Exploring *Arduino*®

Tools and Techniques for Engineering Wizardry

Jeremy Blum

Wiley

Джереми Блум

ИЗУЧАЕМ
ARDUINO

ИНСТРУМЕНТЫ И МЕТОДЫ ТЕХНИЧЕСКОГО ВОЛШЕБСТВА

Санкт-Петербург
«БХВ-Петербург»
2017

УДК 004 ББК
32.973.26 Б71

Блум Джереми

Б71 Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. — СПб.: БХВ-Петербург, 2017. — 336 с.: ил.

ISBN 978-5-9775-3585-4

Книга посвящена проектированию электронных устройств на основе микроконтроллерной платформы Arduino. Приведены основные сведения об аппаратном и программном обеспечении Arduino. Изложены принципы программирования в интегрированной среде Arduino IDE. Показано, как анализировать электрические схемы, читать технические описания, выбирать подходящие детали для собственных проектов. Приведены примеры использования и описание различных датчиков, электродвигателей, сервоприводов, индикаторов, проводных и беспроводных интерфейсов передачи данных. В каждой главе перечислены используемые комплектующие, приведены монтажные схемы, подробно описаны листинги программ. Имеются ссылки на сайт информационной поддержки книги. Материал ориентирован на применение несложных и недорогих комплектующих для экспериментов в домашних условиях.

Для радиолюбителей

УДК 004 ББК
32.973.26

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Перевод с английского	*Виктора Петина*
Редактор	*Леонид Конин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.03.17.
Формат 70x1001Лб. Уел. печ. л. 27,09.
Доп. тираж 2000 экз. Заказ № 1438 "БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.
Отпечатано в АО "Первая Образцовая типография" Филиал "Чеховский Печатный Двор"
142300, Московская область, г. Чехов, ул. Полиграфистов, д. 1 Сайт: www.chpd.ru, E-mail: sales@chpd.ru, тел. +7-499-270-73-59

ISBN 978-1-118-54936-0 (англ.)

ISBN 978-5-9775-3585-4 (рус.)

Оглавление

Моей бабушке,
которая каждый день поощряла меня
и вдохновляла на добрые дела

Об авторе

Джереми Блум недавно защитил магистерскую диссертацию по электронике и вычислительной технике в Корнельском университете, где ранее получил степень бакалавра в той же области. В Корнельском университете он участвовал в разработке и создании нескольких проектов "интеллектуальных" зданий по всему миру, осуществляемых известной организацией Cornell University Sustainable Design, удостоенной высокой оценки в США и других странах (Green Building Councils). Увлечение электроникой дало Джереми возможность применить полученные знания при проектировании автономных систем мониторинга домов, работающих на энергии солнца, революционных волоконно-оптических светодиодных систем освещения и систем контроля интеллектуальных солнечных батарей. Он также помогал в создании бизнес-инкубаторов, ежегодно способствовавших развитию десятков студенческих стартапов.

Джереми разработал отмеченные наградами методы управления протезированием, распознавания жестов, автоматизации зданий. Он спроектировал электронику для 3D-принтера MakerBot Replicator, который используется людьми во всем мире, а также такими известными организациями, как NASA. Джереми также разработал аппаратную часть и программное обеспечение 3D-сканера MakerBot Digitizer. Работая в исследовательской лаборатории Machines Lab, он внес вклад в создание самообучающихся роботов и 3D-принтеров, которые преобразили индивидуальное производство. Результаты этих исследований опубликованы в рецензируемых журналах и представлены на конференциях даже в такой далекой стране, как Индия.

Джереми создает и размещает на YouTube самые популярные в Интернете видеоуроки по ***Arduino***, которые просматривают миллионы людей. Он хорошо известен в международном сообществе программистов как автор проектов и учебных пособий с открытым исходным кодом, которые демонстрировались на канале Дискаве-ри и получили несколько наград на хакатонах. В 2012 году Американский институт инженеров по электротехнике и электронике присвоил Джереми звание "Новое имя в инжиниринге" (New Face of Engineering).

Джереми оказывает инженерные консалтинговые услуги через свою фирму Blum Idea Labs LLC и преподает основы инжиниринга для студентов в Нью-Йорке. Его кредо — улучшение качества жизни людей и нашей планеты через творческие инженерные решения. Вы можете узнать больше о Джереми и его работе на его сайте http://www.jeremyblum.com.

О техническом редакторе

Скотт Фицджеральд — дизайнер и педагог, использующий платформу ***Arduino*** для обучения с 2006 года. С 2005 года он преподает вычислительную технику по программе Interactive Telecommunications Program (ITP) в Нью-Йоркском университете, знакомя художников и дизайнеров с миром микроконтроллеров. Скотт сотрудничает с командой ***Arduino*** в документальном сопровождении новых продуктов и создании обучающих пособий для знакомства с платформой ***Arduino***. Он был техническим редактором второго издания Making Things Talk в 2011 году и является автором книги, которая сопровождает официальный ***Arduino*** Starter Kit в 2012 году.

Благодарности

Прежде всего, я должен поблагодарить моих друзей из издательства Wiley за то, что эта книга увидела свет. В первую очередь, я признателен Мери Джеймс за поощрение моего желания написать книгу и Дженнифер Линн за поддержку при работе над каждой главой. Я также хочу сказать большое спасибо Скотту Фицджеральду за критические замечания при техническом редактировании рукописи.

Без квалифицированной помощи сотрудников компании element14 я не смог бы выпускать обучающих уроков по ***Arduino***, ставших прелюдией данной книги. Особо хочу отметить Сабрину Дейч и Сагар Джефани — замечательных партнеров, с которыми я имел честь работать.

Большую часть книги я писал, обучаясь в магистратуре и работая одновэеменно в двух фирмах. Выражаю огромную благодарность профессорам и коллегам, которые помогали мне справляться с возросшими обязанностями.

Наконец, я хочу поблагодарить мою семью, особенно моих родителей и брата Дэвида, чья постоянная поддержка напоминает мне о важности того, что я делаю.

Введение

Мы живем в прекрасное время. Я люблю говорить, что мы устремлены в будущее. С помощью инструментов, доступных вам сегодня, которые мы рассмотрим в этой книге, появилась реальная возможность изменять окружающий мир по своему желанию. До недавнего времени нельзя было создать устройство с использованием микроконтроллера всего за несколько минут. Как вам, наверное, известно, микроконтроллер является программируемой платформой для управления различными системами с помощью относительно простых команд. С появлением платформы ***Arduino*** возможности разработчиков резко увеличились, и я надеюсь, что ***Arduino*** станет вашим любимым инструментом для исследования электронных схем, программирования, создания систем управления и многого другого.

В этой книге описано много разных проектов на основе плат ***Arduino***, от простого датчика движения до создания беспроводной системы управления с выходом в Интернет. Платформа ***Arduino*** будет отличным вариантом для проектирования микропроцессорных систем как для начинающих, так и для опытных разработчиков. Собрав своими руками конкретные устройства, рассмотренные в данной книге, вы сможете использовать полученный опыт, фрагменты программного кода, принципиальные схемы для создания собственных проектов на основе платформы ***Arduino*** или какой-либо другой.

Для кого эта книга

Эта книга предназначена для энтузиастов ***Arduino***, желающих разрабатывать электронные устройства и писать программы для них. Материал каждой последующей главы опирается на понятия и проекты, описанные ранее. Шаг за шагом вы сможете реализовать все более сложные системы. Если вы что-то забудете, ссылки напомнят, где вы впервые столкнулись с данным вопросом, так что вы сможете легко освежить память. Книга рассчитана на читателя, не обладающего большим опытом в электронике и программировании. По ходу изложения некоторые понятия объясняются более подробно, чтобы глубже разобраться в конкретных теоретических и практических вопросах.

О чем эта книга

В этой книге вы не найдете готовых рецептов. Если при проектировании вы желаете получить четкие инструкции без объяснения последовательности шагов, то эта книга не для вас. Настоящая книга — своего рода введение в мир электроники, информатики и практического применения платформы ***Arduino***, как средства для воплощения ваших идей на конкретных примерах. Здесь вы узнаете не только как собрать готовое устройство, но и как анализировать электрические схемы, читать технические описания, которые позволят вам выбрать подходящие детали при создании собственных проектов. При написании программного обеспечения в каждом примере предоставляется полный программный код, но сначала рассматриваются и объясняются несколько фрагментов, образующих итоговую программу. Такой подход помогает лучше уяснить определенные функции и особенности алгоритма программы. Книга научит принципам цифрового проектирования и специфическим для платформы ***Arduino*** понятиям программирования.

Я надеюсь, что, повторив действующие проекты из данной книги, вы не только научитесь разрабатывать устройства на основе ***Arduino***, но и получите навыки, необходимые для создания более сложных электронных систем и осуществления инженерной деятельности в других областях на различных платформах.

Что вам понадобится

В дополнение к конкретным компонентам для реализации проектов на основе ***Arduino***, перечисленным в начале каждой главы, есть несколько общих инструментов и материалов, которые вам пригодятся при прочтении книги. Это, в первую очередь, компьютер с операционной системой Mac OS X, Windows или Linux и установленной интегрированной средой разработки IDE для ***Arduino***. Рекомендую также приобрести следующие дополнительные инструменты для сборки и отладки устройств:

- паяльник и припой;
- мультиметр;
- набор небольших отверток;
- клеевой пистолет с нагревом.

Электронные ресурсы к книге

Поддерживаемый автором сайт http://www.exploringarduino.com специально предназначен для сопровождения этой книги. На нем вы можете загрузить исходный код примеров и проектов для каждой главы, а также видеоуроки и другие полезные материалы. Издательство Wiley также предоставляет электронные ресурсы для этой книги на сайте wiley.com. Получить доступ к исходным кодам программ можно на вкладке Download code на странице http://www.wiley.com/go/exploringarduino. Вы можете найти данную страницу по коду ISBN (для этой книги 978-1-118-54936-0). В начале каждой главы приведены ссылки на скачивание файлов с листингами программ данной главы. Файлы представлены в виде zip-архивов, после скачивания их необходимо разархивировать.

Дополнительный материал и поддержка

Во время изучения платформы ***Arduino*** у вас неизбежно возникнут вопросы, и возможно, вы столкнетесь с проблемами. За поддержкой всегда можно обратиться к сообществу пользователей ***Arduino***, которое легко найти в Интернете. Вот список полезных ресурсов для разработчиков ***Arduino***:

- официальный сайт проекта ***Arduino*** — http://www.arduino.cc/en/Reference/HomePage;
- моя серия уроков по ***Arduino*** — http://www.jeremyblum.com/category/arduino-tutorials;
- учебные материалы по ***Arduino*** от Adafruit — http://learn.adafruit.com/category/learn-arduino;
- учебные материалы по ***Arduino*** от SparkFun — http://learn.sparkfun.com/tutorials;
- официальный форум ***Arduino*** — http://www.arduino.ee/forum;
- сообщество ***Arduino*** на сайте element 14 — http://www.element14.com/community/groups/arduino.

Если вы исчерпали все эти ресурсы и до сих пор не можете решить свою проблему, свяжитесь со мной через Twitter (@sciguy14), может быть, я смогу помочь. Вы также можете связаться со мной напрямую через контактную страницу на моем сайте (http://www.jeremyblum.com/contact), но я не гарантирую быстрого ответа.

Что такое *Arduino*?

С помощью ***Arduino*** можно реализовать практически любой ваш замысел. Это может быть автоматическая система управления поливом, или веб-сервер, или даже автопилот для мультикоптера. Итак, ***Arduino*** — это платформа для разработки устройств на базе микроконтроллера, на простом и понятном языке программирования в интегрированной среде ***Arduino*** IDE. Добавив датчики, приводы, динамика, добавочные модули (платы расширения) и дополнительные микросхемы, вы можете использовать ***Arduino*** в качестве «мозга» для любой системы управления. Трудно даже перечислить все, на что способна платформа ***Arduino***, потому что возможности ограничены только вашим воображением. Эта книга послужит руководством, знакомящим вас с функциональностью ***Arduino*** путем выполнения большого количества проектов, которые дадут навыки, необходимые для реализации своих собственных разработок. Более подробно об особенностях ***Arduino*** мы расскажем в **главе 1**. Если вы интересуетесь внутренним устройством ***Arduino***, то вам повезло — это платформа с открытым исходным кодом, и все схемы и документация находятся в свободном доступе на сайте ***Arduino***.

О движении Open Source

Если вы новичок в мире открытого исходного кода (Open Source), то я рекомендую познакомиться с основными принципами этого сообщества. Здесь мы не будем вдаваться в подробности, а лишь немного коснемся идеологии данного движения, делающей работу с ***Arduino*** такой привлекательной. Получить более полное представление можно на веб-сайте Ассоциации открытого аппаратного обеспечения: http://www.oshwa.org/definition.

Как уже упоминалось, ***Arduino*** — платформа с открытым исходным кодом, поэтому все схемы и исходный код программ доступны для любого желающего. Это означает, что вы можете не только экспериментировать с ***Arduino***, но и использовать платформу и прилагаемые к ней программные библиотеки в своих проектах, производить и продавать клоны платы ***Arduino***.

Хотя книга ориентирована главным образом на фирменные изделия ***Arduino***, для повторения описанных далее устройств подойдут платы многочисленных сторонних разработчиков. Лицензия ***Arduino*** допускает также коммерческое применение конструкций на основе ***Arduino*** (без указания торговой марки ***Arduino***) в своих проектах. Итак, если вы создали на основе ***Arduino*** оригинальное

устройство и хотите превратить его в коммерческий продукт, вы можете сделать это. Например, вся электронная начинка в проекте MakerBot Replicator 3D-принтер выполнена на основе платформы ***Arduino*** Mega (http://www.thingiverse.eom/thing:16058).
Все примеры программ, которые я написал для этой книги (если не указано иное) на условиях лицензии GNU General Public License (GPL), можно использовать без ограничений для всего, что вы хотите.

Несколько советов читателю

Некоторые из вас, возможно, знакомы с моими популярными видеоуроками по изучению ***Arduino*** и основ электроники на канале YouTube (http://www.youtube.com/sciguy14)[1]. Я отсылаю читателя к ним для более полного раскрытия изложенных тем.
Если вам интересно узнать о том, какие замечательные вещи можно создать, творчески сочетая электронику, микроконтроллеры и информатику, рекомендую посмотреть мое портфолио (http://www.jeremyblum.com/portfolio) с самыми интересными проектами. Как и устройства на основе ***Arduino***, большинство моих разработок соответствуют открытой лицензии, которая позволяет легко дублировать созданное мною для ваших собственных нужд.
Мне будет интересно узнать, как вы примените знания и навыки, полученные при прочтении данной книги. Я призываю вас поделиться ими со мной и с остальным миром. Желаю удачи в ваших экспериментах с ***Arduino***!

Дополнительная информация издательства «БХВ-Петербург» к русскоязычному изданию книги

Для выполнения проектов, описанных в книге, издательство подготовило специальный набор, который включает в себя ***Arduino Uno***, плату прототипирования и необходимые электронные компоненты. Подробную информацию о наборе можно получить по адресу http://www.bhv.ru/books/193108.
Издательство «БХВ-Петербург» выражает благодарность компании «Амперка» за участие в подготовке русскоязычного издания книги. На интернет-ресурсах этой компании вы сможете найти:

- учебные материалы на русском от Амперки — http://wiki.amperka.ru.
- большую часть видеоуроков от автора этой книги Джереми Блума, переведен-ных на русский язык, — http://www.youtube.com/AmperkaRU или http://wiki.amperka.ru/видеоуроки:джереми-блюм

[1] Большая часть видеоуроков по Arduino переведена на русский язык. Локализованную версию можно найти на канале http://www.youtube.com/AniperkaRU. — Примеч. пер.

Часть 1. Общие сведения о платформе *Arduino*

Глава 1. Начало работы, переключаем светодиод из *Arduino*

Список деталей

Для повторения примеров главы вам потребуются следующие детали:

- плата ***Arduino Uno***;
- USB-кабель.

Электронные ресурсы к главе

На странице http://www.exploringarduino.com/content/chl можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/exploringarduino в разделе Downloads.

> ***ПРИМЕЧАНИЕ РОССИЙСКИХ ПЕРЕВОДЧИКОВ***
>
> *Плату* ***Arduino****, а также все электронные компоненты и инструменты можно приобрести в магазине компании "Амперка". Все необходимое для повторения опытов из этой книги можно найти в специальном разделе: http://amperka.ru/jeremy. Используйте кодовое слово* ***JEREMY*** *при покупке товаров из этого раздела для получения скидки. Кроме того, на сайте компании можно найти видеоуроки автора книги, переведенные на русский язык.*

1.1. Знакомство с платформой *Arduino*

Если у вас уже есть некоторое представление о платформе ***Arduino*** и его возможностях, можно начинать более подробное изучение ***Arduino***. В этой главе вы познакомитесь с аппаратными средствами, узнаете о среде и языке программирования, а также напишете первую программу. А при наличии деталей из списка, призеденно-го в начале главы, вы сможете увидеть результат работы программы — мигание светодиода!

> ***ПРИМЕЧАНИЕ***
>
> *Вводный видеоурок по платформе* ***Arduino*** *можно найти на странице www.jeremyblum.com/2011/01/02/arduino-tutorial-series-it-begins/*[1] *и на сайте издательства Wiley.*

При изучении платформы ***Arduino*** для повторения проектов из книги вам потребуются три главных компонента:

- основная плата ***Arduino***;
- платы расширения;
- интегрированная среда разработки ***Arduino*** — ***Arduino*** IDE.

В этой книге рассмотрены преимущественно фирменные платы ***Arduino***. Подойдут и выпускаемые в большом ассортименте клоны ***Arduino*** — платы, совместимые как с аппаратной, так и с программной частью ***Arduino***. Там, где это будет необходимо, вы найдете рекомендации по поводу пригодности тех или иных плат для различных устройств. Большинство проектов базируется на плате ***Arduino Uno***. Сначала мы рассмотрим общие функциональные возможности всех разновидностей плат ***Arduino***, а затем укажем особенности, присущие каждой плате. В результате вы сможете подобрать подходящую плату ***Arduino*** для каждого конкретного проекта.

1.2. Аппаратная часть

Все платы ***Arduino*** содержат основные компоненты, необходимые для программирования и совместной работы с другими схемами (Рисунок 1):

- микроконтроллер Atmel;
- USB-интерфейс для программирования и передачи данных;
- стабилизатор напряжения и выводы питания;
- контакты входов ввода-вывода; индикаторные светодиоды (Debug, Power, Rx, Tx);
- кнопку сброса;
- встроенный последовательный интерфейс программирования (ICSP).

[1] На русском: http://wiki.amperka.ru/видеоуроки:1-первые-шаги

1.3. Микроконтроллеры Atmel

Основной элемент платы ***Arduino***— микроконтроллер Atmel. На большинстве плат ***Arduino***, включая ***Arduino Uno***, установлен микроконтроллер ATmega. На плате ***Arduino Uno***, изображенной на Рисунок 1, вы видите микроконтроллер ATmega 328. Исключением является плата Due, укомплектованная микроконтроллером ARM Cortex.

Микроконтроллер исполняет весь скомпилированный код программы. Язык ***Arduino*** предоставляет доступ к периферийным устройствам микроконтроллера: аналого-цифровым преобразователям (ADC_S), цифровым портам ввода-вывода, коммуникационным шинам (включая I^2C и SPI) и последовательным интерфейсам. На плате все эти порты выведены на штырьковые контакты.

К тактовым контактам микроконтроллера ATmega подключен кварцевый резонатор на 16 МГц.

С помощью кнопки сброса выполнение вашей программы можно перезапустить.

Рисунок 1. Компоненты платы Arduino Uno

Большинство плат ***Arduino*** оснащено светодиодом отладки (Debug), подсоединенным к контакту 13, который позволит реализовать нашу первую программу (мигающий светодиод) без дополнительных компонентов.

1.4. Интерфейсы программирования

Обычно программы микроконтроллера ATmega, написанные на C или Ассемблере загружаются в микроконтроллер через интерфейс ICSP с помощью программатора (Рисунок 2). Возможно, самая важная особенность ***Arduino*** — непосредственное программирование через USB-порт, без дополнительного программатора. Эту функцию обеспечивает загрузчик ***Arduino***, записанный в микроконтроллер ATmega на заводе-изготовителе, и позволяющий загружать пользовательскую программу на плату ***Arduino*** по последовательному порту USART.

В случае ***Arduino Uno*** и Mega 2560 интерфейсом между кабелем USB и контактами USART на основном микроконтроллере служит дополнительный контроллер (ATmega 16U2 или 8U2 в зависимости от версии платы). На плате ***Arduino*** Leonardo установлен основной микроконтроллер ATmega 32U4, имеющий встроенный контроллер USB. В более старых платах ***Arduino*** функцию сопряжения между последовательным портом ATmega и интерфейсом USB выполняла специальная микросхема.

Загрузчик — это фрагмент программного кода, который записан в зарезервированное пространство памяти программы ***Arduino***. Микроконтроллеры AVR обычно программируются с помощью ICSP, который взаимодействует с микроконтроллером через последовательный периферийный интерфейс (SPI). Этот способ предполагает наличие программатора, например, STK500 или ISP MKII (см.

Рисунок 2: AVR программатор ISP MKII

Рисунок 2).
Сразу после включения платы ***Arduino*** запускается загрузчик, который работает в течение нескольких секунд. Если за это время загрузчик получает команду программирования от IDE по последовательному интерфейсу UART, то он загружает программу в свободную область памяти микроконтроллера. Если такая команда не поступает, запускается последняя программа, находящаяся в памяти ***Arduino***.
При подаче команды загрузки от IDE ***Arduino*** вспомогательный контроллер (ATmega 16U2 или 8U2 в случае ***Arduino Uno***) сбрасывает основной микроконтроллер, подготавливая его к загрузке. Затем внешний компьютер начинает отправлять код программы, который микроконтроллер получает через соединение UART.
Загрузчики занимают в памяти довольно много места, потому что они реализуют простое программирование через USB без внешних аппаратных средств. Однако у них есть два основных недостатка:

- они занимают место в памяти (приблизительно 2 Кбайт), которое могло бы пригодиться при написании программ;
- при наличии загрузчика выполнение вашей программы всегда будет задерживаться на несколько секунд при начальной загрузке, поскольку загрузчик обрабатывает запрос на программирование.

Если у вас есть программатор (или другая плата ***Arduino***, запрограммированная как программатор), то можно удалить загрузчик из своего контроллера ATmega и программировать его с помощью внешнего программатора.

1.5. Цифровые и аналоговые контакты ввода-вывода

У контроллеров ***Arduino*** к большинству контактов ввода-вывода можно подключить внешние схемы. Все контакты могут служить цифровыми входами и выходами. Часть контактов ***Arduino*** могут также действовать в качестве аналоговых входов. Многие из контактов работают в режиме мультиплексирования и выполняют дополнительные функции: различные коммуникационные интерфейсы, последовательные интерфейсы, широтно-импульсные модуляторы и внешние прерывания.

1.6. Источники питания

Для большинства проектов достаточно 5-вольтового питания, получаемого по кабелю USB. Однако, при необходимости разработки автономного устройства, схема способна работать от внешнего источника от 6 до 20 В (рекомендуется напряжение 7-12 В). Внешнее питание может подаваться через разъем DC или на контакт Уш.

У ***Arduino*** есть встроенные стабилизаторы на 5 и 3,3 В:

- напряжение 5В используется для всех логических элементов на плате, уровень на цифровых контактах ввода-вывода находится в пределах 0-5В;
- напряжение 3,3В выведено на отдельный контакт для подключения внешних устройств.

1.7. Платы *Arduino*

Мы не будем рассматривать все существующие платы ***Arduino***, т. к. их очень много и постоянно выпускаются все новые с различными функциями. Кратко опишем лишь некоторые из фирменных плат ***Arduino***.

Arduino Uno (Рисунок 3) — основная плата линейки ***Arduino***, она будет использоваться в большинстве примеров книги. Плата укомплектована микроконтроллером ATmega 328 и микросхемой 16U2 преобразователя USB. Микроконтроллер ATmega 328 может быть выполнен в исполнении DIP или SMD.

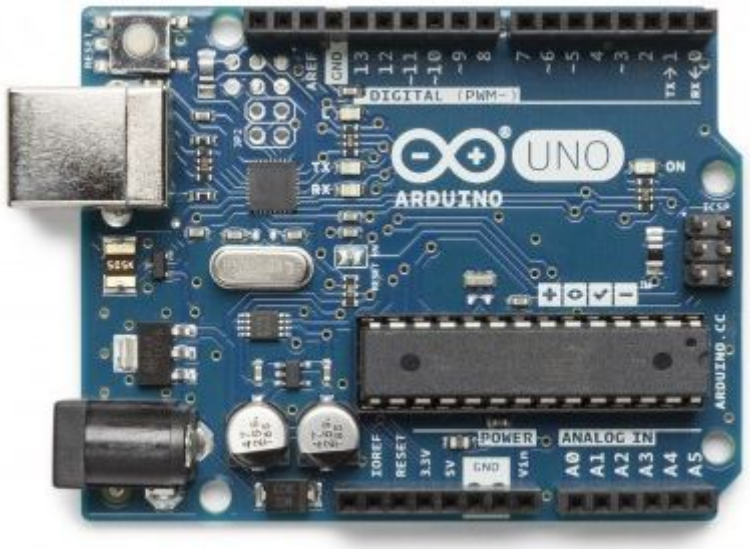

Рисунок 3. Плата Arduino Uno

На плате Leonardo (Рисунок 4) установлен контроллер 32U4 со встроенным интерфейсом USB. Это уменьшает стоимость изделия и дает возможность использовать плату в качестве USB-устройства, например, как эмулятор джойстика или клавиатуры. Вы узнаете, как работать с этими функциями, в главе 6.

На плате ***Arduino*** Mega 2560 (Рисунок 5) установлен контроллер ATmega 2560, имеющий 54 цифровых входа-выхода, что позволяет подключать еще больше устройств. У ***Arduino*** Mega 2560 увеличено число аналоговых входов и последовательных портов (четыре против одного у ***Arduino Uno***).

В отличие от остальных плат ***Arduino***, использующих 8-разрядные контроллеры AVR, плата Due (Рисунок 6) создана на базе 32-разрядного процессора Atmel SAM3X8E ARM Cortex-M3 с тактовой частотой 84 МГц. Отличительные особенности платы: повышенная точность аналого-цифрового преобразователя, настраиваемая частота сигнала ШИМ, отдельные выводы цифроаналогового преобразователя, наличие встроенного последовательного порта.

Конструкция миниатюрной платы ***Arduino*** Nano (Рисунок 7) такова, что ее можно установить в панельку для микросхем.

Плата Mega ADK (Рисунок 8) очень похожа на ***Arduino*** Mega 2560, но у Mega ADK есть дополнительная функциональность интерфейса USB, позволяющая ему соединяться с телефоном на базе Android.

Рисунок 4. Плата Arduino Leonardo

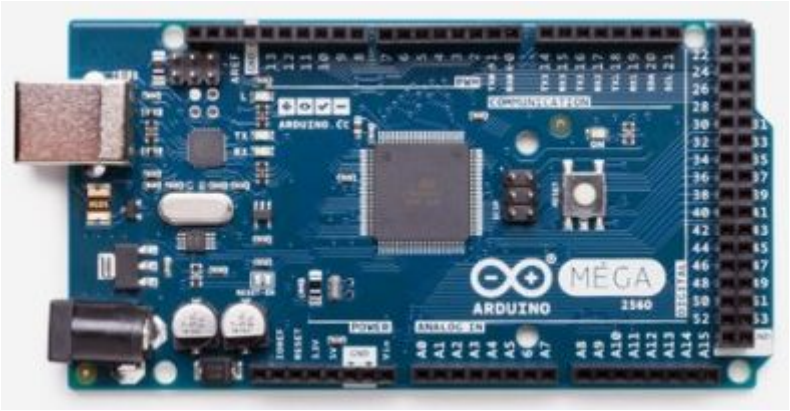

Рисунок 5. Плата Arduino Mega 2560

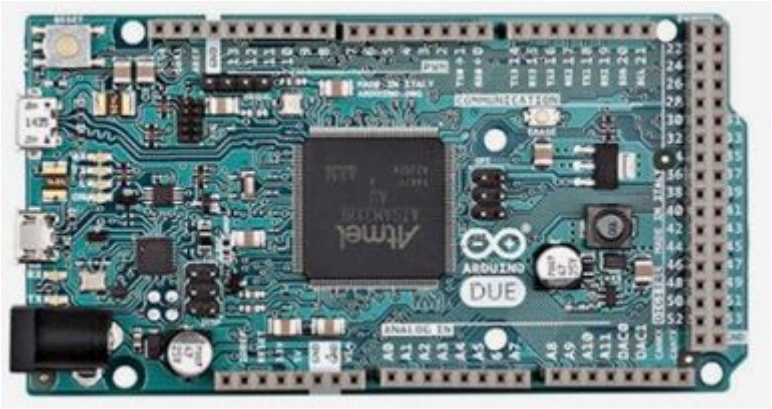

Рисунок 6. Плата Arduino Due

Рисунок 7. Плата Arduino Nano

Рисунок 8. Плата Arduino Mega ADK

Уникальность платы ***Arduino*** LilyPad (Рисунок 9) в том, что она разработана как часть одежды. Ее можно вшить в ткань вместе с датчиками, светодиодами и т. п. Для программирования платы необходим кабель FTDI.

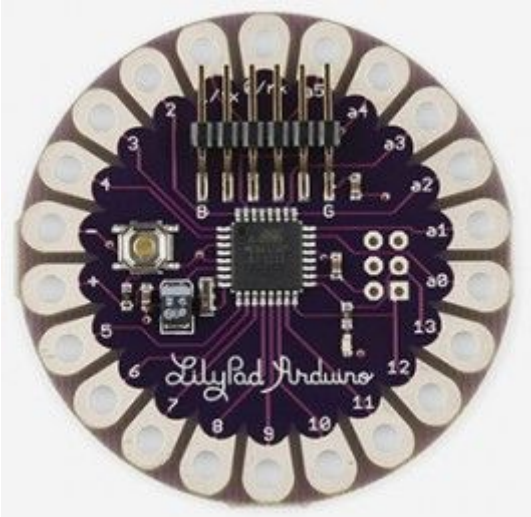

Рисунок 9. Плата Arduino LilyPad

Как уже упоминалось во введении, ***Arduino*** — это открытая платформа. Поэтому в продаже можно найти десятки ***Arduino***-совместимых устройств, которые будут работать с IDE ***Arduino*** и со всеми проектами, описанными в этой книге. Многие используют популярные платы Seeeduino, Adafruit 32U4, SparkFun Pro и миниплаты ***Arduino***. Много сторонних плат разработано для конкретных приложений с дополнительной функциональностью, уже встроенной в плату. Например, ArduPilot— плата для автономного управления квадрокоптером (Рисунок 10). ***Arduino***-совместимые платы также применяются в качестве контроллера MakerBot и 3D-принтера.

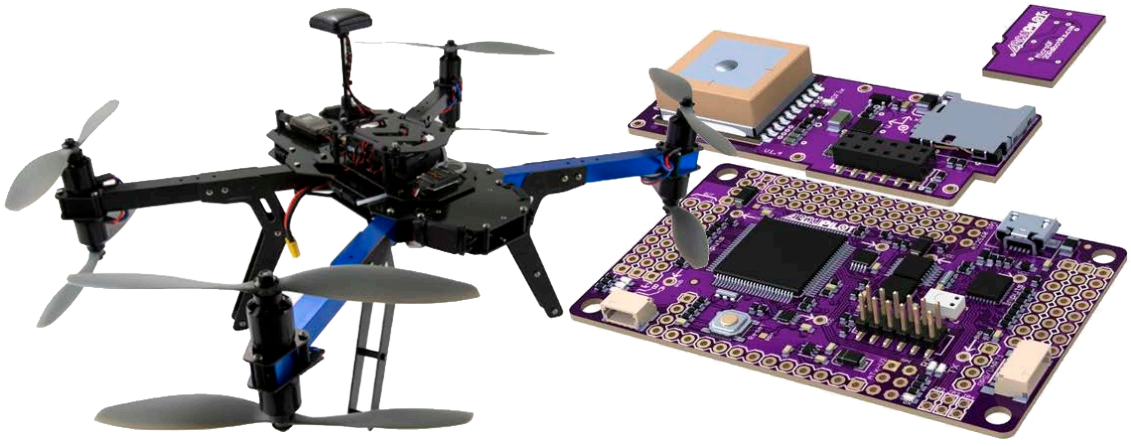

Рисунок 10. Квадрокоптер и контроллер ArduPilot Mega

1.8. Запускаем первую программу

Теперь, когда вы познакомились с аппаратными средствами ***Arduino***, можно установить программное обеспечение и запустить первую программу. Приступим к загрузке программного обеспечения на компьютер.

1.8.1. Загрузка и установка *Arduino* IDE

Зайдите на официальный сайт ***Arduino*** https://www.arduino.cc/ и загрузите последнюю версию ***Arduino*** IDE со страницы «Software» (Рисунок 11).

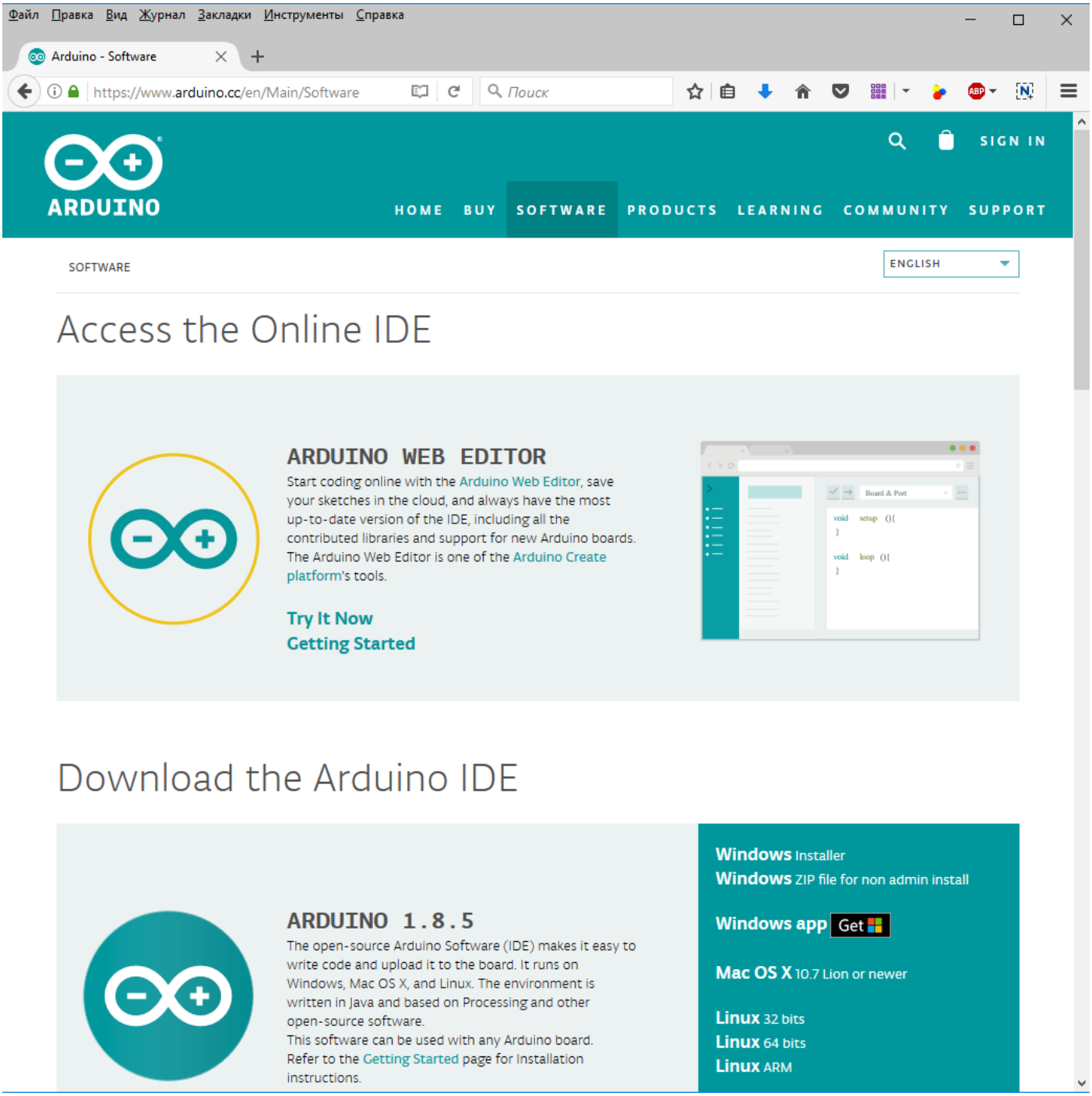

Рисунок 11. Страница загрузки сайта Arduino.cc

После завершения загрузки разархивируйте загруженный файл. В папке вы найдете ***Arduino*** IDE. Новые версии IDE для Windows доступны в форме установщика, который инсталлирует программную среду ***Arduino*** автоматически.

1.8.2. Запуск IDE и подключение к *Arduino*

Подключите ***Arduino*** к компьютеру с помощью кабеля USB, как изображено на Рисунок 12. Компьютеры с операционной системой Mac и Linux установят драйверы автоматически.

Если на вашем компьютере установлена операционная система OS X, то при первом подключении появится сообщение о том, что было добавлено новое сетевое устройство. Нажмите кнопку Network Preferences (Системные настройки -> Сеть). В появившемся окне нажмите кнопку Apply (Подключить). Даже если будет выдано предупреждение Not Configured, устройство можно использовать. После этого выйдите из меню System Preferences (Системные настройки).

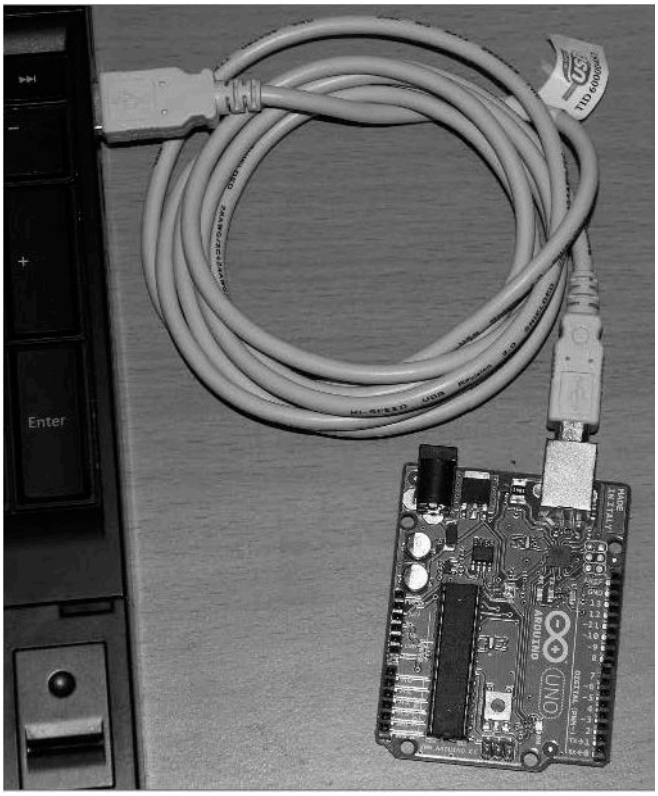

Рисунок 12. Подключение платы к компьютеру посредством кабеля USB

При подключении ***Arduino*** к компьютеру с операционной системой Windows, возможно, придется установить драйверы. При инсталляции ***Arduino*** IDE с помощью установщика для Windows драйверы загружаются автоматически. Если вы устанавливали IDE из zip-файла, то для инсталляции драйверов необходимо выполнить следующие шаги:

- дождитесь сообщения о неудачной попытке автоматической установки драйвера;
- нажмите кнопку Пуск и откройте Панель управления;
- перейдите на вкладку Система и безопасность (System and Security). Затем выберите раздел Система. Когда откроется окно Система, выберите опцию Диспетчер устройств (Device Manager);
- обратите внимание на порты (COM и LPT). Вы увидите открытый порт, в названии которого присутствует ***Arduino***;
- щелкните правой кнопкой мыши и выберите опцию «Обновить драйвер» (Update Driver Software);
- щелкните мышью пункт Browse my computer for Driver software;
- для завершения установки найдите и выберите файл драйвера, расположенный в папке Drivers программного обеспечения для ***Arduino*** (но не в подкаталоге FTDI USB Drivers);
- в результате драйвер для Windows будет установлен.

Теперь запустите ***Arduino*** IDE. Все готово для загрузки первой программы на плату ***Arduino***. Чтобы убедиться в этом, запустим программу Blink, которая будет мигать встроенным светодиодом. На большинстве плат ***Arduino*** есть светодиод, подключенный к цифровому контакту 13. Выполняем последовательность команд File -> Examples -> Basic и выбираем программу Blink. Откроется новое окно с кодом этой программы. Загрузим ее в плату ***Arduino*** в качестве примера, а затем проанализируем, чтобы понять, как писать собственные программы.

Прежде чем загружать программу в плату ***Arduino***, необходимо указать тип платы и номер

последовательного порта. Находим в меню опцию Tools-> Board (Сервис-> Плата) и выбираем из списка плату ***Arduino***. В книге мы используем ***Arduino Uno***, если у вас другая плата, выберите ее наименование.
Затем необходимо указать порт, к которому подсоединена плата. Переходим к опции Tools -> Serial Port (Сервис -> Последовательный порт) и выбираем последовательный порт. На компьютерах с Windows это будет COM*, где * — некоторое число, соответствующее номеру последовательного порта. На компьютерах с Linux и Mac порт обозначен как dev/tty.usbmodem* или /dev/tty.usbserial*, где * — строка алфавитно-цифровых символов.

> ***СОВЕТ***
> *Если в списке присутствует несколько последовательных портов, и вы не можете определить, к какому из них подключена плата* ***Arduino****, отключите плату, чтобы увидеть, какой порт исчезнет из меню, это и есть порт подсоединения платы* ***Arduino****.*

Теперь можно загрузить первую программу. Нажимаем кнопку Upload (Загрузить), расположенную в левом верхнем углу ***Arduino*** IDE. В строке состояния, находящейся внизу, отображается процесс компиляции и загрузки программы. После загрузки программы светодиод, подключенный к выводу 13 ***Arduino***, должен мигать оранжевым цветом с частотой один раз в секунду. Поздравляем! Ваша первая программа работает успешно.

1.8.3. Анализируем программу Blink

Подробно рассмотрим текст программы Blink (Рисунок 13), чтобы понять базовую структуру программ, написанных для ***Arduino***.

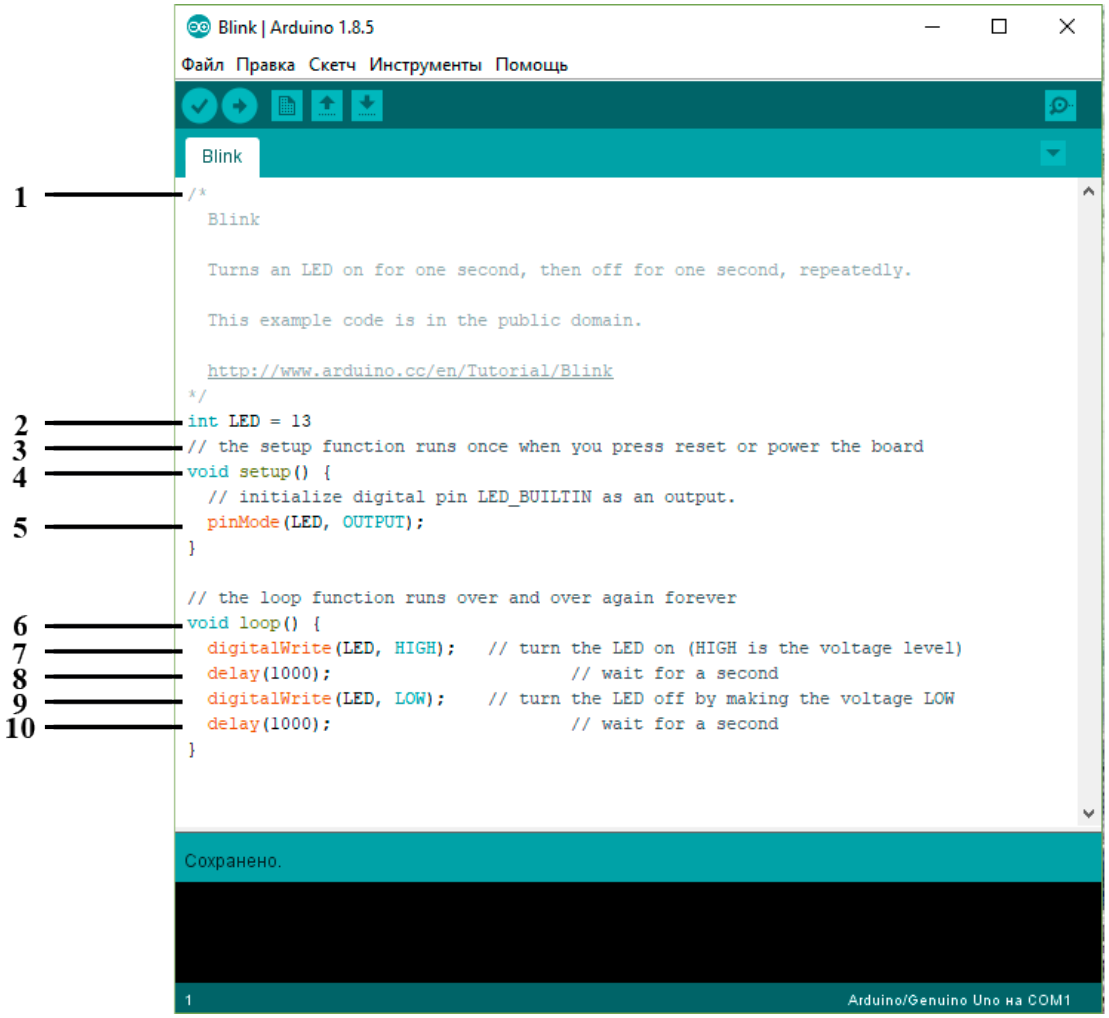

Рисунок 13. Структура программы Blink

Цифрами на рис. 13 обозначено следующее:

1. многострочный комментарий. Комментарии важны для пояснения кода программы. Все, что написано между этими символами, не будет обрабатываться компилятором. Многострочные

комментарии начинаются с /* и заканчиваются */. Многострочные комментарии удобны, когда текст пояснения большой, например, описание программы.

