки проводников схемы, приставив пробники к противоположным концам проводника. При проверке неразрывности цепи не имеет значения, какой щуп подключать к какому концу цепи, поскольку проверяется лишь сам факт соединения этих концов между собой электрически.

Измерение сопротивления

Этот режим измерений обозначается на мультиметре символом, показанным на рис. П.3.

Описанная только что проверка цепи на неразрывность является, по сути, видом измерения сопротивления. В частности, зуммер звучит и при низком сопротивлении измеряемого элемента цепи. Но чтобы узнать значение сопротивления, прибор необходимо переключить в режим измерения сопротивления. Для этого переключатель режимов нужно установить на одну из меток в области, обозначенной символом «омега» Ω, который представляет единицу сопротивления Ом. При этом нужно обязательно убедиться, что на измеряемый резистор или другой элемент не подается питание. Также, в целях повышения точности

Рис. П.2. Символ режима определения неразрывности электроцепи

Рис. П.3. Символ режима измерения сопротивления

измерений, по крайней мере один вывод измеряемого элемента не должен быть ни к чему подключен. Как и многие другие электрические элементы, резистор имеет два вывода. Чтобы измерить его сопротивление, просто прикоснитесь концами щупов к его выводам. Так же, как и при определении неразрывности электроцепи, не имеет значения, какой щуп подключать к какому выводу резистора.

Область измерения сопротивления разбита на несколько диапазонов с разными предельными значениями измерения. Например, чтобы с точностью измерить сопротивление резистора малого номинала, переключатель режимов следует установить на указатель одного из низших пределов — например, на 200 Ω.

Если измеряемое сопротивление выше установленного максимального значения, на дисплее мультиметра будут отображаться символы **[1.]** без нулей. В таком случае просто установите переключатель на следующий более высокий предел и повторите попытку.

Попробуйте измерить сопротивление какого-либо резистора. В частности, попробуйте измерить сопротивление резистора номиналом 330 Ом (оранжевая, оранжевая и коричневая полоски). Какие показания у вас получаются на разных установленных пределах? Реальное сопротивление всех резисторов отличается от указанного номинального в пределах допустимого отклонения, или допуска. Такой допуск обычно составляет 5 процентов и указывается четвертой полоской. Каков процент отклонения измеренного вами значения сопротивления резистора от указанного номинального? Находится ли он в пределах допустимого?

Теперь измерьте сопротивление фоторезистора. При этом держите над фоторезистором руку или какой-либо непрозрачный предмет и замерьте сопротивление фоторезистора при разной высоте над ним затеняющего его объекта.

Измерение напряжения

Этот режим измерений обозначается на мультиметре символом, показанным на рис. П.4.

Рис. П.4. Символ режима измерения напряжения

Напряжение представляет собой разницу электрических потенциалов между двумя точками. Подобно режиму измерения сопротивления, режимы измерения напряжения также имеют несколько диапазонов с разными максимальными пределами.

Вы, наверное, обратили внимание, что режим измерения напряжения в действительности обозначается двумя символами: один — с прямыми линиями, другой — с одной волнистой линией. Символ с двумя прямыми линиями обозначает измерение напряжения постоянного тока, который является наиболее часто используемым типом тока в электронике (по крайней мере, в цифровой электронике). А волнистая линия обозначает переменный ток, примером которого может служить ток в электрической сети вашего дома. Таким образом, для измерения напряжения используется два режима: режим измерения постоянного напряжения и режим измерения переменного напряжения. Поэтому при измерении напряжения нужно в обязательном порядке правильно установить переключатель режимов. В подавляющем большинстве случаев это будет постоянное напряжение. Для большинства проектов этой книги самым лучшим диапазоном измерений будет диапазон с пределом 20 В, поскольку все напряжения в Arduino не превышают 5 В.

Для практики попробуйте измерить напряжение на плате Arduino. Для этого подключите плату Arduino к компьютеру и подсоедините черный щуп к гнезду, обозначенному GND («земля»). Затем касайтесь красным щупом разных точек на плате (при этом соблюдая максимальную осторожность, чтобы случайно не закоротить щупом соседние дорожки, что чревато опасностью вывода платы из строя) или гнезд выводов. Какие показания отображаются для вывода, обозначенного 5 В (5V)? А для вывода 3.3 В (3,3V)?