2. код объявления переменной. Переменная — это ячейка памяти, содержащая информацию. Существуют переменные различных типов. В нашем примере указана переменная типа int, что означает целое число. Целочисленной переменной `LED` присвоено значение 13 — номер цифрового контакта, к которому подключен светодиод на плате ***Arduino***. Всюду в остальной части программы можно использовать переменную `LED`, когда мы хотим управлять контактом 13. Переменные в данном случае удобны, потому что при необходимости поменять контакт ввода-вывода достаточно изменить только одну строку, а остальная часть кода не изменится.
3. однострочный комментарий. Если поместить // на любую строку, компилятор проигнорирует весь текст строки после этого символа. Однострочный комментарий обычно поясняет определенную строку кода.
4. функция `setup()`, одна из двух функций, которые должны быть включены в каждую программу ***Arduino***. Функция — это фрагмент кода, выполняющий определенную задачу. Код в теле функции setup о выполняется один раз в начале программы. Это полезно для установки начальных параметров настройки, назначения режимов портов ввода-вывода, инициализации коммуникационных интерфейсов и т. д.
5. цифровые контакты ***Arduino*** могут быть запрограммированы на ввод или вывод. Сконфигурировать их направление позволяет команда `pinMode()`, имеющая два параметра, указанных в круглых скобках. Первый параметр `pinMode` определяет номер контакта. Поскольку переменная `LED` уже назначена ранее в программе, конфигурация задается для контакта 13. Второй параметр устанавливает направление контакта: `input` (вход) или `output` (выход). По умолчанию все контакты настроены на ввод. Чтобы сконфигурировать их на вывод, следует явно указать значение этого параметра `output`. Поскольку нам нужно управлять светодиодом, контакт 13 должен быть выходом. Настройка конфигурации контакта сохраняется до тех пор, пока вы не измените его назначение на ввод.
6. вторая обязательная функция во всех программах ***Arduino***— `loop()`. Это оператор цикла.
7. функция `digitalWrite()` устанавливает состояние выходного контакта: 5 или 0 В. Если светодиод подсоединен к контакту через резистор, то установка значения логической «1» позволит зажечь светодиод (вы узнаете больше об этом в следующей главе). Первый параметр функции `digitalWrite()` — номер контакта, которым требуется управлять. Второй параметр— значение, которое нужно задать: `HIGH` (5 В) или `LOW` (0 В). Контакт остается в этом состоянии, пока не будет изменен следующей командой `digitalWrite()`.
8. функция `delay()` имеет один аргумент — время задержки выполнения программы в миллисекундах. При вызове `delay()` ***Arduino*** останавливает выполнение программы на определенный интервал времени. В нашем примере задержка равна 1000 мс (1 с). Это приводит к свечению светодиода в течение одной секунды до выполнения следующей команды.
9. здесь вызвана функция `digitalWrite()`, чтобы выключить светодиод, устанавливая состояние контакта в `LOW`.
10. снова делаем задержку на одну секунду, чтобы светодиод был погашен перед повторением цикла.

Вот и все. Не расстраивайтесь, если вы еще не полностью понимаете код программы. В следующих главах мы рассмотрим больше примеров, процесс выполнения программы станет понятнее, и вы сможете написать собственный код.

Резюме

В этой главе вы узнали о следующем:

- Из каких компонентов состоит плата ***Arduino***.
- Как загрузчик ***Arduino*** позволяет запрограммировать плату ***Arduino*** через интерфейс USB.
- Каковы различия между основными платами ***Arduino***.
- Как установить ***Arduino*** IDE и соединить плату ***Arduino*** с компьютером.
- Как загрузить и выполнить первую программу.

Глава 2. Цифровые контакты ввода-вывода, широтно-импульсная модуляция

Список деталей

Для повторения примеров главы понадобятся следующие детали:

- плата ***Arduino Uno***;
- макетная плата;
- перемычки;
- 1 резистор номиналом 10 кОм;
- 3 резистора номиналом 220 Ом;
- кабель USB;
- кнопка;
- одноцветный светодиод 05 мм;
- RGB-светодиод 05 мм с общим катодом.

Электронные ресурсы к главе

На странице http://www.exploringarduino.com/content/ch2 можно загрузить код программ, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118549368.html в разделе Downloads.

Что вы узнаете в этой главе

Проект с мигающим светодиодом, рассмотренный в предыдущей главе, скорее игрушка, чем полезное устройство. Действительно привлекательной для разработчиков делает платформу ***Arduino*** наличие контактов ввода-вывода. К плате можно подключить, например, геркон, и при открытии двери проигрывать какую-либо мелодию или создать электронный сейф, или несложный музыкальный инструмент.

В этой главе вы приступите к разработке новых проектов: познакомитесь с возможностями цифровых входов ***Arduino***, узнаете о подтягивающих (pull-up) и стягивающих (pull-down) резисторах и научитесь управлять цифровыми выходами.

У большинства плат ***Arduino*** нет аналоговых выходов, но их можно эмулировать с помощью широтно-импульсной модуляции (ШИМ). Далее мы расскажем, как сформировать ШИМ-сигнал. Прочитав главу, вы сможете создать ночник на RGB-светодиоде.

> ***ПРИМЕЧАНИЕ***
>
> *Видеоурок данной главы можно посмотреть на интернет-странице http://www.jeremyblum.com/2011/01/10/arduino-tutorial-2-now-with-more-blinky-things/[1].*
>
> *Если вы захотите узнать больше о некоторых аспектах электротехники, затронутых в этой главе, то посмотрите видеофильм, расположенный на интернет-странице http://www.jeremyblum.com/2011/01/17/electrical-engineering-basics-in-arduino-tutorial-3/[2].*

2.1. Цифровые контакты

В Часть 1. Общие сведения о платформе *Arduino* вы узнали, как заставить мигать светодиод, подключенный к цифровому контакту. Продолжим изучать возможности цифровых выходов ***Arduino*** и рассмотрим следующие темы:

- конфигурирование назначения цифровых выводов;
- подключение внешних компонентов;
- новые концепции программирования циклов и констант;
- различие между цифровыми и аналоговыми выходами;
- широтно-импульсная модуляция (ШИМ).

2.2. Подключение внешнего светодиода

Мигающий светодиод из предыдущего примера был встроен в плату ***Arduino***. Теперь настало время выйти за пределы платы и соединить ее контакт 9 с внешним светодиодом. Этот простой пример поможет вам понять, как собирать более сложные внешние цепи, описанные в следующих главах.

[1] На русском: **http://wiki.amperka.ru/видеоуроки:2-кнопки-pwm-функции/**

[2] На русском: **http://wiki.amperka.ru/видеоуроки:3-основы-схемотехники/**

Более того, контакт 9 ***Arduino*** позволяет формировать сигнал широтно-импульсной модуляции, что мы используем далее в этой главе.

2.2.1. Работа с макетной платой

Рассмотрим, что такое макетная плата и как эффективно использовать ее для проектов из этой книги. Макетная плата — удобный инструмент для экспериментов, позволяющий легко собирать простые схемы без изготовления печатных плат и пайки. С двух сторон по всей длине макетной платы расположены красные и синие отверстия. Все красные отверстия соединены между собой и служат, как правило, для подачи питания. Для большинства проектов из этой книги это +5 В. Все синие отверстия тоже электрически соединены друг с другом и играют роль шины заземления. Каждые пять отверстий, расположенных вертикальными рядами, также соединены друг с другом. Посередине есть свободное место для удобства установки компонентов на макетной плате. Электрические соединения отверстий показаны на **Ошибка! Источник ссылки не найден.** орнажевыми линиями.

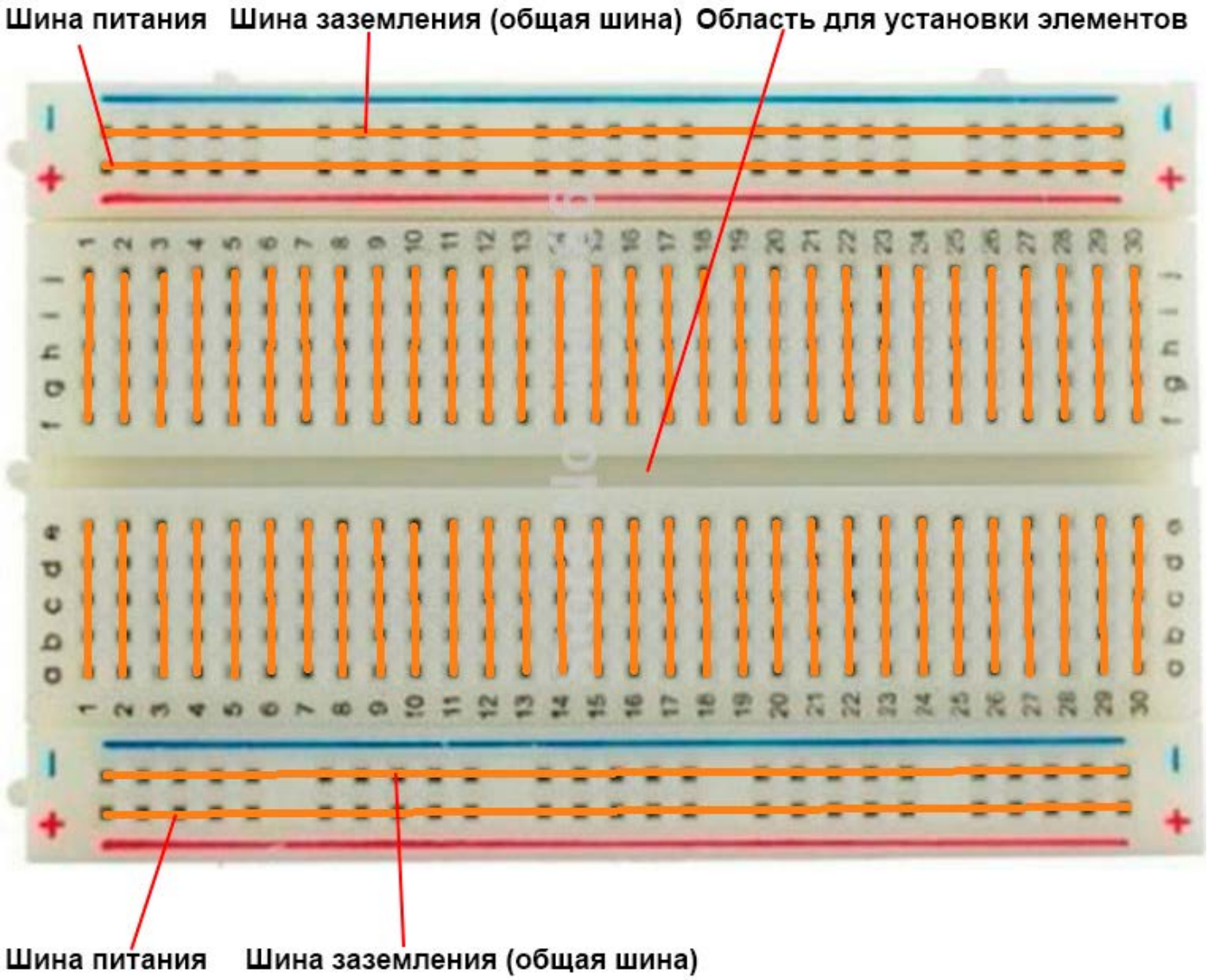

Рисунок 14. Электрические соединения макетной платы

2.3. Подсоединение светодиодов

Светодиоды почти наверняка будут одними из наиболее часто используемых деталей в проектах из данной книги. Подключая светодиоды, необходимо соблюдать правильную полярность. Положительный вывод светодиода называется анодом, отрицательный— катодом. Определить назначение контактов светодиода можно визуально: вывод катода короче, чем анода.

Ток через светодиод течет только в одном направлении: от анода к катоду. Поскольку ток протекает

от положительного полюса к отрицательному, анод светодиода следует подключить к источнику тока (цифровой выход +5 В), а катод — к земле. Резистор может быть подключен последовательно с любым из выводов светодиода. Полярность подключения для резисторов не важна.
Подключать светодиод к контакту 9 ***Arduino*** нужно последовательно с резистором, который выступает в качестве ограничителя тока. Чем больше сопротивление резистора, тем сильнее он ограничивает ток. В этом примере мы применим резистор номиналом 220 Ом. Монтажная схема изображена на Рисунок 15.

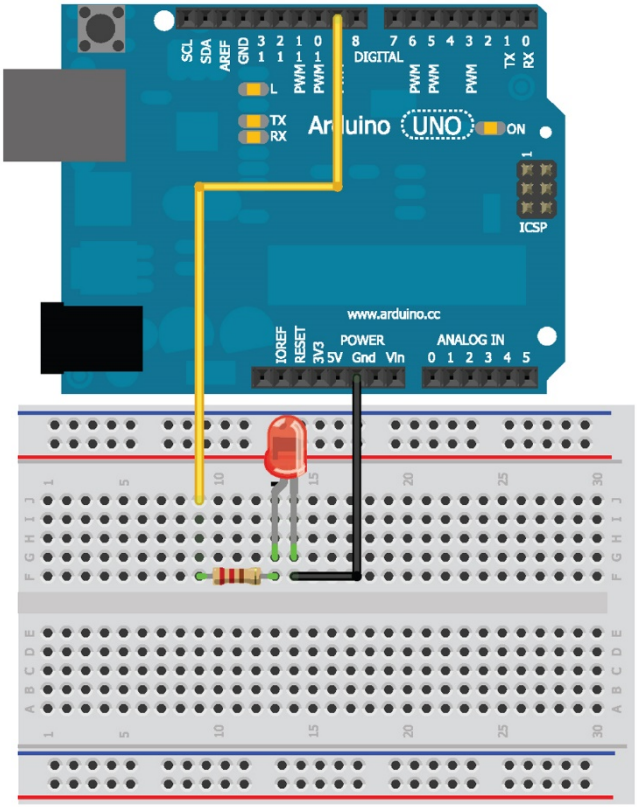

Рисунок 15. Подключение светодиода к плате Arduino Uno

2.3.1. Закон Ома и формула для расчета мощности

Самая главная формула для любого инженера-электрика — это закон Ома, который определяет соотношение между напряжением (измеряется в вольтах), током (измеряется в амперах) и сопротивлением (измеряется в Омах) в цепи. Схема представляет собой замкнутый контур с источником электрической энергии (например, батареей 9 В) и нагрузкой (чем-то, что расходует энергию, как светодиод). Прежде всего, важно понять физический смысл каждого термина:

- напряжение представляет собой разность электрических потенциалов между двумя точками;
- ток течет от точки с более высокой потенциальной энергией, чтобы снизить потенциальную энергию. Пользуясь аналогией, электрический ток можно представить, как поток воды, а напряжение — как высоту перепада. Вода (или ток) всегда течет из точки с большей высотой (более высокое напряжение) к точке с меньшей высотой (или более низкому напряжению). Ток, как вода в реке, всегда будет идти по пути наименьшего сопротивления в цепи;
- по аналогии сопротивление является отверстием для протекания тока. Когда вода (ток) течет через узкую трубу, за одинаковое количество времени проходит меньшее количество, чем через широкую трубу. Узкая труба эквивалентна большему сопротивлению, потому что вода будет течь медленнее. Широкая труба эквивалентна малому сопротивлению, потому что вода (ток) может течь быстрее.

Закон Ома определяется следующим образом:

$$U=I\times R,$$

где ***U*** — напряжение в вольтах; ***I*** — ток в амперах; ***R*** — сопротивление в омах.
В электрической цепи каждый компонент обладает некоторым сопротивлением, что снижает напряжение. Закон Ома очень удобен для подбора значения резистора, подключаемого последовательно со светодиодом. Светодиоды характеризуются определенной величиной падения

напряжения и заданным значением рабочего тока. Чем больше ток через светодиод (не превышая максимально допустимого), тем ярче он светится. Для наиболее распространенных светодиодов максимальный ток равен 20 мА. Типовое значение падения напряжения для светодиода составляет около 2В.
Рассмотрим схему, изображенную на Рисунок 16, и применим закон Ома для подбора резистора R1.

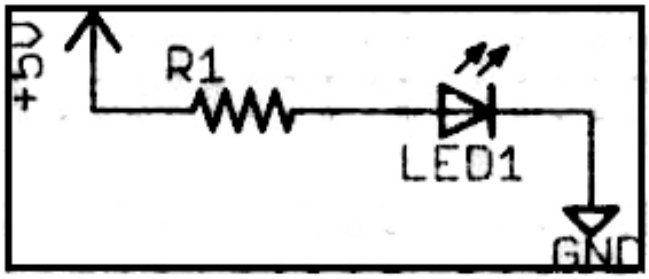

Рисунок 16. Схема включения светодиода

Предположим, что LED1 — стандартный светодиод с прямым током 20 мА и падением напряжения 2 В. Напряжение питания 5В должно перераспределиться между светодиодом и резистором. Поскольку доля светодиода составляет 2В, оставшиеся 3В должны быть приложены к резистору. Зная максимальное значение прямого тока через светодиод (20 мА), можно найти номинал резистора:

$$R = U/I = 3/0{,}02 = 150\ Ом.$$

Таким образом, при сопротивлении резистора 150 Ом через него и светодиод протекает ток 20 мА. По мере увеличения сопротивления ток будет уменьшаться. Резистор 220 Ом обеспечивает достаточную яркость свечения светодиода, к тому же этот номинал очень распространен.
Еще одно важное соотношение — формула для расчета мощности, которая показывает, сколько ватт рассеивается на каждом компоненте. Увеличение мощности рассеивания связано с ростом тепловыделения прибора. Для каждого компонента, как правило, задается максимально допустимая мощность. Максимальная мощность резистора в нашем примере равна 0,125 Вт. Формула для расчета мощности выглядит следующим образом:

$$P = U \times I,$$

где P — мощность, Вт; U — напряжение, В; I— сила тока, А.
Для резистора из схемы на Рисунок 16 при падении напряжения 3 В и силе тока 20 мА мощность равна

$$P = 3 \times 0{,}02 = 0{,}06\ Вт.$$

Поскольку 60 мВт <0,125 Вт= 125 мВт, следовательно, данный резистор не перегреется.

2.4. Программирование цифровых выводов

По умолчанию все внешние контакты ***Arduino*** сконфигурированы как входы. Если необходимо использовать контакт ***Arduino*** как выход, нужно его переконфи^риро-вать, подав соответствующую команду микроконтроллеру.
Каждая программа для ***Arduino*** должна включать две обязательные функции:
`setup()` и **`loop()`**.
В части 1 уже упоминалось, что функция **`setup()`** запускается один раз в начале программы, а **`loop()`** работает как цикл. Поскольку каждый контакт обычно конфигурируется в программе один раз, логично делать это в теле функции **`setup()`**.
Для начала напишем простую программу, которая при запуске сконфигурирует контакт 9 как выход. В программе будут еще две функции: **`pinMode()`** — для конфигурации контакта и `digitalWrite()` — для установки значения `HIGH` (5 В) на этом контакте (Листинг 1).

```
const int LED=9;                // Константа - номер контакта светодиода
void setup()
{
pinMode (LED, OUTPUT);          // Конфигурируем контакт светодиода как выход
digitalWrite(LED, HIGH);        // Устанавливаем значение HIGH на выходе
}
```

```
void loop() {                    // В цикле ничего не выполняем
}
```
Листинг 1. Пример конфигурации контакта

Соберите схему, как показано на Рисунок 15, и загрузите код листинга 2.1 в плату ***Arduino***. Обратите внимание, что в этой программе я использовал оператор инициализации константы перед определением значения контакта ***Arduino***. Обычно для хранения значений, которые могут изменяться во время выполнения программы, предназначены переменные.
Поставив оператор `const` до объявления переменной, вы говорите компилятору, что это переменная «только для чтения» и она не будет изменяться во время выполнения программы. Всем экземплярам переменной `LED` в программе будет присвоено значение 9. В виде констант рекомендуется определять значения, которые не будут меняться при выполнении программы. Далее в некоторых примерах этой главы встретится иная ситуация: значения, которые могут изменяться при выполнении программы.
При объявлении любой переменной необходимо указать ее тип. В нашем случае это целое число (номера контактов всегда будут целыми числами).
Теперь попробуйте изменить программу из главы 7, добавив функцию `digitaiwrite ()` и введя задержку в цикле `loop()`. Экспериментируя со значениями задержки, можно создавать различные эффекты мигания.

2.5. Использование цикла

На практике часто необходимо циклически изменять значения переменных для выполнения заданного алгоритма. В предыдущем примере можно реализовать цикл, чтобы увидеть, как влияют на частоту мигания разные значения задержки. Вы можете реализовать разные скорости мигания, задавая с помощью переменной цикла различные значения задержки. Пример иллюстрирует код из Листинг 2.

```
const int LED=9;                 // Константа номера контакта светодиода
void setup()
{
pinMode (LED, OUTPUT);           // Конфигурируем контакт светодиода как выход
}
void loop()
{
for (int i=100; i<=1000; i=i+100)
{
digitalWrite(LED, HIGH);
delay(i);
digitalWrite(LED, LOW);
delay(i);
}
}
```
Листинг 2. Изменение частоты мигания светодиода

Скомпилируйте код Листинг 2, загрузите его на свою плату ***Arduino*** и посмотрите, что происходит. Теперь разберемся, как это работает.
Оператор `for` всегда содержит три выражения, разделенные точкой с запятой:

- первое выражение присваивает начальное значение переменной-счетчику цикла. В нашем примере переменная `i` получает начальное значение 100;
- второе выражение указывает, когда цикл должен остановиться. Операторы в теле цикла будут выполняться снова и снова, пока условие истинно. Запись `<=` означает меньше или равно. Таким образом, этот цикл будет выполняться тех пор, пока переменная `i` меньше или равна 1000;
- последнее выражение указывает, что должно произойти с переменной `i` каждый раз после выполнения операторов тела цикла. В нашем примере, значение счетчика цикла увеличивается на 100.

Чтобы лучше понять работу оператора `for`, подробно рассмотрим, что происходит за два прохода цикла:

1. Значение переменной `i` равно 100, 100 меньше или равно 1000, значит выполнять код в теле цикла.
2. На контакте 9 установлено значение `HIGH`, светодиод горит 100 мс (текущее значение `i`).
3. На контакт 9 подано значение `LOW`, светодиод потушен 100 мс (текущее значение `i`).
4. В конце цикла значение переменной `i` увеличивается на 100, теперь `i` рауно 200.
5. 200 меньше или равно 1000, цикл повторяется снова.
6. На контакте 9 установлено значение `HIGH`, светодиод горит 200 мс (текущее значение `i`).
7. На контакт 9 подано значение `LOW`, светодиод потушен 200 мс (текущее значение `i`).
8. В конце цикла значение переменной `i` увеличивается на 100, теперь `i` равно 300.
9. Этот процесс повторяется, пока `i` не превосходит 1000 и затем `i` снова принимает значение 100 и все повторяется заново.

Итак, вы разобрались с работой цифровых контактов платы ***Arduino***. Далее мы расскажем, как с помощью ШИМ сформировать аналоговые сигналы на цифровых контактах платы ***Arduino***.

2.6. Широтно-импульсная модуляция с помощью analogWrite()

Вы освоили контроль над цифровыми контактами ***Arduino***. Они очень удобны для переключения светодиодов, управления реле и двигателями постоянного тока. Но что делать, если необходимо вывести напряжение, отличное от 0 и 5 В. С помощью контактов одной только платы ***Arduino Uno*** это невозможно. Придется задействовать цифроаналоговый преобразователь или взять плату ***Arduino*** Due или добавить внешнюю микросхему ЦАП.

Тем не менее, можно сымитировать генерацию аналоговых значений на цифровых контактах с помощью широтно-импульсной модуляции (ШИМ). Для некоторых контактов ***Arduino*** сформировать ШИМ-сигнал можно командой *`analogWrite()`*. Контакты, которые могут выдавать ШИМ-сигнал на определенные периферийные устройства, помечены символом ~ на плате ***Arduino***. На ***Arduino Uno*** контакты 3, 5, 6, 9, 10, 11 поддерживают выдачу ШИМ-сигнала. При наличии ***Arduino Uno*** проверить команду `analogWrite` о можно с помощью схемы, изображенной на рис. 2.1. Если уменьшить напряжение на контакте 9 ***Arduino***, яркость свечения светодиода должна стать меньше, потому что снизится ток, текущий через него. Этого эффекта можно добиться с помощью ШИМ и команды *`analogWrite()`*.

Функция `analogWrite ()` имеет два аргумента: номер контакта и 8-разрядное значение в диапазоне от 0 до 255, устанавливаемое на этом контакте.

В Листинг 3 приведен код программы генерации ШИМ-сигнала на контакте 9 для плавного управления яркостью светодиода.

```
const int LED=9;   //Константа номера контакта светодиода
void setup()
{
pinMode (LED, OUTPUT); //Конфигурируем контакт светодиода
                       //как выход
}
void loop()
{
for (int i=0; i<256; i++)
{
analogWrite(LED, i);
delay(10);
```

```
}
for (int i=255; i>=0; i--)
{
analogWrite(LED, i);
delay(10);
}
}
```

Листинг 3. Плавное изменение яркости светодиода — fade.ino

Что будет происходить со светодиодом при выполнении Листинг 3? Вы будете наблюдать, как свечение светодиода изменяется от тусклого к яркому в одном цикле `for`, а затем от яркого к тусклому в другом цикле `for`. Все это будет происходить в основном цикле `loop()` до бесконечности. Обязательно обратите внимание на различие двух циклов `for`. В первом цикле выражение `i++` является сокращением кода `i=i+1`. Аналогично, запись `i--` эквивалентна коду `i=i-1`. Первый цикл плавно зажигает светодиод до его максимальной яркости, второй — постепенно гасит его.

Во многих случаях ШИМ пригодна для эмуляции аналогового выхода, но, когда требуется неискаженный аналоговый сигнал, этот вариант неприемлем. Например, ШИМ отлично подходит для регулировки скорости двигателя постоянного тока (примеры будут приведены в следующих главах), но не годится для управления аудиоколонками (без дополнительной внешней схемы).

Чтобы понять все тонкости, разберемся, как на самом деле работает ШИМ. Рассмотрим графики, представленные на Рисунок 17.

ШИМ представляет собой изменение скважности (отношения периода к длительности импульса) прямоугольной последовательности импульсов. Скважность можно трактовать как процент времени, когда прямоугольный импульс имеет уровень `HIGH`, ко всему периоду повторения. Скважность 50% означает, что половину периода сигнал имеет высокий уровень, а половину — низкий.

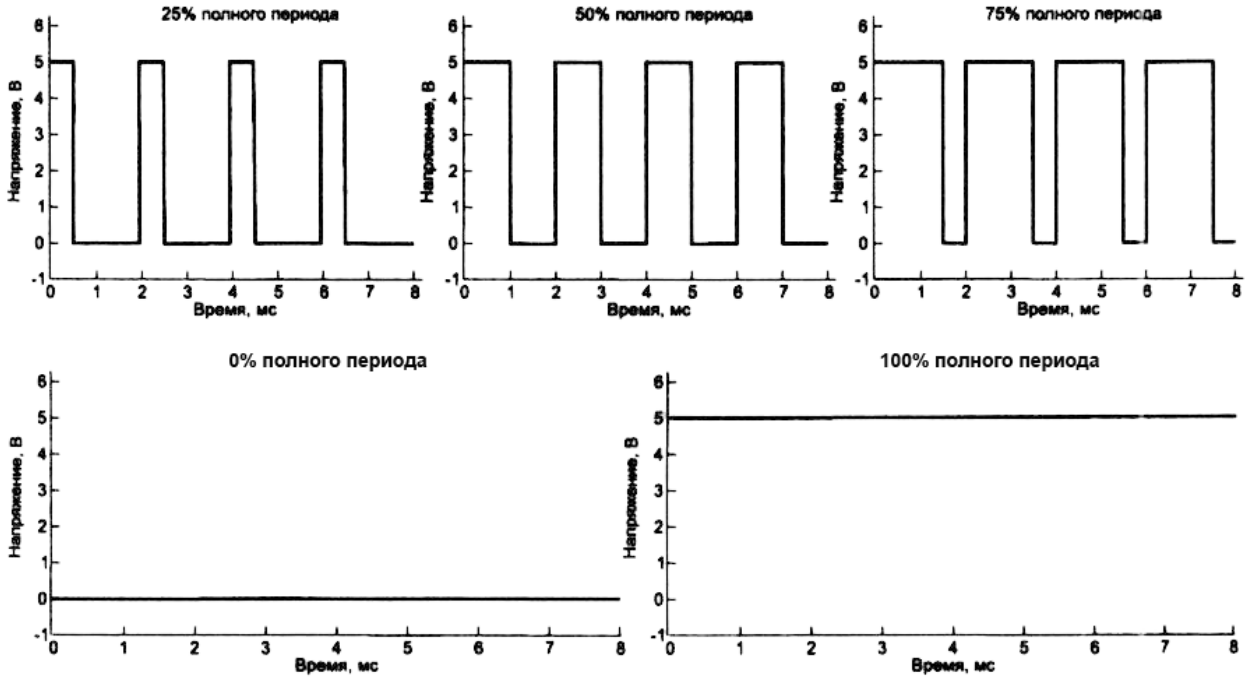

Рисунок 17. ШИМ-сигналы с различной скважностью

Функция *`analogWrite()`* устанавливает скважность последовательности прямоугольных импульсов в зависимости от значения, передаваемого ей:

- значение аргумента *`analogWrite()`*, равное нулю, задает скважность 0% (всегда `LOW`);
- значение 255 — скважность 100% (всегда `HIGH`);
- значение 127 соответствует скважности 50% (половина времени `HIGH`, половина времени `LOW`).

На графиках Рисунок 17 видно, что для сигнала со скважностью 25% значение `HIGH` действует в течение четверти периода, а остальные 75% времени установлено значение `LOW`. Частота прямоугольной последовательности импульсов в случае с ***Arduino*** составляет приблизительно 490 Гц. Другими словами, уровень сигнала меняется от высокого (5 В) к низкому (0 В) приблизительно 490 раз каждую секунду.

Как видим, напряжение, подаваемое на светодиод, на самом деле не понижается, почему же при уменьшении скважности наблюдается спад яркости свечения светодиода? Это связано с особенностью нашего зрения. Если светодиод включается и выключается один раз за 1 мс (при скважности 50%), то вам кажется, что яркость свечения светодиода составляет приблизительно 50% от максимальной, потому что переключение происходит быстрее, чем глаза могут это зафиксировать. Ваш мозг фактически усредняет сигнал и создается впечатление, что светодиод работает на половине яркости.

2.7. Считывание данных с цифровых контактов

Рассмотрим еще одну функцию цифровых контактов. До сих пор мы использовали их в качестве выходов, генерируя цифровой сигнал и ШИМ-сигнал. Следующий шаг— функционирование контактов платы ***Arduino*** в качестве входов. Это позволит подключить, например, переключатели и кнопки для взаимодействия со своим устройством в режиме реального времени. В этом разделе вы научитесь считывать значения на входе, узнаете о стягивающих и подтягивающих резисторах, сможете обрабатывать в программе нажатие кнопки.

2.7.1. Считывание цифровых входов со стягивающим резистором

Изменим схему, изображенную на Рисунок 16. Подключим к цифровому контакту кнопку и стягивающий резистор, в результате схема примет вид, представленный на Рисунок 18.

СОВЕТ

Проверьте, что шины питания и земли обеих плат надежно соединены друг с другом. Тогда в дальнейшем вы сможете легко менять элементы на макетной плате.

Прежде чем написать программу опроса состояния кнопки, важно понять назначение резистора в этой схеме. Почти для всех цифровых входов необходим дополнительный стягивающий (`pull-down`) или подтягивающий (`pull-up`) резисторы для установки "значения по умолчанию" на входном контакте. Представьте себе, что в схеме на рис. 2.5 нет резистора 10 кОм. В этом случае при нажатии на кнопку на выводе будет значение `HIGH`. Но что происходит, когда кнопка не нажата? В такой ситуации входной контакт не привязан ни к чему, как говорят, "висит в воздухе". А поскольку вывод физически не подключен ни к 0 В, ни к 5 В, чтение значения может дать неожиданный результат. Электрические помехи на близлежащих выводах могут привести к тому, что значение напряжения будет колебаться между `HIGH` и `LOW`. Чтобы предотвратить это, стягивающий резистор подключают так, как показано на Рисунок 18.

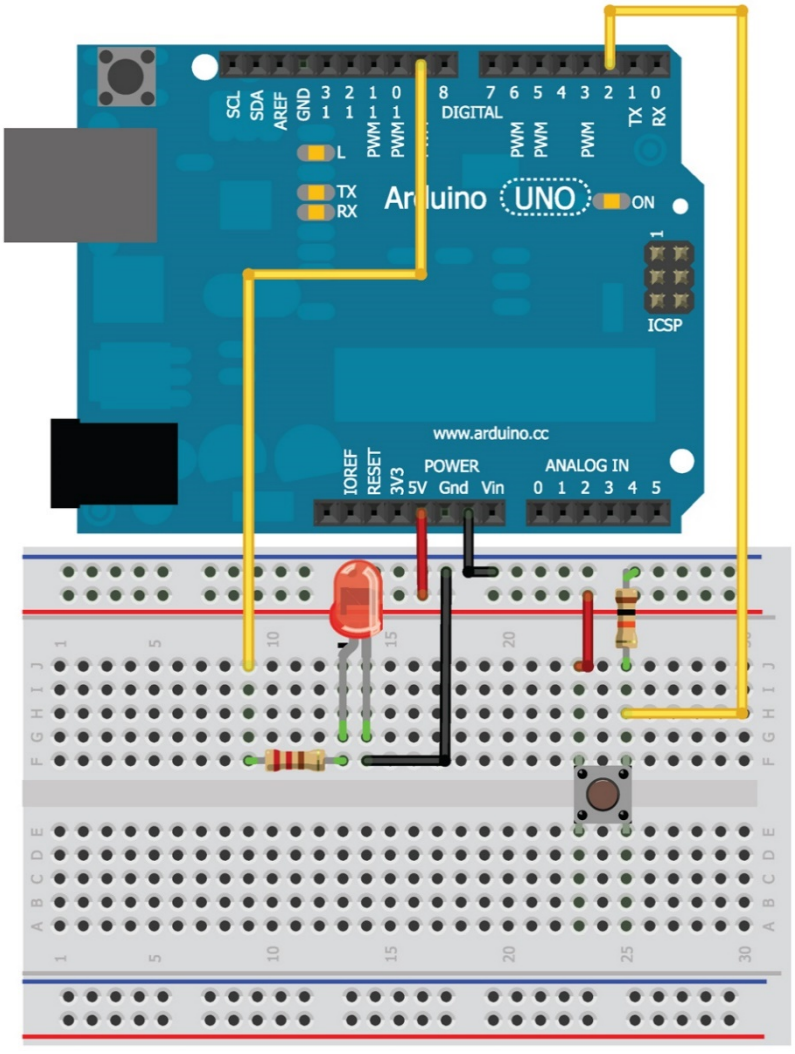

Рисунок 18. Подключение кнопки и светодиода к плате Arduino

Посмотрим, что происходит, когда кнопка не нажата, а входной контакт подключен через стягивающий резистор 10 кОм к земле. Через резистор протекает ток утечки и на входном контакте будет установлено значение напряжения LOW. 10 кОм — довольно распространенный номинал для стягивающего резистора. При нажатии на кнопку входной контакт оказывается напрямую связан с шиной 5 В. Теперь ток может течь двумя путями:

- через практически нулевое сопротивление нажатой кнопки к шине 5 В;
- через высокое сопротивление резистора на землю.

В соответствии с законом Ома ток всегда будет идти по пути наименьшего сопротивления. Большая часть тока будет протекать через замкнутую кнопку и на входе установится уровень HIGH.

> ***ПРИМЕЧАНИЕ***
> *В рассмотренном примере используется стягивающий резистор, но возможна установка и подтягивающего резистора, подключенного к шине 5 В, тогда кнопка должна быть соединена с землей. В таком случае на входном контакте будет значение HIGH при отпущенной кнопке и значение LOW, когда кнопка нажата.*

Стягивающие и подтягивающие резисторы важны, потому что они гарантируют, что кнопка не создаст короткое замыкание между 5 В и землей при нажатии и что входной контакт не останется в «подвешенном» состоянии.
Теперь напишем программу для рассмотренной схемы. Светодиод должен гореть, пока кнопка нажата, и быть выключенным, когда кнопка отжата (Листинг 4).

Листинг 4. Включение светодиода с помощью кнопки — LED_buttton.ino

```
const int LED=9;   // Контакт 9 для подключения светодиода
const int BUTTON=2; // Контакт 2 для подключения кнопки

void setup()
{
     pinMode (LED, OUTPUT);       // Сконфигурировать контакт светодиода
                                  // как выход
     pinMode (BUTTON, INPUT);     // Сконфигурировать контакт кнопки
                                  //как вход
}

void loop()
{
     if (digitalRead(BUTTON) == LOW)
     {
          digitalWrite(LED, LOW);
     }
     else
     {
          digitalWrite(LED, HIGH);
     }
}
```

В коде Листинг 4 реализованы некоторые новые элементы: функция digitalRead() и оператор if /else. Константа button типа int добавлена для контакта кнопки. Кроме того, в функции setup() конфигурируем контакт button как вход. Это необязательно, т. к. выводы ***Arduino*** являются входами по умолчанию. Функция digitalRead() считывает значение сигнала на входе. Если кнопка нажата, digitalRead() возвращает значение HIGH (лог. 1). Если кнопка не нажата, то получаем LOW (лог. 0).

Проверяем содержимое внутри оператора `if()`. Если условие внутри оператора `if()` истинно (кнопка не нажата, `digitalRead()` == `LOW`), вызываем функцию `digitalWrite` (`LED`, `LOW`) (гасим светодиод). В противном случае (кнопка нажата) выполняем код после оператора `else` (включаем светодиод функцией `digitalWrite`(LED, `HIGH`)).
Вот и все! Загружаем данный код на плату ***Arduino*** и убеждаемся, что все работает, как и ожидалось.

2.8. Устранение «дребезга» кнопок

Удобно ли держать кнопку постоянно нажатой для свечения светодиода? Гораздо лучше иметь возможность нажать кнопку один раз, чтобы включить светодиод, и нажав ее еще раз, выключить. При таком варианте, для горения светодиода кнопку не придется удерживать нажатой. К сожалению, сделать это не так легко, как кажется. Нельзя просто считывать значение сигнала на входе, необходимо учитывать явление, называемое дребезгом контактов.
Обычные кнопки представляют собой механические устройства с пружинным контактом. При нажатии на кнопку сигнал не просто меняется от низкого до высокого, он на протяжении нескольких миллисекунд неоднократно меняет свое значение, прежде чем установится уровень `LOW`. Отличие ожидаемого процесса от реального иллюстрируют осциллограммы сигнала с кнопки, приведенные на Рисунок 19.

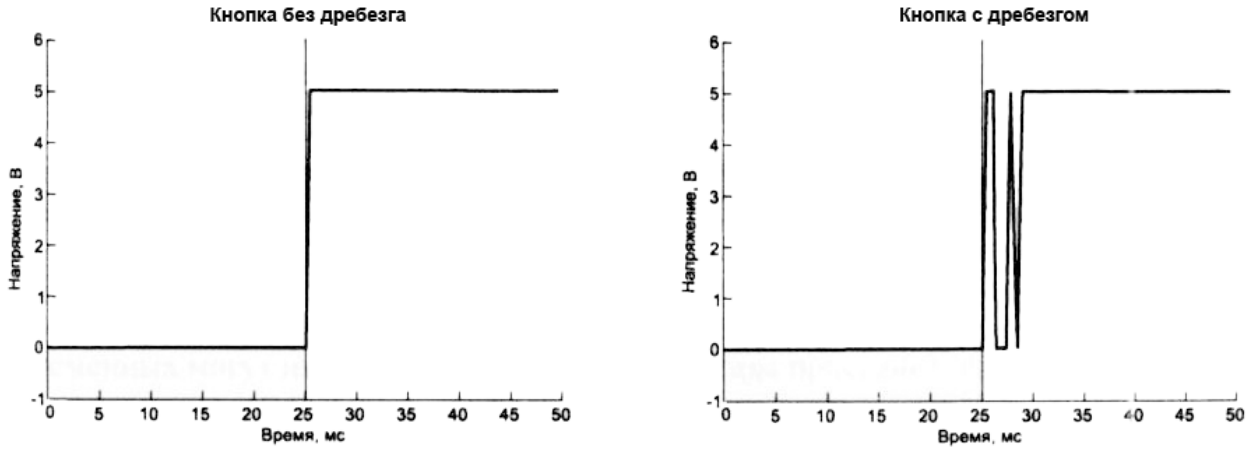

Рисунок 19. Эффект дребезга кнопок

Кнопка была физически нажата в течение 25 мс. Предположение, что состояние кнопки можно определить, считав значение с входа контакта (график слева) неверно. Кнопка фактически возвращается вверх-вниз, пока значение не установится (график справа). Теперь, зная, как ведет себя кнопка, можно написать программу для кнопки с дребезгом, которая фиксирует изменение состояния кнопки, некоторое время ждет и затем снова читает состояние переключателя. Алгоритм работы такой программы можно записать следующим образом:

1. Сохраняем предыдущее и текущее состояния кнопки (при инициализации `LOW`).
2. Считываем текущее состояние кнопки.
3. Если текущее состояние кнопки отличается от предыдущего, ждем 5 мс, потому что кнопка, возможно, изменит свое состояние.
4. Подождав 5 мс, считываем состояние кнопки и делаем его текущим состоянием кнопки.
5. Если предыдущее состояние кнопки было `LOW`, а текущее— `HIGH`, переключаем состояние светодиода.
6. Устанавливаем предыдущее состояние кнопки в качестве текущего.
7. Возвращаемся к шагу 2.

Данный алгоритм — прекрасный пример для изучения функций. Функция — это оператор, который может принимать входные аргументы, выполнять фрагмент кода с их использованием и, возможно, возвращать результат. Не зная этого, вы уже встречали функции в программах. Например, `digitalWrite()` — это функция, которая принимает в качестве аргументов номер контакта и

значение (HIGH или LOW), и устанавливает это значение на контакте. Чтобы упростить программу, можно определить свои собственные функции для инкапсуляции действий, которые придется повторять неоднократно.
Процесс выполнения программы представляет собой многократное повторение шагов. Напишем функцию для устранения дребезга контактов, которую можно вызывать неоднократно. Наша функция будет принимать предыдущее состояние кнопки в качестве входных данных, выполнять противодребезговую защиту и выводить установившееся состояние кнопки. Основной цикл программы переключает состояние светодиода при каждом нажатии кнопки. Загрузите код Листинг 5 в плату ***Arduino*** и посмотрите, как он работает.

Листинг 5. Подавление дребезга кнопки — debounce.ino

```
const int LED=9;              //Контакт 9 для подключения светодиода
const int BUTTON=2;           //Контакт 2 для подключения кнопки
boolean lastButton = LOW;     //Переменная для сохранения предыдущего состояния
кнопки
boolean currentButton = LOW; //Переменная для сохранения текущего состояния
кнопки
boolean ledOn = false;        //Текущее состояние светодиода (включен/выключен)

void setup()
{
  pinMode (LED, OUTPUT);      //Сконфигурировать контакт светодиода как выход
  pinMode (BUTTON, INPUT);    //Сконфигурировать контакт кнопки как вход
}
/*
* Функция сглаживания дребезга
* принимает в качестве аргумента предыдущее состояние кнопки
* и выдает фактическое.
*/
boolean debounce(boolean last)
{
  boolean current = digitalRead(BUTTON);   //Считать состояние кнопки
  if (last != current)                     //Если изменилось...
  {
    delay(5);                              //Ждем 5 мс
    current = digitalRead(BUTTON);         //Считываем состояние кнопки снова
  }
  return current;                          //Возвращаем состояние кнопки
}

void loop()
{
  currentButton = debounce(lastButton);    //чтение состояние дребезга
  if (lastButton == LOW && currentButton == HIGH)     //Если нажатие…
  {
    ledOn = !ledOn;                        //переключить значение светодиода
  }
  lastButton = currentButton;              //сбросить значение кнопки

  digitalWrite(LED, ledOn);                //изменить состояние светодиода

}
```

Теперь рассмотрим текст Листинг 5 подробнее. Сначала заданы номера контактов для подключения кнопки и светодиода. Затем объявлены три глобальные логические переменные, которые будут

изменяться в программе (значение глобальной переменной можно менять в любой части программы). Каждой из трех переменных присвоены начальные значения (LOW, LOW и false). Далее в программе значения этих переменных могут изменяться с помощью оператора присваивания =.
Рассмотрим функцию подавления дребезга кнопки boolean debounce(). Эта функция принимает логическую переменную (имеющую только два состояния: true/false, HIGH/LOW, вкл./выкл., 1/0) предыдущего состояния кнопки и возвращает текущее значение состояния кнопки. Внутри функции текущее состояние кнопки сравнивается с предыдущим с помощью оператора != (не равно). Если состояния отличаются, то кнопка, возможно, нажата. Затем ожидаем 5 мс (этого достаточно, чтобы состояние кнопки стабилизировалось после дребезга), прежде чем проверить состояние кнопки снова. Затем вновь проверяем состояние кнопки. Как вы помните, функции могут возвращать результат. Данная функция возвращает текущее значение булевой локальной переменной, которая объявлена и используется только в функции debounce(). Когда функция debounce() вызывается из основного цикла, возвращенное значение записывается в глобальную переменную currentButton, которая была определена в начале программы.
После вызова функции debounce() и установки значения переменной currentButton происходит сравнение текущего и предыдущего значений состояния кнопки с помощью оператора && (логический оператор "И", означающий, что выражение в скобках выполнится, только если истинно каждое из равенств, разделенных оператором &&).
Если ранее состояние кнопки было LOW, а теперь HIGH, значит, кнопка была нажата и нужно инвертировать значение переменной ledOn. Это действие выполняет оператор «!» перед переменной ledOn. Цикл закончен, обновляем предыдущую переменную состояния кнопки и изменяем состояние светодиода.
Программа изменяет состояние светодиода после каждого нажатия кнопки. При отсутствии проверки дребезга кнопки результаты будут непредсказуемыми.

2.9. Создание управляемого ночника на RGB-светодиоде

Вы уже знаете, как управлять цифровыми выходами, как создать противодребезговую защиту для кнопки, как менять яркость светодиода с помощью ШИМ-сигнала. Теперь подключим к плате ***Arduino*** трехцветный RGB-светодиод и создадим ночник, цвет которого будет меняться при нажатии на кнопку. В RGB-светодиоде можно смешивать цвета, изменяя широтно-импульсной модуляцией яркость каждого из них.
В устройстве используем RGB-светодиод с четырьмя выводами, один из которых является катодом, общим для всех трех диодов, а остальные — аноды для диодов каждого цвета. Подключите RGB-светодиод проводами к трем ШИМ-контактам платы ***Arduino*** через токоограничивающие резисторы, как показано на Рисунок 20.
Вы можете настроить циклическое переключение цветов светодиода при каждом нажатии на кнопку. В данном случае удобно добавить функцию для установки цвета светодиода в следующее состояние. В программе, представленной в листинге 2.6, определено семь цветов и состояние, когда светодиод не горит. С помощью функции *analogWrite()* можно задать свои цветовые комбинации. Единственное отличие цикла loop о от предыдущего примера — увеличение числа состояний светодиода (по кругу от 0 до 7).
Загрузите программу в плату и поэкспериментируйте с разноцветным ночником. Поменяйте цвет RGB-светодиода, изменив значения в функции *analogWrite()* на свои собственные.