Измерение тока

Этот режим измерений обозначается на мультиметре символом, показанным на рис. П.5.

Рис. П.5. Символ режима измерения тока

Ток представляет собой поток зарядов в замкнутой электрической цепи. Сила тока определяется как скорость перемещения зарядов по цепи и измеряется в амперах (А). Чтобы измерить скорость перемещения зарядов и, таким образом, измерить силу тока в цепи, необходимо сделать разрыв цепи в точке, в которой нужно измерить силу тока, и подсоединить щупы мультиметра к обоим концам разрыва. При этом переключатель режима надо установить на измерение тока, а также на диапазон с необходимым пределом. Если ожидается, что ток в точке измерения будет выше 200 мА, установите переключатель на значение 10 А и также вставьте красный щуп в гнездо, обозначенное 10 А. Если вы не уверены, какой может быть сила тока в измеряемой точке, это будет самый безопасный диапазон, с которого начинать измерение. Установка неправильного диапазона может повредить мультиметр.

Например, чтобы измерить ток, проходящий через простую цепь из светодиода и резистора, щупы можно вставить в разрыв цепи между светодиодом и резистором, как показано на рис. П.6. При этом путь тока должен проходить через мультиметр. Поскольку это последовательная цепь, ток можно измерять в разрыве в любой точке цепи: до светодиода, после светодиода или после резистора.

При измерении тока важно, чтобы измеряемый ток не превышал установленного предела измерений мультиметра. Сведения о максимально допустимой величине тока, который может измерять ваш мультиметр, можно найти в документации на него. Превышение этого предела может вызвать перегорание плавкого предохранителя мультиметра. Но не переживайте, если это случится, поскольку новый предохранитель стоит совсем недорого. Чтобы заменить перегоревший предохранитель, придется открыть заднюю крышку корпуса мультиметра, отвинтив крепежные шурупы. При подключении красного щупа в стандартный разъем mAVΩ можно безопасно измерять ток величиной до 200 мА.

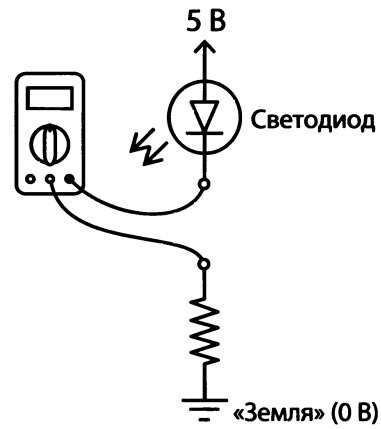


Рис. П.6. Подключение мультиметра в разрыв цепи для измерения тока

Работа с паяльником

Навыки пайки являются одними из самых основных навыков, которыми следует владеть при создании прототипов электронных проектов. Процесс пайки состоит из нанесения на спаиваемые компоненты расплавленного специального металла, называемого *припоем*, так, чтобы он обволакивал эти компоненты, и затем позволения ему остыть, надежно скрепив компоненты. Пример пайки показан на рис. П.7.

Обычный припой (рис. П.8) представляет собой проволоку из сплава олова и свинца и плавится при сравнительно низкой температуре. В частности, современные припои плавятся при температуре около 180 °С, что приблизительно равно температуре при выпечке печенья. Большинство припоев, используемых для пайки электронных деталей, имеют сердечник из *флюса*. Флюс — это специальное вещество, способствующее надежной адгезии (прилипанию) припоя к спаиваемым деталям. При плавлении припоя флюс помогает удалять грязь со спаиваемых поверхностей и улучшает их обволакивание припоем.

Рис. П.8. Катушка припоя

Припой плавится с помощью *паяльника*. Большинство паяльников имеют длину около 20–25 см и состоят из двух основных частей: деревянной или пластмассовой ручки и заключенного в металлический кожух нагревательного элемента с жалом (рис. П.9).