Листинг 6. Управляемый ночник на светодиоде — rgb_nightlight.ino

```
const int BLED=9;                //Контакт 9 для вывода BLUE RGB-светодиода
const int GLED=10;               //Контакт 10 для вывода GREEN RGB-светодиода
const int RLED=11;               //Контакт 11 для вывода RED RGB-светодиода
const int BUTTON=2;              //Контакт 2 для входа кнопки

boolean lastButton = LOW;        //Предыдущий статус кнопки
boolean currentButton = LOW;     //Текущий статус кнопки
int ledMode = 0;                 //Значение статуса RGB-светодиода
```

```
void setup()
{
  pinMode(BLED, OUTPUT);         //Сконфигурировать BLUE-контакт
                                 //светодиода как выход
  pinMode(GLED, OUTPUT);         //Сконфигурировать GREEN-контакт
                                 //светодиода как выход
  pinMode(RLED, OUTPUT);         //Сконфигурировать RED-контакт
                                 //светодиода как выход
  pinMode(BUTTON, INPUT);        //Сконфигурировать контакт кнопки
                                 //как вход
}
      /*
      *  Функция сглаживания дребезга
      * принимает в качестве аргумента предыдущее состояние кнопки
      * и выдает фактическое.
      */
boolean debounce(boolean last)
{
  boolean current = digitalRead(BUTTON);   //Считать состояние кнопки
  if (last != current)                     //Если изменилось...
  {
    delay(5);                              //Ждем 5 мс
    current = digitalRead(BUTTON);         //снова считываем состояние кнопки
  }
  return current;                          //Возвращаем состояние кнопки
}
      /*
      *  Выбор режима светодиода.
      * Передача номера режима и установка заданного режима светодиода.
      */
void setMode(int mode)
{
  //Красный
  if (mode == 1)
  {
    digitalWrite(RLED, HIGH);
    digitalWrite(GLED, LOW);
    digitalWrite(BLED, LOW);
  }
  //Зелёный
  else if (mode == 2)
  {
    digitalWrite(RLED, LOW);
    digitalWrite(GLED, HIGH);
    digitalWrite(BLED, LOW);
  }
  //Синий
  else if (mode == 3)
  {
    digitalWrite(RLED, LOW);
    digitalWrite(GLED, LOW);
    digitalWrite(BLED, HIGH);
  }
```

```
  //Пурпурный (Красный + Синий)
  else if (mode == 4)
  {
    analogWrite(RLED, 127);
    analogWrite(GLED, 0);
    analogWrite(BLED, 127);
  }
  //Бирюзовый (Синий + Зеленый)
  else if (mode == 5)
  {
    analogWrite(RLED, 0);
    analogWrite(GLED, 127);
    analogWrite(BLED, 127);
  }
  //Оранжевый (Зеленый + Красный)
  else if (mode == 6)
  {
    analogWrite(RLED, 127);
    analogWrite(GLED, 127);
    analogWrite(BLED, 0);
  }
  //Белый (Зеленый + Красный + Синий)
  else if (mode == 7)
  {
    analogWrite(RLED, 85);
    analogWrite(GLED, 85);
    analogWrite(BLED, 85);
  }
  //Выключен (mode = 0)
  else
  {
    digitalWrite(RLED, LOW);
    digitalWrite(GLED, LOW);
    digitalWrite(BLED, LOW);
  }
}

void loop()
{
  currentButton = debounce(lastButton);    //Чтение статуса дребезга кнопки
  if (lastButton == LOW && currentButton == HIGH)     //Если нажата кнопка
  {
    ledMode++;                          //Инкремент переменной статуса светодиоде
  lastButton = currentButton;           //Сброс значения кнопки
           //Если пройдены все режимы – сброс счётчика на 0
  if (ledMode == 8) ledMode = 0;
  setMode(ledMode);                     //Изменить режим светодиода
}
```

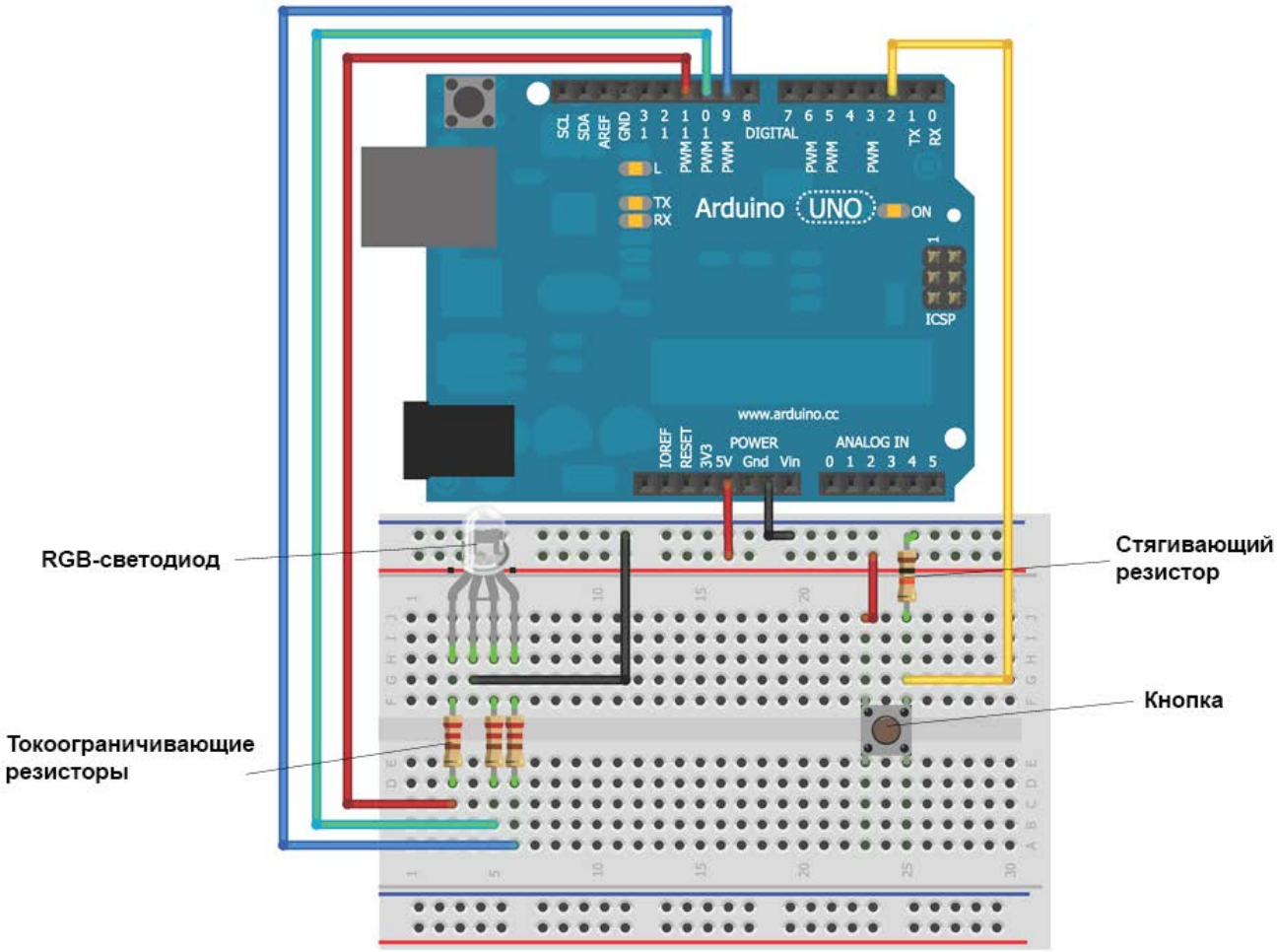

Рисунок 20. Монтажная схема ночника

На первый взгляд может показаться, что программа слишком велика. Но в основном это комбинация фрагментов кода, который вы уже встречали в данной главе.
Вы можете самостоятельно изменить этот проект. Например, добавить кнопки для управления каждым выводом RGB-светодиода. Или реализовать дополнительный режим мигания каждым цветом, взяв код из части 1.
Возможности для творчества безграничны.

Резюме

В этой главе вы узнали о следующем:

- Как работать с макетной платой.
- Как выбрать резистор для ограничения тока светодиода.
- Как подключить внешний светодиод к плате ***Arduino***.
- Как использовать ШИМ, как замену аналогового вывода.
- Как считывать состояние кнопки.
- Как подавить дребезг кнопки.
- Для чего нужны подтягивающий и стягивающий резисторы.

Глава 3. Опрос аналоговых датчиков

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;
- макетная плата;
- перемычки;
- потенциометр 10 кОм;
- 2 резистора номиналом 10 кОм;
- 3 резистора номиналом 220 Ом;
- кабель USB;
- фоторезистор;
- датчик температуры TMP36 (или любой другой аналоговый датчик на 5 В);
- RGB-светодиод с общим катодом.

Электронные ресурсы к главе

На странице https://www.exploringarduino.com/content/ch3/ можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118549368.html в разделе Downloads.

Что вы узнаете в этой главе

Наш мир является аналоговым. Хотя нередко можно услышать фразу про мир «цифровых технологий», большинство наблюдаемых явлений вокруг нас имеет аналоговый характер. Мир предполагает бесконечное число возможных состояний, будь то солнечный свет, или температура океана, или концентрация загрязняющих веществ в воздухе. Эта глава посвящена методам преобразования аналоговых величин в цифровые значения, которые могут быть проанализированы микроконтроллером ***Arduino***.

Далее вы узнаете о различиях между аналоговыми и цифровыми сигналами и о способе конвертации одних в другие, мы также рассмотрим аналоговые датчики, которые могут взаимодействовать с платой ***Arduino***. Основываясь на материале предыдущей главы, вы сможете добавить датчик света для автоматического изменения настроек ночника. Вы также узнаете, как отправлять аналоговые данные с платы ***Arduino*** на компьютер через последовательный интерфейс, что открывает огромные возможности для создания более сложных систем, способных передавать данные об окружающей среде на компьютер.

> ***ПРИМЕЧАНИЕ***
>
> *Видеоурок данной главы можно посмотреть на интернет-странице https://www.jeremyblum.com/2011/01/24/arduino-tutorial-4-analog-inputs/[1].*
>
> *Если вы хотите узнать больше о различиях между аналоговым и цифровым сигналами, посмотрите видеофильм, расположенный на интернет-странице https://www.jeremyblum.com/2010/06/20/lets-get-digital-or-analog/.*

3.1. Понятие об аналоговых и цифровых сигналах

Данные об окружающем мире все устройства неизбежно получают в аналоговом виде. Вспомните ночник из предыдущей главы. Для управления цифровым входом там была кнопка. Переключатель — это цифровое устройство, он имеет только два возможных состояния: включено или выключено, `HIGH` или `LOW`, 1 или 0, и т. д. Цифровая информация представляет собой серию бинарных (цифровых) данных. Каждый бит принимает только одно из двух возможных значений.

Но мир вокруг нас редко представляет информацию только двумя способами. Выгляните в окно. Что вы видите? Если это дневное время, вы, вероятно, видите солнечный свет, деревья, колышущиеся на ветру, и возможно, проезжающие машины и гуляющих людей. Все это нельзя отнести к двоичным данным. Солнечный свет не просто включен или выключен, его яркость варьируется в течение дня. Точно так же у ветра не два единственных состояния, он все время дует порывами с различной скоростью.

[1] На русском: **http://wiki.amperka.ru/видеоуроки:4-аналоговые-входы**.

3.2. Сравнение аналоговых и цифровых сигналов

Графики на Рисунок 21 показывают, чем отличаются друг от друга аналоговые и цифровые сигналы. Слева прямоугольные импульсы, амплитуда которых принимает только два значения: 0 и 5 вольт. Точно так же, как с кнопкой из предыдущей главы: только `HIGH` или `LOW`. Справа изображен фрагмент косинусоидального сигнала. Несмотря на то, что его амплитуда находится в тех же границах (0 и 5 вольт), аналоговый сигнал принимает бесконечное число значений между этими двумя.

Аналоговые сигналы нельзя представить конечным числом состояний, теоретически они могут иметь бесконечное число значений в пределах некоторого диапазона. Допустим, солнечный свет— это аналоговый сигнал, который нужно измерить. Естественно, есть разумный диапазон, в пределах которого меняется освещенность (измеряется в люксах — световом потоке на единицу площади). Можно обоснованно ожидать значение показаний между 0 люкс (для совершенно черного) и 130 000 люкс на прямом солнечном свете. Если бы измерительный прибор был абсолютно точен, то можно получить бесконечное число значений в данном диапазоне.

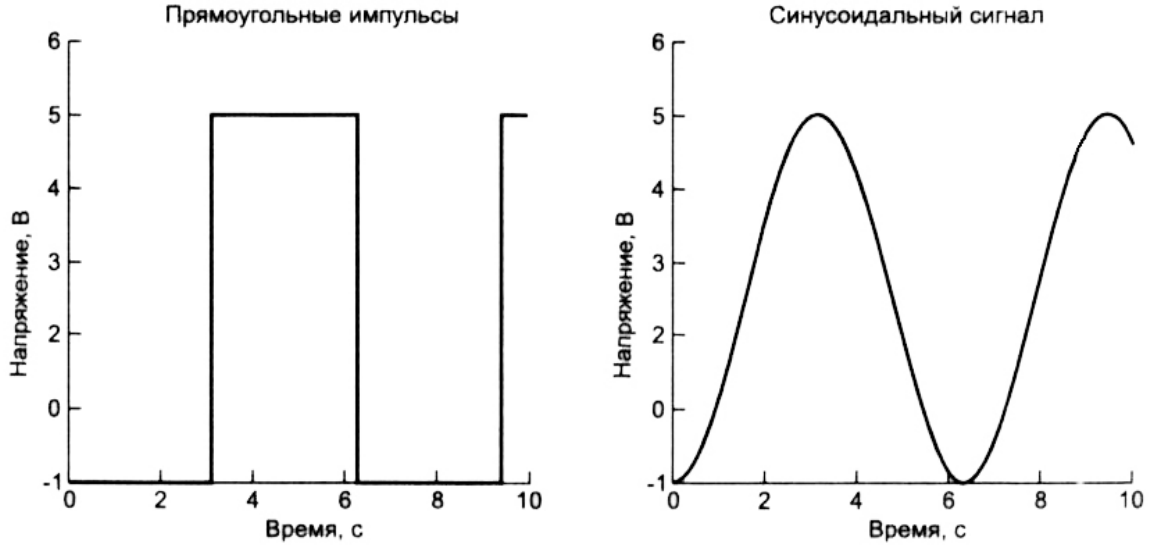

Рисунок 21. Аналоговые и цифровые сигналы

Компьютерная система никогда не может оперировать с бесконечным числом десятичных разрядов для аналогового значения, потому что объем памяти и производительность компьютера ограничены. Как же тогда соединить интерфейс цифрового контроллера ***Arduino*** с аналоговым реальным миром? Это делает аналого-цифровой преобразователь (АЦП), который преобразует аналоговые значения в цифровые с заданной точностью.

3.3. Преобразование аналогового сигнала в цифровой

Предположим, что вы хотите измерить освещенность в своей комнате. Хороший светочувствительный датчик выдает выходное напряжение, которое зависит от освещенности комнаты. Когда в помещении абсолютно темно, устройство выдало бы 0 В, а при максимальной освещенности — 5 В. Промежуточные значения соответствуют средним освещенностям. Но как эти значения считает плата ***Arduino***, чтобы узнать, насколько светло в комнате? Преобразовать аналоговые значения напряжения в числа, которые может обрабатывать контроллер, позволяет аналого-цифровой преобразователь ***Arduino***.

Точность АЦП зависит от его разрядности. На плате ***Arduino Uno*** установлен 10-разрядный АЦП. Это означает, что АЦП может разделить аналоговый сигнал на 2^{10} различных значений. Следовательно, ***Arduino*** может присвоить 2^{10}= 1024 аналоговых значений, от 0 до 1023.

Опорное напряжение определяет максимальное напряжение на входе АЦП, его значение соответствует коду 1023. При нулевом входном напряжении АЦП выдает на выходе 0, при входном напряжении 2,5 В на выходе будет значение 512 (половина от 1023), при входном напряжении 5 В выходной код равен 1023. Чтобы лучше понять это, посмотрите на графики для трехразрядного АЦП, изображенные на рис. 3.2. В принципе, опорное напряжение АЦП можно изменить, но в наших устройствах опорным будет напряжение 5 В.

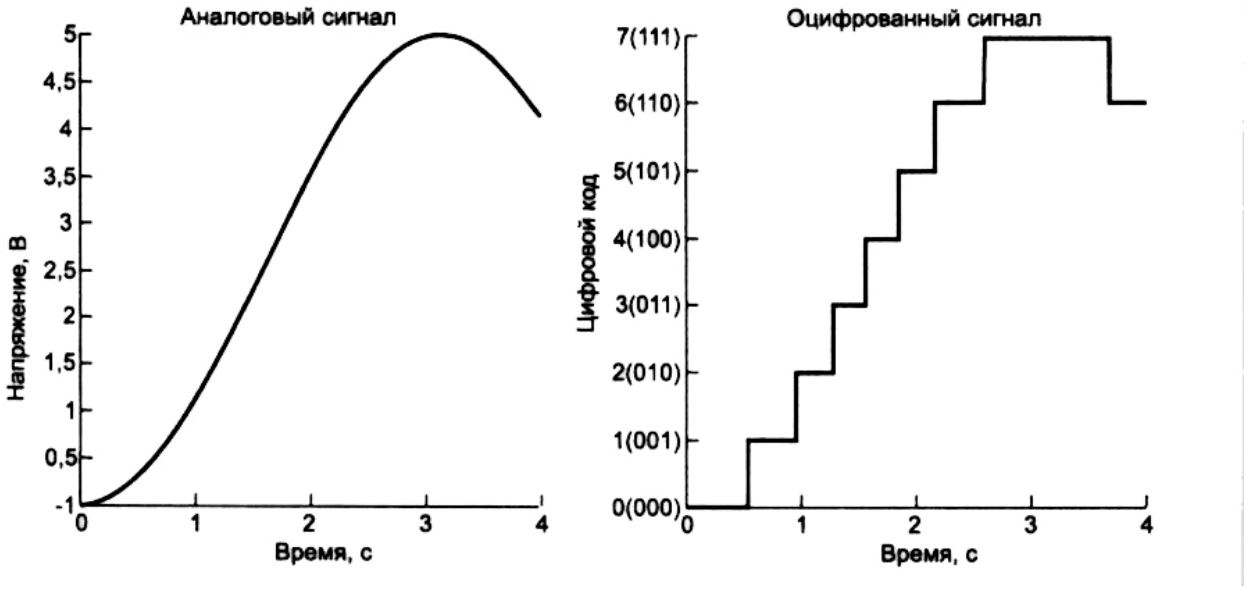

Рисунок 22. Трехразрядное аналого-цифровое преобразование

У трехразрядного АЦП 3 бита разрешения, поскольку 2^3= 8, следовательно, у него есть 8 уровней, от 0 до 7. Любому аналоговому значению, которое поступает на вход такого АЦП, на выходе соответствует код от 0 до 7. На Рисунок 22 показано, что уровни входного напряжения преобразуются в выходные дискретные цифровые коды, с которыми может оперировать микроконтроллер. Чем выше разрядность, тем больше уровней, которые доступны для представления каждого значения. Как упоминалось, у ***Arduino Uno*** АЦП имеет 1024 уровней, а не 8, как на Рисунок 22.

> ***ПРИМЕЧАНИЕ***
> *Если вы хотите узнать больше об использовании нестандартного (или внешнего) опорного напряжения, посетите страницу на официальном сайте* ***Arduino:*** *https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/.*

3.4. Считывание аналоговых датчиков с помощью *Arduino*. Команда *analogRead()*

Теперь, когда вы понимаете, как преобразовать аналоговые сигналы в цифровые коды, можно начать писать программы и разрабатывать схемы. У различных плат ***Arduino*** разное число аналоговых контактов. Для чтения аналоговых значений предусмотрена Функция `analogRead()`.
Мы начнем с простых экспериментов с потенциометром и аналоговым датчиком. Затем вы узнаете, как работают делители напряжения и как можно сделать свои собственные аналоговые датчики из компонентов, сопротивление которых зависит от каких-нибудь внешних факторов.

3.5. Чтение данных с потенциометра

Самый простой аналоговый датчик, с которого можно получить аналоговый сигнал, — это потенциометр. Их используют в стереосистемах, звуковых колонках, термостатах и в других изделиях. Потенциометры действуют как регулируемые делители напряжения и снабжены ручкой-регулятором. Они бывают разных размеров и форм, но всегда имеют три вывода.
Подключите один крайний вывод потенциометра к земле, а другой к шине 5 В. Потенциометры симметричны, так что не имеет значения, с какой стороны вы подключите шину питания, а с какой землю. Средний вывод соедините с аналоговым контактом 0 на плате ***Arduino***. Как правильно подключить потенциометр к ***Arduino***, показано на Рисунок 23. При повороте ручки потенциометра аналоговый входной сигнал будет плавно меняться от 0 до 5 В. В этом можно убедиться с помощью

мультиметра. Переведите мультиметр в режим измерения напряжения, подсоедините его, как показано на Рисунок 24, и следите за показаниями, поворачивал ручку потенциометра. Красный (положительный) щуп мультиметра должен бьгъ подключен к среднему контакту потенциометра, а черный (отрицательный) щуп — к земле.

ПРИМЕЧАНИЕ

Потенциометр и мультиметр внешне могут выглядеть не так, как показано на Рисунок 24.

Прежде чем использовать потенциометр для управления другим оборудованием, посмотрим, как считать значение сопротивления потенциометра с помощью АЦП и передать через последовательный порт ***Arduino*** для просмотра значений на компьютере. Для чтения значения аналогового входа предусмотрена функция `analogRead()`, для вывода значений в последовательный порт ***Arduino*** IDE— функция `Serial.pritln()`. Наберите и загрузите в плату ***Arduino*** программу из Листинг 7.

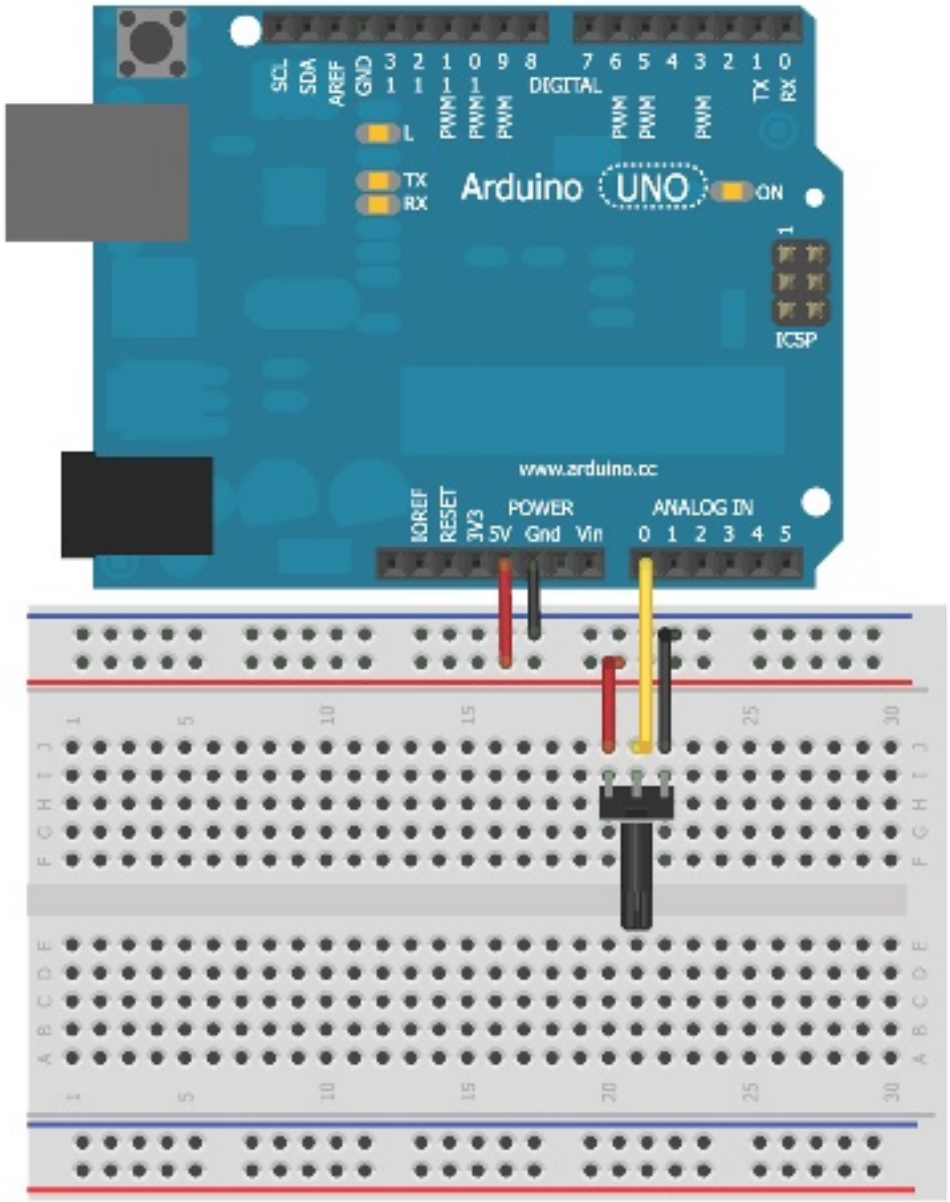

Рисунок 23. Подключение потенциометра

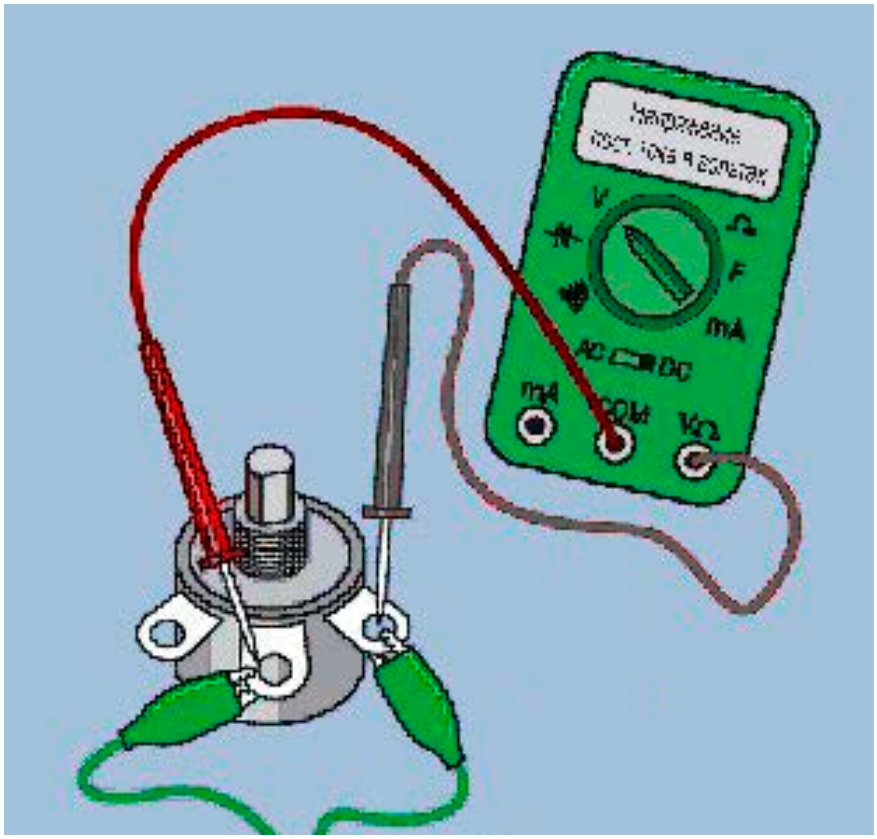

Рисунок 24. Измерение напряжения с помощью мультиметра

Листинг 7. Программа чтения данных потенциометра — pot.ino

```
//Программа чтения данных с потенциометра
const int POT=0;  //Аналоговый вход 0 для подключения потенциометра
int val = 0;      //Переменная для хранения значения потенциометра

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  val = analogRead(POT);
  Serial.println(val);
  delay(500);
}
```

Подробно функционирование последовательного интерфейса обмена данными мы рассмотрим в последующих главах. А сейчас достаточно знать, что сначала необходимо инициировать последовательное соединение, вызвав функцию `Serial.begin()`, единственный аргумент которой задает скорость передачи данных в бодах. Скорость передачи данных определяет количество битов, передаваемых в секунду. Высокая скорость передачи позволяет передавать больше данных за меньшее время, но может привести к ошибкам в некоторых системах связи. В наших примеэах выбрана скорость 9600 бод.

В каждой итерации цикла переменная `val` получает аналоговое значение, считанное командой `analogRead()` с входа, соединенного со средним контактом потенциометра (в нашем случае это вход ao). Далее это значение функция `Serial.println()` выводит в последовательный порт, соединенный с компьютером. Затем следует задержка в полсекунды (чтобы числа выводились не быстрее, чем вы можете их прочитать).

После загрузки на плату ***Arduino*** вы заметите, что светодиод tx, расположенный на плате, мигает

каждые 500 мс (по крайней мере, так должно быть). Этот индикатор показывает, что плата ***Arduino*** передает данные через последовательный USB-интерфейс на компьютер. Для просмотра данных подойдут любые терминальные программы, но в ***Arduino*** IDE есть встроенный монитор последовательного порта, для запуска которого нажмите кнопку, обведенную красным кружком на Рисунок 25.

После запуска монитора последовательного порта на экране компьютера появляется окно с отображением потока передаваемых чисел. Поверните ручку потенциометра, и вы увидите, что выводимые значения меняются. Если повернуть ручку в одном направлении, числа начинают приближаться к 0, если в другом — к 1023. Пример отображения данных показан на Рисунок 26.

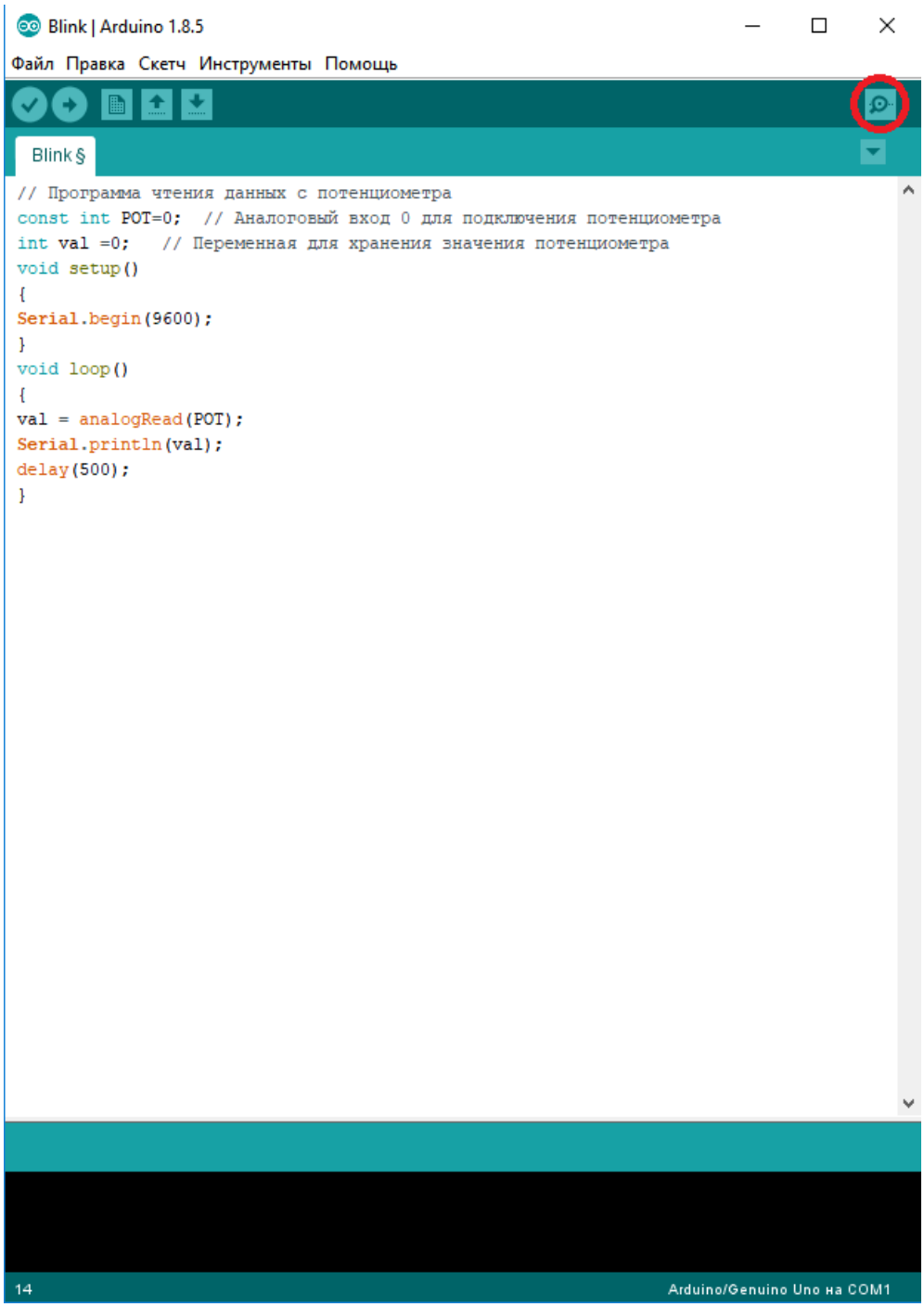

Рисунок 25. Кнопка запуска монитора последовательного порта

ПРИМЕЧАНИЕ

Если выводятся непонятные символы, убедитесь, что скорость передачи данных установлена правильно. В программе порт инициализирован на скорость 9600 бод, такое же значение необходимо установить в настройках монитора последовательно порта.

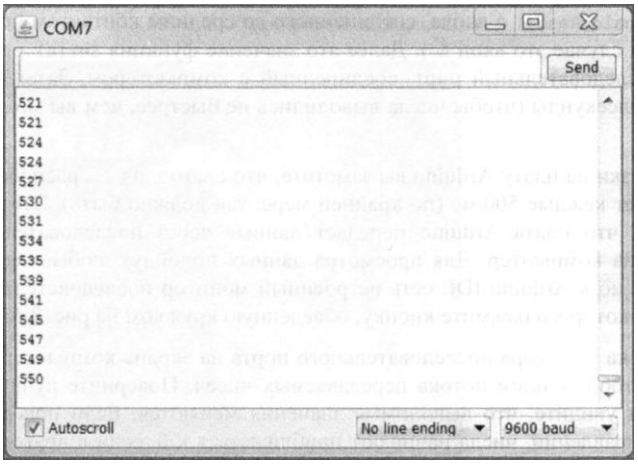

Рисунок 26. Вывод данных в последовательный порт

Итак, вы настроили получение аналоговых данных и смогли их изменять с помощью потенциометра, интересно, не правда ли? Но это только первый шаг. Далее вы узнаете об иных типах аналоговых датчиков и о том, как с их помощью управлять другим оборудованием. В этой главе будем снова управлять светодиодом, но в следующих главах рассмотрим двигатели и другие устройства.

3.6. Использование аналоговых датчиков

Хотя на контакте потенциометра можно получить значение аналогового напряжения, он на самом деле не является датчиком в традиционном смысле. Потенциометры «чувствуют» лишь поворот ручки, но это не слишком интересно. Но есть настоящие датчики, они выдают значения на аналоговом выходе, соответствующие «реальному» действию. Примеры датчиков:

- акселерометры для обнаружения наклона (применяются в смартфонах и планшетах);
- магнитометры для фиксации магнитных полей (необходимы при создании цифровых компасов);
- инфракрасные датчики для определения расстояния до объекта;
- датчики для измерения температуры.

Многие из этих датчиков подключают аналогично потенциометру: два контакта питания (VCC и GND) и один к аналоговому входу платы ***Arduino***. Для следующего эксперимента вы можете выбрать любой датчик из списка:

- Инфракрасный датчик расстояния Sharp. (Описание датчика приведено на странице http://www.exploringarduino.com/parts/IR-Distance-Sensor, разъем для подключения — http://www.exploringarduino.com/parts/JST-Wire.)
  Инфракрасные датчики Sharp измеряют расстояние от датчика до объекта. По мере удаления объекта напряжение на выходе датчика уменьшается. Рисунок на странице 5 технического описания датчиков Sharp (скачать можно по адресу http://exploringarduino.com/wp-content/uploads/2013/06/GP2Y0A-datasheet.pdf) показывает связь между выходным напряжением и расстоянием до объекта.
- Датчик температуры TMP36. (Описание приведено на странице http://www.exploringarduino.com/parts/TMP36.)
  Датчик температуры TMP36 позволяет легко преобразовать выходной уровень напряжения в показания температуры в градусах Цельсия. Каждые 10 мВ выходного напряжения соответствуют 1°C. Формула для преобразования выходного напряжения (в мВ) в температуру (в °C) выглядит так:
  $$T = (U_{вых} — 500) / 10.$$
  Смещение 500 мВ необходимо для работы с температурами ниже 0°C. Эта зависимость приведена на Рисунок 27.
- Трехосевой аналоговый акселерометр. (Описание приведено на странице http://www.exploringarduino.com/parts/TriAxis-Analog-Accelerometer.)
  Трехосевые акселерометры предназначены для определения ориентации объекта. Аналоговые акселерометры выдают значения, соответствующие смещению объекта по каждой оси: X, У и

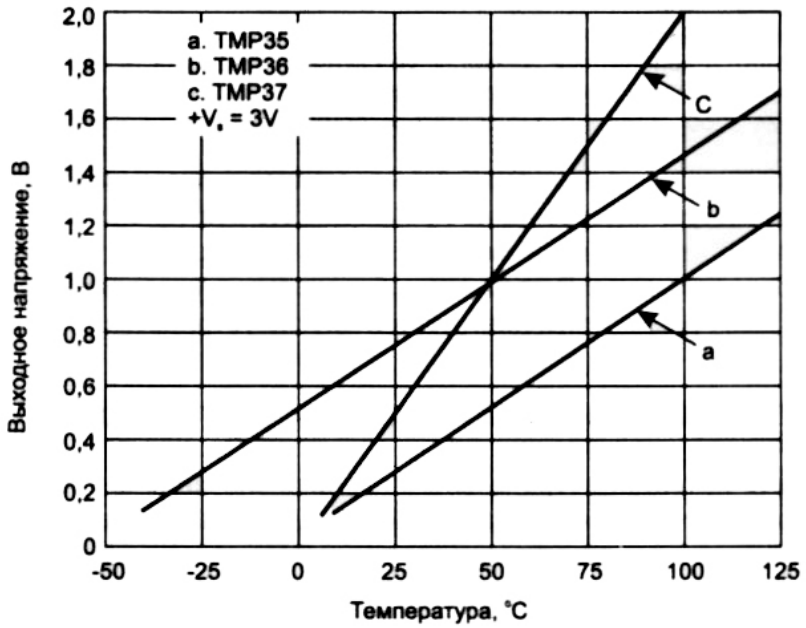

Рисунок 27. Зависимость выходного напряжения от температуры для различных датчиков

Z (для каждой оси разном контакте). С помощью тригонометрических преобразований и закона всемирного тяготения можно определить позицию объекта в трехмерном пространстве. Напряжение питания многих акселерометров равно 3,3 В, поэтому для получения правильных значений в программе нужно предусмотреть установку опорного напряжения `analogReference()`, а вывод питания акселерометра подсоединить к контакту 3,3 В платы ***Arduino***.

- Двухосевой аналоговый гироскоп. (Описание датчика приведено на странице http://www.exploringarduino.com/parts/DualAxis-Analog-Gyroscope.) Гироскопы, в отличие от акселерометров, нечувствительны к силе тяжести. Напряжение на их аналоговом выходе изменяется в соответствии с угловым ускорением вокруг оси. Гироскопы особенно полезны для обнаружения поворота. Посмотрите пример взаимодействия гироскопа с платой ***Arduino*** в моем проекте SudoGlove (http://www.jeremyblum.com/portfolio/sudoglove-hardware-controller/). Перчатка-манипулятор, которую я разработал, распознает движения руки и способна управлять музыкальным синтезатором или радиоуправляемым автомобилем. Напряжение питания многих гироскопов составляет 3,3 В.

Если вы выбрали датчик, перейдем к примеру его использования.

3.7. Работа с аналоговым датчиком температуры

Рассмотрим простой пример работы с датчиком температуры TMP36, упомянутым в предыдущем разделе. Вы можете выбрать любой аналоговый датчик из приведенного ранее списка или взять какой-нибудь другой. Последовательность действий, описанная далее, практически одинакова для любого аналогового датчика.

Для начала подсоедините к плате ***Arduino Uno*** RGB-светодиод, как в Глава 2. Цифровые контакты ввода-вывода, широтно-импульсная модуляция, и датчик температуры к выходу A0 (Рисунок 28).

На основе этой схемы создадим простую систему, сигнализирующую об изменении температуры. RGB-светодиод будет гореть зеленым, когда температура находится в пределах допустимого диапазона, красным, когда станет жарко, и синим, когда становится холодно.

Прежде всего, определите приемлемый для вас температурный диапазон. Используя программу из Листинг 7, определите аналоговые значения для верхнего и нижнего порогов температуры. Для меня нижний порог комфортной температуры составляет 20 °C, что соответствует аналоговому значению 143. У вас эта цифра может быть другой. Следите за показаниями в мониторе последовательного порта при наступлении нижнего и верхнего предела температуры. Эти значения можно получить из графика на Рисунок 27 или из формулы (Формула 1), связывающей температуру (в °C) с входным напряжением (в мВ):

Формула 1

Температура (°C) × 10 = Напряжение (мВ) — 500.

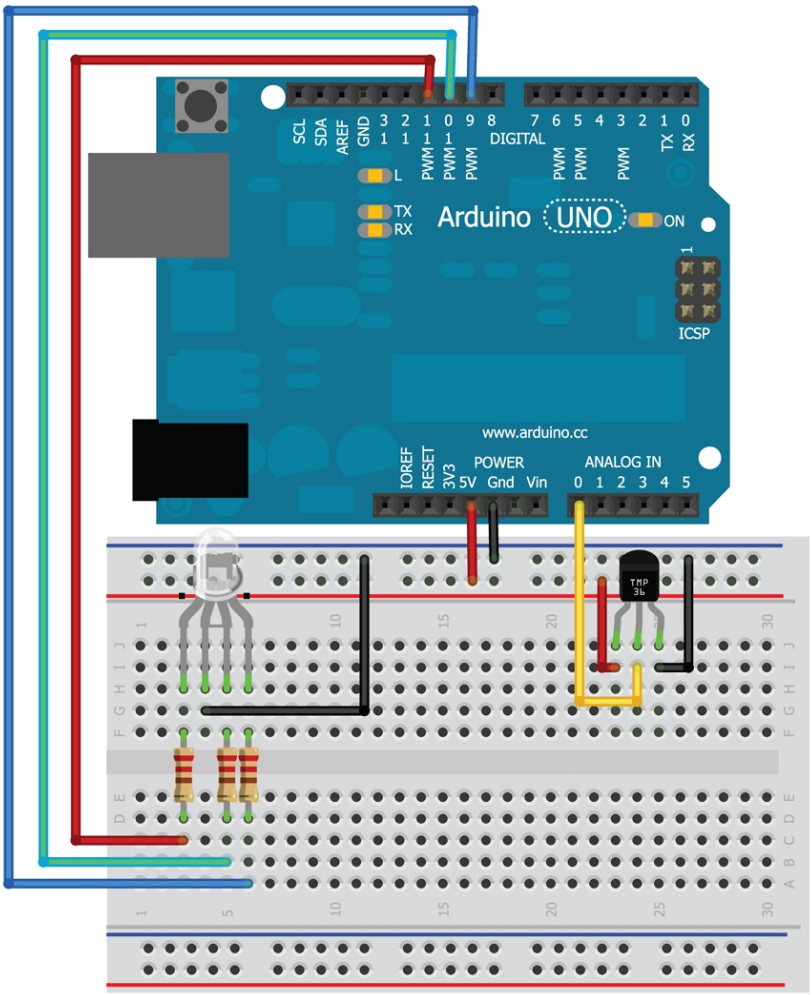

Рисунок 28. Схема подключения датчика температуры

Напряжение 700 мВ соответствует температуре 20°C. Расчет по формуле (или просто анализ показаний монитора последовательного порта) дает для 22°C цифровое значение 147, для 18°C — 139. Эти величины выберем для нижнего и верхнего значений комфортной температуры, чтобы изменять цвет светодиода. Функция `analogRead()` будет считывать показания датчика температуры, `digitalWrite()` — устанавливать цвет светодиода.

> **СОВЕТ**
>
> *Рекомендую вам не копировать Листинг 8, а попробовать написать текст программы самостоятельно, чтобы убедиться в своих силах. Сравните свой результат с приведенным далее.*

Листинг 8. Программа температурного оповещателя — tempalert.ino

```
// Температурный оповещатель
const int BLED=9;          //Контакт 9 для вывода BLUE RGB-светодиода
const int GLED=10;         //Контакт 10 для вывода GREEN RGB-светодиода
const int RLED=11;         //Контакт 11 для вывода RED RGB-светодиода
const int TEMP=0;          //Контакт A0 для подключения датчика температуры

const int LOWER_BOUND=139; //Нижний порог
const int UPPER_BOUND=147; //Верхний порог
```

```
int val = 0;                //Переменная для чтения аналогового значения


void setup()
{
  pinMode (BLED, OUTPUT);  //Сконфигурировать BLUE контакт светодиода
                           //как выход
  pinMode (GLED, OUTPUT);  //Сконфигурировать GREEN контакт светодиода
                           //как выход
  pinMode (RLED, OUTPUT);  //Сконфигурировать RED контакт светодиода
                           //как выход
}

void loop()
{
  val = analogRead(TEMP);

  if (val < LOWER_BOUND)
  {
    digitalWrite(RLED, LOW);
    digitalWrite(GLED, LOW);
    digitalWrite(BLED, HIGH);
  }
  else if (val > UPPER_BOUND)
  {
    digitalWrite(RLED, HIGH);
    digitalWrite(GLED, LOW);
    digitalWrite(BLED, LOW);
  }
  else
  {
    digitalWrite(RLED, LOW);
    digitalWrite(GLED, HIGH);
    digitalWrite(BLED, LOW);
  }
}
```

В коде листинге 8 нет ничего принципиально нового, он сочетает в себе все изложенное ранее о системах, взаимодействующих с окружающей средой и платой ***Arduino***.

3.8. Использование переменных резисторов для создания собственных аналоговых датчиков

Благодаря достижениям в области физики, мы имеем множество материалов, способных изменять сопротивление в результате физического воздействия. Например, проводящие краски изменяют свое сопротивление при изгибе и скручивании, полупроводники меняют сопротивление под действием света (фоторезисторы), сопротивление некоторых материалов зависит от нагрева и охлаждения (термисторы). Это всего лишь несколько примеров, которые позволят создать свои собственные аналоговые датчики.

Поскольку упомянутые датчики меняют сопротивление, а не напряжение, в схеме потребуется создать делитель напряжения, чтобы можно было измерить изменение сопротивления.

3.9. Резистивный делитель напряжения

Резистивный делитель напряжения состоит из двух резисторов, от соотношения сопротивлений которых зависит выходное напряжение. Так, если один из резисторов переменный, то на выходе

можно получить изменение напряжения. Другой резистор определяет чувствительность схемы, если это подстроечный резистор, то чувствительность можно корректировать.

Рассмотрим нерегулируемый резистивный делитель (Рисунок 29) и напряжение на его выходе. Обозначение A0 на Рисунок 29— это аналоговый вход A0 на плате ***Arduino***. Зависимость выходного напряжения делителя от входного:

$$U_{вых} = U_{вх}(R2/(R1 + R2)).$$

В нашем случае на вход делителя подано напряжение 5 В, а выход подключен к аналоговому контакту A0 платы ***Arduino***. Если R1 и R2 одинаковы (как, например, 10 кОм), то 5 В делится пополам, и на аналоговом входе будет 2,5 В. Проверьте это, подставив значения в формулу:

$U_{вых}$ = 5 В (10 кОм / (10 кОм + 10 кОм)) = 2,5 В.