Существует много разных типов и стилей паяльников. Самые дешевые стоят около 10 долларов (60 рублей) и обычно имеют постоянный нерегулируемый нагрев. Но рекомендуется приобрести для работы паяльник с каким-либо типом

Рис. П.7. Пайка

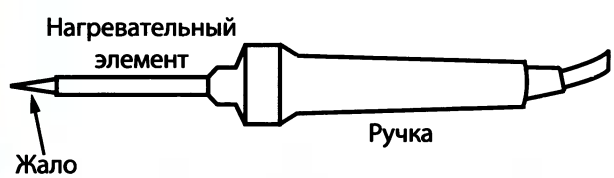


Рис. П.9. Типичный паяльник

регулировки нагрева жала. Оптимальная рабочая температура жала паяльника составляет около 345 °С. При слишком высокой температуре жало будет быстро окисляться и загрязняться, а при слишком низкой температуре припой не станет плавиться. Наличие регулировки температуры жала решает эти две проблемы и стоит нескольких лишних рублей.

Будьте осторожны при работе с паяльником — после включения жало нагревается очень быстро до температуры плавления металла. Это очень высокая температура! Всегда беритесь за паяльник со стороны ручки и никогда со стороны нагревательного элемента, даже когда вы думаете, что он выключен.

Также следует защитить рабочую поверхность стола, положив на нее кусок картона, коврик для резки или кусок фанеры или доски. Прежде чем приступать к работе с паяльником, всегда наденьте какую-либо защиту для глаз. Горячий припой и флюс имеют свойство иногда разбрызгиваться. Хотя эти брызги совсем небольшие, они достаточно велики, чтобы повредить ваши глаза. Так что лучше обезопасить их от этой неприятности.

Разогревание паяльника

Для работы с паяльником включите его и дайте время, чтобы жало нагрелось до рабочей температуры. В зависимости от типа паяльника это может занять от около 30 секунд до пары минут. Пока паяльник разогревается, убедитесь, что он установлен на подставке таким образом, чтобы жало не касалось стола или другой поверхности, на которой вы работаете.

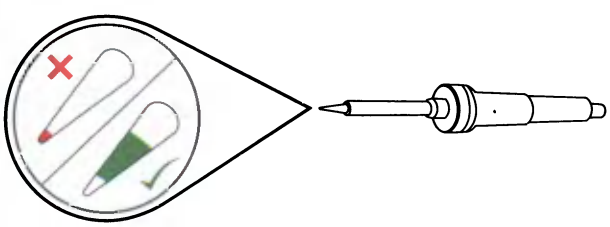


Рис. П.10. Температура стороны жала намного выше температуры кончика жала

Неправильно: Не используйте самый кончик жала.

Правильно: Используйте сторону жала, ближнюю к его кончику, — «сладкое пятнышко».

Разогретый надлежащим образом паяльник должен легко плавить припой. Проверьте это, коснувшись кусочка припоя стороной жала, ближней к его кончику. Это самая горячая область жала (рис. П.10), и именно ее нужно использовать для разогрева компонентов и плавки припоя. Если припой сразу же начал плавиться, жало достаточно горячее, и можно приступать к работе.

Советы по улучшению навыков пайки

Вопреки тому, что вы можете предполагать, при пайке мы не касаемся жалом паяльника непосредственно самого припоя, чтобы расплавить его. Вместо этого мы разогреваем жалом спаиваемые компоненты в течение около 2–3 секунд, а затем касаемся припоем нагретой точки, в результате чего он расплавится. Расплавленный припой всегда стремится к источнику тепла и оседает на самой горячей точке компонента. Если плавить припой, касаясь им непосредственно жала паяльника, он может собраться в каплю на жале и отказаться переходить на спаиваемые детали. Если такое случится, просто очистите жало паяльника (см. об этом далее) и повторите попытку. Удерживайте паяльник в своей «рабочей» руке со стороны ручки таким образом, как будто бы это карандаш. В другой руке удерживайте отрезок припоя. Будьте осторожны, чтобы не держаться за припой слишком близко к его расплавляемой области, чтобы не обжечь пальцы.

Прикоснитесь стороной жала паяльника к спаиваемым деталям. При этом жало должно касаться обеих спаиваемых деталей, чтобы они обе разогревались равномерно (см. рис. П.7). Отсчитайте три секунды: «раз одна тысяча, два одна тысяча, три одна тысяча»[2].

[2] Английский метод отсчета секунд: one one-thousand, two one-thousand, three one-thousand. Вроде бы должен работать и по-русски. Или пользуйтесь секундомером.

Затем, продолжая касаться жалом компонентов, прикоснитесь кончиком отрезка припоя к спаиваемому месту. Помните, что расплавленный припой стремится к источнику тепла.

Когда соединение содержит достаточно припоя, удалите остальной припой, но удерживайте жало на месте еще одну секунду. Это позволит припою равномерно распределиться и осесть в месте пайки. Удалите паяльник от места пайки и установите его обратно на подставку.

Поверхность застывшего припоя в хорошей пайке должна быть гладкой и немного блестящей. Место пайки выводов деталей в отверстиях печатной платы часто выглядит как конус вулкана или шоколадная конфетка Hershey's kiss[3]. Чтобы научиться хорошо паять, необходима практика, поэтому если ваша пайка не выглядит гладкой и блестящей, попробуйте еще нагреть место соединения, чтобы снова дать припою растечься и осесть на деталях, или же добавьте еще чуток припоя.

На рис. П.11 показаны примеры правильного паяного соединения и некоторые распространенные огрехи пайки и возможные способы их устранения.

[3] См. **https://en.wikipedia.org/wiki/Hershey's_Kisses**.

A	**Правильная пайка.** Припой растекся равномерно по месту пайки сверху вниз, полностью обволакивая соединение
B	**Огрех:** Припой собрался в каплю над контактной площадкой печатной платы, не создавая контакта с ней. **Решение:** Повторно нагрейте все соединение: как вывод детали, так и контактную площадку. Возможно, не помещает добавить еще немного припоя.
C	**Огрех:** Слишком мало припоя, соединение слабое. **Решение:** Повторно нагрейте соединение и добавьте припоя, чтобы получилось правильное соединение, как в позиции A.
D	**Огрех:** Некачественное соединение. И до чего же уродливое... **Решение:** Повторно нагрейте соединение и добавьте припоя. Перемещайте жало паяльника вокруг соединения, чтобы равномерно нагреть все спаиваемые детали, и чтобы припой растекся по всему соединению.
E	**Огрех:** Слишком много припоя, получилась перемычка между двумя выводами. **Решение:** Уберите косичкой (см. далее) излишек припоя.

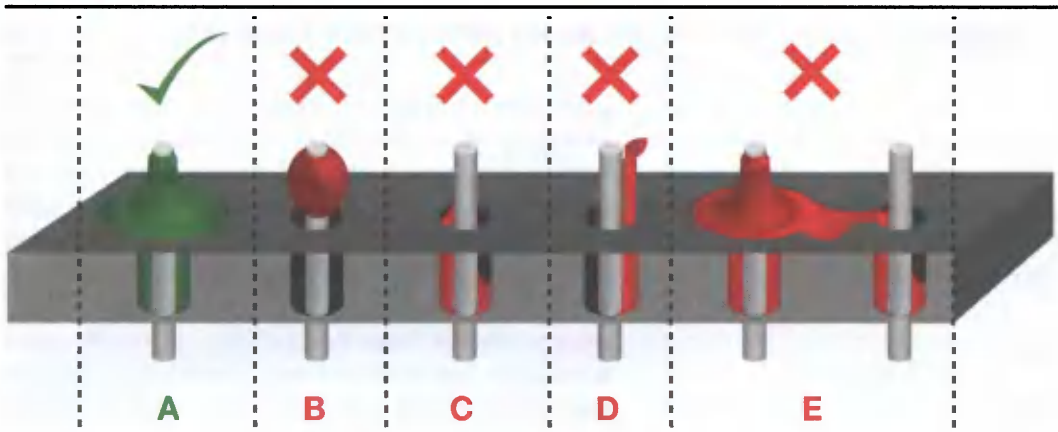


Рис. П.11. Правильное паяное соединение и наиболее распространенные огрехи пайки и способы их устранения

Очистка паяльника

Один из секретов получения хороших паяных соединений заключается в чистоте жала паяльника. Жало рекомендуется очищать перед каждым использованием паяльника. Это можно делать, нагрев паяльник и потерев жало о латунный скребок (наподобие скребка из нержавеющей стали для мытья посуды) или о мокрую губку, чтобы стереть излишек припоя или накопившиеся окисления.