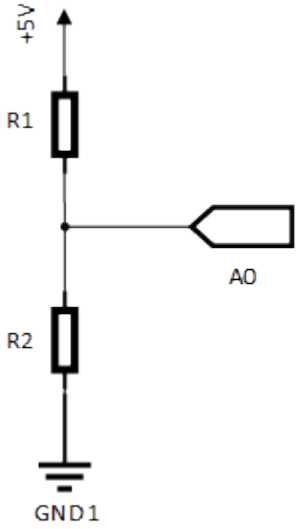

Рисунок 29. Простой делитель напряжения

Рисунок 30. Фоторезистор

Теперь предположим, что один из этих резисторов переменный, например, фоторезистор (Рисунок 30). Сопротивление фоторезистора зависит от интенсивности падающего на него света. Я использовал фоторезистор с номинальным сопротивлением 200 кОм. В полной темноте его сопротивление около 200 кОм, при ярком свете оно падает почти до нуля. От того, какой резистор (Rl или R2) поменять на фоторезистор, и от номинала постоянного резистора будет зависеть масштаб и точность показаний. Попробуйте поэкспериментировать с различными конфигурациями и посмотрите через монитор последовательного порта, как меняются показания. В качестве примера заменим R1 на фоторезистор, а R2 возьмем постоянным с номиналом 10 кОм (Рисунок 31). Для данного упражнения можно оставить на плате RGB-светодиод и подключить его как одноцветный.

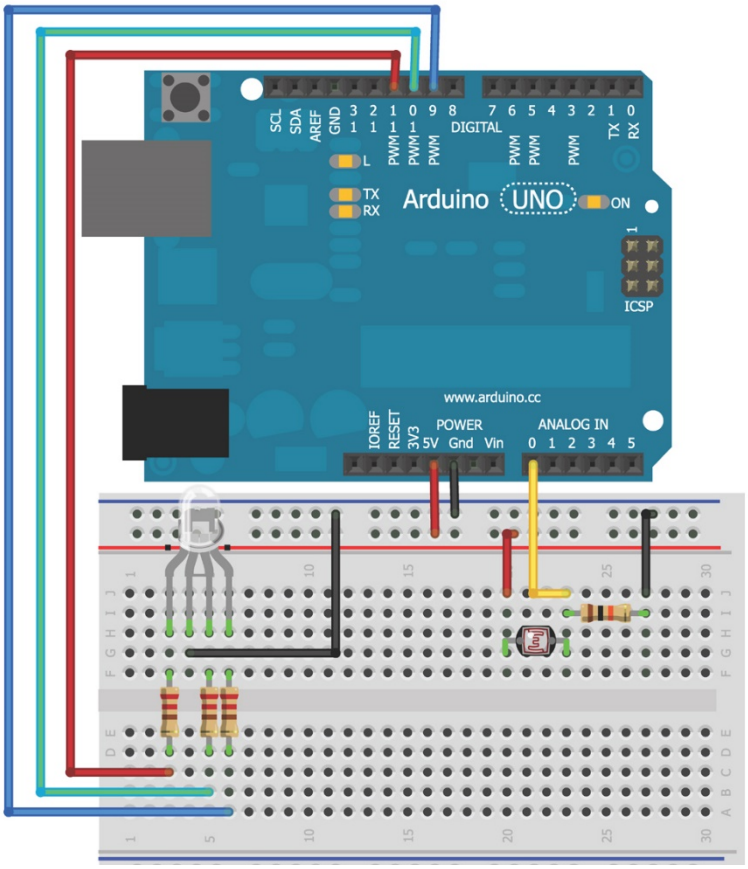

Рисунок 31. Подключение фоторезистора

Загрузите программу считывания аналоговых данных и выдачи результата в последовательный порт (см. листинг 3.1) и поменяйте освещенность фоторезистора. Вы не сможете получить весь диапазон значений от 0 до 1023, потому что у

фоторезистора никогда не будет нулевого сопротивления. В результате вы определите минимальное и максимальное значения напряжения на выходе. Эти данные потребуются, чтобы сделать "интеллектуальный" ночник, который будет светить более ярко в темном помещении, и наоборот. Выберите аналоговые значения для вашей комнаты, соответствующие темноте и максимальной освещенности. У меня это были значения 200 (темнота) и 900 (максимальное освещение). У вас могут быть другие цифры. Они зависят от условий освещения, значения резистора R2 и характеристик фоторезистора.

3.10. Управление аналоговыми выходами по сигналу от аналоговых входов

Напомним, что функция `analogwrite()` позволяет изменять яркость светодиода. Но не забывайте, аргумент этой функции 8-разрядный, т. е. находится в диапазоне от 0 до 255, в то время как АЦП выдает значения от 0 до 1023. В языке программирования ***Arduino*** есть удобные функции для пропорционального преобразования значений от одного диапазона к другому: *`map()`* и *`constrain()`*. Синтаксис функции *`map()`* выглядит следующим образом:

`output = map(value, fromLow, fromHigh, toLow, toHigh).`

Здесь `value`— преобразуемое значение (напряжение на аналоговом входе). `fromLow` и `fromHigh` — это нижняя и верхняя границы текущего диапазона. В нашем примере это минимальная и максимальная освещенность в помещении (200 и 900). `toLow` и `toHigh`— нижняя и верхняя границы нового диапазона. Аргумент функции `anaiogWrite()` должен быть в диапазоне от 0 до 255. Но мы хотим меньшей освещенности сопоставить большую яркость светодиода, т. е. минимальным значениям на аналоговом входе должны соответствовать максимальные значения на выводах светодиода. Удобно, что функция `map()` делает это автоматически. Функция `map()` осуществляет линейное отображение. Например, если `fromLow` и `fromHigh` равны 200 и 900, соответственно, а `toLow` и `toHigh` равны 255 и 0, то 550 превратится в 127, потому что 550 находится посередине между 200 и 900, а 127 посередине между 255 и 0. Следует учесть, что функция `map()` не ограничивает значения, если они выходят за границы диапазона. Если `value` окажется меньше 200 (для нашего примера), то `output` будет больше 255. Это неудобно, т. к. передать функции `analogWrite()` значение, превышающее 255, нельзя. Для ограничения значений есть функция `constrain()`, синтаксис которой выглядит следующим образом:

`output = constrain(value, min, max).`

При передаче значения из функции `map()` в функцию `constrain()` можно установить аргумент `min` равным 0 и `max` — 255, тогда величины, выходящие за рамки этого диапазона, будут ограничены. Теперь все готово, чтобы написать программу управляемого ночника. Посмотрим, как будет выглядеть окончательно наш проект (Листинг 9).

Листинг 9. Программа управляемого ночника — nightlight.ino

```
//Автоматический ночник
const int RLED=9;          // Контакт 9 для ШИМ-вывода RED RGB-светодиода
const int LIGHT=0;         // Контакт A0 для входа фоторезистора
const int MIN_LIGHT=200;   //Нижний порог освещенности
const int MAX_LIGHT=900;   //Верхний порог освещенности
int val =0;                // Переменная для сохранения считанного
                           // аналогового значения
void setup()
{
pinMode(RLED, OUTPUT);     // Сконфигурировать RED-контакт светодиода
                           //как выход
}
void loop()
{
val = analogRead(LIGHT);                    //Чтение показаний фоторезистора
val = map(val, MIN_LIGHT, MAX_LIGHT, 255, 0);   // Вызов функции map()
val = constrain(val, 0, 255);                   // Ограничение границ
analogWrite(RLED, val);                         // Управление светодиодом
```

```
}
```

Обратите внимание, что в листинге переменная `val` используется повторно. В принципе, можно задать и другую переменную. В таких функциях, как `map()`, предыдущее значение переменной `val` служит в качестве аргумента и после завершения выполнения функции перезаписывается заново.
Загрузите программу в плату ***Arduino*** и посмотрите, работает ли ночник, как ожидалось. Чувствительность ночника можно отрегулировать, подобрав минимальную и максимальную границы комфортного диапазона с помощью монитора последовательного порта. Подумайте, как можно реализовать в этой программе выбор цвета ночника, воспользовавшись сведениями из предыдущей главы. Попробуйте добавить кнопку для выбора цвета светодиода и фоторезистор для регулировки яркости каждого цвета.

Резюме

В этой главе вы узнали следующее:

- Чем отличаются аналоговые сигналы от цифровых.
- Как преобразовать аналоговые сигналы в цифровые.
- Как считать аналоговый сигнал с потенциометра.
- Как вывести на экран данные, используя монитор последовательного порта.
- Как взаимодействовать через интерфейс с аналоговыми датчиками.
- Как создать собственные аналоговые датчики.
- Как ограничить значения для управления аналоговыми выходами.

Часть 2. Управление окружающей средой

Глава 4. Использование транзисторов и управляемых двигателей

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;
- USB-кабель;
- батарея 9 В;
- разъем для батареи 9 В;
- стабилизатор напряжения L4940V5;
- электролитический конденсатор 22 мкФ;
- электролитический конденсатор 0,1 мкФ;
- керамический конденсатор 1 мкФ;
- 4 синих светодиода;
- 4 резистора номиналом 1 кОм;
- биполярный *n-p-n* транзистор PN2222;
- диод 1N4004;
- перемычки;
- провода;
- ИК-датчик расстояния Sharp GP2Y0A41SK0F ИК с кабелем;
- стандартный серводвигатель;
- двигатель постоянного тока;
- макетная плата;
- потенциометр;
- драйвер двигателя SN754410.

Электронные ресурсы к главе

На странице https://www.exploringarduino.com/content/ch4/ можно загрузить код программ, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы https://www.wiley.com/en-ru/Exploring+Arduino%3A+Tools+and+Techniques+for+Engineering+Wizardry-p-9781118549360 в разделе Downloads.

Что вы узнаете в этой главе

Теперь вы уже можете получать информацию из окружающей среды. Но как управлять этим миром? Мигание светодиода и автоматическая регулировка яркости ночника уже неплохо, но вы можете сделать гораздо больше. Двигатели и приводы, а также транзисторы позволят осуществлять с помощью ***Arduino*** реальные физические действия. Соединяя двигатели с платой ***Arduino***, можно управлять роботами, создавать механические манипуляторы, перемещать датчики и делать многое другое. В этой главе вы узнаете, как запускать двигатели постоянного тока, как работать с транзисторами и управлять серводвигателями. Освоив это, вы сможете создать датчик расстояния, способный определять расположение близлежащих объектов. Этот датчик идеально подходит, например, для установки на автономном движущемся роботе. По завершении главы вы приобретете навыки, достаточные для разработки по-настоящему интерактивного устройства.

ПРИМЕЧАНИЕ

Если вы хотите узнать больше о двигателях и транзисторах, смотрите видеофильм, расположенный на интернет-странице https://www.jeremyblum.com/2011/01/31/arduino-tutorial-5-motors-and-transistors/ [1]

ВНИМАНИЕ!

Для питания двигателей постоянного тока потребуется батарея 9 В, т. к. для работы электродвигателя требуется больше мощности, чем может выдать плата ***Arduino****. Это напряжение не опасно, но при неправильном подключении можно повредить электронные компоненты. При повторении примеров тщательно проверяйте схемы и следуйте*

[1] На русском: http://wiki.amperka.ru/видеоуроки:5-моторы-и-транзисторы.

инструкциям. Избегайте коротких замыканий, не пытайтесь соединить два источника напряжения друг с другом. Следите за тем, чтобы источники напряжения 5 и 9 В не оказались подключены к одной шине питания макетной платы.

4.1. Двигатели постоянного тока

Вал двигателя постоянного тока вращается при подаче постоянного напряжения на его контакты. Подобные двигатели можно встретить во многих бытовых приборах, например, в радиоуправляемых автомобилях, в приводе DVD-плеера. Такие двигатели бывают разного размера и обычно стоят недорого. Регулируя напряжение, подаваемое на двигатель, можно менять скорость его вращения. Переключая полярность приложенного напряжения, можно изменять направление вращения. Это делают, используя H-мост, о котором вы узнаете далее в этой главе.
Щеточные двигатели постоянного тока состоят из неподвижных магнитов (статора) и вращающейся обмотки (ротора). Электроэнергию подводят через контакты "щетки", поэтому двигатели называются щеточными. В отличие от электродвигателей постоянного тока других типов (таких, например, как шаговые двигатели), щеточные электродвигатели дешевле и скорость вращения легко регулировать. Однако их срок службы невелик, потому что щетки со временем изнашиваются.

4.2. Борьба с выбросами напряжения

Двигатели постоянного тока обычно требуют ток больше, чем может выдать встроенный в ***Arduino*** блок питания, к тому же они могут создавать опасные выбросы напряжения. Для решения этой проблемы необходимо научиться эффективно изолировать двигатель постоянного тока от платы ***Arduino*** и подключать его к отдельному источнику питания. Транзистор позволит безопасно включать двигатель, а также управлять его скоростью с помощью методов ШИМ, рассмотренных в главе 2. Прежде чем собирать схему подключения двигателя постоянного тока, изображенную на Рисунок 32, рассмотрим основные компоненты схемы:

- Q1 — *n-p-n* биполярный плоскостной транзистор действует как ключ, включая и выключая внешний источник питания 9 В. Существуют два типа биполярных плоскостных транзисторов: *n-p-n* и *p-n-p*. Мы будем применять транзисторы типа *n-p-n*. Говоря упрощенно, *n-p-n* транзистор представляет собой переключатель, управляемый напряжением, что позволяет подавать или отключать ток;
- R1 — резистор номиналом 1 кОм, соединяющий контакт платы ***Arduino*** с базой транзистора;

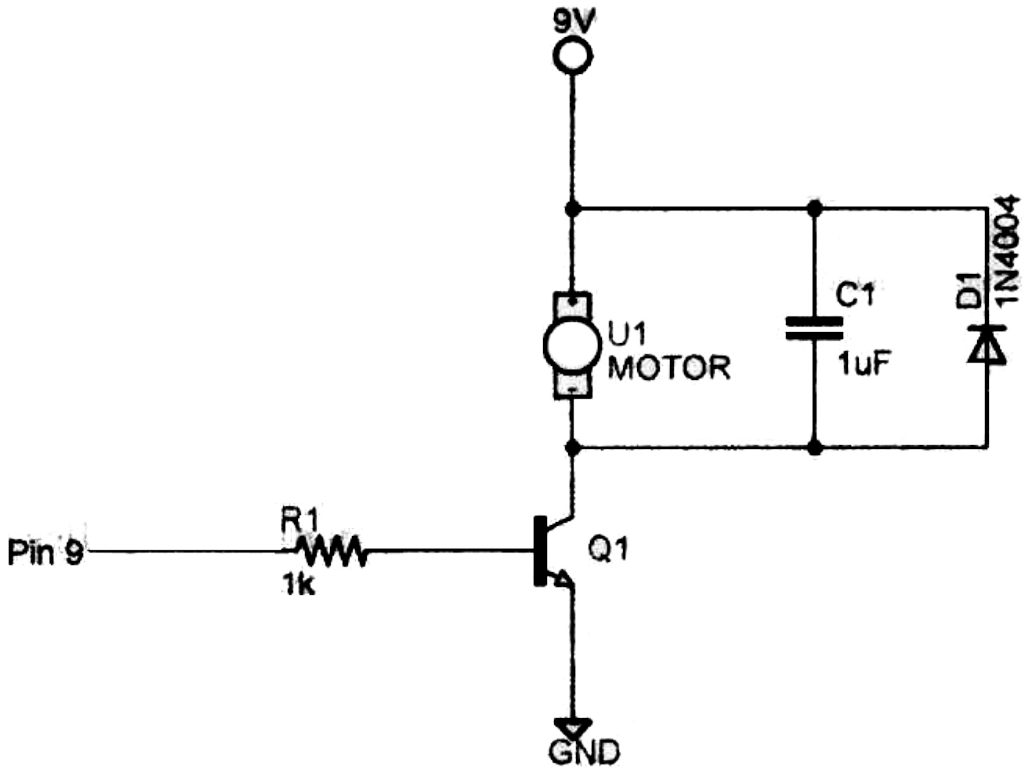

Рисунок 32. Схема включения двигателя постоянного тока

- U1 — двигатель постоянного тока;
- C1 — конденсатор для фильтрации помех, вызванных работой двигателя;
- D1 — диод для защиты блока питания от обратного напряжения.

4.3. Использование транзистора в качестве переключателя

Транзисторы применяются во многих устройствах: от усилителей до компонентов центрального процессора в компьютерах и смартфонах. У нас транзистор будет работать в качестве простого электрически управляемого переключателя. Каждый биполярный транзистор имеет три контакта (Рисунок 33): эмиттер (Е), коллектор (К) и базу (Б).

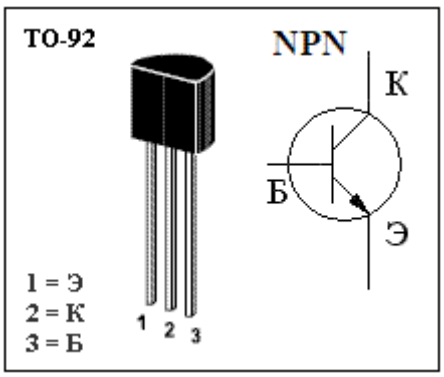

Рисунок 33. Биполярный n-p-n транзистор

Между коллектором и эмиттером течет большой ток, величина которого зависит от малого тока базы. Изменяя ток базы, мы можем регулировать ток через транзистор и менять скорость вращения двигателя. Напряжения 5 В, подаваемого на выход ***Arduino***, достаточно для включения транзистора. Используя ШИМ, можно управлять скоростью вращения двигателя. Поскольку механические детали двигателя обладают инерцией, быстрое переключение транзистора под действием ШИМ-сигнала с разной скважностью приведет к плавной регулировке скорости вращения.

4.4. Назначение защитных диодов

Одна из проблем электродвигателей постоянного тока — наличие противо-ЭДС. В двигателе есть обмотки, в которых создается магнитный поток. При работе двигателя энергия магнитного поля запасается в обмотках. При выключении электродвигателя на концах обмотки возникает выброс напряжения обратной полярности, опасный для источника питания. Предотвратить воздействие электрических выбросов на внешние цепи можно с помощью защитных диодов. Подключив защитный диод, можно быть уверенным, что он устранит выброс напряжения при выключении двигателя.

4.5. Назначение отдельного источника питания

В схеме, изображенной на Рисунок 32, двигатель подключен к отдельному источнику напряжением 9 В, а не к контакту 5 В разъема USB. Для данного примера подойдет также внешний источник с напряжением 5 В. Внешний источник питания необходим по двум причинам:

- уменьшается вероятность повреждения платы ***Arduino*** при неправильном подключении электродвигателя;
- ток и напряжение могут быть больше, чем обеспечивает встроенный в ***Arduino*** источник питания.

Некоторые двигатели постоянного тока потребляют ток, больший, чем может выдать плата ***Arduino***. Кроме того, рабочее напряжение многих двигателей превышает 5 В. Хотя они и будут вращаться при напряжении 5-вольтовом питании, но достичь заданной скорости вращения могут только при питании 9 или 12 В (в зависимости от технических характеристик двигателя).

ВНИМАНИЕ!

*Обратите внимание, что необходимо соединить землю отдельного источника питания с землей **Arduino**. Это обеспечит общую точку между уровнями напряжения в двух частях схемы.*

4.6. Подключение двигателя

Теперь, когда мы рассмотрели все тонкости управления щеточным двигателем постоянного тока,

установим его макетную плату и подключим. Соберите схему, изображенную на рис. 4.1, а затем проверьте правильность монтажа по рис. 4.3. Важно научиться хорошо читать электрические схемы без использования графического макета.

Перед включением питания проверьте следующее:

- убедитесь, что земля от батареи 9 В соединена с землей платы ***Arduino***, для этого можно общую шину земли на макетной плате, как показано на рис. 4.3;
- убедитесь, что провод питания +9 В не подключен к проводу питания +5 В;
- убедитесь, что транзистор подключен правильно;
- убедитесь, что диод включен правильно; конденсатор керамический, для него полярность не имеет значения.

Пришло время заставить двигатель вращаться. На вал двигателя можно прикрепить кусочек клейкой ленты, чтобы оценить скорость вращения. Перед написанием программы необходимо проверить работу схемы. Подсоединим батарею, подадим питание на плату ***Arduino*** через USB-кабель, подключим базу транзистора к выводу +5 В, это имитирует высокий логический уровень на выходном контакте ***Arduino***. Вал двигателя должен начать вращаться. При подключении базы транзистора к земле двигатель останавливается. Если это не так, проверьте правильность монтажа. Если все работает как описано, переходим к следующему шагу — программированию.

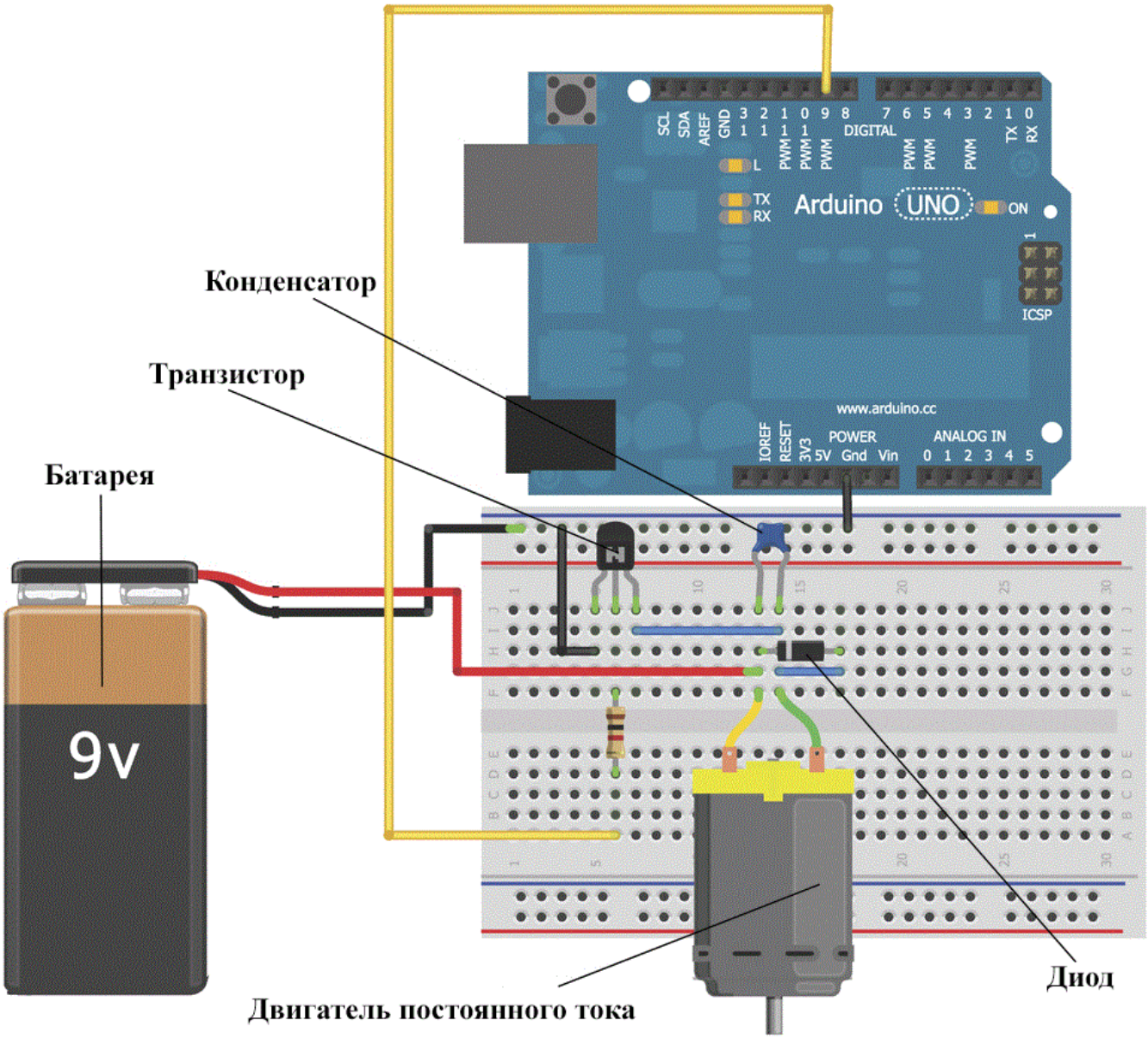

Рисунок 34. Подключение двигателя постоянного тока

4.7. Управление скоростью вращения двигателя с помощью ШИМ

Программа для управления скоростью вращения двигателя (Листинг 10) будет похожа на программу регулировки яркости светодиодов ночника из *Глава 3*. Опрос аналоговых датчиков. Появление на выходе платы ***Arduino*** ШИМ-сигнала вызывает быстрый запуск и остановку двигателя с разным периодом, что эквивалентно изменению скорости вращения.

Листинг 10. Автоматическое управление скоростью двигателя — motor.ino

```
// Пример управления скоростью вращения двигателя
const int M0T0R=9;    // Вывод 9 Arduino для подключения двигателя
void setup()
{
      pinMode (MOTOR, OUTPUT);
}
void loop()
{
      for (int i=0; i<256; i++)
      {
            analogWrite(MOTOR, i);
            delay(10);
      }
      delay(2000);
      for (int i=255; i>=0; i--)
      {
            analogWrite(MOTOR, i);
            delay(10);
      }
      delay(2000) ;
}
```

Если все собрано правильно, то после запуска программы скорость вращения двигателя сначала плавно увеличится, а затем уменьшится. На основе описанного проекта можно сконструировать, например, макет движущегося робота.

Воспользуемся нашими знаниями аналоговых датчиков и попробуем вручную управлять скоростью вращения вала двигателя с помощью потенциометра. Подсоединим потенциометр к аналоговому входу A0, как показано на Рисунок 35. Обратите внимание, что контакт ***Arduino*** 5 В необходимо соединить с шиной питания на макетной плате.

Теперь изменим программу так, чтобы управлять скоростью вращения двигателя, регулируя положение ручки потенциометра. При полностью выведенном движке потенциометра двигатель остановлен, при полностью введенном — вал двигателя вращается с максимальной скоростью. Не забывайте, что контроллер ***Arduino*** работает очень быстро, цикл повторяется несколько тысяч раз в секунду! Поэтому малейшее изменение положения движка потенциометра сразу же сказывается на частоте вращения двигателя. Описанный алгоритм реализует программный код, приведенный в листинге 4.2. Загрузите программу в плату ***Arduino*** и регулируйте скорость вращения двигателя с помощью потенциометра.

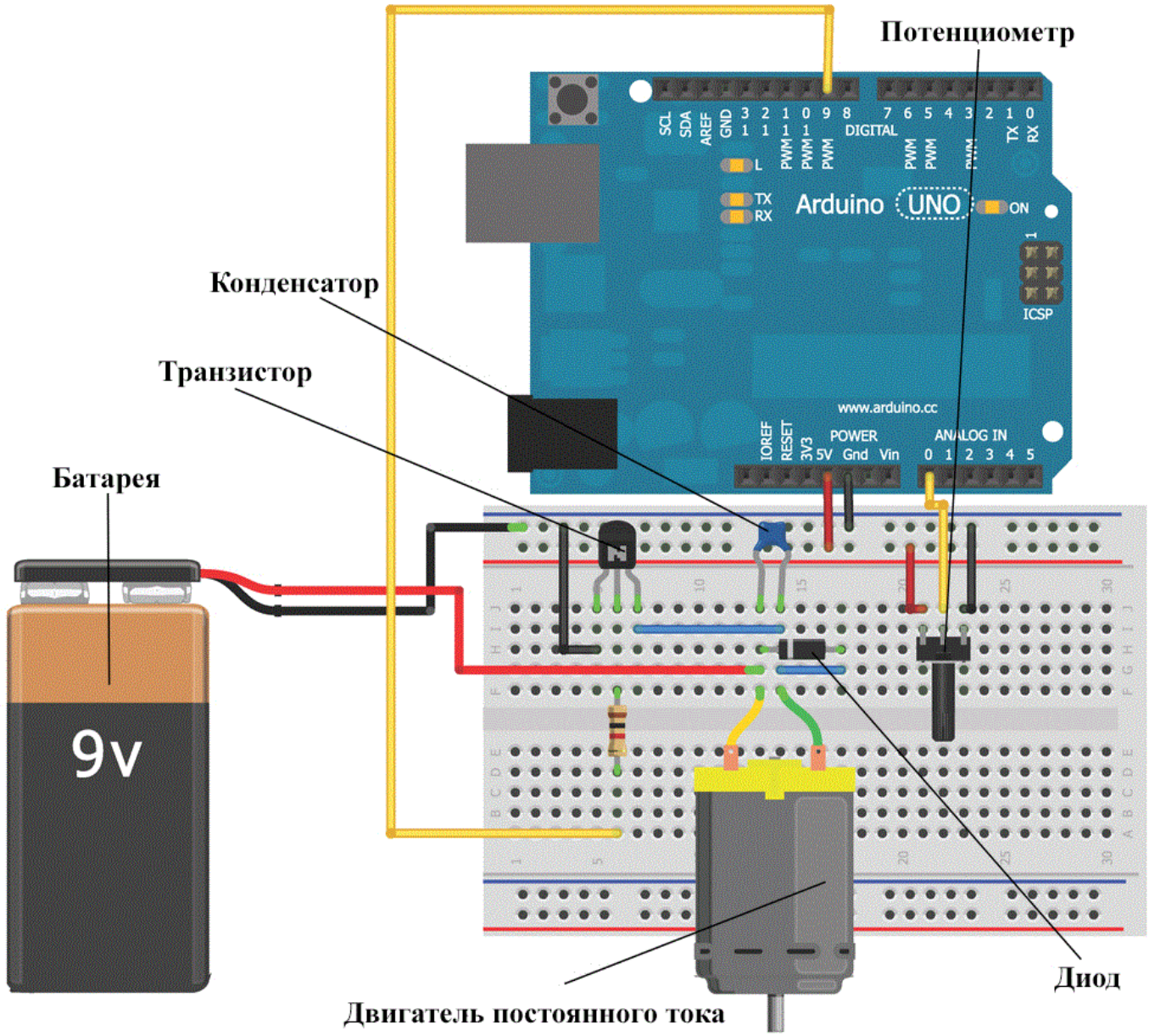

Рисунок 35. Схема подключения потенциометра для регулировки скорости вращения

Листинг 11. Регулирование скорости двигателя — motor_pot.ino

```
const int M0T0R=9;    // Вывод 9 Arduino для подключения двигателя
const int POT=0;      // Вывод A0 Arduino для подключения потенциометра
int val = 0;
void setup()
{
pinMode (MOTOR, OUTPUT);
}
void loop()
{
val = analogRead(POT);
val = map(val, 0, 1023, 0, 255);
analogWrite(MOTOR, val)
}
```

4.8. Управление направлением вращения двигателя постоянного тока с помощью H-моста

Вы научились изменять скорость вращения двигателя постоянного тока. Это позволит управлять

движением робота, но лишь в том случае, если он будет двигаться только вперед. Однако любой двигатель постоянного тока способен вращаться в двух направлениях. Для изменения направления вращения применим устройство, называемое Н-мостом. Принцип работы Н-моста поясняет схема, изображенная на Рисунок 36.

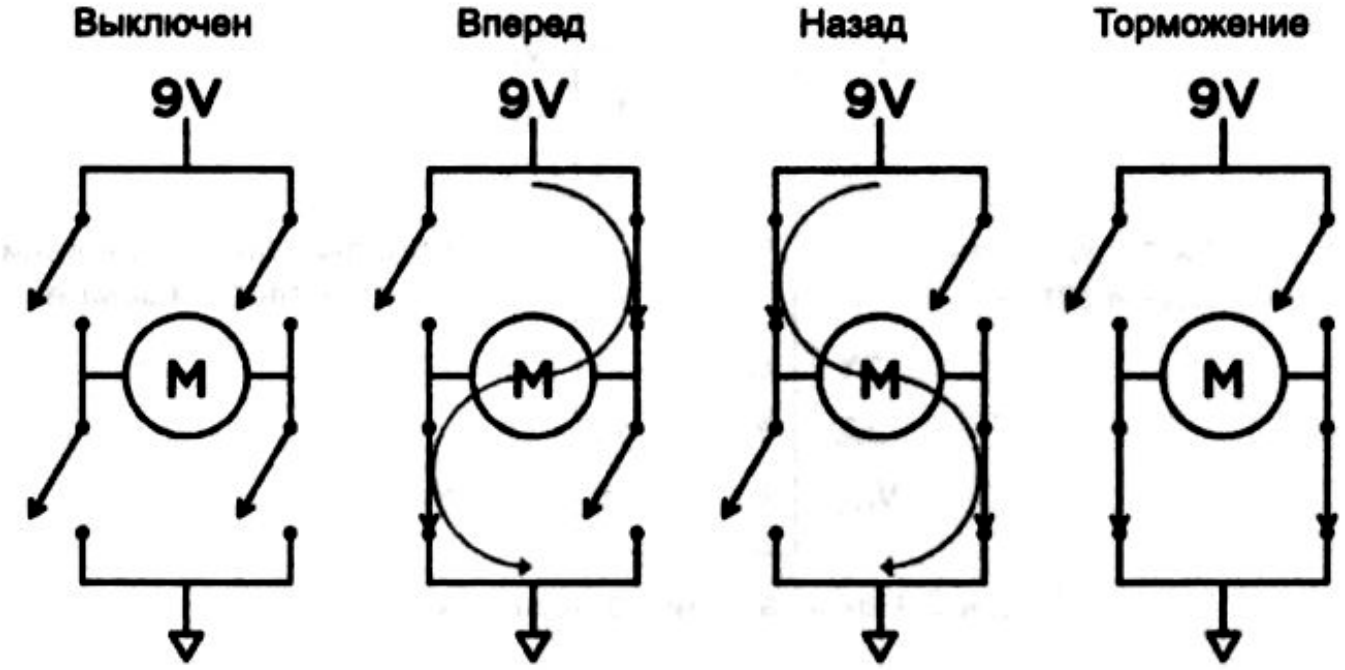

Рисунок 36. Схема работы Н-моста

Почему схема называется Н-мостом? Ответ прост. Обратите внимание, что изображение двигателя в сочетании с четырьмя переключателями похоже на прописную букву «Н». Хотя на схеме изображены просто выключатели, на самом деле это транзисторы, подобные тем, которые были в предыдущем примере. В реальной схеме Н-моста также есть некоторые дополнительные цепи, в том числе защитные диоды.

Н-мост может находиться в четырех основных состояниях: «выключен», «торможение», «вперед» и «назад». В выключенном состоянии все выключатели разомкнуты и двигатель не вращается. В состоянии «вперед» два выключателя замкнуты, в результате через обмотку двигателя течет ток и вал вращается. В состоянии «назад» ток течет в противоположном направлении, и направление вращения вала обратное. Если Н-мост находится в состоянии торможения, то обмотка замкнута, вращение замедляется и двигатель останавливается.

Необходимо помнить об опасности короткого замыкания цепей Н-моста. Что произойдет, если все четыре выключателя будут замкнуты? Это вызовет короткое замыкание между шиной +9 В и землей. В результате батарея очень быстро нагреется, что может привести к ее разрыву. Кроме того, короткое замыкание может повредить Н-мост или другие элементы схемы.

Для нашего эксперимента выберем микросхему SN754410 — четырехканальный драйвер Н-полумоста, которая имеет встроенную защиту от короткого замыкания.

4.9. Сборка схемы Н-моста

Настала пора собрать схему Н-моста. Возьмем микросхему SN754410— четырехканальный драйвер Н-полумоста. Два драйвера Н-полумоста образуют полный драйвер Н-моста, как на Рисунок 36. Для управления одним электродвигателем постоянного тока используются два из четырех драйверов Н-полумоста. Если вы хотите сделать, например, радиоуправляемый автомобиль, можно управлять двумя колесами с помощью одной микросхемы SN754410. На Рисунок 37 приведена цоколевка микросхемы SN754410.

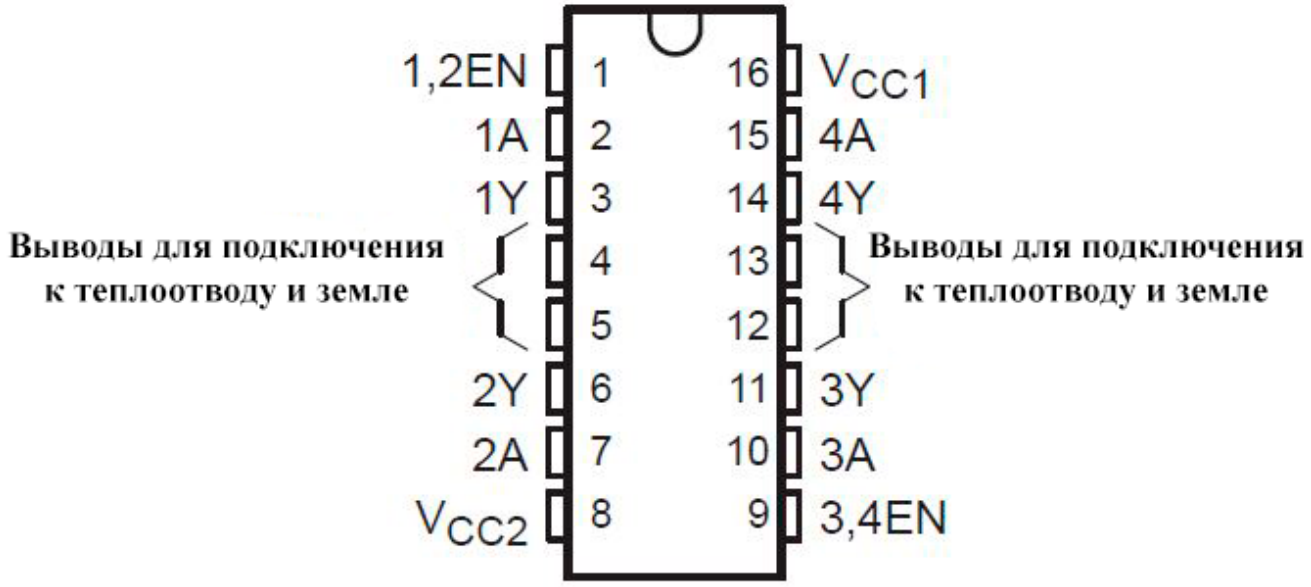

Рисунок 37. Цоколевка микросхемы SN754410

Нумерация контактов на микросхемах начинается с левого верхнего угла и идет против часовой стрелки. На корпусе всегда имеется метка у контакта 1 (полукруг, точка или что-то другое).
Соответствие состояний входов и выходов драйвера SN754410 иллюстрирует Таблица 1 (условные обозначения в таблице: H— высокий уровень; L— низкий уровень; X — безразличное состояние; Z — высокоимпедансное состояние).

Таблица 1. Состояния входов и выходов драйвера SN754410

Вход		Выход
A	**EN**	**Y**
И	И	И
L	H	L
X	L	Z

Рассмотрим назначение контактов микросхемы SN754410:

- GND (контакты 4, 5, 12, 13)— выводы для подключения к земляной шине монтажной платы;
- V_{CC2} (контакт 8) — напряжение питания двигателя (подсоедините к 9 В);
- V_{CCl} (контакт 16) — напряжение питания микросхемы (подсоедините к 5 В);
- 1Y и 2Y (контакты 3 и 6) — выходы для подключения первого двигателя;
- 1A и 2A (контакты 2 и 7) — коммутация первого двигателя, эти выводы соединены с управляющими контактами ***Arduino***;
- 1,2 EN (контакт 1)— включение и отключение левого драйвера. Данный вывод соединен с ШИМ-контактами на плате ***Arduino***, что позволяет динамически регулировать скорость двигателей;
- 3Y и 4Y (контакты 11 и 14) — выходы для подключения второго двигателя;
- 3A и 4A (контакты 10 и 15)— коммутация второго двигателя, эти выводы соединены с управляющими контактами ***Arduino***;
- 3,4 EN (контакт 9) — вывод включения или отключения правого драйвера. Он соединен с ШИМ-контактами на ***Arduino***, что позволяет динамически регулировать скорость двигателей.

Проверьте монтаж по Рисунок 38. Потенциометр подключим позже.
Прежде чем приступать к программированию, проверим работоспособность схемы. Подключите один из входных контактов (2 или 7) микросхемы Н-моста к шине 5 В, другой к земле. Двигатель начнет

вращаться. Поменяйте подключение контактов 2 и 7, двигатель будет вращаться в другую сторону.

ВНИМАНИЕ!

Во время переключения контактов отключите батарею, чтобы случайно не вызвать короткое замыкание моста.

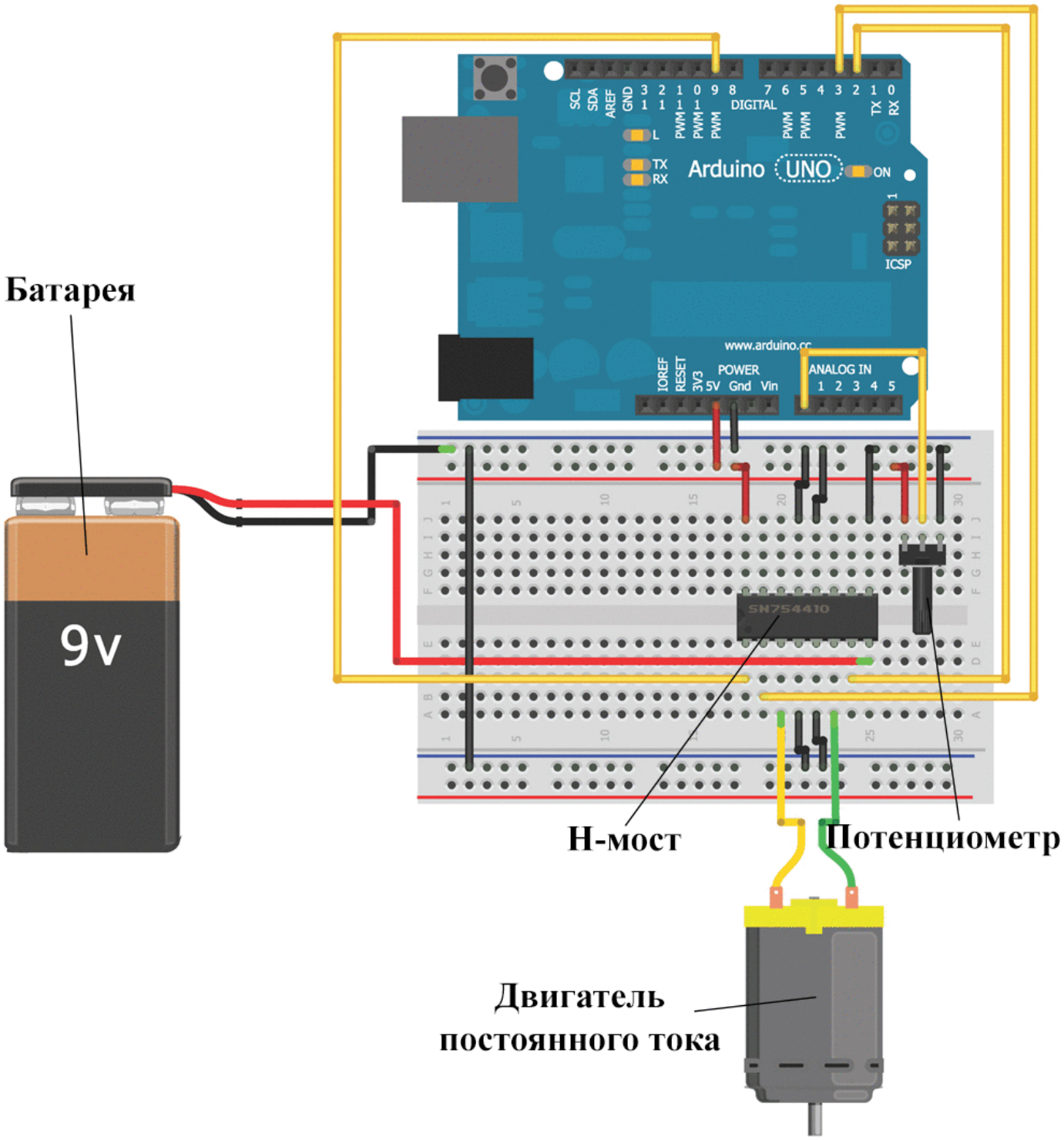

Рисунок 38. Схема подключения Н-моста

4.10. Управление работой Н-моста

Напишем программу для управления скоростью и направлением вращения двигателя с помощью потенциометра и драйвера Н-моста. Установка движка потенциометра в среднее положение приводит к остановке двигателя, при перемещении движка вправо скорость вращения вала двигателя увеличивается, перемещение движка влево от среднего положения приводит к увеличению скорости вращения вала двигателя в обратном направлении. В программе будут три вспомогательные функции: первая — для остановки двигателя, вторая — для вращения двигателя с заданной скоростью и третья — для вращения двигателя с заданной скоростью в обратном направлении.

Анализируя Рисунок 36, делаем следующие выводы:

1. Для вращения двигателя один из выключателей должен быть замкнут, другой разомкнут.
2. Чтобы двигатель вращался в обратном направлении, замкнутый в п. 1 выключатель должен быть разомкнут, а разомкнутый — замкнут.
3. Для остановки двигателя оба выключателя должны быть разомкнуты.

ПРИМЕЧАНИЕ

Перед изменением состояния выключателей всегда отключайте ток, чтобы не вызвать короткого замыкания Н-моста.

Сначала напишем код функций для выполнения описанных действий (Листинг 12).

Листинг 12. Вспомогательные функции для управления двигателем

```
// Вращение двигателя вперед с заданной скоростью (диапазон 0-255)
void forward(int rate)
{
      digitalWrite(EN, LOW);
      digitalWrite(MC1, HIGH);
      digitalWrite(MC2, LOW);
      analogWrite(EN, rate);
}
// Вращение двигателя в обратном направлении с заданной скоростью
//(диапазон 0-255)
void reverse(int rate)
{
      digitalWrite(EN, LOW);
      digitalWrite(MC1, LOW);
      digitalWrite(MC2, HIGH);
      analogWrite(EN, rate);
}
// Остановка двигателя
void brake()
{
      digitalWrite(EN, LOW);
      digitalWrite(MC1, LOW);
      digitalWrite(MC2, LOW);
      digitalWrite(EN, HIGH)
}
```

Обратите внимание, что в начале каждой функции на контакте `EN` всегда устанавливается низкий уровень, и затем задаются значения на входах блока управления `MC1` и `MC2`. После установки значений на входах `MC1` и `MC2` можно снова включить ток. Подавая сигнал ШИМ на вход `EN`, можно управлять скоростью двигателя. Значение переменной `rate` должно быть в диапазоне от 0 до 255. Основной цикл программы (Листинг 13) считывает данные с потенциометра и в зависимости от результата вызывает требуемую функцию.

Листинг 13. Программа вызова вспомогательных функций

```
void loop()
{
      val = analogRead(POT);
      // Движение вперед
      if (val > 562)
      {
            velocity = map(val, 563, 1023, 0, 255);
            forward(velocity);
      }
      // Движение назад
      else if (val < 462)
      {
            velocity = map(val, 461, 0, 0, 255);
            reverse(velocity);
      }
      // Остановка
      else
      {
            brake();
      }
}
```

Сигнал с аналогового входа преобразуется в цифровое значение в диапазоне от 0 до 1023. Чтобы лучше понять принцип управления, обратимся к Рисунок 39.