Если жало настолько загрязнилось, что указанные методы очистки не работают, его можно очистить с помощью специальной смеси слабой кислоты и припоя Tip Tinner and Cleaner (TOL-13246). Для этого разогрейте паяльник, вставьте жало паяльника в состав для чистки и лужения и удерживайте в течение около 10–15 секунд, чтобы дать составу разъесть окисления и загрязнения. Затем извлеките паяльник и вытрите жало о губку. Повторите процедуру, если необходимо. После этого жало паяльника должно быть покрыто тонким слоем припоя.

> **Примечание**
>
> Некоторые типы припоя содержат свинец, который является токсичным металлом. Но независимо от типа используемого вами припоя, настоятельно рекомендуется мыть руки после выполнения паяльных работ.

Советы по работе с паяльником

На рис. П.12 показано еще несколько советов по работе с паяльником.

Правильно: Жало паяльника касается одновременно как вывода компонента, так и кольца контактной площадки.

Правильно: Удерживая жало паяльника в контакте с выводом и кольцом контактной площадки, подавайте припой в место пайки..

Неправильно: Не расплавляйте припой на жале паяльника, нанося его затем жалом на соединение.

Правильно: Очищайте жало паяльника, когда на нем накопятся окисления.

Рис. П.12. Дополнительные советы и рекомендации по работе с паяльником

Умение работать с паяльником должно присутствовать в арсенале навыков любого любителя электроники. Если вы решите сделать свои прототипы постоянно действующими и прочными, монтаж пайкой обеспечит намного более надежные и долговременные соединения.

Дополнительные инструменты для паяльных работ

Далее кратко рассматриваются несколько дополнительных инструментов, использование которых поможет вам получать качественные паяные соединения на постоянной основе. Это инструменты для удерживания спаиваемых деталей, очистки соединений и удаления излишков припоя.

«Третья рука»

«Третья рука» — это, по сути, зажим для удерживания спаиваемых деталей, и он станет одним из наиболее полезных помощников при выполнении паяльных работ. Существует много разновидностей «третьей руки», но большинство из них представляют собой просто пару зажимов типа «крокодил» на стойке с тяжелым основанием, с помощью которых можно закрепить спаиваемые детали, освобождая руки для держания паяльника и припоя. Многие варианты «третьей руки» оснащены увеличительным стеклом и небольшим держателем для паяльника. Пример такой «третьей руки» показан на рис. П.13.

Рис. П.13. «Третья рука» с увеличительным стеклом и держателем для паяльника

Флюс-аппликатор

Одним из секретов получения качественных паяных соединений является чистота спаиваемых деталей, что обеспечивается нанесением на место пайки флюса. Флюс — это чистящая жидкость небольшого уровня кислотности, которая часто изготавливается на основе сосновой канифоли. Флюс-аппликатор (рис. П.14) работает наподобие чернильного маркера — просто нажимаем кончиком маркера на спаиваемое соединение, пока на него не перейдет небольшая капля флюса. После этого приложите жало паяльника непосредственно в точку пайки и внесите припой. С использованием флюса припой расплавится намного быстрей и будет иметь лучшую адгезию со спаиваемыми деталями.

Рис. П.14. Аппликатор с водорастворимым флюсом

Флюс творит чудеса при пайке, но он слегка едкий, поэтому следите за тем, чтобы свести к минимуму контакт флюса с кожей, и мойте руки сразу же после работы с ним.

Косичка для удаления припоя

Иногда случается, что в место пайки наносится слишком много припоя, или припой попадает в место, где его не должно быть. С удалением излишков припоя вам помогут следующие два приспособления. Первым из них является косичка из тонких медных проводов, показанная на рис. П.15.

Косичкой пользуются следующим образом. Положите конец косички поверх места с лишним припоем и приложите к косичке жало паяльника. Когда косичка нагреется до достаточной температуры, она расплавит находящийся под ней припой и впитает его в себя. Убрав косичку, мы уберем вместе с ней и лишний припой.

Но нужно следить за тем, чтобы удерживать жало паяльника на косичке при ее снятии. Если убрать паяльник от косички слишком рано, она припаяется к соединению. В таком случае просто снова нагрейте косичку и удалите ее.

Рис. П.15. Косичка для удаления припоя

Вакуумный отсос

Вторым инструментом для удаления лишнего припоя является вакуумный отсос. Этот инструмент похож на большой шприц. При нажатии на поршень шприца он сжимает находящуюся под ним пружину, и поршень фиксируется в нижнем положении защелкой. При нажатии на кнопку освобождения защелки под воздействием пружины поршень шприца резко поднимается вверх, создавая вакуум в носике.

Разогрейте место с лишним припоем, чтобы он полностью расплавился и стал текучим. Продолжая разогревать припой, быстро приложите носик отсоса к лужице припоя и нажмите кнопку освобождения поршня. Создаваемый поршнем вакуум должен всосать расплавленный припой в корпус отсоса.

Если убрать лишний припой с первого раза не получилось, повторите попытку. Использование этого инструмента требует определенной сноровки и быстроты действий. Иногда полезно добавить дополнительный припой к тому, который нужно убрать.

Полосатые резисторы

Резисторы бывают самых разных номиналов. Но как определить сопротивление резистора, если на нем нет никаких цифр или текста?

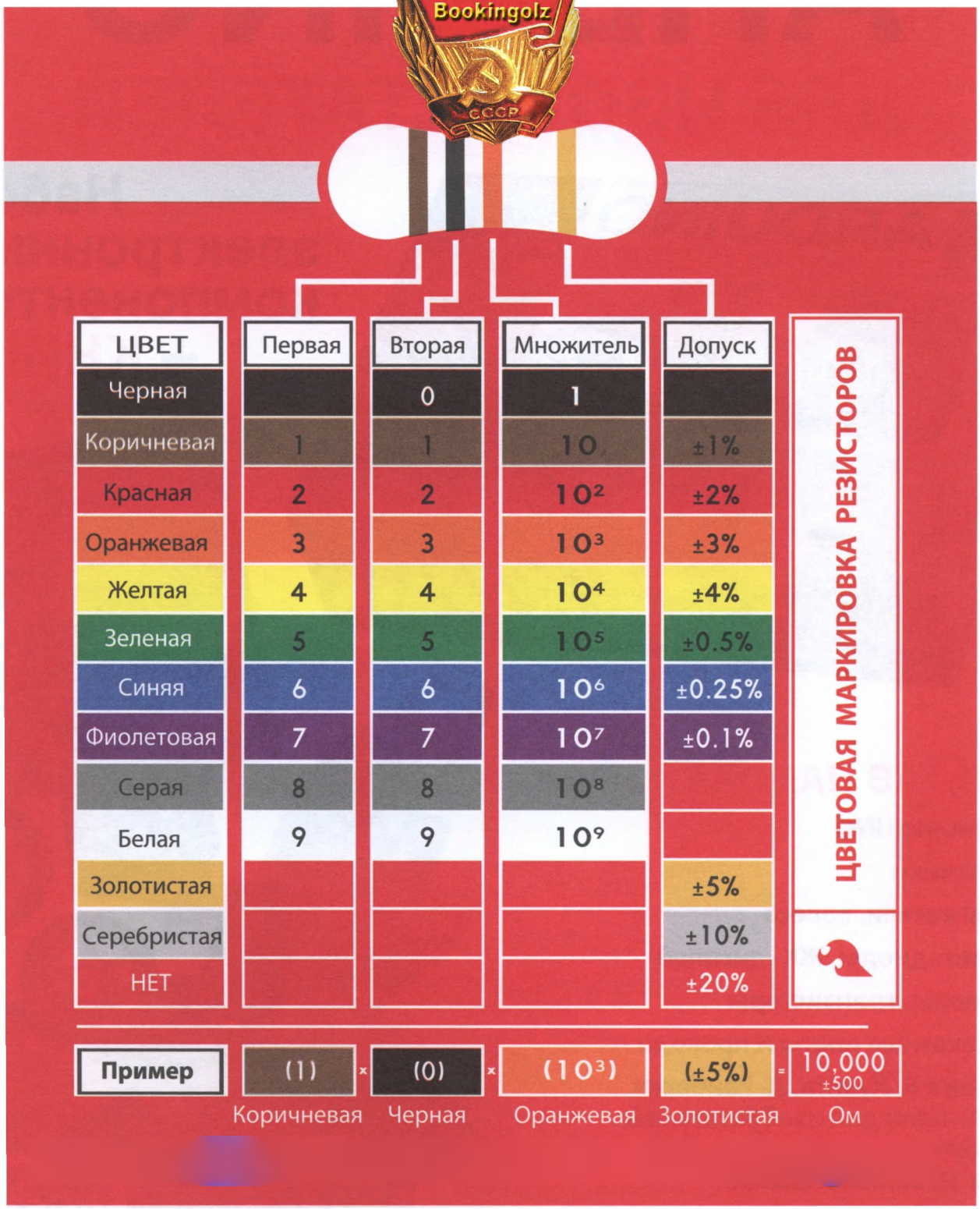
Значения сопротивлений резисторов обозначаются с помощью системы цветных полос. Принцип работы этой системы показан на рис. П.16.