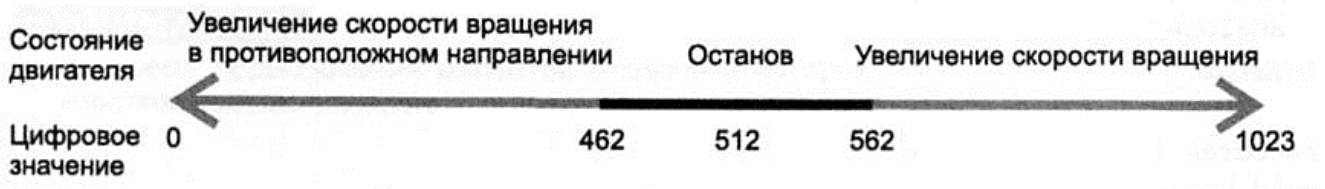

Рисунок 39. Принцип управления двигателем.

При значениях сигнала от потенциометра в диапазоне от 462 до 562 (100 отсчетов в районе средней точки) вызывается функция `break()` для остановки двигателя, при значениях от 562 до 1023 — функция запуска двигателя в прямом направлении `forward()`, при значениях от 0 до 462 — функция запуска двигателя в обратном направлении `reverse()`. Функция `map()` знакома вам из предыдущей главы. При определении обратной скорости значению потенциометра 461 соответствует значение скорости 0, а значению потенциометра 0 соответствует значение скорости 255. Функция `map()` инвертирует значения так, что на вход они подаются в обратном порядке. Объединив цикл `loop()` со вспомогательными функциями и начальной установкой `setup()`, получим полный код программы управления скоростью и направлением движения двигателя с помощью потенциометра (Листинг 14).

Листинг 14. Программа управления двигателем с помощью потенциометра

```
// Управление двигателем с помощью H-моста
const int EN=9;       //Вход включения двигателя EN
const int MC1=3;      //Вход 1 управления двигателем
const int MC2=2;      //Вход 2 управления двигателем
const int POT=0;      //Аналоговый вход 0 для подключения потенциометра

int val = 0;          //Переменная для хранения значения потенциометра
int velocity = 0;     //Переменная для хранения скорости двигателя (0-255)
```

```
void setup()
{
    pinMode(EN, OUTPUT);
    pinMode(MC1, OUTPUT);
    pinMode(MC2, OUTPUT);
    brake();          //Остановка двигателя при инициализации
}

void loop()
{
    val = analogRead(POT);

     //Движение вперед
    if (val > 562)
    {
        velocity = map(val, 563, 1023, 0, 255);
        forward(velocity);
    }

     //Движение назад
    else if (val < 462)
    {
        velocity = map(val, 461, 0, 0, 255);
        reverse(velocity);
    }

     //Остановка
    else
    {
        brake();
    }
}

     //Движение двигателя вперед с заданной скоростью (диапазон 0-255)
void forward (int rate)
{
    digitalWrite(EN, LOW);
    digitalWrite(MC1, HIGH);
    digitalWrite(MC2, LOW);
    analogWrite(EN, rate);
}

     //Движение двигателя в обратном направлении с заданной скоростью
     //(диапазон 0-255)
void reverse (int rate)
{
    digitalWrite(EN, LOW);
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, HIGH);
    analogWrite(EN, rate);
}

     //Остановка двигателя
void brake ()
```

```
{
    digitalWrite(EN, LOW);
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, LOW);
    digitalWrite(EN, HIGH);
}
```

Загрузите программу в плату ***Arduino*** и запустите на выполнение. Все работает, как ожидалось? Если нет, еще раз внимательно проверьте монтаж.
В качестве упражнения подключите к драйверу H-моста SN754410 второй двигатель постоянного тока и напишите программу управления двумя двигателями.

4.11. Управление серводвигателем

Двигатели постоянного тока прекрасно действуют в качестве моторов, но очень неудобны для точных работ, т. к. не имеют обратной связи. Другими словами, без какого-нибудь внешнего датчика нельзя узнать положение вала двигателя постоянного тока. Серводвигатели (или сервоприводы), напротив, отличаются тем, что с помощью команд можно установить их в определенное положение, в котором они будут находиться до поступления новых команд. Это важно, когда необходимо некоторую систему переместить в определенное положение. В этом разделе вы узнаете о серводвигателях и их управлении с помощью ***Arduino***.

4.11.1. Стандартные сервоприводы и сервоприводы вращения

Проще всего приобрести стандартные сервоприводы. Они имеют фиксированный диапазон углов поворота (обычно от 0 до 180°) и содержат соединенный с приводным валом потенциометр, который определяет угол поворота сервопривода. Управление сервоприводом происходит подачей прямоугольного импульса. Длительность импульса (в случае стандартного сервопривода) определяет угол поворота.
Если удалить потенциометр, получится сервопривод непрерывного вращения, который сможет вращаться постоянно, при этом длительность импульса определяет скорость вращения.
Далее будем использовать стандартные сервоприводы, вал которых поворачивается на определенный угол. При желании вы можете поэкспериментировать с сервоприводами непрерывного вращения либо удалив из стандартного потенциометр, либо купив готовый сервопривод постоянного вращения.

4.11.2. Принцип работы серводвигателя

В отличие от двигателей постоянного тока, серводвигатели имеют три контактных провода:

- питание (обычно красного цвета);
- земля (обычно коричневого или черного цвета);
- сигнальный вход (обычно белый или оранжевый).

Провода имеют цветовую маркировку и обычно расположены в том же порядке, как на Рисунок 40. Некоторые производители меняют порядок расположения проводов, поэтому перед применением сервопривода желательно ознакомиться с документацией.

Рисунок 40. Сервоприводы

Окраска проводов может отличаться, но приведенные цветовые сочетания наиболее распространены (обратитесь к документации конкретного сервопривода, если не уверены).

Как и двигатели постоянного тока, серводвигатели требуют для работы ток больше, чем выдает встроенный источник питания ***Arduino***. Хотя иногда удается запустить один-два сервопривода от блока питания платы ***Arduino***. У серводвигателей, в отличие от двигателей постоянного тока, есть дополнительный сигнальный провод для установки угла поворота вала. Питание и земляной провод серводвигателя нужно подсоединить к источнику постоянного напряжения.

Сервоприводы управляются по сигнальной линии с помощью прямоугольных импульсов регулируемой длительности. Для стандартного сервопривода подача импульса длительностью 1 мс приводит к установке сервопривода в положение 0°, импульс длительностью 2 мс устанавливает сервопривод в положение 180°, импульса 1,5 мс — 90°. После того как импульс подан, вал сервопривода устанавливается в определенную позицию и остается там до поступления следующей команды. Тем не менее, чтобы постоянно поддерживать точное положение вала сервопривода, необходимо отправлять сигнальные импульсы каждые 20 мс. Библиотека ***Arduino*** Servo, которую мы будем использовать для управления серводвигателями, позаботится об этом.

Чтобы лучше понять, как управлять серводвигателями, изучим графики, приведенные на Рисунок 41. В примерах, изображенных на Рисунок 41, импульс подается каждые 20 мс. Длительность импульса возрастает от 1 до 2 мс, при этом угол поворота серводвигателя (показанный справа от графика импульсов) увеличивается от 0 до 180°.

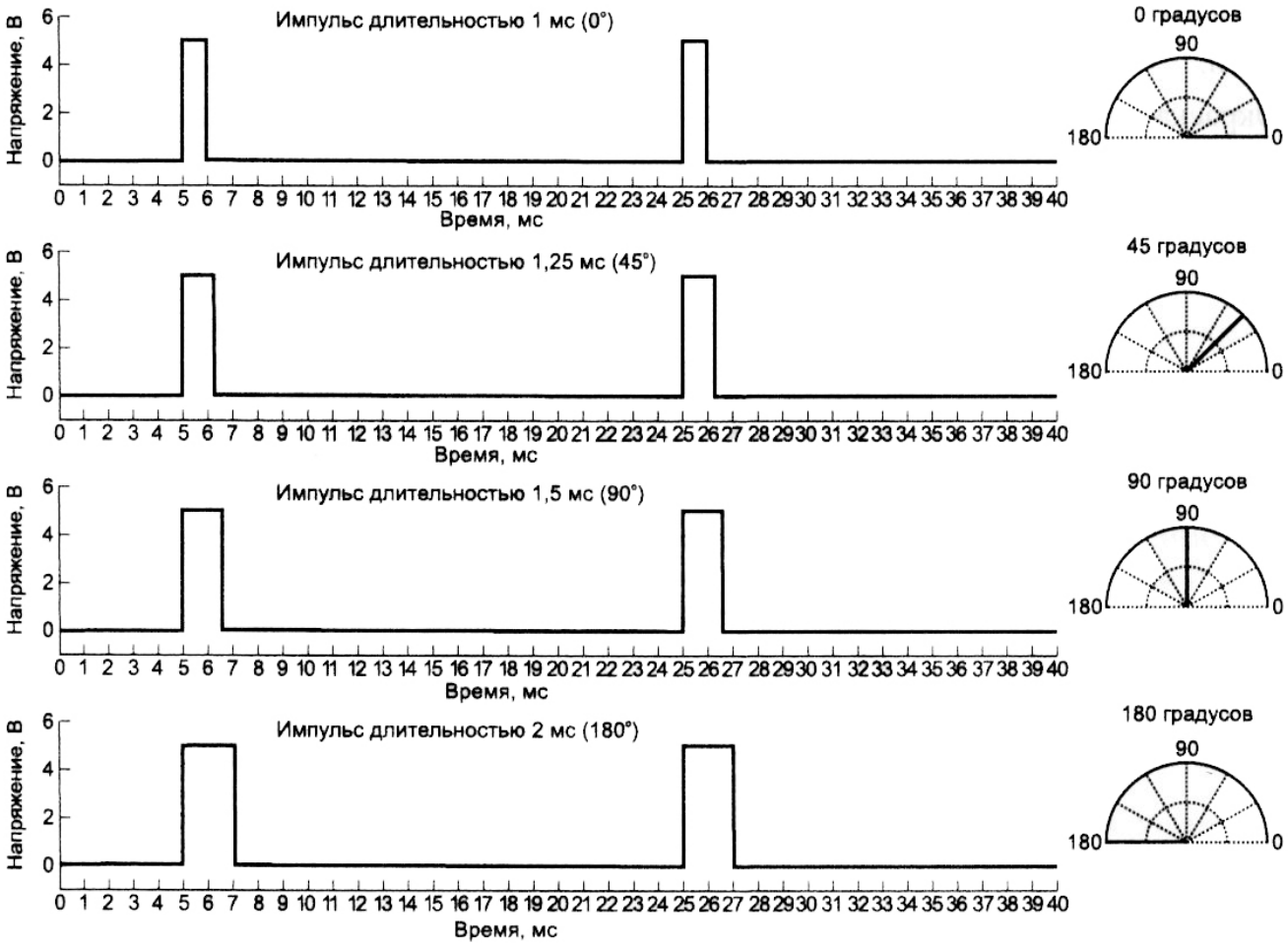

Рисунок 41. Временные диаграммы управления серводвигателем

Как упоминалось ранее, для работы серводвигателя требуется ток, больший, чем выдает встроенный в ***Arduino*** блок питания. Однако большинство серводвигателей работает от напряжения 5 В, а не 9 или 12 В как двигатели постоянного тока. Несмотря на это, необходим отдельный блок питания серводвигателя.

Мы рассмотрим, как с помощью источника 9 В и стабилизатора напряжения получить напряжение 5 В для питания сервопривода. Интегральный стабилизатор напряжения — чрезвычайно простое устройство, у которого обычно три контакта:

- вход;
- выход;
- заземление.

Заземляющий вывод соединен как с землей входного, так и с землей выходного напряжения. Для работы стабилизатора входное напряжение должно быть выше, чем выходное, причем величина выходного напряжения фиксирована в зависимости от типа стабилизатора.

Падение напряжения приводит к нагреву, поэтому необходимо позаботиться о теплоотводе (например, алюминиевом радиаторе). Для наших экспериментов возьмем 5-вольтовый стабилизатор напряжения L4940V5, который способен выдавать ток до 1,5 А. Схема включения стабилизатора приведена на Рисунок 42.

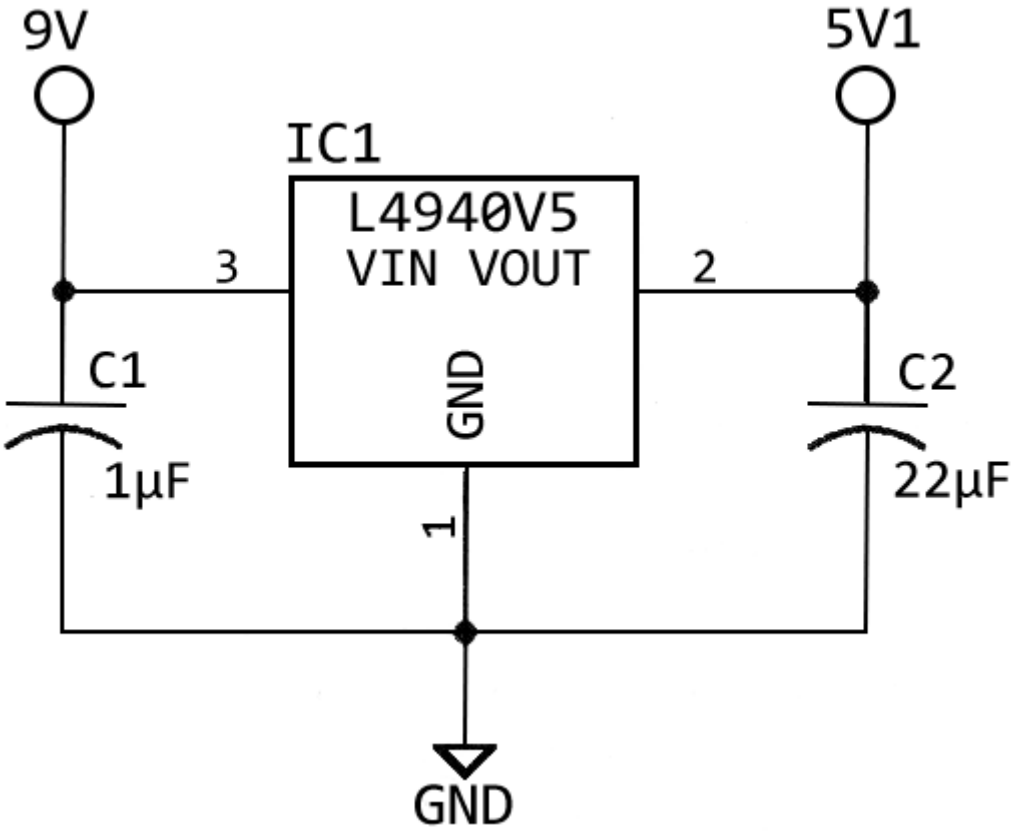

Рисунок 42. Схема включения стабилизатора напряжения

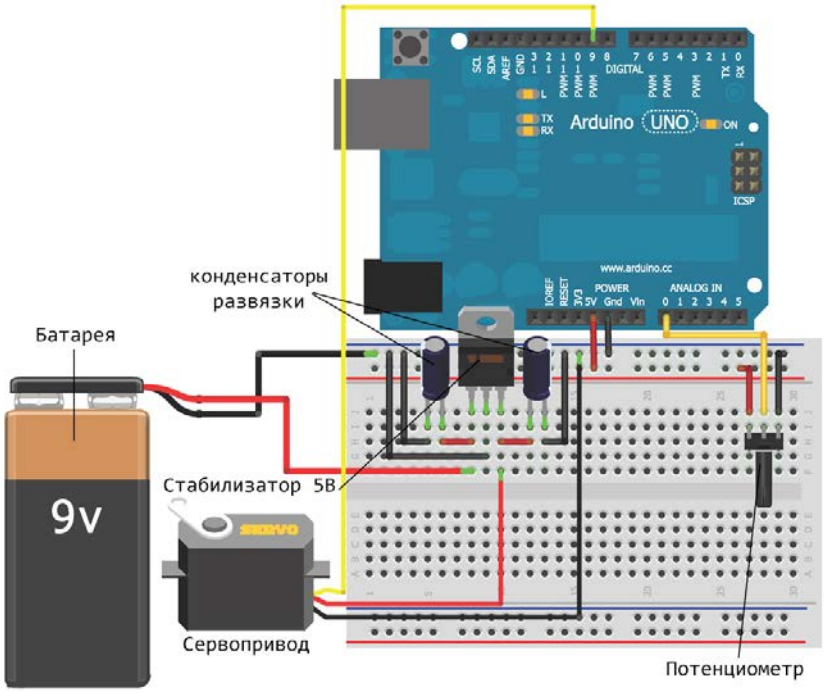

Рисунок 43. Схема подключения сервопривода

Обратите внимание на конденсаторы на входе и выходе стабилизатора, они устраняют пульсации напряжения. Схема и номиналы конденсаторов подойдут для большинства типов стабилизаторов. Имейте в виду, что выход стабилизатора не нужно соединять с шиной 5 В платы ***Arduino***. Соединить следует только шины земли стабилизатора и платы ***Arduino***.
Пришло время подсоединить сервопривод. Сверяясь с монтажной схемой, подсоедините потенциометр к аналоговому входу A0, сигнальный вход сервопривода к цифровому входу 9 платы ***Arduino***, убедитесь, что стабилизатор выдает напряжение 5 В. При соединении элементов обратите внимание на правильность подключения контактов стабилизатора и полярность включения конденсаторов (Рисунок 43). Соединив все элементы, можно перейти к следующему разделу, чтобы узнать, как программировать контроллер сервопривода.
Разберемся, почему серводвигателю нужен внешний источник, если он работает, как и плата ***Arduino***, от напряжения 5 В. При питании ***Arduino*** от USB для самой платы ***Arduino*** и подключенных к ней устройств максимально возможный ток равен 500 мА. В неподвижном положении сервоприводы потребляют малый ток. Но при выполнении команд сервоприводы потребляют ток в несколько сотен миллиампер, что приводит к скачкам напряжения. Кроме того, при недостаточном напряжении питания вал сервопривода перемещается неустойчиво. Поэтому для сервопривода необходим отдельный источник питания.

4.12. Контроллер серводвигателя

В ***Arduino*** IDE предусмотрена библиотека Servo для упрощения управления сервоприводами. Чтобы работать с библиотекой, необходимо подключить ее к нашей программе. Затем следует прикрепить объект servo к определенному выводу ***Arduino*** и задать угол вращения. Обо всем остальном позаботится библиотека. Самый простой способ проверить функционирование сервопривода— управление позицией вала с помощью потенциометра. Значение 0 потенциометра соответствует повороту сервопривода на 0°, значение 1023— повороту на 180°. Загрузите код, приведенный в Листинг 15, в плату ***Arduino***, чтобы проверить все в действии.

Листинг 15. Управление положением серводвигателя с помощью потенциометра — servo.ino

```
//Управление положением серводвигателя с помощью потенциометра

#include <Servo.h>

const int SERVO=9;    //Вывод 9 для подключения сигнального провода сервопривода
const int POT=0;      //Подключение потенциометра к аналоговому входу A0

Servo myServo;
int val = 0;          //Переменная для чтения показаний потенциометра

void setup()
{
    myServo.attach(SERVO);
}

void loop()
{
    val = analogRead(POT);       //Чтение данных потенциометра
    val = map(val, 0, 1023, 0, 179);  //Преобразование к нужному диапазон
    myServo.write(val);          // Установить положение сервопривода
    delay(15);                   // Ожидание сервопривода
}
```

Оператор `include`, указанный в начале программы, добавляет функционал библиотеки `Servo`. Оператор `servo myServo` создает объект сервопривода с именем `myServo`. В том месте программы, где требуется действие с сервоприводом, будет ссылка на объект `myServo`. В функции `setup()` вы

инициализируете сервопривод, присоединяя его к контакту 9 ***Arduino***. Можно подсоединить к ***Arduino*** несколько сервоприводов, создав несколько объектов `Servo` и назначив каждому свой контакт ***Arduino***. В цикле `loop()` считывается текущее значение потенциометра, масштабируется до диапазона значений сервопривода и формируется импульс для установки вала сервопривода в соответствующую позицию. Задержка на 15 мс гарантирует, что вал сервопривода фиксируется, прежде чем поступит новая команда.

4.13. Создание радиального датчика расстояния

В завершение этой главы применим знания, полученные ранее, для создания дальномера. Система состоит из инфракрасного (ИК) датчика расстояния, установленного на серводвигателе, и четырех светодиодов. Четыре позиции вала серводвигателя панорамируют датчик по периметру комнаты, что позволяет примерно определить расстояние до объектов в каждой из четырех областей. Яркость четырех светодиодов меняется в зависимости от расстояния до объекта в каждой области.

Так как инфракрасный свет является частью электромагнитного спектра, невидимой для человеческого глаза, подобная система может использоваться для создания "ночного видения". ИК-датчик расстояния работает следующим образом. Излучение ИК-светодиода воспринимается фотоприемником, который расположен рядом со светодиодом. Таким образом определяется расстояние до объекта, которое преобразуется в аналоговое напряжение и далее анализируется с помощью микроконтроллера. Даже если в комнате темно и расстояние до объекта неизвестно, дальномер позволит его узнать, потому что он работает в диапазоне, невидимом для человеческого глаза.

Разные модели ИК-датчиков могут иметь различные интерфейсы. Если ваш датчик отличается от рассмотренного в этом примере, необходимо ознакомиться с документацией, чтобы убедиться, что он является аналоговым.

> ***ПРИМЕЧАНИЕ***
>
> *Вы можете посмотреть демонстрационный видеоклип работы датчика расстояния на сайте https://www.exploringarduino.com/content/ch4/.*

Прикрепите термоклеем датчик расстояния на вал серводвигателя, как показано на Рисунок 44. Я предпочитаю термоклей, потому что он прочно крепит и при необходимости достаточно легко удаляется. Тем не менее, вы можете также воспользоваться суперклеем, шпатлевкой или клейкой лентой.

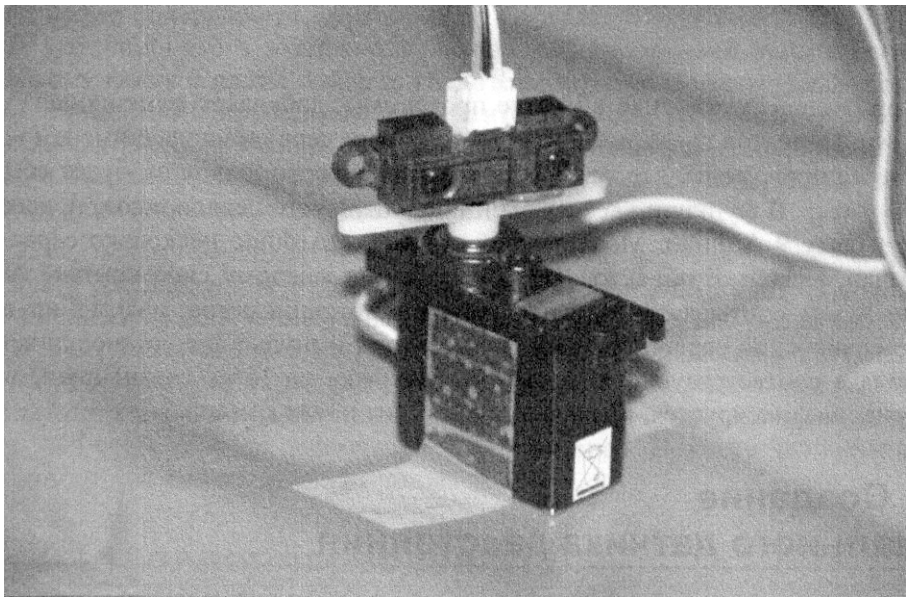

Рисунок 44. ИК-датчик расстояния, установленный на сервоприводе

Затем подключаем сервопривод к контакту 9 платы ***Arduino***, для питания сервопривода используем стабилизатор напряжения на 5 В. ИК-датчик расстояния соединяем с аналоговым входом A0. Четыре светодиода подключаем к контактам 3, 5, 6, и 11 через резисторы номиналом 1 кОм. На плате ***Arduino Uno*** предусмотрено шесть выводов ШИМ, но контакты 9 и 10 нельзя задействовать для создания ШИМ-сигналов, потому что аппаратный таймер, обеспечивающий ШИМ, занят библиотекой `Servo`. При желании увеличить число светодиодов, необходимо взять плату ***Arduino*** Mega или реализовать собственное программное обеспечение для формирования ШИМ.

Монтаж компонентов выполняйте согласно Рисунок 45. Я использовал синие светодиоды, но вы можете выбрать светодиоды любого другого цвета. ИК-датчик расстояния присоедините к сервоприводу, как показано на Рисунок 44.

Последний шаг — программирование датчика. Алгоритм работы системы следующий:

1. Поворот вала сервопривода в одну из четырех позиций.
2. Измерение расстояния.
3. Преобразование его в значение, которое подходит для управления светодиодом.
4. Изменение яркости соответствующего светодиода.
5. Выбор следующей позиции вала сервопривода.
6. Возврат к шагу 1.

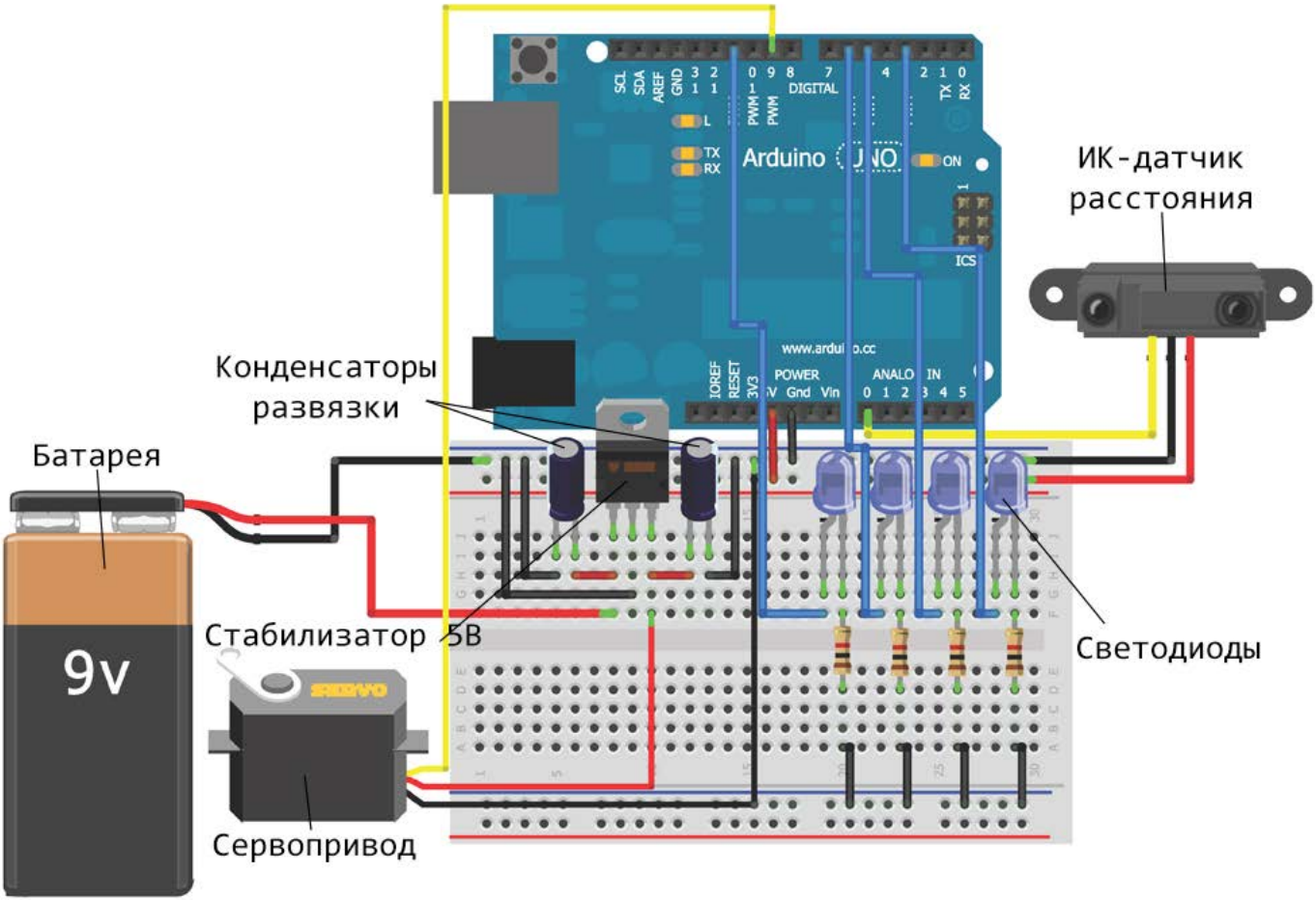

Рисунок 45. Схема подключения датчика расстояния

Код программы приведен в Листинг 16. Создайте в ***Arduino*** IDE новый проект, скопируйте этот код и загрузите его в плату ***Arduino***.

Листинг 16. Программа ИК-датчика расстояния

```
//ИК-датчик расстояния
#include <Servo.h>

 const int SERVO  =9;      //Вывод 9 для подключения сигнального провода
 const int IR     =0;      //Подключение ИК-датчика расстояния
                           //к аналоговому входу A0
 const int LED1   =3;      //Вывод светодиода 1
 const int LED2   =5;      //Вывод светодиода 2
 const int LED3   =6;      //Вывод светодиода 3
 const int LED4   =11;     //Вывод светодиода 4

Servo myServo;             //Создание объекта Servo
int dist1 = 0;             //Расстояние в первой области
```

```
int dist2 = 0;              //Расстояние во второй области
int dist3 = 0;              //Расстояние в третьей области
int dist4 = 0;              //Расстояние в четвертой области

void setup()
{
    myServo.attach(SERVO); //Подключение сервопривода
     // Назначение контактов подключения четырех светодиодов как выходы
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
}

void loop()
{
    //Поворот вала сервопривода по четырем позициям и изменения светодиодов
    dist1 = readDistance(15);   //Измерение расстояния ИК-датчиком
                                //при повороте сервопривода на 15 градусов
    analogWrite(LED1, dist1);   //Регулировка яркости светодиода
    delay(300);                 //Задержка перед следующим измерением

    dist2 = readDistance(65);   //Измерение расстояния ИК-датчиком
                                //при повороте сервопривода на 65 градусов
     analogWrite(LED2, dist2);  //Регулировка яркости светодиода
    delay(300);                 //Задержка перед следующим измерением

    dist3 = readDistance(115);  //Измерение расстояния ИК-датчиком
                                //при повороте сервопривода на 15 градусов
    analogWrite(LED3, dist3);   //Регулировка яркости светодиода
    delay(300);                 //Задержка перед следующим измерением

    dist4 = readDistance(165);  //Измерение расстояния ИК-датчиком
                                //при повороте сервопривода на 15 градусов
    analogWrite(LED4, dist4);   //Регулировка яркости светодиода
    delay(300);                 //Задержка перед следующим измерением
}

int readDistance(int pos)
{
    myServo.write(pos);         //Поворот в заданное положение
    delay(600);                 //Ожидание, пока серводвигатель двигается
    int dist = analogRead(IR);  //Чтение данных с датчика расстояния

    dist = map(dist, 50, 500, 0, 255);     //Преобразование к нужному диапазону
    dist = constrain(dist, 0, 255);        //Наложить ограничение значениям
    return dist;                           //Выдача значения,
                                           //соответствующего расстоянию

}
```

В программе есть функция `readDistance()`, которая поворачивает вал сервопривода на определенный угол, измеряет расстояние, масштабирует его, а затем возвращает полученное значение в цикл `loop()`. Какой диапазон значений выбрать для светодиодов, зависит от конкретной ситуации.

У меня для самого дальнего объекта датчик выдавал значение 50, до ближайших — 500.
После загрузки кода и запуска программы система должна функционировать, как на демонстрационных видеоклипах, перечисленных в начале главы.

Резюме

В этой главе вы узнали следующее:

- Как работают двигатели постоянного тока.
- Что двигатели являются индуктивными нагрузками, которые следует снабдить надлежащей защитой и схемой питания, чтобы безопасно взаимодействовать с платой ***Arduino***.
- Как управлять скоростью и направлением вращения двигателя с помощью ШИМ и Н-моста.
- Что серводвигатели точно позиционируются и управляются с помощью ***Arduino*** библиотеки `Servo`.
- Как создать вторичный источник питания 5 В от батареи 9 В с помощью стабилизатора напряжения.
- Что ИК-датчики получают инфракрасный сигнал, отраженный от объекта, и возвращают аналоговые значения, соответствующие расстоянию до данного объекта.
- Что комментарии важны для отладки и совместного использования программ.

Глава 5. Работаем со звуком

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;
- USB-кабель;
- 4 кнопки;
- 5 резисторов номиналом 10 кОм;
- 1 резистор номиналом 150 Ом;
- перемычки;
- провода;
- макетная плата;
- потенциометр 10 кОм;
- динамик 8 Ом.

Электронные ресурсы к главе

На странице https://www.exploringarduino.com/content/ch5/ можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать по ссылке: https://www.exploringarduino.com/code/ch5.zip.

Что вы узнаете в этой главе

Как известно, у человека пять органов чувств. Мы не будем задействовать вкус при общении с электронными компонентами, ведь никому не придет в голову облизывать ***Arduino***! Запах нам тоже не пригодится, если вы почувствуете запах от платы, то скорее всего схема сгорела. Остаются осязание, зрение и слух. При работе с потенциометром и кнопками требуется осязание, а при включении светодиодов — зрение. Без употребления остался только слух. Эта глава посвящена созданию звука с помощью ***Arduino***, так что теперь ваши устройства обретут собственный «голос».

Генерировать звук с помощью ***Arduino*** можно несколькими способами. Самый простой способ— использование функции `tone()`, которую мы рассмотрим в данной главе. Существуют также различные дополнительные платы, подключаемые к основной плате ***Arduino*** с помощью штыревых разъемов и расширяющие музыкальные возможности ***Arduino***. Некоторые из плат расширения мы рассмотрим з последующих главах. Если у вас плата ***Arduino*** Due, то для генерации звуков подойдет встроенный цифроаналоговый преобразователь (ЦАП).

5.1. Свойства звука

Перед тем как приступить к генерации звука с помощью ***Arduino***, вы должны понимать, что такое звук и как люди воспринимают его. В этом разделе мы расскажем о свойствах звуковых волн, воспроизведении музыки, речи и других звуков.

Звук распространяется по воздуху в виде волны. Работа звуковых колонок, удар в барабан или колокол создают вибрацию воздуха. Частицы воздуха за счет колебаний передают энергию все дальше и дальше. Волна давления передается от источника к вашей барабанной перепонке через реакцию вибрирующих частиц. Теперь посмотрим, как эти знания помогут нам сгенерировать звуки с помощью платы ***Arduino***?

Вы можете управлять двумя свойствами этих колеблющихся частиц: частотой и амплитудой. Под частотой понимают скорость вибрации частиц воздуха, а амплитуда представляет собой размах их колебаний. В физическом смысле звуки с большой амплитудой громче, чем с малой. Тон высокочастотных звуков выше (например, сопрано), а низкочастотных— ниже (например, бас). Рассмотрим график на Рисунок 46, на котором изображены синусоидальные звуковые волны с различными амплитудами и частотами.

На Рисунок 46 изображены графики, соответствующие трем фортепианным нотам: низкой, средней и высокой. В качестве примера рассмотрим ноту «До» первой октавы с частотой 261,63 Гц. Громкоговоритель, гитарная струна или фортепиано, при воспроизведении этой ноты генерирует звуковую волну, совершающую 261,63 колебаний в секунду. Можно рассчитать период колебания волны (1/261,63 = = 3,822 мс), что соответствует полному колебанию на графике. Плата ***Arduino*** позволяет задать период для меандра, устанавливая таким образом тембр каждой ноты. Важно отметить, что ***Arduino*** не может на самом деле создать синусоидальную волну, которая

распространена в реальном мире. Меандр является цифровым периодическим сигналом — это мгновенное переключение между двумя уровнями: высоким и низким (см. Рисунок 21). В результате по-прежнему возникает волна давления, обусловливающая звук, но звучание не вполне соответствует синусоидальной волне.

Что касается амплитуды, ею можно управлять, изменяя ток через динамик. Подключение потенциометра последовательно с динамиком позволяет регулировать уровень громкости звука.

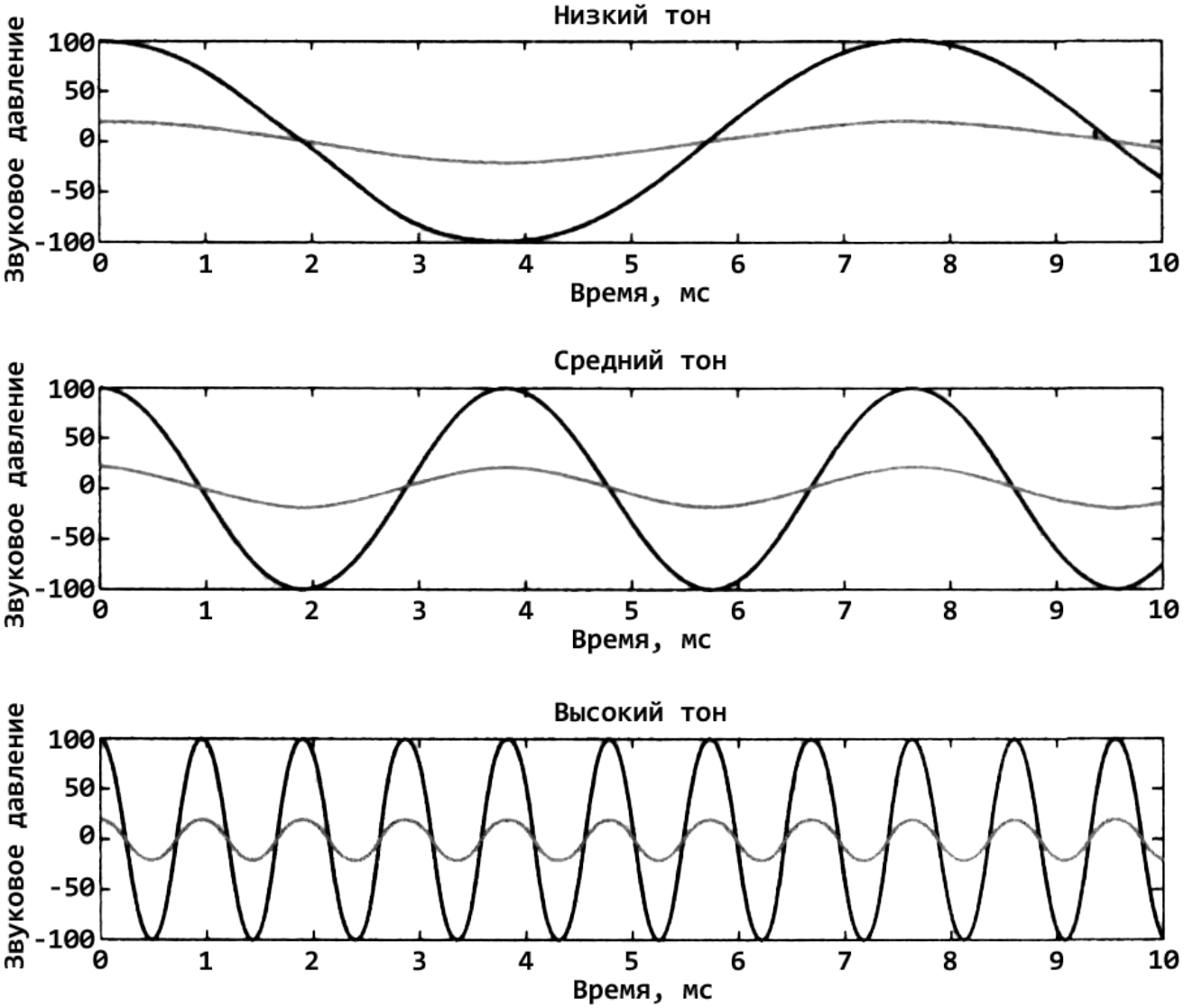

Рисунок 46. Звуковые волны с различной частотой и амплитудой

5.2. Как динамик воспроизводит звук

Динамики, как и двигатели, которые мы рассмотрели в предыдущей главе, используют электромагниты для преобразования электрического сигнала в механическое перемещение. Внимательно исследуйте металлическую деталь на задней стенке динамика. Заметили что-то необычное? К ней прилипают металлические предметы, потому что это магнит. Все станет понятно, если посмотреть на Рисунок 47, иллюстрирующий устройство динамика.

Перед постоянным магнитом размещена звуковая катушка. Когда вы подаете на нее электрический сигнал синусоидальной формы (или меандр, в случае ***Arduino***), переменный ток создает магнитное поле, которое заставляет звуковую катушку перемещать диффузор вверх и вниз. Эти возвратно-поступательные движения заставляют вибрировать диффузор, и из динамика раздается звук.

5.3. Использование функции *`tone()`* для генерации звуков

В ***Arduino*** IDE есть встроенная функция для генерации звуков произвольной частоты. Функция `tone()` формирует меандр с заданной частотой и выдает его на выбранный вами выходной контакт ***Arduino***.

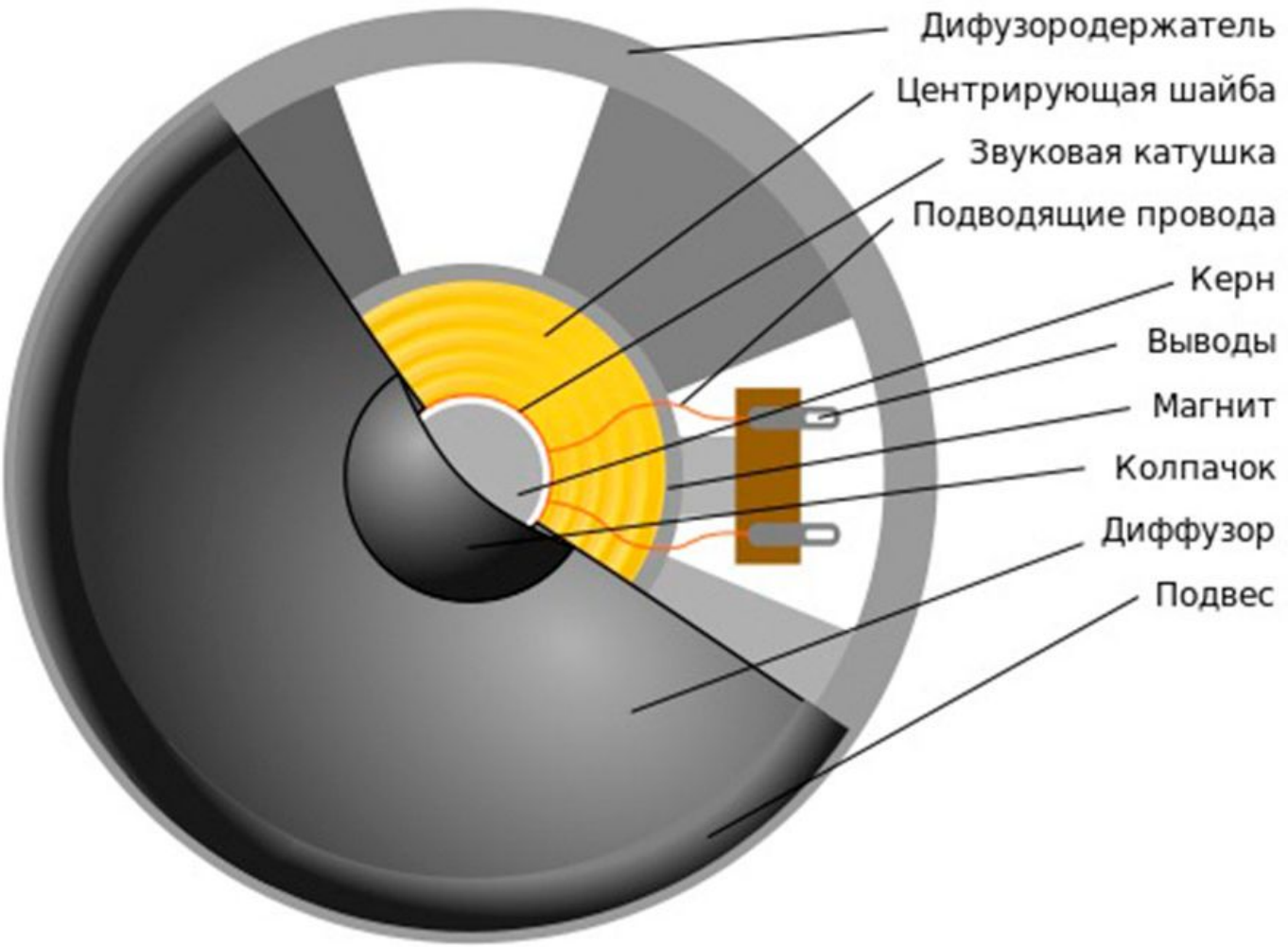

Рисунок 47. Устройство динамика

Аргументы `tone()`:

- первый аргумент устанавливает номер контакта ***Arduino*** для генерации волны;
- второй аргумент задает частоту сигнала;
- третий (необязательный) аргумент определяет продолжительность звучания; если этот аргумент не установлен, звук продолжается до тех пор, пока не вызвана функция `noTone()`.

Функция `tone()` взаимодействует с одним из аппаратных таймеров контроллера ATmega, поэтому ее можно вызвать и продолжать работать с ***Arduino***, а звук будет играть в фоновом режиме.

В следующих разделах вы узнаете, как создавать произвольные звуковые последовательности. Вы можете подать звуковой сигнал функцией `tone()` в ответ на различные события (нажатие кнопок, получение определенных значений с датчиков расстояния, акселерометров и т. д.). В конце главы мы расскажем, как создать простое пятикнопочное пианино.

5.4. Включение файла заголовка

Когда дело доходит до воспроизведения музыкальных звуков, полезно создать заголовочный файл, определяющий частоты для музыкальных нот. Это делает программу более понятной при составлении простых музыкальных мелодий. Те, кто знаком с нотными знаками, знают, что ноты обозначаются буквами. В ***Arduino*** IDE есть специальный файл, содержащий значения частот для всех нот. Не ищите его в каталогах, а просто зайдите на сайт https://www.exploringarduino.com/code/ch5.zip и скачайте на рабочий стол (находится в папке *music*). Затем в ***Arduino*** IDE создайте пустой новый файл. Как вы, наверное, заметили, ***Arduino*** IDE создает новый файл внутри папки с одноименным названием. Добавляя в эту папку новые файлы, вы можете включать их в свою программу, в результате код будет лучше структурирован. Скопируйте файл *`pitches.h`*, сохраненный на рабочем столе, в папку,

созданную ***Arduino*** IDE, для нового проекта. Теперь заново откройте в ***Arduino*** IDE этот файл. Обратите внимание на две вкладки (см. Рисунок 48).

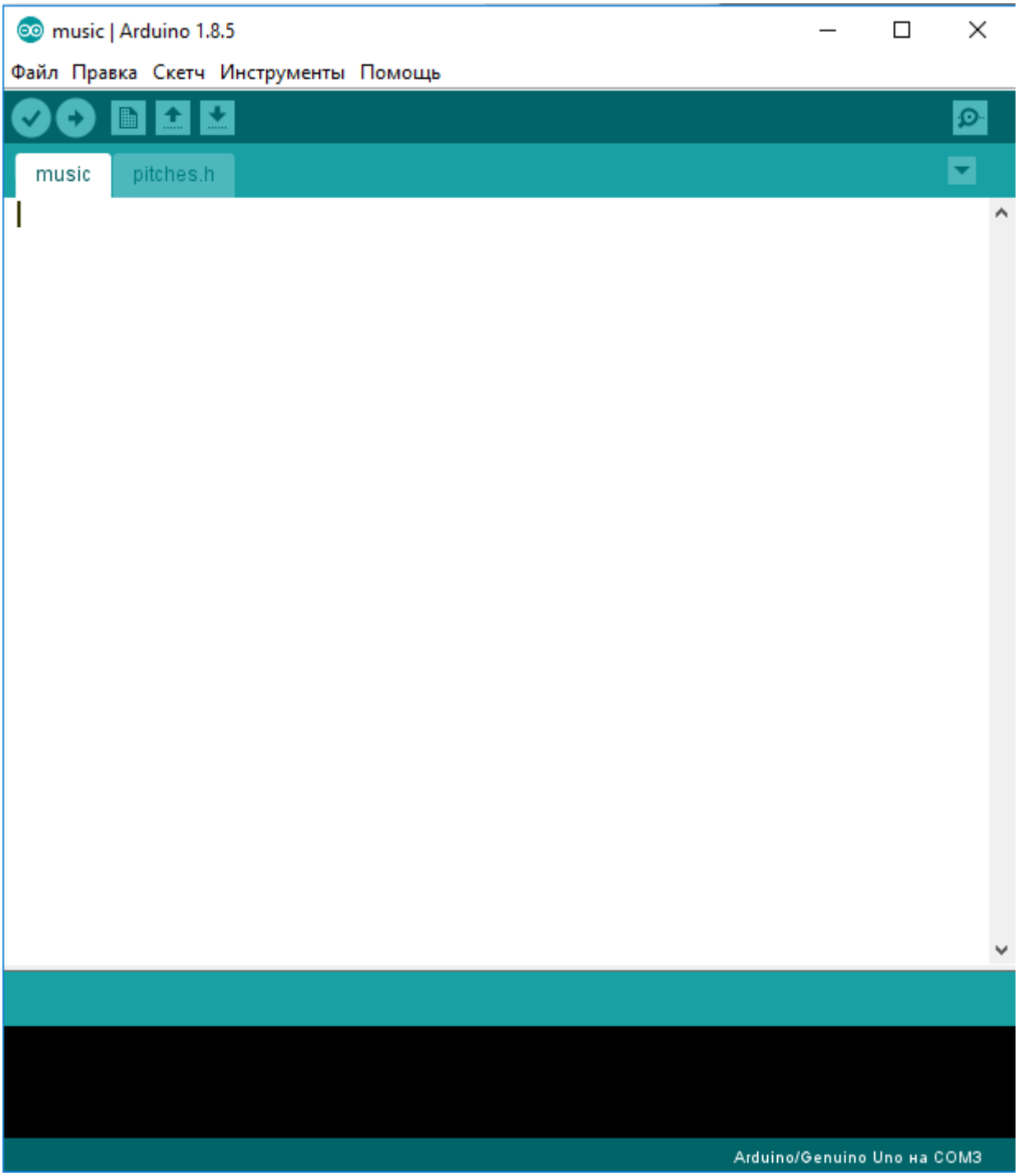

Рисунок 48. Окно Arduino IDE с двумя вкладками заголовочных файлов

Перейдите на вкладку `pitches.h`, чтобы увидеть содержимое файла. Обратите внимание, что это всего лишь список операторов определений, которые задают соответствие названий нот и значений частот. Чтобы использовать эти определения при компиляции программы для ***Arduino***, необходимо сообщить компилятору, где искать данный файл. Сделать это легко. Просто добавьте соответствующую строку кода в начало файла *.ino:

```
#include "pitches.h"
```

Для компилятора это, по существу, то же самое, что копирование и вставка содержимого файла заголовка в начало основного файла. Тем не менее, код становится аккуратнее и проще для чтения. В следующих разделах, при написании программ рекомендуем использовать данный заголовочный файл для определения высоты тона (частоты ноты).

5.5. Подключение динамика

Теперь, когда включен файл заголовка для нот, можно собрать схему и написать программу, которая будет воспроизводить звуки. Электрическая схема очень проста — нужно лишь соединить динамик с выходными контактами ***Arduino***. Однако при подключении необходимо помнить о токоограничивающих резисторах.

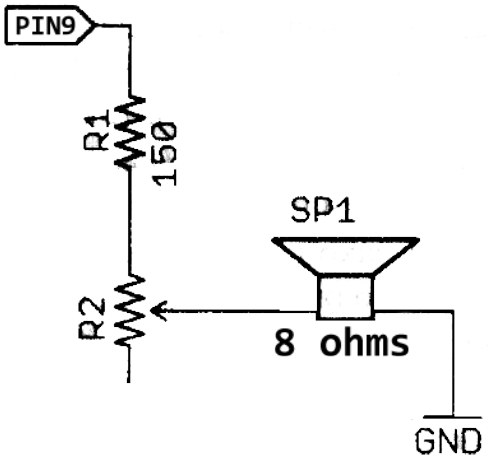

Рисунок 49. Схема включения динамика и регулятора громкости

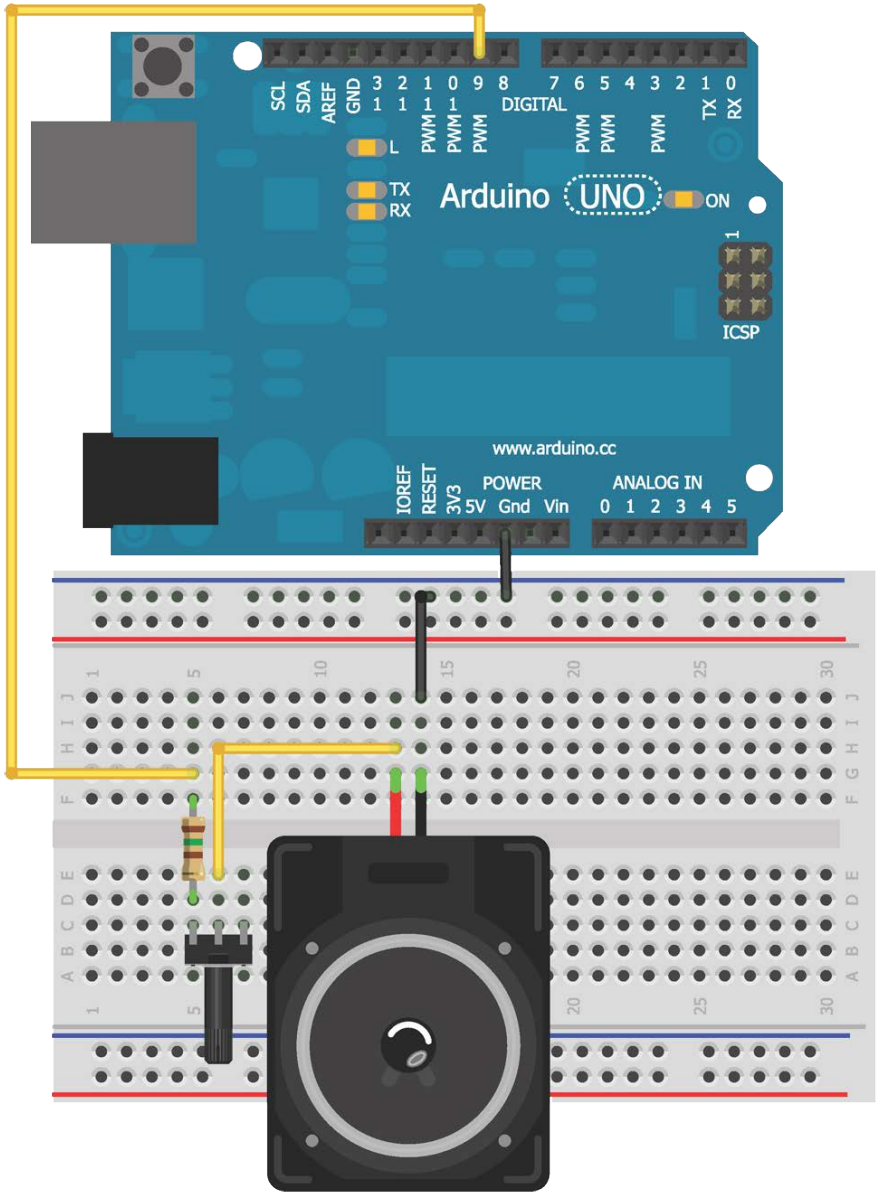

Рисунок 50. Монтажная схема подключения динамика

Так же, как и при подключении светодиодов, необходимо поставить токоограничивающий резистор последовательно с динамиком. В предыдущих главах упоминалось, что каждый вывод ***Arduino*** может выдать ток не более 40 мА. Внутреннее сопротивление нашего динамика равно 8 Ом (как и для большинства имеющихся в продаже динамиков). Это сопротивление обмоток провода, которые составляют электромагнит. Напомним, что закон Ома гласит U = IR. Выходное напряжение для вывода ***Arduino*** 5 В и ток не должен превышать 40 мА. Отсюда определяем, что минимальное сопротивление должно быть R = 5 В/40 мА = 125 Ом. Поскольку сопротивление динамика 8 Ом, то минимальное сопротивление токоограничивающего резистора получается 125 - 8 = 117 Ом. Ближайший номинал резистора 150 Ом. Регулируя сопротивление, можно изменять громкость динамика. Сделаем это проще, включив последовательно с резистором потенциометр (Рисунок 49). На схеме R1 — резистор 150 Ом, R2 — потенциометр 10 кОм.

Обратите внимание, что, в отличие от предыдущих случаев, здесь подключены только два вывода потенциометра: средний идет на динамик, а один из крайних соединен последовательно с токоограничивающим резистором 150 Ом. При повороте ручки потенциометра сопротивление цепи увеличивается и громкость снижается.

Подключите динамик к плате ***Arduino*** в соответствии со схемой, изображенной на Рисунок 49, и проверьте правильность монтажа по Рисунок 50.

Полярность включения динамиков не имеет значения. Схема собрана, и можно приступать к написанию музыки.

5.6. Создание мелодии

Для формирования мелодии очень удобно использовать массивы. Вывод последовательности звуков реализуется простым циклом перебора массива нот с отправкой текущего значения на динамик.

5.6.1. Использование массивов

Массив представляет собой упорядоченную последовательность элементов одного типа, которые связаны между собой. Группировка таких элементов— идеальный формат для перебора. Массив можно представить, как нумерованный список. Каждый элемент массива имеет индекс, который указывает его местоположение в списке. В нашем примере в массиве хранится звуковая последовательность — список нот, которые будем воспроизводить по порядку.

В ***Arduino*** при объявлении массива необходимо указывать его размер. Вы можете это сделать, либо явно указав размерность массива, либо заполнив массив всеми значениями. Например, если требуется, чтобы массив содержал четыре элемента типа `int`, объявляем его в программе так:

```
int numbers[4];
```

При необходимости можно инициализировать массив значениями при объявлении. При объявлении массива таким способом указывать число элементов массива необязательно, предполагается, что длина массива равна числу объявленных элементов:

```
// Оба варианта объявления верны
int numbers[4] = {-7, 0, 6, 234};
int numbers[] = {-7, 0, 6, 234};
```

Обратите внимание, что массивы индексируются с нуля. Доступ к элементу массива можно получить, поставив после имени массива в квадратных скобках соответствующее значение индекса. Например, если требуется установить яркость светодиода, подключенного к выводу 9 ***Arduino***, равной значению третьего элемента в массиве, то можно сделать это следующим образом:

```
analogWrite(9,numbers[2]);
```

Обратите внимание, что индекс 2 представляет собой третье значение в массиве, поскольку нумерация

начинается с нуля. Изменить одно из значений массива можно так:

```
numbers[2] = 10;
```

Далее массивы потребуются нам, чтобы создать структуру, которая может содержать последовательность нот, воспроизводимую на динамике.

5.6.2. Создание массивов нот и определение их длительности звучания

Для хранения информации о мелодии, которую вы хотите воспроизвести, создадим два массива одинаковой длины. Первый содержит перечень нот, а второй — список длительности звучания для каждой ноты в миллисекундах. Затем, перебирая индексы этих массивов, воспроизведем мелодию.
Пользуясь музыкальными навыками, которые я приобрел на уроках в средней школе, я сочинил короткую мелодию (Листинг 17).

Листинг 17. Пример мелодии

```
//Массив нот
int notes[] = {
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_E4, NOTE_D4, NOTE_C4, NOTE_B4, NOTE_A4, NOTE_B4, NOTE_C4, NOTE_D4,
 NOTE_E4, NOTE_E3, NOTE_A4, 0
};

//Массив длительностей звучания нот в мс
int times[] = {
 250, 250, 250, 250,
 250, 250, 250, 250,
 125, 125, 125, 125, 125, 125, 125, 125,
 250, 250, 250, 250
};
```

Обратите внимание, что длина обоих массивов одинакова (20 элементов). Некоторые ноты задаются в виде нулевых значений, — это музыкальные паузы. Длительность звучания каждой ноты берем из второго массива. Для тех, кто знаком с теорией музыки, обратите внимание, что я задал длительность четвертных нот 250 мс, а восьмых — 125 мс.
Сначала попробуйте воспроизвести мою мелодию, а затем попытайтесь создать свою собственную!

> ***ПРИМЕЧАНИЕ***
> *Послушать аудиозапись можно на странице https://www.exploringarduino.com/content/ch5/ или на сайте издательства Wiley.*

5.6.3. Написание программы воспроизведения звука

Осталось написать программу для воспроизведения мелодии. С помощью цикла выбираем значения нот и их длительность из массивов и реализуем воспроизведение каждой ноты. Поскольку вы, вероятно, не захотите слушать мелодию снова и снова, можно выполнить цикл в функции `setup()`. Чтобы возобновить воспроизведение, нажмите кнопку Reset. В Листинг 18 приведена полная программа проигрывателя ***Arduino***.

Листинг 18. Проигрыватель мелодий — `music.ino`

```
//Проигрывание мелодии на динамике

#include "pitches.h" //Заголовочный файл со значениями частоты нот

const int SPEAKER=9;  //Вывод подключения динамика

//Массив нот
```

```
int notes[] = {
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_E4, NOTE_D4, NOTE_C4, NOTE_B4, NOTE_A4, NOTE_B4, NOTE_C4, NOTE_D4,
 NOTE_E4, NOTE_E3, NOTE_A4, 0
};

//Массив длительностей звучания нот в мс
int times[] = {
 250, 250, 250, 250,
 250, 250, 250, 250,
 125, 125, 125, 125, 125, 125, 125, 125,
 250, 250, 250, 250
};


void setup()
{
 //Воспроизвение каждой ноты нужной длительностью
 for (int i = 0; i < 20; i++)
 {
  tone(SPEAKER, notes[i], times[i]);
  delay(times[i]);
 }
}

void loop()
{
 //Чтобы повторить воспроизведение, необходимо нажать кнопку Reset
}
```

Если вы захотите создать свою собственную мелодию, проследите, чтобы массивы нот и длительностей имели равный размер, и правильно задайте верхнюю границу для цикла перебора `for()`. Поскольку функция `tone()` может работать в фоновом режиме, важно определить задержку `delay()`, чтобы следующая нота не звучала, пока не закончится воспроизведение предыдущей.

Резюме

В этой главе вы узнали следующее:

- Как динамики создают вибрацию воздуха, которая распространяется в пространстве и воспринимается нашей барабанной перепонкой в виде звука.
- Что изменение электрического тока индуцирует магнитное поле, которое генерирует звук из громкоговорителя.
- Как создавать звуки произвольной частоты и длительности с помощью функции `tone()`.
- Что язык программирования ***Arduino*** поддерживает массивы, что удобно для перебора последовательностей данных.
- Что громкость можно регулировать потенциометром, соединенным последовательно с динамиком.

Глава 6. USB и последовательный интерфейс

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;
- плата ***Arduino Leonardo***;
- USB-кабель A - B (для ***Uno***);
- USB-кабель A - микро B (для Leonardo);
- светодиод;
- RGB-светодиод с общим катодом;
- резистор номиналом 150 Ом;
- 3 резистора номиналом 220 Ом;
- 2 резистора номиналом 10 кОм;
- кнопка;
- фоторезистор;
- датчик температуры TMP36;
- двухкоординатный джойстик (SparkFun, Parallax или Adafruit);
- перемычки;
- провода;
- макетная плата;
- потенциометр.

Электронные ресурсы к главе

На странице https://www.exploringarduino.com/content/ch6/ можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы https://www.wiley.com/go/exploringarduino в разделе ***Downloads***.

Что вы узнаете в этой главе

Обычно для загрузки программ из компьютера в микроконтроллер нужны внешние аппаратные средства, такие как программатор AVR ISP MKII. Замечательная особенность любой платы ***Arduino***— возможность запрограммировать ее через USB-интерфейс. Это позволяет программировать ***Arduino*** без специального программатора. В плату ***Arduino*** уже встроен программатор, что дает возможность напрямую подключаться к интегрированному универсальному синхронно/асинхронному приемопередатчику ATmega (USART). Через этот интерфейс можно обмениваться данными между ***Arduino*** и компьютером или между ***Arduino*** и другими устройствами, поддерживающими протокол (включая другие платы ***Arduino***).

В этой главе рассматривается все, что вам необходимо знать о подсоединении ***Arduino*** к компьютеру через USB и передаче данных между ними. У различных плат ***Arduino*** разные возможности последовательного соединения, и мы рассмотрим проекты с каждым из них, чтобы ознакомившись со всеми, затем использовать их максимально эффективно. Обратите внимание, что в списке деталей в начале главы указано несколько плат ***Arduino***. В зависимости от типа платы вы можете решить, какие разделы главы читать, какие примеры выбрать для выполнения и какие из перечисленных деталей понадобятся.

6.1. Реализация последовательного интерфейса в *Arduino*

Как уже упоминалось во введении к данной главе, в разных платах ***Arduino*** последовательный интерфейс выполнен по-разному. Различия есть как в аппаратной, так и в программной реализации: неодинаковы типы микросхем преобразователей и перечень поддерживаемых функций. Сначала мы рассмотрим различия аппаратных интерфейсов на платах ***Arduino***.

ПРИМЕЧАНИЕ

Чтобы узнать больше о последовательном интерфейсе, посмотрите видеоурок на странице https://www.jeremyblum.com/2011/02/07/arduino-tutorial-6-serial-communication-and-processing/[1] или на сайте издательства Wiley.

Для начала необходимо понять разницу между последовательным портом и USB. Если вы молоды, то, наверное, даже не сталкивались с последовательным портом (или RS-232), т. к. его давно уже вытеснил USB-интерфейс. Внешний вид стандартного последовательного порта изображен на Рисунок 51.

Фирменные платы ***Arduino*** снабжены последовательным портом и подключаются к компьютеру с помощью 9-контактного разъема. В настоящее время еще можно встретить компьютеры, оснащенные такими портами, хотя давно существуют адаптеры от RS232 к USB. У микроконтроллера ATmega328, который установлен на плате ***Arduino Uno***, есть один аппаратный последовательный порт. Он соединен с контактами Tx (передача) и Rx (прием), к которым можно получить доступ на цифровых выводах 0 и 1. Как мы узнали в *Глава 1*. Начало работы, переключаем светодиод из *Arduino*, плата ***Arduino*** снабжена загрузчиком, который позволяет программировать ее по последовательному интерфейсу. Это как раз те выводы, которые «мультиплексированы» (т. е. выполняют более одной функции), они используются и как линии приема-передачи кабеля USB. Но последовательный порт и USB-интерфейс несовместимы. В ***Arduino*** эта проблема решается двумя способами. Первый — применение дополнительной микросхемы-преобразователя (так сделано на платах ***Arduino Uno***). Второй способ — использование микроконтроллера, имеющего встроенный USB-интерфейс (например, микроконтроллер 32U4 в ***Arduino*** Leonardo).

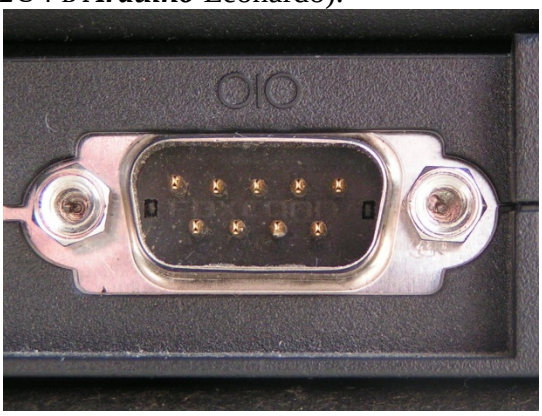

Рисунок 51. Последовательный порт

6.2. Платы *Arduino* с внутренним или внешним преобразователем FTDI

На многих платах ***Arduino*** (и их клонах) установлена дополнительная интегральная схема для преобразования USB в последовательный порт. FTDI — популярный чип, выполняющий единственную функцию: конвертирование между последовательным портом и USB. Когда компьютер подключается к микросхеме FTDI, она появляется в системе как "Virtual Serial Port", и доступ к нему аналогичен 9-проводному порту прямо в вашем компьютере. Плата ***Arduino Nano*** с установленной микросхемой преобразователя FTDI изображена на Рисунок 52.

ПРИМЕЧАНИЕ

Для обеспечения правильного взаимодействия компьютера с адаптером FTDI необходимо установить драйверы. Найти последние версии драйверов для Windows, OS X и Linux можно на странице http://www.ftdichip.com/Drivers/VCP.htm. Ссылка на данную страницу есть на сайте Exploring Arduino.

Иногда для уменьшения размера платы чип FTDI встраивают в кабель (USB-кабель с чипом FTDI изображен на Рисунок 53) или устанавливают на дополнительной плате адаптера (Рисунок 54).
Плата ***Arduino*** со съемным адаптером FTDI целесообразна для проектов, в которых нет необходимости

[1] На русском: http://wiki.amperka.ru/видеоуроки:6-serial-и-processing.

подключаться к компьютеру через USB. В результате уменьшится стоимость и габариты готового устройства.

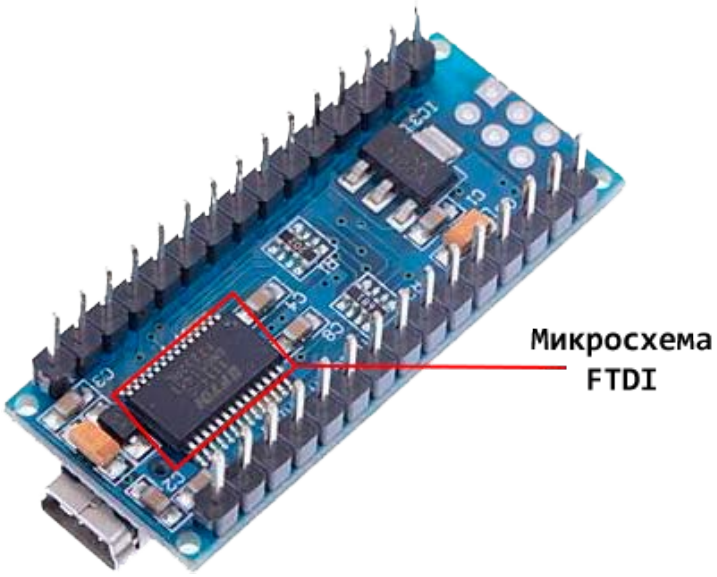

Рисунок 52. Плата Arduino-Nano с чипом FTDI

Рисунок 53. Кабель FTDI

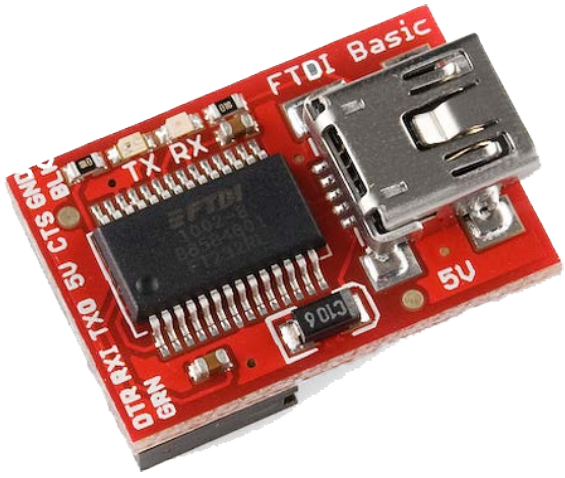

Рисунок 54. Плата адаптера FTDI Sparkfun

Далее перечислены платы ***Arduino*** со встроенным чипом FTDI. Большинство из них уже не выпускается, тем не менее, продается еще много клонов этих плат:

- ***Arduino Nano;***
- ***Arduino Extreme;***
- ***Arduino NG;***
- ***Arduino Diecimila;***
- ***Arduino Duemilanove;***
- ***Arduino Mega (original).***

А вот список плат, рассчитанных на работу с внешним адаптером FTDI:

- ***Arduino Pro;***
- ***Arduino Pro Mini;***
- ***LilyPad Arduino;***
- ***Arduino Fio;***
- ***Arduino Mini;***
- ***Arduino Ethernet.***

6.3. Платы Arduino с дополнительным микроконтроллером для преобразования USB в последовательный порт

Плата ***Arduino Uno*** была первой платой, где для преобразования USB в последовательный порт применен дополнительный контроллер. Функционирует все точно так же, но с небольшими техническими различиями. На Рисунок 55 изображен последовательный адаптер 8U2, установленный в ***Arduino Uno*** (в новых версиях используется преобразователь 16U2).

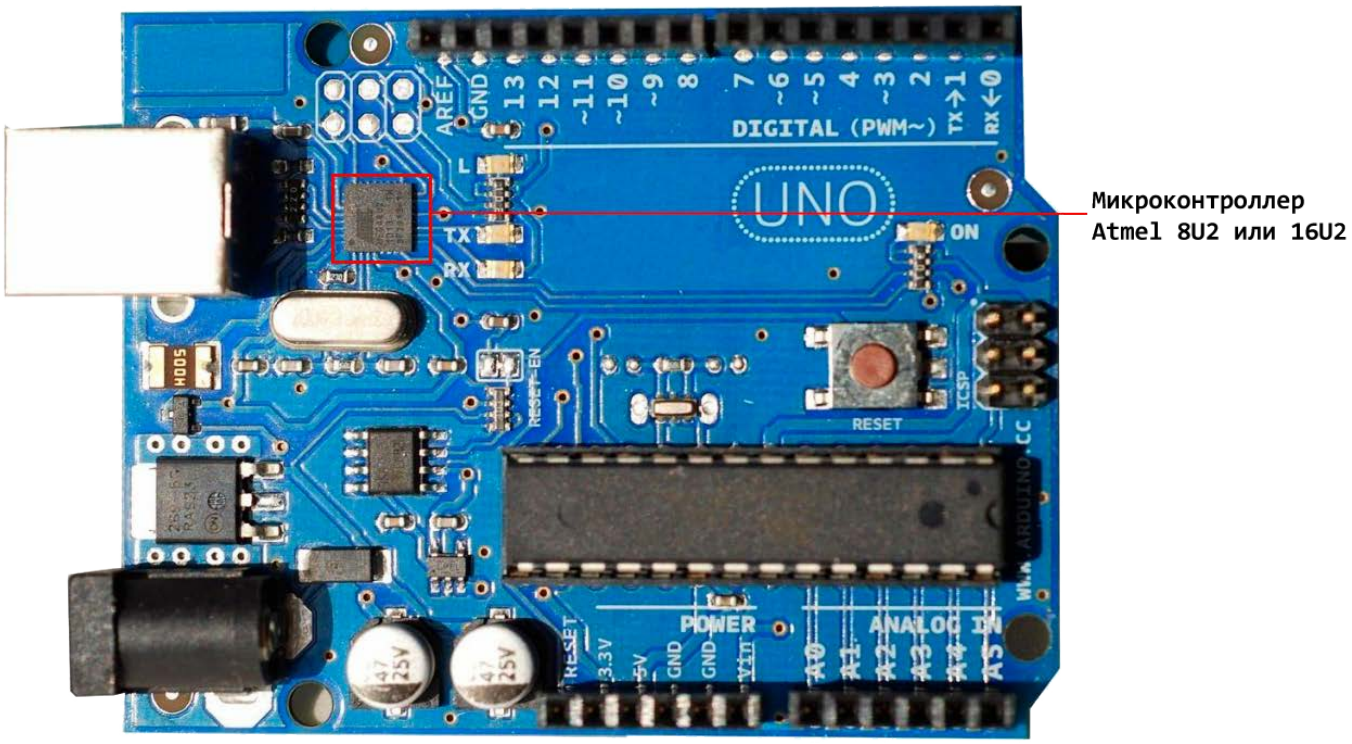

Рисунок 55. Чип 8U2 на плате Arduino Uno

Вот краткий перечень различий:

- в операционной системе Windows для плат с новым адаптером требуется специальный драйвер. Этот драйвер поставляется в комплекте с загруженной версией ***Arduino*** IDE (драйверы для операционных систем OS X и Linux не нужны);
- использование второго микроконтроллера в качестве адаптера позволило при подключении передавать в компьютер идентификатор производителя и код продукта. Ранее при подключении платы ***Arduino*** к компьютеру она определялась как последовательное устройство. Плата ***Arduino*** с адаптером 8U2 и 16L2 определяется компьютером как устройство ***Arduino***;

- вспомогательный микроконтроллер можно перепрограммировать (он работает на прошивке LUFA, которая эмулирует конвертер USB), следовательно, есть возможность изменить прошивку, чтобы плата ***Arduino*** определялась, например, как джойстик, клавиатура или MIDI-устройство. При замене прошивки плату ***Arduino*** придется программировать через программатор, например, AVR ISP MKII.

Вот список плат ***Arduino*** со вспомогательным микроконтроллером для преобразования USB в последовательный порт:

- ***Arduino Uno***;
- ***Arduino Mega 2560***;
- ***Arduino Mega ADK*** (на основе 2560);
- ***Arduino Due*** (эту плату можно запрограммировать непосредственно).

6.4. Платы *Arduino* с микроконтроллером, снабженным встроенным интерфейсом USB

Плата ***Arduino*** Leonardo была первой платой, имеющей только одну микросхему, выполняющую функции и программируемого пользователем микроконтроллера, и интерфейса USB. На плате ***Arduino*** Leonardo (и ее клонах) установлен микроконтроллер 32U4, поддерживающий прямую передачу через USB. Это дает несколько преимуществ.

Во-первых, уменьшается стоимость платы, потому что на ней меньше компонентов и короче программа первоначальной загрузки платы. Во-вторых, плата способна эмулировать не только последовательный порт, но и другие устройства (такие как клавиатура, мышь или джойстик). В-третьих, обычный порт USART на ATmega не мультиплексирован с выводами интерфейса USB, поэтому возможен параллельный обмен данными как с главным компьютером, так и с внешним последовательным устройством (таким как модуль GPS).

Вот список плат ***Arduino***, снабженных микроконтроллером со встроенным интерфейсом USB:

- ***Arduino Due*** (ее также можно запрограммировать через вспомогательный микроконтроллер);
- ***LilyPad Arduino USB***;
- ***Arduino Esplora***;
- ***Arduino Leonardo***;
- ***Arduino Micro***.

6.5. Платы *Arduino* с возможностями USB-хоста

Некоторые платы ***Arduino*** обладают возможностями USB-хоста, что позволяет подсоединить к ним традиционные USB-устройства (клавиатуры, мыши, телефоны на базе Android). Естественно, для поддержки этих устройств потребуются дополнительные драйверы. Например, нельзя просто так соединить веб-камеру с ***Arduino*** Due и сразу же ожидать получения фотографий. ***Arduino*** Due поддерживает класс USB Host, что позволяет подключить к USB-порту клавиатуру или мышь. ***Arduino*** Mega ADK поддерживает протокол Android Open Accessory Protocol (AOA), что упрощает обмен данными между ***Arduino*** и устройством на базе Android. Прежде всего, это нужно для управления вводом-выводом ***Arduino*** из приложения, работающего на устройствах на базе Android.

Возможности USB-хоста поддерживают две платы: ***Arduino Due*** и ***Arduino Mega ADK*** (на основе Mega 2560).

6.6. Опрос *Arduino* с компьютера

Основная функция, которую обеспечивает последовательный интерфейс, — вывод данных с ***Arduino*** в терминал компьютера. В предыдущих главах об этом уже упоминалось. В этом разделе более детально остановимся на данном вопросе и позже рассмотрим примеры приложений, которые отвечают на данные, которые вы отправляете, вместо того, чтобы просто выводить их в терминал. Этот процесс одинаков для всех плат ***Arduino***.

6.6.1. Пример вывода данных

Для вывода данных в терминал существуют три функции:

- `Serial.begin(baud_rate);`
- `Serial.print("Message");`
- `Serial.println("Message"),`

где `baud rate` и `Message` — переменные, задаваемые пользователем.

Как вы уже знаете, функция `serial.begin()` вызывается один раз в начале программы в `setup()`, чтобы настроить последовательный порт для связи. После этого можно вызвать функции `Serial.print()` и `Serial.println()` для передачи данных в последовательный порт. Единственное различие между ними состоит в том, что функция `Serial.println()` добавляет символ перевода в конце строки. Чтобы поэкспериментировать с этими функциями, соберем простую схему, подключив потенциометр к контакту A0 платы ***Arduino***, как показано на Рисунок 56.

После подключения потенциометра загрузите простую программу, приведенную в Листинг 19, которая выводит показания потенциометра в виде абсолютного значения и в процентах.

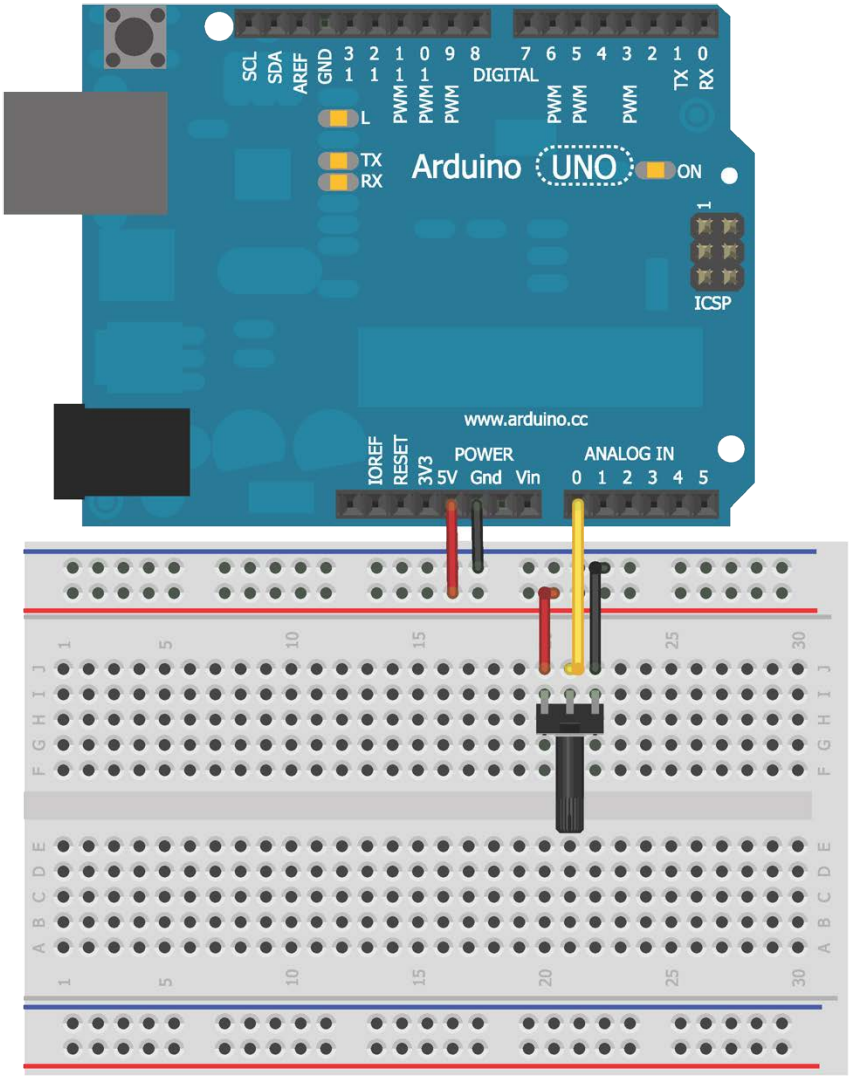

Рисунок 56. Схема подключения потенциометра

Листинг 19. Тестовая программа вывода значений потенциометра в последовательный порт — pot.ino

```
//Тестовая программа вывода значений потенциометра
// в последовательный порт
const int POT=0;           //Подключение потенциометра к аналоговому входу A0

void setup()
{
  Serial.begin(9600);      //Инициализация последовательного порта
                           // на скорости 9600
```

```
}

void loop()
{
  int val = analogRead(POT);              // Чтение данных с потенциометра
  int per = map(val, 0, 1023, 0, 100);    //Перевод в процентное значение
  Serial.print("Analog Reading: ");
  Serial.print(val);                      //Вывод аналогового значения
  Serial.print("  Percentage: ");
  Serial.print(per);                      //Вывод значения в процентах
  Serial.println("%");                    //Печать знака «%» и перевод строки
  delay(1000);                            //Ожидание 1 сек перед
                                          // получением новых данных
}
```

6.6.2. Использование специальных символов

Вы также можете передавать различные «специальные» (или управляющие) символы, которые позволяют изменить форматирование последовательных данных при выводе на печать. Управляющий символ начинается с обратного слэша (\), за которым следует символ команды. Есть много специальных символов, но два из них представляют наибольший интерес: табуляция и переход на новую строку. Чтобы вставить символ табуляции, нужно добавить к строке управляющую последовательность `\t`. Символ перехода на новую строку вставляют, добавляя к строке `\n`. Это особенно полезно, если вы хотите перейти на новую строку в начале передаваемой строки, а не в конце, как делает функция `Serial.println()`. Если, по некоторым причинам, вы на самом деле хотите напечатать последовательность символов `\n` или `\t` в строке, это можно сделать с помощью последовательностей `\\n` или `\\t`, соответственно. Листинг 20 представляет собой модификацию Листинг 19, но с использованием управляющих символов для отображения данных в табличной форме.

Листинг 20. Табличная разметка с помощью управляющих символов — pot_tabular.ino

```
//Табличная разметка с использованием управляющих символов
const int POT=0;                  //Подключение потенциометра
                                  //к аналоговому входу A0

void setup()
{
  Serial.begin(9600);             //Инициализация последовательного порта
                                  //на скорости 9600
}

void loop()
{
  Serial.println("\nAnalog Pin\tRaw Value\tPercentage");
  Serial.println("-----------------------------------------");
  for (int i = 0; i < 10; i++)
  {
    int val = analogRead(POT);            //Чтение данных потенциометра
    int per = map(val, 0, 1023, 0, 100);  //Перевод в процентное значение

    Serial.print("A0\t\t");
    Serial.print(val);                    //Вывод аналогового значения
    Serial.print("\t\t");
    Serial.print(per);                    //Вывод процентного значения
    Serial.println("%");                  //Печать знака «%» и перевод строки
```

```
    delay(1000);                          //Ожидание 1 сек перед
                                          //получением новых данных

  }
}
```

При повороте движка потенциометра данные, выдаваемые в последовательный порт, должны выглядеть примерно так, как на Рисунок 57.

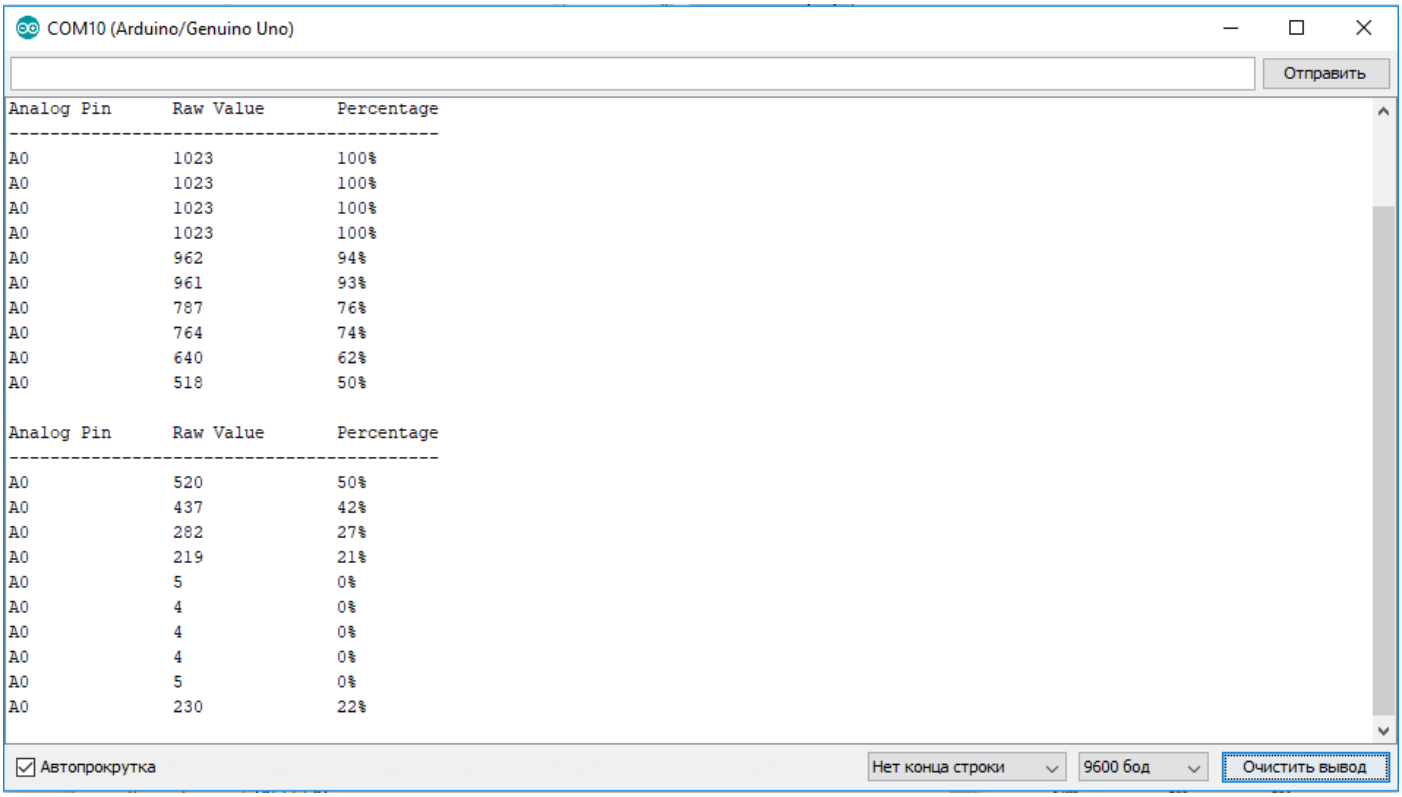

Рисунок 57. Скриншот терминала последовательного порта сданными в виде таблицы

6.6.3. Изменение представлений типа данных

Функции Serial.print () и `Serial.println()` позволяют вывести данные в конкретном формате. Есть опции для представления данных в различных форматах, включая шестнадцатеричный, восьмеричной и двоичный. По умолчанию принят десятичный формат `ASCII`. У функций `Serial.print()` и `Serial.println()` есть дополнительный второй параметр, который определяет формат выводимых данных. В Таблица 2 приведены примеры вывода одинаковых данных (число 23) в различных форматах и их отображения в мониторе последовательного порта.

Таблица 2. Тип последовательных данных

Формат	Код	Отображение данных
Decimal	Serial.println(23);	23
Hexadecimal	Serial.println(23,HEX);	17
Octal	Serial.println(23,OCT);	27
Binary	Serial.println(23,BIN);	00010111

6.6.4. Общение с *Arduino*

Что хорошего в общении с ***Arduino***, если разговор идет только в одном направлении? Теперь, когда мы разобрались, как ***Arduino*** посылает данные на компьютер, давайте попробуем это сделать. Вы, наверное, заметили, что в окне монитора последовательного порта ***Arduino*** IDE сверху есть поле ввода текста, а снизу — раскрывающееся меню (Рисунок 58).

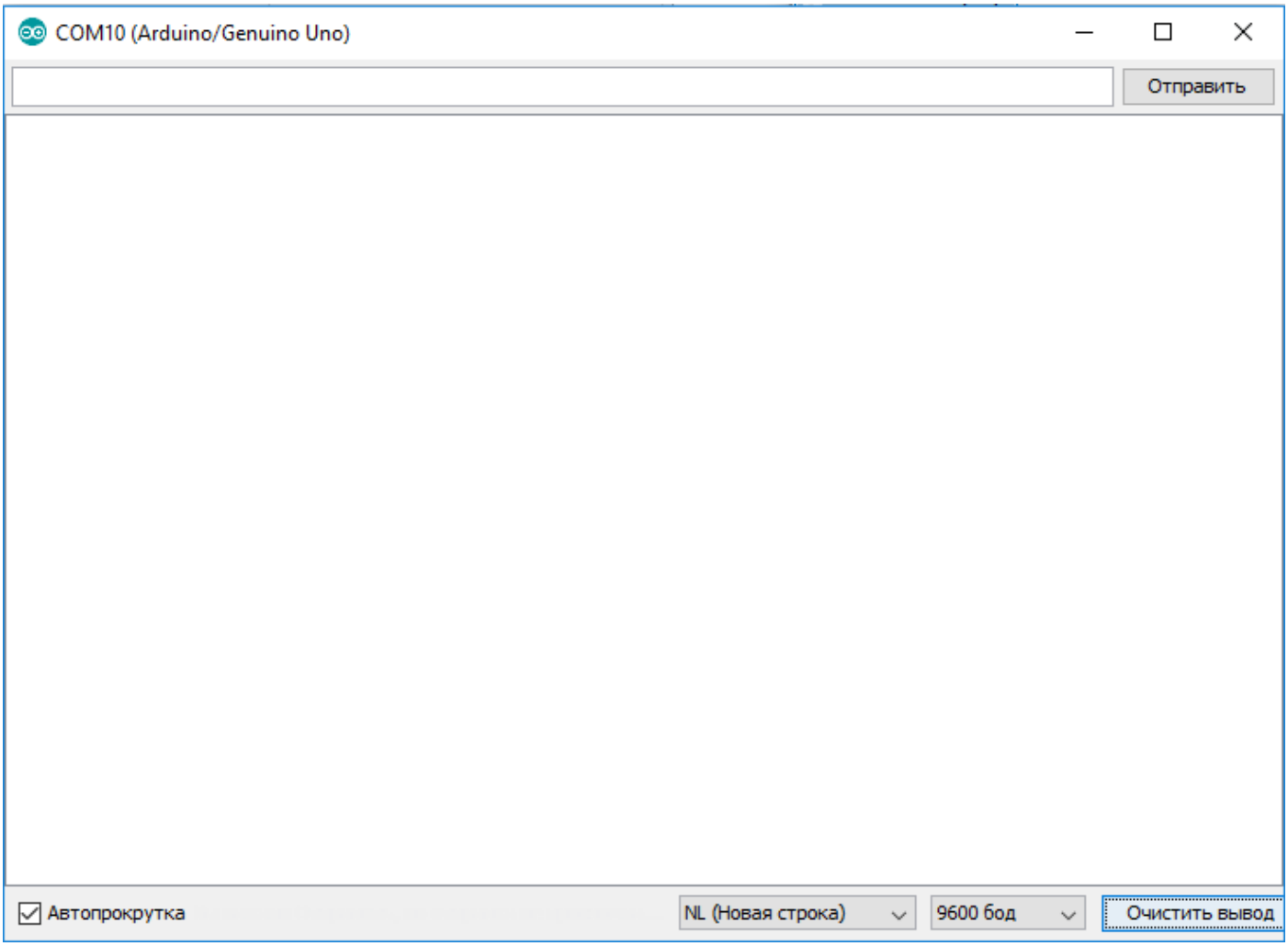

Рисунок 58. Скриншот терминала последовательного порта с полем ввода текста и выбранной опцией конца строки в раскрывающемся меню

Во-первых, убедитесь, что в раскрывающемся списке выбран пункт **Newline**. Опция в выпадающем меню определяет, что добавляется к вашим командам при отправке их в ***Arduino***. Примеры в следующих разделах предполагают, что вы выбрали пункт **Newline**, который только добавляет \n в конец ваших данных, посылаемых из поля ввода текста в верхней части окна монитора последовательного порта.