Обычно резисторы имеют четыре или пять цветных полос. Последняя полоса обозначает допустимое отклонение в процентах действительного значения сопротивления резистора от номинального значения. Большинство резисторов имеют допуск величиной 5 процентов, который обозначается золотистой полосой. Это означает, что действительное значение резистора может отклоняться на 5 процентов от указанного номинального значения. Например, действительное сопротивление резистора с номинальным сопротивлением 10 кОм и допуском 5 процентов может быть где-то в диапазоне от 9,5 до 10,5 кОм. При этом считается, что такой резистор имеет сопротивление 10 кОм.

Маркировочные полосы читаются слева направо, с полосой допуска (обычно золотистой или серебристой), расположенной справа. В случае резисторов с четырьмя маркировочными полосами первые две полосы обозначают базовое значение, а третья — степень множителя 10. Для резисторов с пятью маркировочными полосами базовое значение обозначается первыми тремя полосами, а четвертая полоса обозначает множитель.

Определим, например, сопротивление резистора, обозначенное коричневой, черной и оранжевой полосами. Согласно таблице, представленной на рис. П.16, коричневая полоса обозначает 1, а черная 0. Таким образом, базовое значение равно 10. Третья полоса, оранжевая, обозначает степень 3 множителя 10, то есть 10^3 или 1000. Умножая базовое значение 10 на множитель 1000, получаем 10 000 Ом или 10 кОм. Наконец, четвертая полоса, золотистая, обозначает допуск 5 процентов.

Чтобы справка по цветовому коду резисторов всегда была у вас под рукой, можно вырвать из книги страницу с таблицей и разместить ее в легкодоступном месте. Не волнуйтесь — мы не скажем библиотекарше.



ЦВЕТОВАЯ МАРКИРОВКА РЕЗИСТОРОВ

ЦВЕТ	Первая	Вторая	Множитель	Допуск
Черная		0	1	
Коричневая	1	1	10	±1%
Красная	2	2	10^2	±2%
Оранжевая	3	3	10^3	±3%
Желтая	4	4	10^4	±4%
Зеленая	5	5	10^5	±0.5%
Синяя	6	6	10^6	±0.25%
Фиолетовая	7	7	10^7	±0.1%
Серая	8	8	10^8	
Белая	9	9	10^9	
Золотистая				±5%
Серебристая				±10%
НЕТ				±20%

Пример	(1)	×	(0)	×	(10^3)	(±5%)	=	10,000 ±500
	Коричневая		Черная		Оранжевая	Золотистая		Ом

Рис. П.16. Шпаргалка цветовой маркировки резисторов