В отличие от некоторых других терминальных программ, монитор последовательного порта ***Arduino*** IDE при нажатии клавиши «Enter» или кнопки «Отправить» посылает всю командную строку. Другие последовательные терминалы, например, Putty (https://www.exploringarduino.com/), посылают символы сразу после их ввода.

6.6.5. Чтение информации из компьютера или другого последовательного устройства

Запустив монитор последовательного порта ***Arduino*** IDE, вы можете вручную отправлять команды в ***Arduino***. Далее рассмотрим, как отправлять последовательности символов и как создать простой графический интерфейс для отправки команд.

У последовательного порта ***Arduino*** есть буфер. Другими словами, вы можете отправить несколько байт данных сразу, и ***Arduino*** поставит их в очередь и обработает их в том порядке, как задумано в вашей программе. Скорость передачи данных не особенно критична, а вот отправка слишком большого объема данных может вызвать переполнение буфера и потерю информации.

Плата Arduino в качестве транслятора данных

Проще всего заставить плату ***Arduino*** реагировать на все, что передано. Для этого плата ***Arduino*** должна запоминать, а затем выводить каждый полученный символ. Чтобы сделать это, потребуются две новые функции:

- `Serial.available()` — возвращает число символов (или байтов), которые на данный момент сохранены во входном последовательном буфере ***Arduino***. Каждый раз, когда это число

отлично от нуля, символ считывается и отправляется обратно на компьютер;

- `Serial.read()` — читает и возвращает следующий символ, который находится в буфере.

Обратите внимание, что каждый вызов к функции `Serial.read()` возвратит из буфера только один байт, поэтому вы должны повторять эту функцию, пока `Serial.available()` возвращает ненулевое значение. Каждый раз, когда функция `Serial.read()` получает байт, буфер очищается, и можно принимать следующий байт. Теперь можно загрузить программу из Листинг 21 на плату ***Arduino***.

Листинг 21. Эхо последовательного порта — echo.ino

```
//Эхо каждого символа

char data;            //Текущий входящий символ

void setup()
{
  Serial.begin(9600); //Инициализация последовательного порта на скорости 9600
}

void loop()
{
//Вывод только при получении данных
  if (Serial.available() > 0)
  {
    data = Serial.read();  //Чтение байта из буфера
    Serial.print(data);    //Вывод в последовательный порт
  }
}
```

Запустите монитор последовательного порта и напечатайте что-нибудь в поле ввода текста. Как только вы нажмете кнопку «Отправить», все, что вы набрали, вернется назад в компьютер и отобразится в окне монитора последовательного порта. Поскольку в меню выбрана опция `Newline`, для каждой команды добавляется символ перевода строки (`\n`), следовательно, каждый ответ отображается с новой строки. Поэтому вместо функции `Serial.println()` в Листинг 21 указана функция `Serial.print()`.

Различие между char и int

При отправке алфавитно-цифрового символа через монитор последовательного порта мы на самом деле не отправляем "`5`" или "`A`". Мы посылаем байт, который компьютер интерпретирует как символ. В случае последовательной связи для представления всех букв, цифр, символов и специальных команд используется кодировка `ASCII`. Основной набор символов `ASCII` (Таблица 3) представляет собой 7-битовый набор, содержащий 128 символов и команд.

При получении значения, отправленного с компьютера, данные должны быть считаны, как `char` (см. Листинг 21). Даже если вы ожидаете из последовательного терминала отправки числа, вам необходимо прочитать байт, как символ, а затем при необходимости конвертировать его. Если вы измените программу Листинг 21, объявив переменную `data` как `int`, то, отправив значение 5, в последовательный монитор вернется значение 53 (десятичное представление символа "5"). Вы можете убедиться в этом, посмотрев Таблица 3.

Тем не менее, в ***Arduino*** часто необходимо отправлять числовые значения. Как это сделать? Существует несколько способов. Во-первых, можно просто сравнить сами символы. Если вы хотите зажечь светодиод при отправке из монитора цифры 1, то можно непосредственно сравнить значения символов:

```
if(Serial.read () = '1')
```

Одинарные кавычки вокруг 1 означают, что единица должна рассматриваться как символ. Второй вариант заключается в преобразовании каждого входящего байта в целое путем вычитания символа '`0`':

```
int val = Serial.read() - '0'
```
Такой способ не годится для чисел больше 9, потому что они содержат несколько цифр. В этом случае нас выручит функция `parseInt()`, входящая в ***Arduino*** IDE, которая извлекает из последовательного потока данных целые числа. Далее мы подробно рассмотрим описанные методы.

Таблица 3. Набор символов ASCII (American Standard Code for Information Interchange)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	32	20	(sp)	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Таблица 4. Сокращения, использованные в таблице символов ASCII

Форматирование	
BS	Backspace (Возврат на один символ). Указывает на движение механизма печати или курсора дисплея назад на одну позицию.
HT	Horizontal Tabulation (Горизонтальное Табулирование). Указывает на движение механизма печати или курсора дисплея до следующей предписанной 'позиции табуляции'.
LF	Line Feed (Перевод строки). Указывает на движение механизма печати или курсора дисплея к началу следующей строки (на одну строку вниз).

VT	Vertical Tabulation (Вертикальное Табулирование). Указывает на движение механизма печати или курсора дисплея к следующей группе строк.
FF	Form Feed (Перевод страницы). Указывает на движение механизма печати или курсора дисплея к исходной позиции следующей страницы, формы или экрана.
CR	Carriage Return (Перевод каретки). Указывает на движение механизма печати или курсора дисплея к исходной (крайней левой)позиции текущей строки.
Передача данных	
SOH	Start of Heading (Начало Заголовка). Используется для указания начала заголовка, который может содержать информацию о маршрутизации или адрес.
STX	Start of Text (Начало Текста). Указывает на начало текста и одновременно на конец заголовка.
ETX	End of Text (Конец Текста). Используется при завершении текста, который был начат с символа STX.
ENQ	Enquiry (Запрос). Запрос идентификационных данных (типа "Кто Вы?") от удаленной станции.
ACK	Acknowledge (Подтверждение). Приемное устройство передает этот символ отправителю в качестве подтверждения успешного приема данных.
NAK	Negative Acknowledgement (Неподтверждение). Приемное устройство передает этот символ отправителю в случае отрицания (неудачи) приема данных.
SYN	Synchronous/Idle (Синхронизация). Используется в синхронизированных системах передачи. В моменты отсутствия передачи данных система непрерывно посылает символы SYN для обеспечения синхронизации.
ETB	End of Transmission Block (Конец Блока Передачи). Указывает на конец блока данных для коммуникационных целей. Используется для разбиения на отдельные блоки больших объемов данных.
Разделительные знаки при передаче информации	
FS	File Separator (Разделитель файлов).
GS	Group Separator (Разделитель групп).
RS	Record Separtator (Разделитель записей).
US	Unit Separator (Разделитель элементов).
Другие символы	
NUL	Null. (No character- нет данных). Используется для передачи в случае отсутствия данных.
BEL	Bell (Звонок). Используется для управления устройствами сигнализации.
SO	Shift Out. Указывает, что все последующие кодовые комбинации должны интерпретироваться согласно внешнему набору символов до прихода символа SI.
SI	Shift In. Указывает, что последующие кодовые комбинации должны интерпретироваться согласно стандартному набору символов.
DLE	Data Link Escape (Переключение). Изменение значения идущих следом символов. Используется для дополнительного контроля или для передачи произвольной комбинации бит.
DC1, DC2, DC3, DC4	Device Controls (Контроль Устройства). Символы для управления вспомогательными устройствами (специальными функциями).
CAN	Cancel (Отмена). Указывает, что данные, который предшествовали этому символу в сообщении или блоке, должны игнорироваться (обычно в случае обнаружения ошибки).
EM	End of Medium (Конец Носителя). Указывает на физический конец ленты или другого носителя информации
SUB	Substitute (Заместитель). Используется для подмены ошибочного или недопустимого символа.
ESC	Escape (Расширение). Используется для расширения кода, указывая на то, что последующий символ имеет альтернативное значение.
(sp)	Space (Пробел). Непечатаемый символ для разделения слов или перемещения механизма печати или курсора дисплея вперед на одну позицию.
DEL	Delete (Удаление). Используется для удаления (стирания) предыдущего знака в сообщении

Отправка одиночных символов для управления светодиодом

Начнем с написания программы, которая использует простое сравнение символов для управления состоянием светодиода. Вы будете отправлять символ «1», чтобы включить светодиод, и «0», чтобы выключить его. Соедините светодиод с контактом 9 платы ***Arduino***, как показано на Рисунок 59.

Как мы уже говорили, при отправке одиночного символа необходимо выполнить простое символьное сравнение. Каждый раз, когда символ окажется в буфере, мы сравниваем его с «0» или «1» и выполняем нужные действия. Загрузите в ***Arduino*** код Листинг 22 и поэкспериментируйте с отправкой нуля или единицы из последовательного терминала.

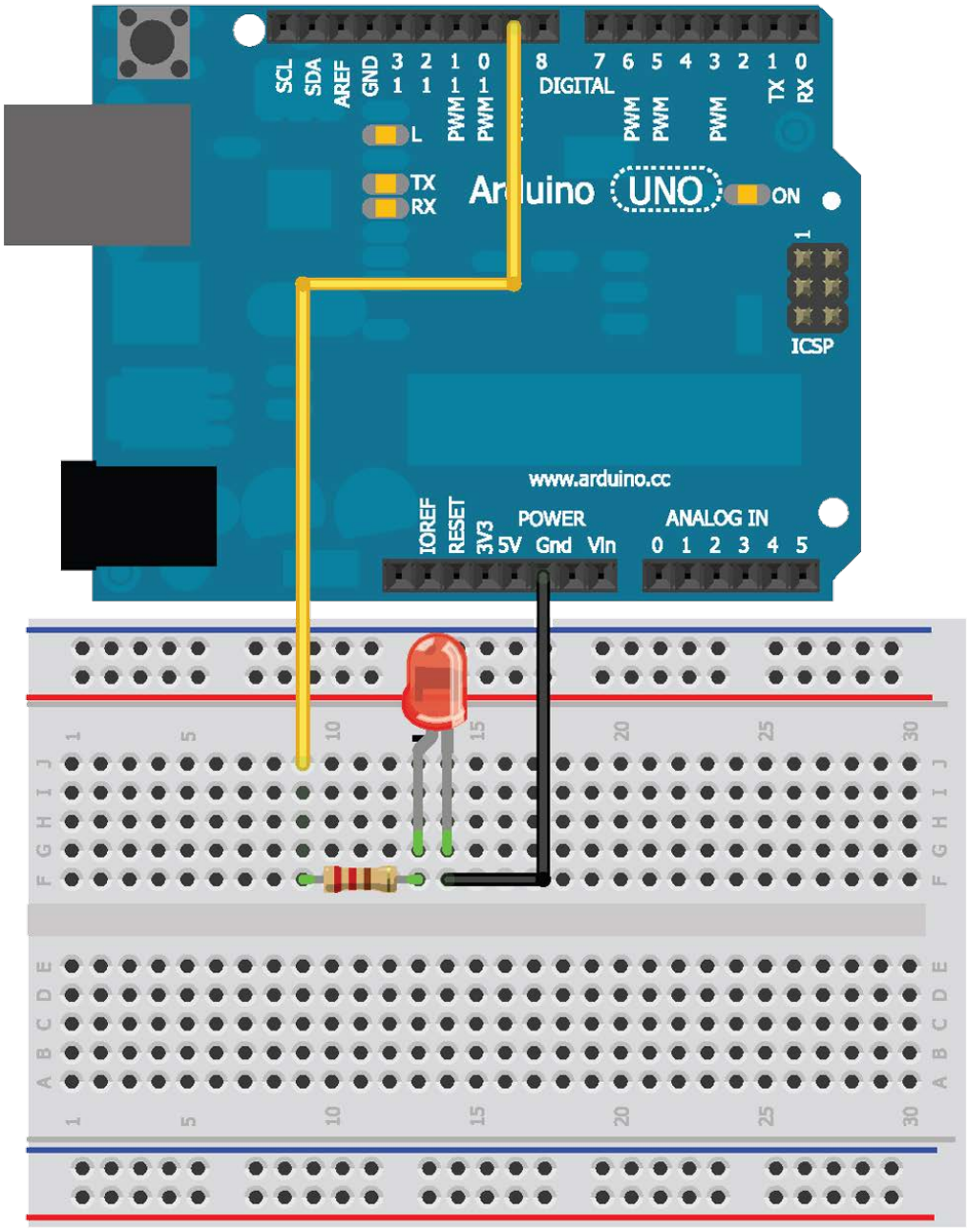

Рисунок 59. Подсоединение светодиода к контакту 9 платы Arduino

Листинг 22. Управление светодиодом последовательной отправкой символов — single_char_control.ino

```
//Переключение состояния светодиода отправкой одиночного символа
const int LED=9;

char data; //Переменная для хранения получаемого символа
```

```
void setup()
{
  Serial.begin(9600); //Инициализация последовательного порта на скорости 9600
  pinMode(LED, OUTPUT);
}

void loop()
{
  //Если в буфере есть символ
  if (Serial.available() > 0)
  {
    data = Serial.read(); //Чтение байта из буфера
    //Включение светодиода
    if (data == '1')
    {
      digitalWrite(LED, HIGH);
      Serial.println("LED ON");
    }
    //Выключение светодиода
    else if (data == '0')
    {
      digitalWrite(LED, LOW);
      Serial.println("LED OFF");
    }
  }

}
```

Обратите внимание, что вместо простого оператора `else` применен оператор `else if`. Это необходимо, потому что в опции терминала установлена отправка символа перевода строки при каждой передаче. Получив символ перевода строки, функция `Serial.read()` определит, что он не равен «`0`» или «`1`», в результате со светодиодом ничего не произойдет. Если бы мы использовали оператор `else`, то отправка и «`0`» и «`1`» приводила бы к отключению светодиода. При отправке «`1`» светодиод будет включен и немедленно выключен снова, когда будет получена последовательность `\n`!

Отправка последовательности цифр для управления RGB-светодиодом

Отправка одного управляющего символа позволяет управлять единственным цифровым контактом, но что делать, если требуются более сложные алгоритмы управления? В этом разделе мы рассмотрим передачу нескольких групп цифр, разделенных запятой, для управления множеством устройств. Чтобы разобраться в этой задаче, соберем схему подключения RGB-светодиода с общим катодом, как показано на Рисунок 60.

Для управления RGB-светодиодом мы посылаем три отдельные 8-битовые значения (0-255), чтобы задать яркость каждого цвета. Например, чтобы установить максимальную яркость всех цветов, нужно послать "255,255,255". Перечислим проблемы, возникающие при реализации данной задачи:

- необходимо различать цифры и запятые;
- последовательность символов нужно привести к целочисленному формату, чтобы использовать функцию `analogWrite()`;
- числовое значение может состоять из одной, двух или трех цифр.

К счастью, в ***Arduino*** IDE есть очень удобная функция для анализа последовательности цифр— `serial.parseInt()`. При каждом вызове этой функции она ожидает прихода в буфер последовательности цифр и затем преобразует ее в целое число. Обнаружив символ запятой, функция считывает первое и второе значения, третье значение будет считано, когда поступит символ перевода строки.

Чтобы посмотреть, как действует эта функция, загрузите программу Листинг 23 на плату ***Arduino***.

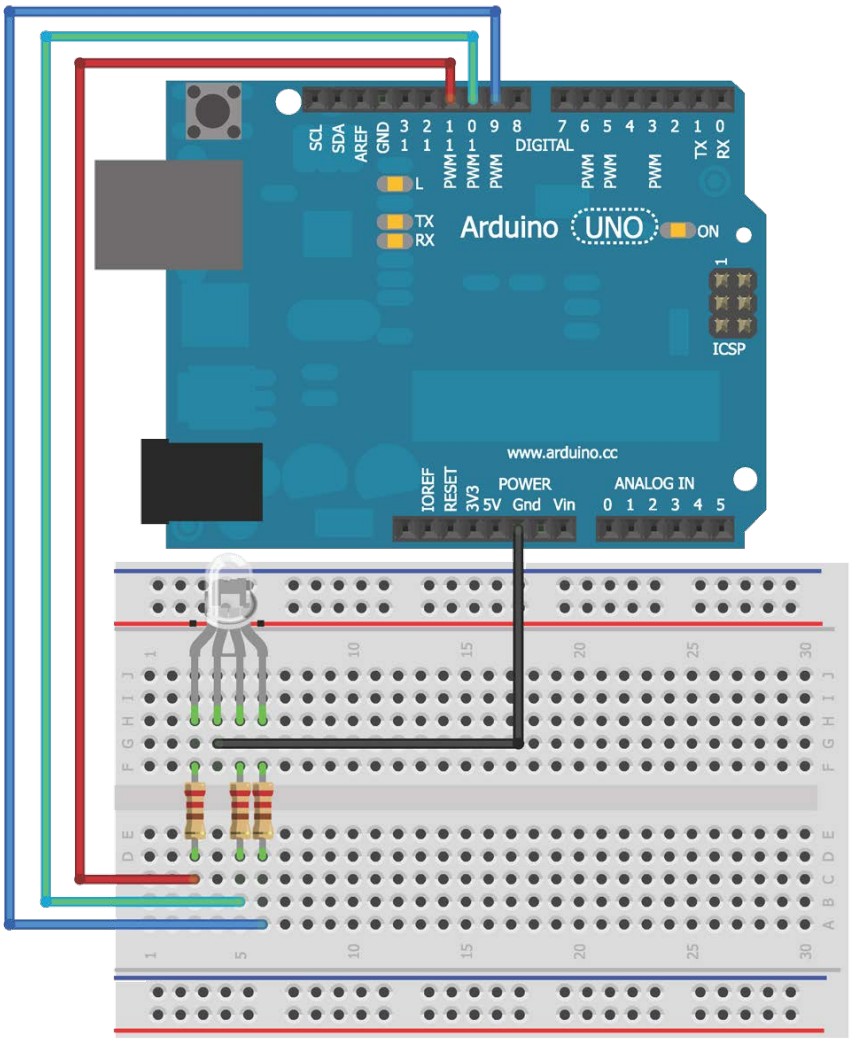

Рисунок 60. Подключение RGB-светодиода к Arduino

Листинг 23. Последовательное управление RGB-светодиодом — list_control.ino

```
// Отправка многосимвольных значений

// Константы выводов RGB-светодиода
const int RED   =11;
const int GREEN =10;
const int BLUE  =9;

// Переменные значений выводов RGB
int rval = 0;
int gval = 0;
int bval = 0;

void setup()
{
  Serial.begin(9600); //Инициализация последовательного порта на скорости 9600

  //Установить контакты на выход OUT
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
}
```

```
void loop()
{
  // Пока в буфере есть данные
  while (Serial.available() > 0)
  {
    rval = Serial.parseInt();    // Первое число
    gval = Serial.parseInt();    // Второе число
    bval = Serial.parseInt();    // Третье число

    if (Serial.read() == '\n')  // Конец передачи
    {
      //Установка яркости светодиода
      analogWrite(RED, rval);
      analogWrite(GREEN, gval);
      analogWrite(BLUE, bval);
    }
  }
}
```

Программа осуществляет поиск трех целочисленных значений, пока не обнаружит символ перевода строки (\n). Как только это произойдет, полученные значения переменных используются для установки яркости светодиодов. Откройте монитор последовательного порта и введите три значения от 0 до 255, разделенные запятой, например, "200,30,180". Попробуйте получить различные цвета, вводя разные цифры.

6.7. Создаем компьютерное приложение

В конце концов, вам надоест «общение» с платой ***Arduino*** через монитор последовательного порта ***Arduino*** IDE. Гораздо удобнее работать с приложением, написанным на каком-либо языке программирования, имеющем библиотеки для обмена по последовательному порту. Вы можете использовать ваш любимый язык программирования для написания программы, которая посылает последовательные команды к плате ***Arduino*** и получает данные, передаваемые от ***Arduino*** к компьютеру.

В этой книге выбран язык программирования Processing, потому что он очень похож на язык ***Arduino***, с которым вы уже знакомы. На самом деле, язык программирования ***Arduino*** основан на Processing! Другие популярные языки программирования, для которых существуют обширные библиотеки для работы с последовательным портом, — Python, PHP, Visual Basic, C и т. п. Сначала выясним, как читать передаваемые последовательные данные в Processing, а затем узнаем, как с его помощью создать простой графический интерфейс пользователя (GUI) для отправки команд на плату ***Arduino***.

6.7.1. Интерфейс Processing

Интерфейс программирования на Processing достаточно прост, и он похож на тот, с которым вы уже знакомы при программировании ***Arduino***. В этом разделе вы установите Processing, а затем напишете простой графический интерфейс для отображения данных, передаваемых с платы ***Arduino*** через последовательный порт. После этого вы реализуете связь в обратном направлении, чтобы управлять платой ***Arduino*** из приложения, установленного на вашем компьютере.

6.7.2. Установка Processing

Сначала необходимо установить Processing на вашей машине. Делается это так же, как и в главе 1, когда устанавливали ***Arduino*** IDE. Зайдите на страницу https://processing.org/download/ (или найдите ссылку на скачивание на https://www.exploringarduino.com/) и скачайте заархивированный пакет в соответствии с вашей операционной системой. Затем распакуйте его и программа будет готова к работе! Запустите Processing, и вы должны увидеть окно программы, которое выглядит как на рис. 6.12.

6.7.3. Плата *Arduino* управляет приложением на Processing

Для первого эксперимента с приложением на Processing подключим к плате ***Arduino*** потенциометр, чтобы управлять цветом окна на вашем компьютере. Соберите схему, изображенную на Рисунок 56.

Плата ***Arduino*** должна послать аналоговые значения от потенциометра к компьютеру. Тот факт, что вы сейчас отправляете последовательные данные в приложение на Processing, не имеет никакого влияния, как вы передаете их.

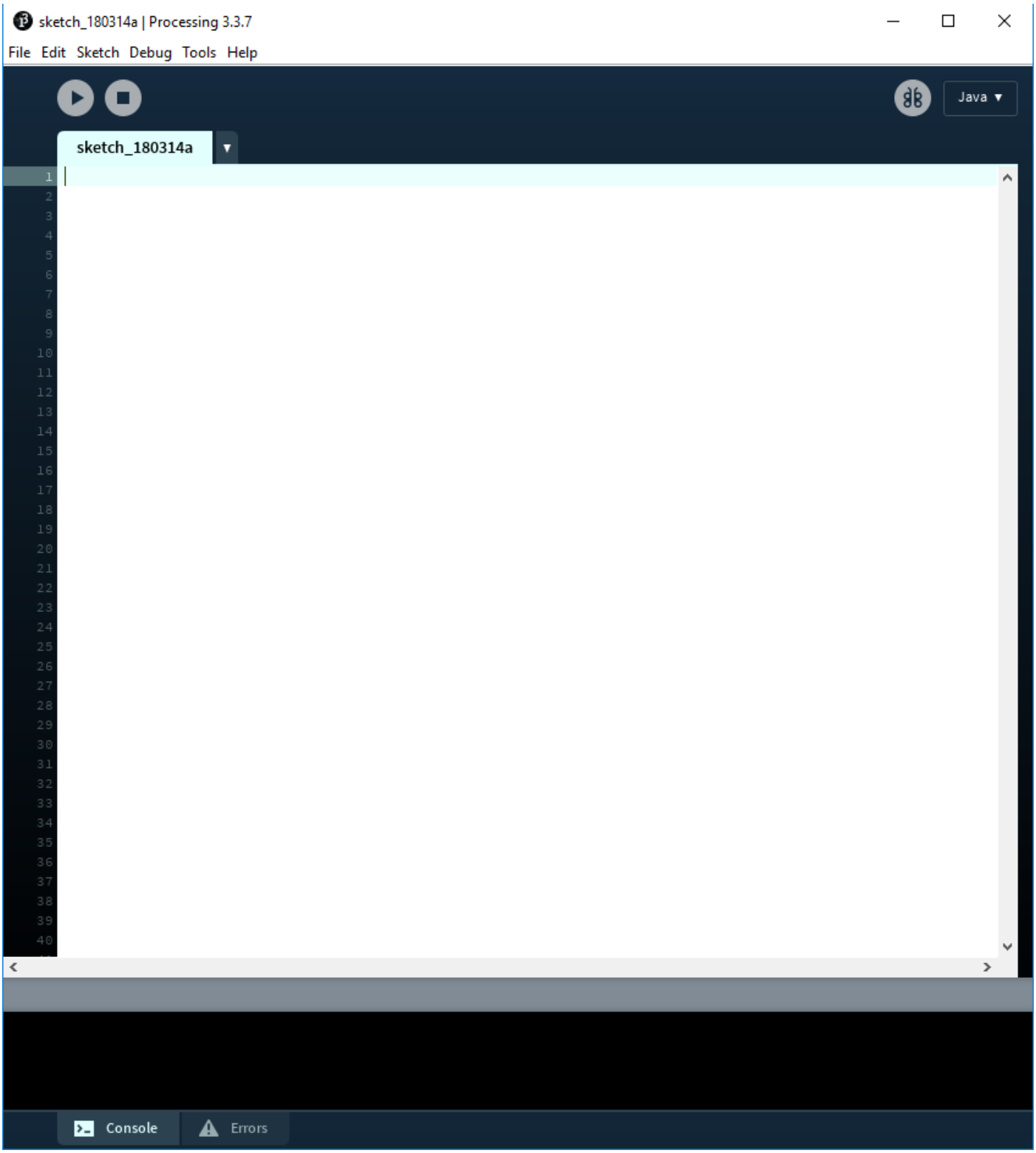

Рисунок 61. Графический интерфейс Processing. Узнаете?

Скопируйте код из Листинг 24 и загрузите его на плату ***Arduino***. ***Arduino*** посылает обновленное значение от потенциометра в последовательный порт компьютера каждые 50 мс. Если отправлять данные быстрее, приложение на Processing не будет успевать их обрабатывать и буфер последовательного порта на вашем компьютере переполнится.

Листинг 24. Программа для отправки данных в компьютер — pot_to_processing/arduino_read_pot

```
// Отправка данных от потенциометра на компьютер

const int POT=0;        // Подключение потенциометра к выводу A0

int val;                // Переменная для хранения значения потенциометра

void setup()
{
  Serial.begin(9600);  // Инициализация последовательного порта на скорости 9600
}

void loop()
{
  val = map(analogRead(POT), 0, 1023, 0, 255);     // Чтение значения потенциометра
  Serial.println(val);                             // Отправка значения
  delay(50);                                       // Задержка 50 мс
}
```

Теперь напишем код для обработки входящих данных. Программа из Листинг 25 считывает данные из буфера последовательного порта и регулирует яркость цвета на экране вашего компьютера в зависимости от полученного значения. Скопируйте Листинг 25 и внесите одно важное изменение. В программе на Processing должен быть указан конкретный последовательный порт получения данных. Замените "COM3" на номер вашего последовательного порта. Помните, что в Linux и Mac он будет выглядеть примерно как /dev/ttyUSB0. Если вы не уверены, скопируйте точное название порта из ***Arduino*** IDE.

Листинг 25. Код Processing для получения данных и изменения цвета экрана — pot_to_processing/processing_display_color

```
//Программа на Processing для чтения переменной и изменения цвета экрана

//Подключение и инициализация библиотеки Serial
import processing.serial.*;
Serial port;

float brightness = 0; // Переменная для хранения значения потенциометра

void setup()
{
  size(500,500);                    // Размер окна
  port = new Serial(this, "COM3", 9600); // Инициализация последовательного порта

  port.bufferUntil('\n');             // Символ конца строки
}

void draw()
{
  background(0,0,brightness); // Перерисовать окно
}

void serialEvent (Serial port)
{
  brightness = float(port.readStringUntil('\n')); // Получить переменную
}
```

После загрузки кода в Processing IDE и правильной инициализации последовательного порта убедитесь, что монитор последовательного порта ***Arduino*** IDE не открыт. К конкретному последовательному порту может иметь доступ только одна программа на компьютере. В окне Processing-приложения нажмите кнопку **Выполнить** (кнопка с изображением треугольника в верхнем левом углу окна), при этом должно появиться небольшое окно. При повороте движка потенциометра цвет окна должен меняться от черного к синему (Рисунок 62).

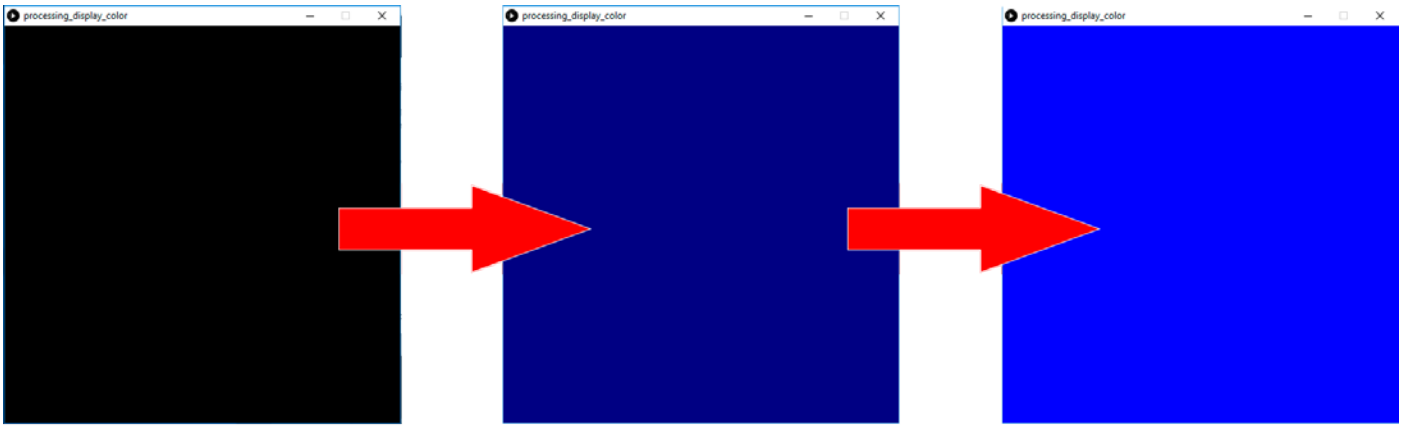

Рисунок 62. Изменение цвета окна Processing-приложения

Теперь, когда вы видели результат, для лучшего понимания рассмотрим, как это работает. В отличие от ***Arduino***, библиотека `serial` не импортируется автоматически. Объявив `import processing.serial.*` и `Serial port`, вы импортируете библиотеку `serial` и создаете объект `serial` под названием `port`.

Как и в ***Arduino***, в ***Processing*** тоже есть функция `setup()`, которая работает только в начале программы. В этом примере функция `setup()` задает последовательный порт и создает окно размером 500x500 пикселов командой `size (500,500)`. Команда `port = new serial (this, "COM3", 9600)` сообщает Processing о создании экземпляра последовательного порта. Экземпляр (под названием `port`) будет работать и общаться на COM3 (или другом последовательном порту) со скоростью 9600 бод. Скорость обмена данными между платой ***Arduino*** и программой на компьютере должна быть задана одинаковой, в противном случае возникнут ошибки. Команда `port.bufferUntil('\n')` сообщает Processing-приложению о буферизации последовательного ввода до получения символа новой строки (`'\n'`).

Вместо `loop()` в Processing есть другие специальные функции. В нашем примере использованы функции `draw()` и `serialEvent()`. Функция `draw()` похожа на `loop()` в ***Arduino***, она работает непрерывно и обновляет экран. Функция `background()` задает цвет окна, имеет три аргумента для установки значений красного, зеленого и синего компонентов цвета. В нашем примере значение с потенциометра управляет интенсивностью синего компонента, а красный и зеленый установлены в нуль. При желании программу можно переделать, чтобы регулировать интенсивность других компонентов. Поскольку интенсивности RGB цветов являются 8-битовыми значениями от 0 до 255, значения потенциометра масштабируются функцией `map()` перед передачей.

Функция `serialEvent()` вызывается всякий раз, когда выполняется условие `bufferUntil()`, указанное в функции `setup()`. При каждом появлении символа перевода строки срабатывает функция `serialEvent()`. Передаваемые данные считываются в виде строки с помощью `port.readStringUntil('\n')`. Строка представляет собой последовательность символов и ее необходимо преобразовать в число с плавающей точкой, вызвав функцию `float()`. Результат присваивается переменной `brigthness`, определяющей цвет фона окна приложения.

Чтобы остановить программу и закрыть последовательный порт, нажмите кнопку **Стоп** в окне Processing-приложения (это квадратик, расположенный рядом с кнопкой **Выполнить**).

ПРИМЕЧАНИЕ

SudoGlove — это перчатка-манипулятор для передачи команд на радиоуправляемый автомобиль и другую аппаратуру. Я разработал на Processing дисплей для графического отображения значений датчиков, удобный при отладке SudoGlove. Больше узнать об

этом можно на www.sudoglove.com. Скачать исходный код приложения можно по адресу www.jeremyblum.com/2011/03/25/processing-based-sudoglove-visual-debugger/. Этот исходный код есть также на сайте издательства Wiley.

6.7.4. Отправка данных из Processing-приложения в *Arduino*

Следующий шаг— передать данные из Processing-приложения на плату ***Arduino***. Подсоедините RGB-светодиод к плате ***Arduino***, как показано на Рисунок 60, и загрузите в нее программу считывания строки из трех значений, разделенных запятыми для установки красной, зеленой и синей интенсивности (см. Листинг 23). Теперь сделаем так, чтобы вместо отправки трехсимвольной строки из последовательного монитора можно было выбрать цвет, используя палитру цветов.

Загрузите и запустите код из Листинг 26 в Processing-приложении, не забыв установить свое значение для последовательного порта, как в предыдущем примере. Сопутствующие файлы из папки данных (data) будут загружены автоматически. Файл hsv.jpg (изображение для выбора цвета) есть среди электронных ресурсов к этой главе. Скачайте и поместите его в папку data в том же каталоге, что и программа. Processing-приложение по умолчанию сохраняет файлы проекта в папке Мои документы. Структура папки проекта показана на Рисунок 63.

Листинг 26. Processing-приложение для установки цвета RGB-светодиода — processing_control_rgb/processing_control_RGB

```
import processing.serial.*; //Подключение библиотеки serial
PImage img;              //Объект «картинка»
Serial port;             //Последовательный порт объекта

void setup()
{
  size(640,256); //Размер изображения HSV
  img = loadImage("hsv.jpg"); //Загрузка фоновой картинки
  port = new Serial(this, "COM9", 9600); //Открыть последовательный порт


}

void draw()
{
  background(0);    //Черный фон
  image(img,0,0);   //Наложение изображения
}

void mousePressed()
{
  color c = get(mouseX, mouseY); //Получить RGB-цвет по позиции курсора мыши
  String colors = int(red(c))+","+int(green(c))+","+int(blue(c))+"\n"; //Сформировать строку значений цвета
  print(colors); //Вывод для отладки
  port.write(colors); //Отправка переменной в Arduino
}
```

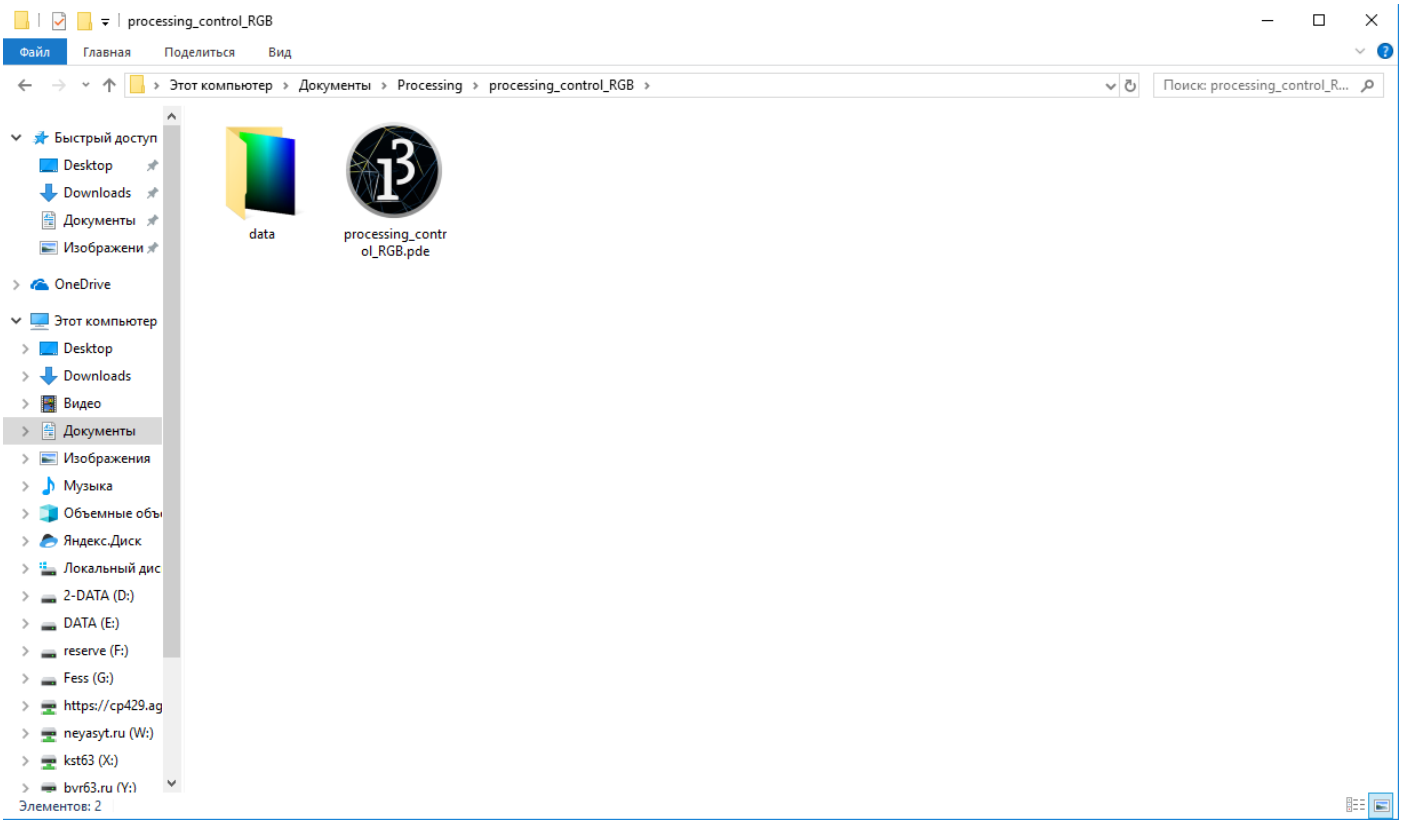

Рисунок 63. Структура папки проекта

При запуске программы вы увидите всплывающее окно, показанное на Рисунок 34. Укажите желаемый оттенок, и его значения RGB будут переданы в плату ***Arduino*** для управления цветом RGB-светодиода. Команды, отправляемые в ***Arduino***, выводятся также в монитор последовательного порта, чтобы облегчить отладку приложения.

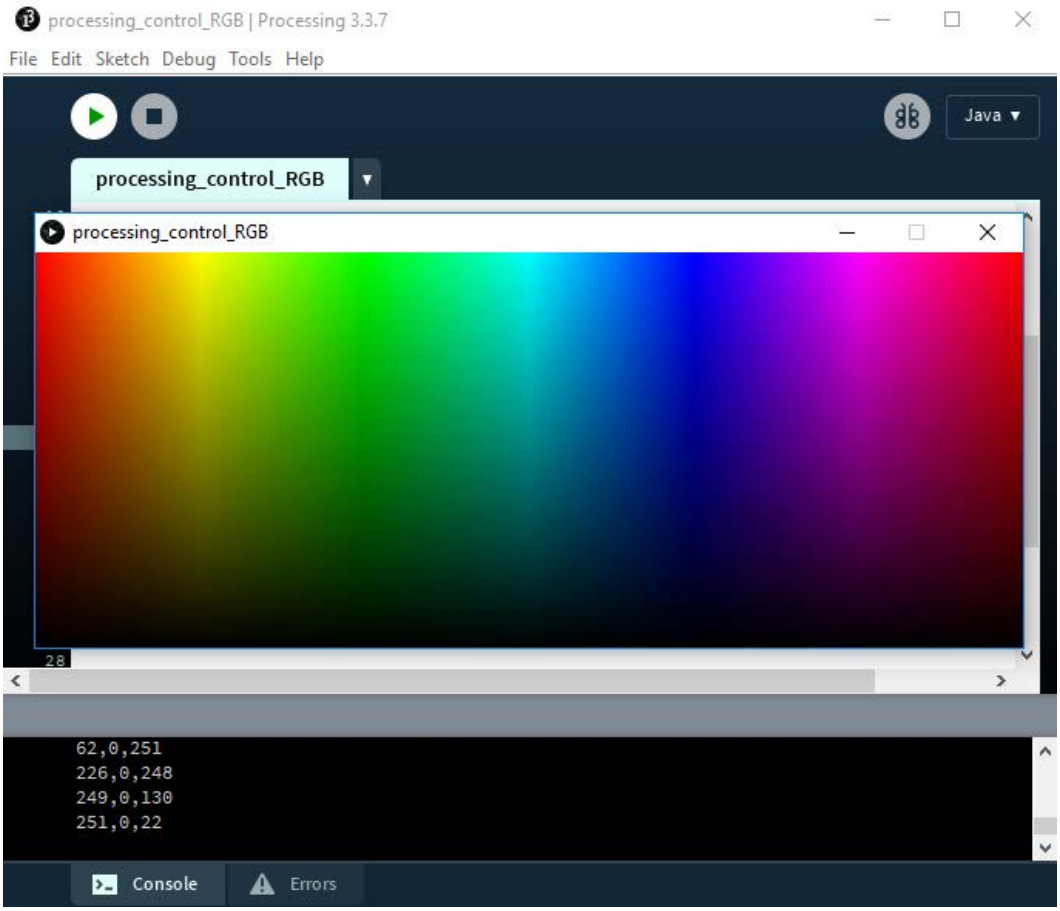

Рисунок 64. Экран выбора цвета в Processing

Теперь посмотрим на код программы и проанализируем, как он работает. Как и раньше, импортируется библиотека `serial` и создается объект `port`. Создается также объект `PImage` с именем `img`. Он будет формировать фоновое изображение. В процедуре `setup()` инициализируется последовательный порт, создается окно в размер изображения, а само изображение импортируется в объект `img` вызовом функции `img=LoadImage("hsv.jpg")`.
В функции `draw()` изображение загружается в окно командой `image(img,0,0)` с двумя аргументами: `Img`— само изображение, а `0,0` — координаты для размещения изображения (верхний левый угол окна приложения).
При нажатии кнопки мыши вызывается функция `mousePressed()`. Цвет пиксела, где вы щелкнули мышью, сохраняется в объекте `color` с именем `c`. Метод `get()` сообщает приложению, откуда можно получить цвет (в данном случае из координат курсора мыши). Затем преобразуем значение в строку, состоящую из целых значений красного, зеленого и синего компонентов цвета. Эти значения также выводятся в монитор последовательного порта.
Включите плату ***Arduino*** и загрузите в нее код из Листинг 23. Запустите ***Processing***-приложение и настройте цвет светодиода, подключенного к ***Arduino***, выбирая цвет из палитры.

6.8. Изучаем особенности работы с *Arduino Leonardo* (и другими платами на основе процессора 32U4)

Leonardo, как и другие платы, построенные на микроконтроллерах с поддержкой интерфейса USB, обладают уникальной способностью эмулировать такие устройства, как, например, клавиатура и мышь. Далее рассмотрим этот вопрос подробнее. При реализации подобных функций нужно соблюдать осторожность. Если, например, вы напишете программу, которая эмулирует мышь и перемещает курсор по экрану, то могут возникнуть проблемы при нажатии кнопки **«Загрузить»** в ***Arduino IDE***. В этом разделе мы опишем несколько приемов, которые позволят избежать подобных проблем.

> ***СОВЕТ***
> *Если плата «зависла» в режиме эмуляции мыши или клавиатуры, чтобы перепрограммировать ее, нажмите и отпустите кнопку* `Reset`*, удерживая нажатой кнопку* ***«Загрузить»*** *в* ***Arduino*** *IDE.*

При первом подключении платы Leonardo к компьютеру необходимо установить драйверы, как и для ***Arduino Uno*** (см. Глава 1. Начало работы, переключаем светодиод из *Arduino*). Инструкции для установки Leonardo можно найти по адресам https://www.arduino.cc/en/Guide/ArduinoLeonardoMicro#toc8 или https://www.exploringarduino.com/

6.8.1. Эмуляция клавиатуры

Благодаря уникальной возможности Leonardo эмулировать USB-устройства, плату ***Arduino*** легко превратить в клавиатуру. В результате можно отправлять комбинации клавиш в виде команд компьютеру или записывать данные непосредственно в файл, открытый на компьютере.
Плата Leonardo может эмулировать USB-клавиатуру, отправляя коды нажатия клавиш и их комбинаций. Рассмотрим эти возможности. Напишем простую программу, которая записывает данные от нескольких аналоговых датчиков в файл, разделяя их символом двоеточия (формат CSV), который затем можно открыть в Excel или Google SpreadSheeets для построения графика.
Вызовем любой текстовый редактор, создадим пустой документ и сохраним его в файле с расширением csv. Для этого в диалоговом окне **«Сохранить»** можно выбрать тип файла **«Все файлы»** и вручную ввести имя файла с расширением, например, data.csv.
Затем соберем простую схему, как показано на Рисунок 65. Устройство будет следить за показаниями температуры и освещенности, поступающими от аналоговых датчиков, с которыми мы уже встречались в *Глава 3*. Опрос аналоговых датчиков. Кроме датчиков, в схеме есть кнопка для включения и выключения записи и светодиод, который будет показывать, идет ли в настоящее время запись данных.

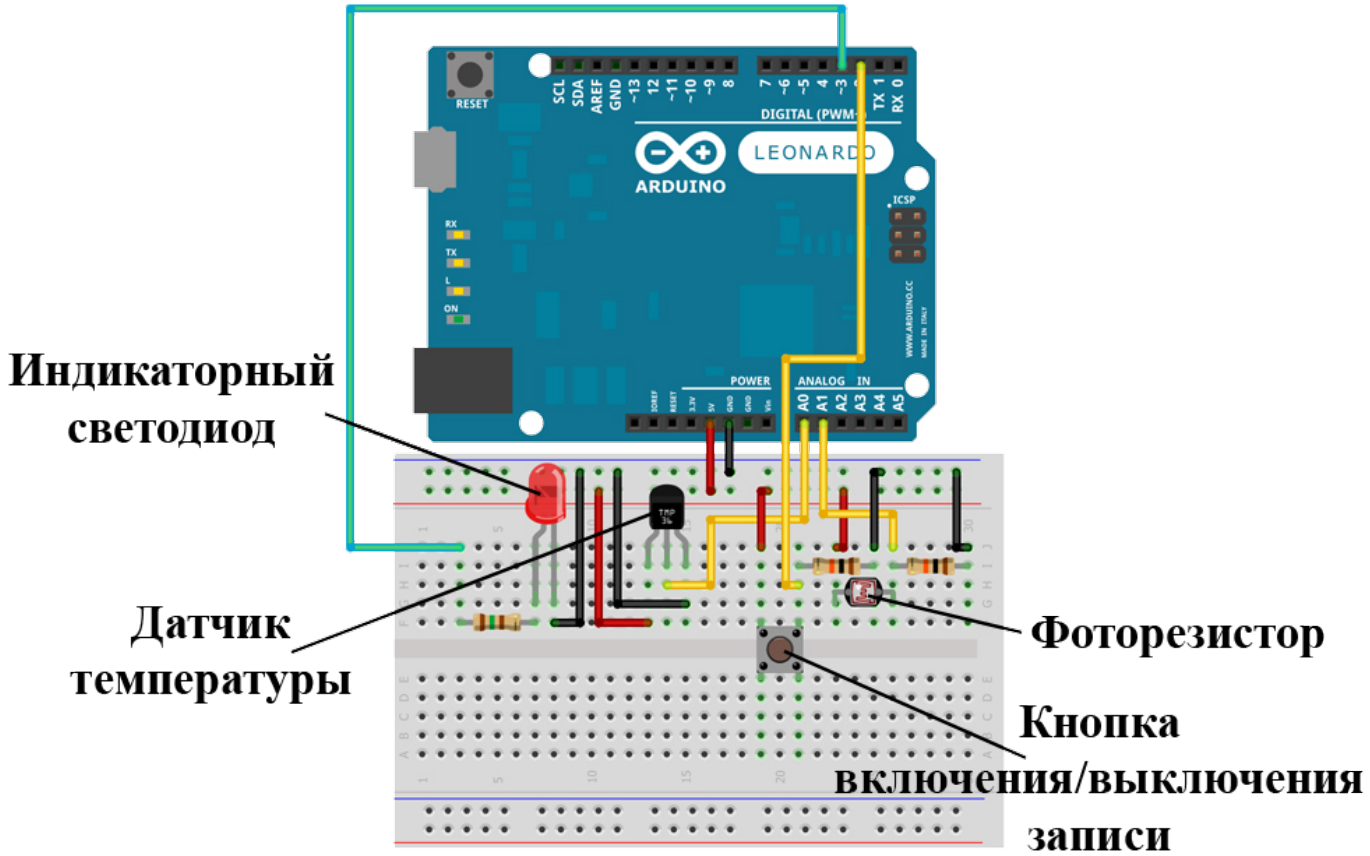

Рисунок 65. Схема подключения датчиков температуры и освещенности

Кнопка для включения/выключения записи снабжена функцией устранения дребезга (см. пункт 2.8. Устранение «дребезга» кнопок). В процессе записи плата ***Arduino*** опрашивает датчики и раз в секунду отправляет в компьютер данные, разделенные запятой. При этом загорается светодиодный индикатор. Поскольку ***Arduino*** постоянно опрашивает состояние кнопки, то задержку 1000 мс перед каждой отправкой данных формирует не функция `delay()`, а функция `millis()`, которая возвращает количество миллисекунд с последнего сброса платы ***Arduino***. Вы можете посылать данные каждый раз, когда функция `millis()` выдает значение, кратное 1000 мс, фактически создавая задержку между посылками, равную 1 с. Это реализует оператор остатка деления по модулю (%). Если, например, вы выполните операцию `1000%1000`, то результат будет равен нулю, т. к. 1000/1000= 1 с нулевым остатком. `1500%1000` вернет 500, потому что 1500/1000 = 1 с остатком 500. Выполняя деление по модулю `millis()` на 1000, получим нулевой результат каждый раз, когда `millis()` выдает значение, кратное 1000, т. е. каждую секунду.
Изучите код Листинг 27 и загрузите его на плату ***Arduino Leonardo***. Убедитесь, что вы выбрали опцию ***Arduino Leonardo*** из меню **Инструменты → Board** в ***Arduino IDE***.

Листинг 27. Запись данных освещенности и температуры — csv_logger.ino

```
//Запись данных температуры и освещенности
const int TEMP   =0;            //Датчик температуры к аналоговому входу 0
const int LIGHT  =1;            //Датчик освещенности к аналоговому входу 1
const int LED    =3;            //Светодиод к выводу 13
const int BUTTON =2;            //Кнопка к выводу 2

boolean lastButton = LOW;       //Последнее состояние кнопки
boolean currentButton = LOW;    //Текущее состояние кнопки
boolean running = false;        //По умолчанию запись выключена
int counter = 1;                //Индекс записываемых данных

void setup()
{
```

```
  pinMode (LED, OUTPUT);          //Контакт светодиода как выход OUTPUT
  Keyboard.begin();               //Запуск эмуляции клавиатуры
}

void loop()
{
  currentButton = debounce(lastButton);    //Чтение начального состояния кнопки
  if (lastButton == LOW && currentButton == HIGH)//Если она была нажата ...
    running = !running;                    //Переключить статус записи

  lastButton = currentButton;              //Установить статус кнопки

  if (running)                             //Если логгер запущен
  {
    digitalWrite(LED, HIGH);               //Включить светодиод
    if (millis() % 1000 == 0)              //Если прошло 1000 мс
    {
      int temperature = analogRead(TEMP);  //Чтение данных с датчика температуры
      int brightness = analogRead(LIGHT);  //Чтение данных
                                           //с датчика освещенности
      Keyboard.print(counter);             //Вывод индекса данных
      Keyboard.print(",");                 //Вывод разделителя
      Keyboard.print(temperature);         //Вывод температуры
      Keyboard.print(",");                 //Вывод разделителя
      Keyboard.println(brightness);        //Вывод освещенности
                                           //и символа новой строки
      counter++;                           //Инкремент индекса
    }
  }
  else
  {
    digitalWrite(LED, LOW);                //Если логгирование не работает
                                           //светодиод выключен
  }
}

/*
* Функция противодребезговой защиты
* Передача предыдущего состояния кнопки, и получение обратно текущее состояние
кнопки подавления дребезга контактов.
*/
boolean debounce(boolean last)
{
  boolean current = digitalRead(BUTTON);   //Чтение состояния кнопки
  if (last != current)                     //Если состояние изменилось...
  {
    delay(5);                              //Ждем 5 мс
    current = digitalRead(BUTTON);         //Повторное чтение состояния кнопки
  }
  return current;                          //Возврат текущего состояния кнопки
}
```

Подробнее рассмотрим некоторые новые функциональные возможности, реализованные в этой программе. Как и ранее при инициализации последовательного порта, клавиатура инициализируется

оператором Keyboard.begin() в функции setup(). В цикле loop() ***Arduino*** проверяет состояние кнопки и запускает программу устранения дребезга. При нажатии на кнопку значение переменной статуса записи инвертируется. Это достигается применением оператора «!» к переменной running. Когда программа находится в режиме записи, отправка данных выполняется раз в 1000мс, благодаря описанному ранее приему. Функции эмулированной клавиатуры и последовательного порта очень похожи. Команда Keyboard.print() отправляет строку в компьютер. После получения данных аналоговых датчиков программа передает данные в компьютер в виде нажатия клавиш. Благодаря команде Keyboard.println() ***Arduino*** эмулирует нажатие клавиши <Enter> (или <Return>) после отправки строки. Индекс данных и оба аналоговых значения при выводе разделяются запятой.

Установите курсор на строке в текстовом документе и нажмите кнопку включения режима записи. Вы должны увидеть, что документ начнет заполняться данными. Закройте ладонью датчик освещенности или возьмите в руку датчик температуры, значения должны измениться. Затем нажмите кнопку еще раз, чтобы остановить отправку данных. После сохранения файла его можно импортировать в электронную таблицу, чтобы построить график. Как это сделать, показано в видеоуроке к главе.

> ***ПРИМЕЧАНИЕ***
> *Посмотрите демонстрационный видеоклип со страницы* *https://www.exploringarduino.com/content/ch6/*

6.8.2. Отправка команд для управления компьютером

Плата Leonardo пригодна и для эмуляции нажатия комбинаций клавиш. На компьютерах с операционной системой Windows нажатие комбинации клавиш <Windows>+<L> блокирует экран компьютера (в Linux существует комбинация <Ctrl>+<Alt>+<L>). Можно, например, по сигналу от датчика освещенности заблокировать компьютер, когда выключается свет. В OS X для блокировки компьютера предусмотрены комбинации <Control>+<Shift>+<Eject> или <Control>+<Shift>+ +<Power>, которые Leonardo не может сформировать, т. к. невозможно смоделировать нажатие клавиш <Eject> и <Power>. Рассмотрим, как заблокировать компьютер с Windows. Подойдет схема, показанная на Рисунок 65, хотя будет использоваться только датчик.
Запустите предыдущую программу при нескольких различных уровнях освещенности и посмотрите на изменение показаний датчика. С учетом полученных данных нужно выбрать пороговое значение освещенности, ниже которого компьютер следует заблокировать (в моей комнате при выключенном свете показания датчика равны 300, а при включенном — 700, я выбрал пороговое значение 500). Когда значение от датчика станет ниже порогового, на компьютер будет отправлена команда блокировки. Возможно, для вашего помещения потребуется другое значение порога.
Загрузите код Листинг 28 на плату ***Arduino***. Подберите порог срабатывания путем анализа данных при различной освещенности. Если порог окажется неправильным, то компьютер может быть заблокированным, как только вы запустите его!

Листинг 28. Блокировка компьютера по сигналу отдатчика освещенности — lock_computer.ino

```
//Блокировка компьютера при выключении света
const int LIGHT     =1;                    //Датчик освещенности на контакт 1
const int THRESHOLD =500;                  //Значение с датчика освещенности
                                           //для блокировки компьютера

void setup()
{
  Keyboard.begin();
}

void loop()
{
  int brightness = analogRead(LIGHT);     //Чтение данных датчика
```

```
  if (brightness < THRESHOLD)
  {
    Keyboard.press(KEY_LEFT_GUI);
    Keyboard.press('l');
    delay(100);
    Keyboard.releaseAll();
  }
}
```

После запуска программы попробуйте выключить свет в комнате. Ваш компьютер должен заблокироваться.

> ***ПРИМЕЧАНИЕ***
> *Вы можете посмотреть демонстрационный видеоклип, расположенный на странице https://www.exploringarduino.com/content/ch6/*

В этом примере реализованы две новые функции для эмулятора клавиатуры: `Keyboard.press()` и `Keyboard.releaseAll()`. Запуск `Keyboard.press()` эквивалентен удержанию клавиши нажатой. Если вы хотите сэмулировать нажатие клавиш `<Windows>` и `<L>`, запустите `Keyboard.press()` для каждой клавиши. Выдержав паузу, вызовите функцию `Keyboard.releaseAll()`, чтобы завершить нажатие комбинации клавиш. Список специальных клавиш можно найти на сайте https://www.arduino.cc/en/Reference/KeyboardModifiers.

6.8.3. Эмуляция мыши

С помощью двухкоординатного джойстика и нескольких кнопок можно превратить плату ***Arduino*** Leonardo в мышь. Джойстик будет перемещать курсор мыши, а кнопки будут выполнять функции левой, средней и правой кнопок мыши. Как и для клавиатуры, в языке ***Arduino*** есть много встроенных функций, позволяющих реализовать функциональность мыши.

Сначала соберите схему с джойстиком и кнопками, как показано на Рисунок 66. Не забывайте, что кнопки нужно снабдить подтягивающими резисторами. Джойстик подключается к аналоговым выводам 0 и 1. Джойстики содержат два потенциометра, присоединенные к рукоятке. При перемещении рукоятки джойстика в направлении x меняется сопротивление одного из потенциометров, в направлении y — другого.

На Рисунок 66 изображен джойстик ***SparkFun***, но подойдет любой (в демонстрационном видеоклипе показан джойстик ***Parallax***). В зависимости от типа джойстика, возможно, потребуется скорректировать диапазон значений функцией `map()` или поменять x и y в коде программы.

Собрав схему, можно загрузить программу на плату **Leonardo**. Скопируйте и запустите код из Листинг 29 и поуправляйте курсором с помощью джойстика и кнопок; курсор на экране компьютера должен реагировать соответствующим образом.

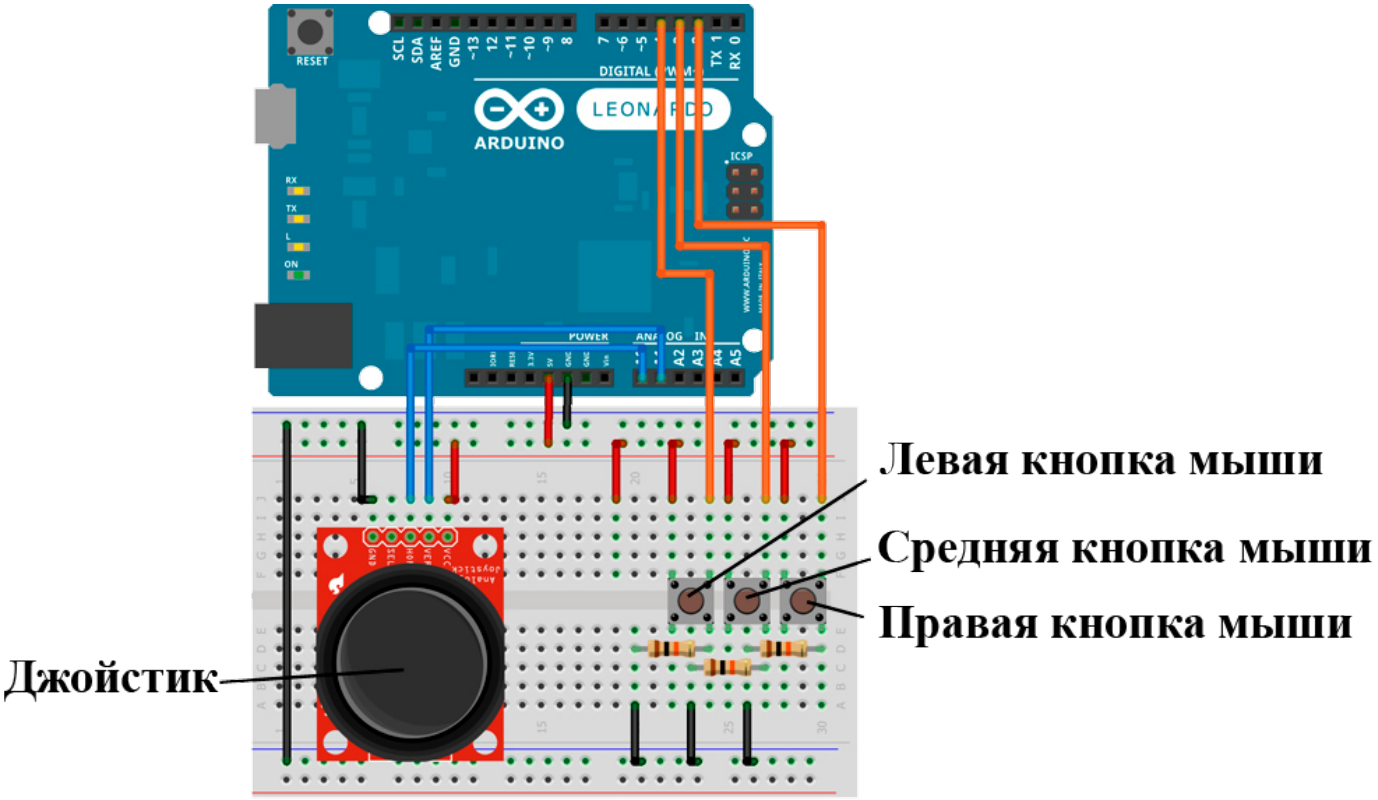

Рисунок 66. Схема мыши на основе джойстика и платы Leonardo

Листинг 29. Программа управления курсором мыши для Leonardo — mouse.ino

```
// Создаем мышь!

const int LEFT_BUTTON   =4;          //Вход для левой кнопки мыши
const int MIDDLE_BUTTON =3;          //Вход для средней кнопки мыши
const int RIGHT_BUTTON  =2;          //Вход для правой кнопки мыши
const int X_AXIS        =0;          //Аналоговый вход для оси x джойстика
const int Y_AXIS        =1;          //Аналоговый вход для оси y джойстика

void setup()
{
  Mouse.begin();
}

void loop()
{
  int xVal = readJoystick(X_AXIS);    // Получить отклонение
  //джойстика по оси x
  int yVal = readJoystick(Y_AXIS);    // Получить отклонение
  // джойстика по оси y

  Mouse.move(xVal, yVal, 0);          //Перемещаем мышь

  readButton(LEFT_BUTTON, MOUSE_LEFT);     // Чтение состояния левой кнопки
  readButton(MIDDLE_BUTTON, MOUSE_MIDDLE); // Чтение состояния средней кнопки

  readButton(RIGHT_BUTTON, MOUSE_RIGHT);   // Чтение состояния правой кнопки

  delay(5);                                //Это контролирует отзывчивость
}
```

```
//Чтение значения джойстика, масштабирование
int readJoystick(int axis)
{
  int val = analogRead(axis);          //Чтение аналогового значения
  val = map(val, 0, 1023, -10, 10);    //чтение карты и масштабирование значения

  if (val <= 2 && val >= -2)           //Создание мертвой зоны для остановки
дрейфа мыши
    return 0;

  else                                 //Вернуть значение
    return val;
}

// Чтение состояния кнопок и отправка команд мыши
void readButton(int pin, char mouseCommand)
{
  //Если кнопка нажата, эмулируем нажатие, если она еще не была нажата
  if (digitalRead(pin) == HIGH)
  {
    if (!Mouse.isPressed(mouseCommand))
    {
      Mouse.press(mouseCommand);
    }
  }
  //Отпустить нажатие мыши
  else
  {
    if (Mouse.isPressed(mouseCommand))
    {
      Mouse.release(mouseCommand);
    }
  }
}
```

Это, безусловно, один из наиболее сложных примеров из всех, рассмотренных до сих пор. Подробно разберем его, чтобы понять новые функции и ход выполнения программы.

В начале программы определены контакты для подключения кнопок и джойстика, в функции `setup()` подключена библиотека эмулятора мыши. В цикле `loop()` непрерывно опрашиваются контакты джойстика и выдаются значения для движения курсора мыши. Контакты кнопок также проверяются и при нажатии кнопки сигнал передается в компьютер.

Функция `readJoystick()` считывает и масштабирует значения от джойстика. По каждой координате джойстик выдает ряд значений от 0 до 1024, полученных от АЦП. Но курсор мыши перемещается по относительным координатам и передача `Mouse.move()` нулевого значения соответствует отсутствию движения по этой оси. Передача положительного значения для оси x будет перемещать курсор мыши вправо, а отрицательного— влево. Чем больше величина, тем дальше будет перемещаться курсор. Таким образом, в функции `readJoystick()` значения от 0 до 1023 масштабируем к диапазону от - 10 до 10.

Для устранения дрейфа предусмотрен небольшой запас в районе нуля, где курсор мыши должен быть неподвижен. Это связано с тем, что во время нахождения рукоятки джойстика в среднем положении фактическое значение может колебаться вокруг 512. Мы должны быть уверены, что при отпускании джойстика курсор мыши не будет двигаться самопроизвольно. Значения x и y передаются функции `Mouse.move()`, что приводит к перемещению курсора на экране. Третий аргумент функции

`Mouse.move()` определяет движение колеса прокрутки.
Функция `readButton()` служит для определения состояния каждой из трех кнопок. Функция определяет текущее состояние мыши с помощью команды `Mouse.isPressed()` и опрашивает мышь через функции `Mouse.press()` и `Mouse.release()`.

ПРИМЕЧАНИЕ

Демонстрационный видеоклип эмулятора мыши для управления компьютером с помощью джойстика можно посмотреть на странице https://www.exploringarduino.com/content/ch6/

Резюме

В этой главе вы узнали следующее:

- Как подключить плату ***Arduino*** к компьютеру через USB-преобразовагель последовательного порта.
- Как осуществляется преобразование USB-интерфейса в последовательный порт на различных платах ***Arduino***.
- Как можно отправлять данные с ***Arduino*** в компьютер через USB-интерфейс.
- Как форматировать отправляемые данные с помощью специальных символов.
- Что последовательные данные передаются в виде символа, который можно преобразовать в целые числа различными способами.
- Как отправлять данные в виде списков с разделителями-запятыми и преобразовывать их в команды с помощью встроенных функций.
- Как можно передать данные из ***Arduino*** внешнему приложению на языке Processing.
- Что можно отправлять данные из Processing-приложения на периферийные устройства, подключенные к ***Arduino***.
- Что плата ***Arduino*** Leonardo может эмулировать клавиатуру и мышь.

Глава 7. Сдвиговые регистры

Список деталей
Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;
- USB-кабель A - B (для ***Uno***);
- 8 красных светодиодов;
- 3 желтых светодиода;
- 5 зеленых светодиодов;
- 8 резисторов номиналом 220 Ом;
- сдвиговый регистр SN74HC595N в DIP-корпусе;
- инфракрасный датчик расстояния GP2Y0A41SK0F с кабелем;
- перемычки;
- макетная плата.

Электронные ресурсы к главе
На странице https://www.exploringarduino.com/content/ch7/ можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы https://www.wiley.com/go/exploringarduino в разделе **Downloads**.

Что вы узнаете в этой главе
Чем дальше вы продвигаетесь в создании новых устройств на основе ***Arduino***, тем чаще возникает мысль: «Что будет, если закончатся контакты платы ***Arduino***?» Например, в одном из популярных проектов плата ***Arduino*** управляет множеством мигающих светодиодов. Осветите свою комнату! Иллюминируйте свой компьютер! Украсьте светодиодами свою собаку! Последнее, пожалуй, перебор. Но проблема остается. Как быть, если вы захотите переключать 50 светодиодов (или управлять другими цифровыми выходами), а все контакты ввода-вывода уже задействованы? Вот тут и пригодятся сдвиговые регистры, позволяющие расширить возможности платы ***Arduino*** и отказаться от покупки более дорогого микроконтроллера с дополнительными контактами ввода-вывода. В этой главе вы узнаете, как работать со сдвиговыми регистрами, рассмотрите программное обеспечение и оборудование, необходимые для того, чтобы расширить возможности цифровых выходов платы ***Arduino***. Подробно описанные примеры познакомят вас со сдвиговыми регистрами и помогут разобраться в проектировании устройств с большим количеством цифровых выходов.
В этой главе, как и в большинстве предыдущих, в качестве платформы используем ***Arduino Uno***. Для экспериментов подойдет и любая другая плата ***Arduino***, но выбор всегда должен быть обоснованным и оптимально соответствовать конкретному проекту. Вы спросите, почему бы не взять ***Arduino*** с большим количеством контактов, например, ***Mega 2560*** или ***Due***? Конечно, это совершенно разумный способ для реализации устройств, где требуется множество контактов. Тем не менее вы, как инженеры, всегда должны помнить при проектировании обо всех нюансах. Когда вычислительной мощности ***Arduino Uno*** вполне хватает, но недостаточно цифровых выходов, можно просто добавить несколько сдвиговых регистров. Это будет дешевле и компактнее, чем выбор более мощной платы. Однако программный код окажется сложнее, и возможно потребуется больше времени для его отладки.

7.1. Что такое сдвиговый регистр

Сдвиговый регистр — это устройство, которое принимает поток последовательных битов и одновременно выводит их значения на параллельных контактах ввода-вывода. Регистры сдвига часто применяются для управления большим количеством светодиодов, например, семисегментными индикаторами или светодиодными матрицами. Прежде чем обсуждать взаимодействие сдвиговых регистров с ***Arduino***, посмотрим на рисунок 67, где изображены входы и выходы регистра сдвига. Далее будет показано, как состояние входов влияет на выходы.

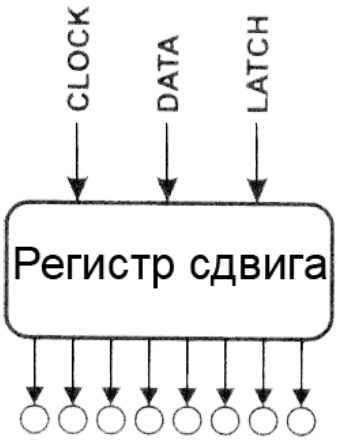

Рисунок 67. Входы и выходы регистра сдвига

Восемь кругов схематично изображают светодиоды, соединенные с восемью выходами сдвигового регистра. Три входа— это линии для подключения сдвигового регистра к плате ***Arduino***.

7.2. Последовательная и параллельная передача данных

Существуют два способа передачи нескольких битов данных. Напомним, что ***Arduino***, как и все микроконтроллеры, представляет собой цифровое устройство, т. е. «понимает» только «0» и «1». Если вы хотите управлять восьмью светодиодами, необходимо передать 8 бит информации. Ранее мы это делали в параллельном режиме с помощью функций digitalWrite() и analogWrite(). В качестве примера параллельной передачи данных, предположим, что нам необходимо включить 8 светодиодов, подключенных к 8 цифровым выходам. При этом все биты будут переданы на цифровые контакты примерно в одно время. В *главе 6* была описана последовательная передача данных, при которой за фиксированный интервал времени передается 1 бит данных. Сдвиговые регистры позволяют легко конвертировать последовательные и параллельные методы передачи данных. Эта глава посвящена регистрам последовательно-параллельного сдвига (SIPO), которые называют просто регистрами сдвига. С их помощью можно принимать данные последовательно, а выводить параллельно на выходы регистра. Кроме того, можно каскадировать сдвиговые регистры и управлять множеством цифровых выходов, используя всего три контакта ***Arduino***.

7.3. Сдвиговый регистр 74HC595

В качестве сдвигового регистра мы будем применять микросхему 74HC595, цоколевка которой изображена на рис. 68.

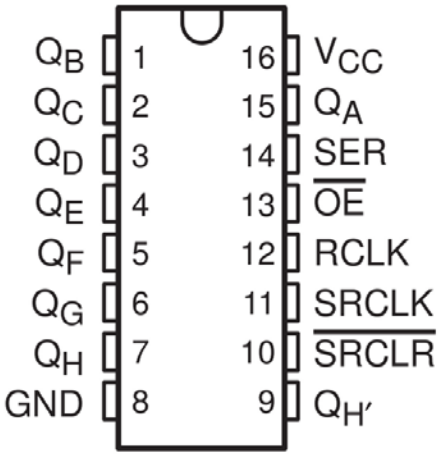

Рисунок 68. Цоколевка микросхемы 74HC595

7.3.1. Назначение контактов сдвигового регистра

Назначение контактов сдвигового регистра:

- QA — QH — восемь параллельных выходов сдвигового регистра;
- GND — соединяется с земляной шиной платы ***Arduino***;
- SER— это вход данных (DATA на рис. 67). По этому входу передаются 8 последовательных битов данных для установки значений на параллельных выходах;
- SRCLK— это тактовый вход (CLOCK на рис. 67). При подаче импульса высокого напряжения HIGH на этот вход происходит считывание одного бита данных с входа DATA в сдвиговый регистр. Для получения всех 8 битов данных необходимо подать 8 импульсов на этот контакт;
- RCLK — это вход (LATCH на рис 67) называется также защелкой и служит для одновременного вывода последовательных данных на параллельные выходы.

Выводы $\overline{\text{SRCLR}}$ и $\overline{\text{OE}}$ не используются в примерах из книги, но они могут пригодиться вам в других проектах, поэтому посмотрим, каково их назначение. $\overline{\text{OE}}$ — разрешение вывода данных на параллельные выходы. Черта сверху означает, что активный уровень для этого входа — низкий. Когда на этом входе низкий уровень, параллельные выходы будут включены, когда высокий — выключены. В наших примерах контакт $\overline{\text{OE}}$ подключен к земле, поэтому параллельные выходы постоянно находятся во включенном состоянии. Вы можете соединить его с контактом ввода-вывода ***Arduino***, чтобы одновременно включать или выключать все светодиоды. $\overline{\text{SRCLR}}$ — это вход сброса. Подача на него напряжения низкого уровня очищает содержимое регистра сдвига. В наших примерах он подключен к шине 5 В, чтобы предотвратить очистку сдвигового регистра.

7.3.2. Принцип действия сдвиговых регистров

Регистр сдвига является синхронным устройством, он принимает данные по нарастающему фронту тактового сигнала. Каждый раз, когда сигнал на входе CLOCK меняется с низкого на высокий, все значения, хранящиеся в восьми выходных ячейках, смещаются на одну позицию. Данные из последней ячейки либо сбрасываются, либо передаются на выход QH' (при каскадном подключении микросхемы 74HC595). Одновременно последовательные данные на входе DATA сдвигаются на одну позицию. За восемь тактов предыдущие значения в ячейках регистра уходят, а новые загружаются. Подача высокого уровня на вход LATCH выводит значения, хранящиеся в ячейках, на выходы регистра. Этот процесс проиллюстрирован на рисунке 69.

Предположим, вы хотите включить некоторые из светодиодов, подключенных к параллельным выходам сдвигового регистра, например, к выходам QA, QC, QE, QG. В двоичном представлении на параллельных выходах должно быть значение 10101010. Теперь посмотрим, как действует регистр. Установим низкий уровень на входе LATCH, чтобы значения на параллельных выходах не изменялись во время загрузки новых данных в ячейки регистра. Затем, подавая импульсы на вход CLOCK, значения на входе DATA загружаем и сдвигаем по ячейкам. После загрузки в регистр всей последовательности данных, устанавливаем на входе LATCH высокий уровень для вывода значений из ячеек на параллельные выходы.

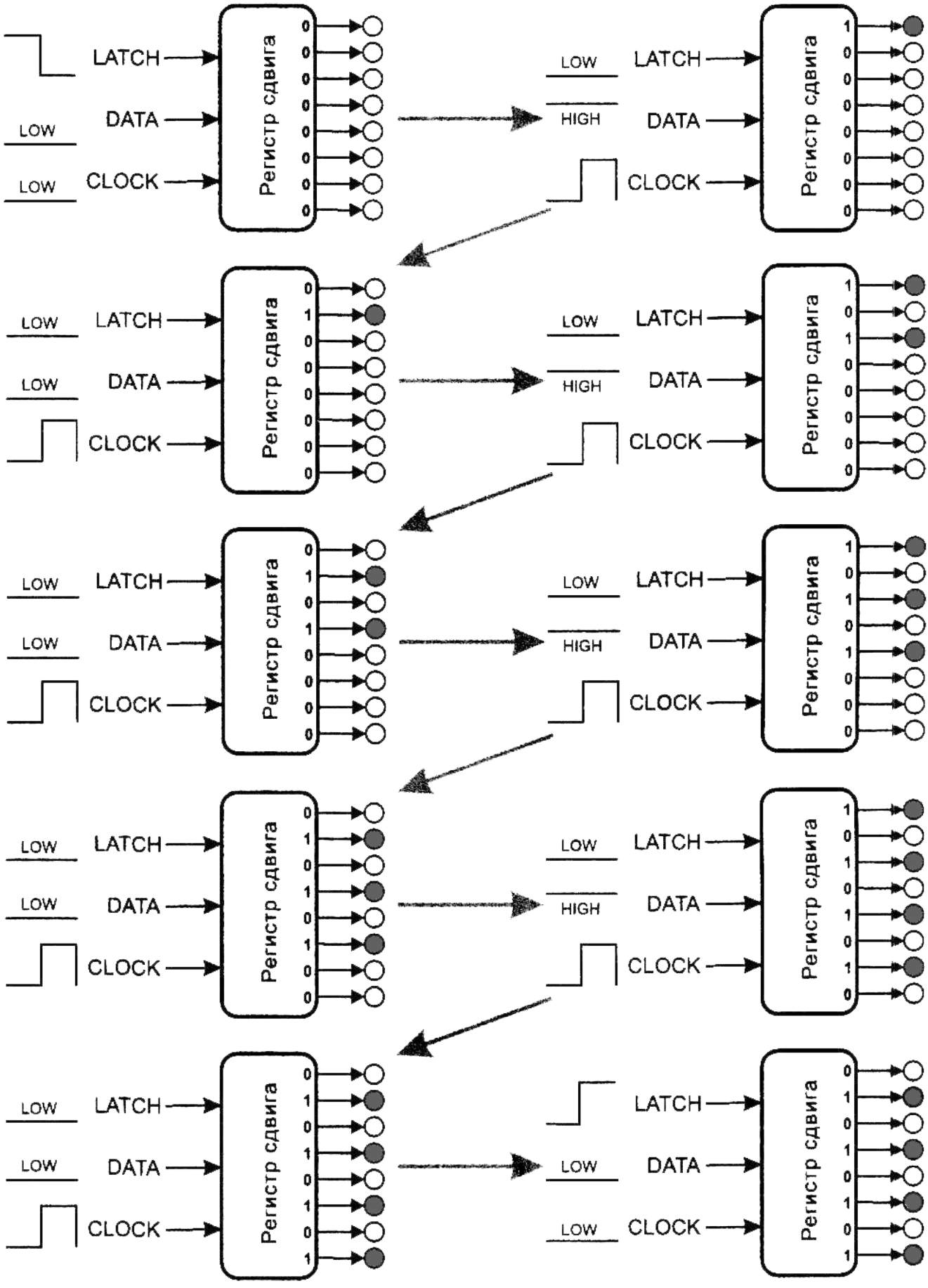

Рисунок 69. Перемещение данных по сдвиговому регистру

7.3.3. Передача данных из ***Arduino*** в сдвиговый регистр

Теперь можно написать программу для передачи данных из ***Arduino*** в сдвиговые регистры. Воспользуемся встроенной в ***Arduino*** IDE функцией `shiftOut()` для поразрядной выдачи данных на контакт платы ***Arduino***. Эта функция принимает четыре аргумента:

- номер контакта DATA;
- номер контакта CLOCK;
- порядок выдачи битов;
- значение, выдаваемое на выход.

Например, если вы хотите зажечь светодиоды как в предыдущем примере, можно вызвать функцию `shiftOut()` следующим образом:

```
shiftOut(DATA, CLOCK, MSBFIRST, B10101010);
```

Константы `DATA` и `CLOCK` — номера контактов ***Arduino*** для передачи данных. Аргумент `MSBFIRST` показывает, что самый старший бит (крайний слева бит в двоичном числе) будет отправлен первым. Можно передавать данные с параметром `LSBFIRST`, который начнет передачу с младшего бита. Последний параметр — это передаваемые данные. Знак «`B`» перед числом указывает ***Arduino*** IDE, что данные представлены в двоичном виде.

Теперь соберем схему устройства. Подключим регистр сдвига к плате ***Arduino*** следующим образом: вывод DATA к контакту 8 платы, LATCH к контакту 9, CLOCK к контакту 10.

Не забывайте, что светодиоды нужно подключать через токоограничительные резисторы (220 Ом). При монтаже сверяйтесь с рисунком 70.

Теперь, зная, как работают сдвиговые регистры и используя встроенную в ***Arduino*** IDE функцию `shiftOut()`, мы можем написать программу управления светодиодами (листинг 30).

Листинг 30. Программа управления светодиодами с помощью сдвигового регистра — alternate.ino

```
const int SER   =8;    //Последовательного выход сдвигового регистра DATA
const int LATCH =9;    //Контакт для подключения вывода Latch
const int CLK   =10;   //Контакт для подключения вывода Clock

void setup()
{
  //Установите контакты как выводы
  pinMode(SER, OUTPUT);
  pinMode(LATCH, OUTPUT);
  pinMode(CLK, OUTPUT);

  digitalWrite(LATCH, LOW);                //Latch – низкий
  shiftOut(SER, CLK, MSBFIRST, B10101010); //Старший бит – первый
  digitalWrite(LATCH, HIGH);               //Latch – высокий (Показать шаблон)
}

void loop()
{
  //Ничего не делать
}
```

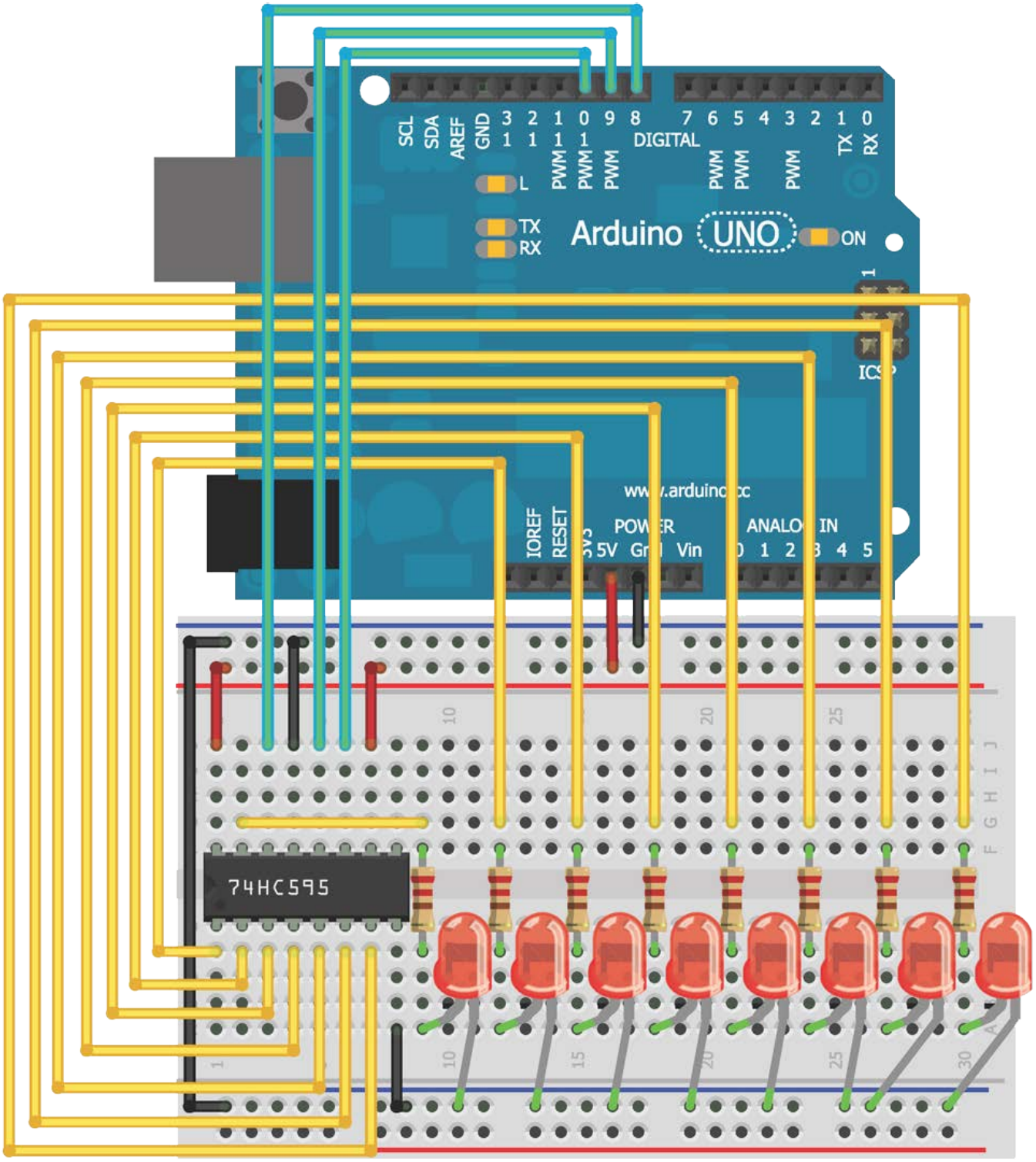

Рисунок 70. Подключение 8 светодиодов к сдвиговому регистру

Поскольку регистр сдвига фиксирует полученные данные, отправляем их только один раз в функции `setup()`. Полученные значения хранятся до следующего изменения. Эта программа выполняет те же шаги, что показаны на рис. 69. На выводе LATCH устанавливается низкий уровень, восемь битов данных передаются функцией `shiftOut()` в ячейки, а затем на LATCH подается высокий уровень, чтобы вывести значения из ячеек на параллельные выходы регистра.

Как видим, нам удалось получить восемь цифровых выходов из трех портов ввода-вывода. Уже неплохо. Но что делать, если нужно еще больше? Это возможно! Последовательно подключая несколько сдвиговых регистров, теоретически можно добавить сотни цифровых выходов, задействуя только три контакта ***Arduino***. Но в таком случае потребуется внешнее питание. Посмотрите на рисунок 70, там есть неиспользованный контакт QH'. Когда значения смещаются по ячейкам, они на самом деле не отбрасываются, а поступают на вывод QH'. Подключив выход QH' к входу DATA другого

сдвигового регистра, а также соединив выводы LATCH и CLOCK обоих регистров, можно создать 16-разрядный регистр сдвига, который управляет 16 выводами. Добавляя все больше и больше регистров сдвига, каждый из которых подсоединен к предыдущему, вы получите сколь угодно много выводов из платы ***Arduino***.
Можете попробовать подключить еще один регистр сдвига и дважды вызвать функцию `shiftOut()` (за один раз функция `shiftOut()` может обрабатывать только 8 разрядов данных).

7.3.4. Преобразование между двоичным и десятичным форматами

В листинге 30 данные для включения светодиодов передавались в виде двоичной строки. Это позволяет наглядно отобразить включенные и выключенные светодиоды. Тем не менее, данные можно записать в десятичном виде, преобразовав их из двоичного (base2) в десятичный (base10) формат. Каждый разряд двоичного числа (начиная с младшего, самого правого) представляет следующую степень числа 2. Преобразовать двоичное представление числа в десятичное очень просто. Рассмотрим этапы преобразования двоичного кода в десятичный (Таблица 5).

Таблица 5. Преобразования двоичного кода в десятичный

1	0	1	0	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1×128+0×64+1×32+0×16+1×8+0×4+1×2+0×1=170							

Двоичное значение каждого бита представляет собой увеличивающуюся степень числа 2. В нашем случае биты 7, 5, 3, 1 установлены в единицу. Таким образом, чтобы найти десятичный эквивалент, вы складываете 2^7, 2^5, 2^3 и 2^1. Полученное десятичное значение равно 170. Можно доказать, что это эквивалентно, подставив десятичное значение в код из листинга 30.
Замените строку с `shiftOut()` на следующую:

```
shiftOut(DATA, CLOCK, MSBFIRST, 170);
```

и убедитесь, что результат будет таким же, как и для двоичного представления.

7.4. Создание световых эффектов с помощью сдвигового регистра

В предыдущем примере мы создали с помощью регистра сдвига и светодиодов неподвижное изображение. Гораздо интереснее отображать динамичную информацию. В следующих примерах используем сдвиговые регистры для создания анимационных эффектов «бегущего всадника» и «гистограммы».

7.4.1. Эффект «бегущий всадник»

«Бегущий всадник» — анимационный эффект, когда горящий светодиод сначала движется в одну сторону, а затем в обратную. Соберем схему из предыдущего примера. С помощью функции `shiftOut()` можно очень быстро обновлять данные в ячейках сдвигового регистра, чтобы создавать динамичные световые анимации. В нашем примере необходимо зажигать по одному светодиоду сначала слева направо, затем справа налево. На временной диаграмме (Рисунок 71) показано, как светодиоды будут гореть на каждом шаге, и приведены десятичные значения для включения конкретной комбинации светодиодов.
В программе передаем функции `shiftOut()` десятичные значения комбинаций светодиодов. В цикле выбираем значение из массива и отправляем в сдвиговый регистр. Код программы приведен в листинге 31.

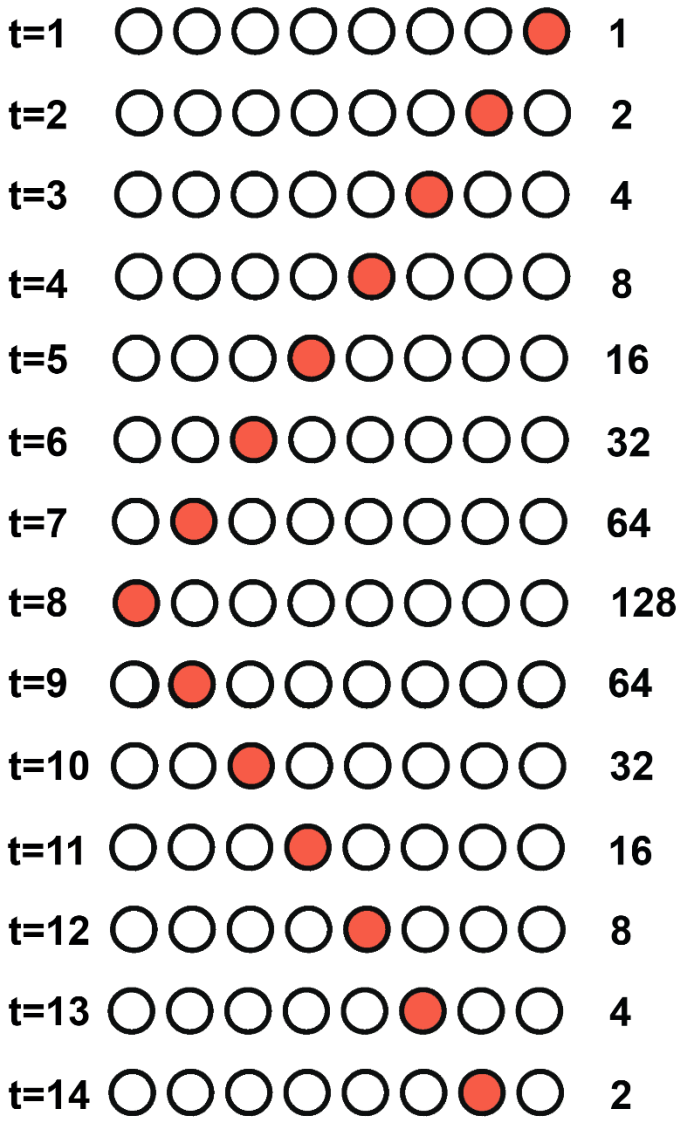

Рисунок 71. Пошаговое представление эффекта «бегущий всадник»

Листинг 31. Световой эффект «бегущий всадник» — lightrider.ino

```
//Создание световой анимации «бегущий всадник»

const int SER   =8;    //Последовательный выход сдвигового регистра DATA
const int LATCH =9;    //Контакт для подключения вывода Latch
const int CLK   =10;   //Контакт для подключения вывода Clock

//Последовательность включения светодиодов
int seq[14] = {1,2,4,8,16,32,64,128,64,32,16,8,4,2};

void setup()
{
  //Установить контакты как выводы
  pinMode(SER, OUTPUT);
  pinMode(LATCH, OUTPUT);
  pinMode(CLK, OUTPUT);
}

void loop()
{
  for (int i = 0; i < 14; i++)
```

```
  {
    digitalWrite(LATCH, LOW);              //Latch – низкий. Старт отправления
    shiftOut(SER, CLK, MSBFIRST, seq[i]); //Старший бит – первый
    digitalWrite(LATCH, HIGH);             //Latch – высокий. Стоп отправки
    delay(100);                            //Скорость анимации (задержка)
  }
}
```

Регулируя величину задержки, можно менять скорость анимации. Попробуйте изменить значения в массиве последовательностей, чтобы получить различные комбинации включения светодиодов.

> ***ПРИМЕЧАНИЕ***
>
> *Для просмотра демонстрационного видеоклипа программы lightrider зайдите по адресу https://www.exploringarduino.com/content/ch7/. Видеоклип также доступен на сайте издательства Wiley.*

7.4.2. Отображение данных в виде гистограммы

Добавив ИК-датчик расстояния в предыдущую схему, можно создать светящуюся гистограмму, которая показывает, насколько близко вы находитесь. Для большей наглядности возьмите несколько светодиодов разного цвета. Принципиальная схема устройства с разноцветными светодиодами и ИК-датчиком расстояния приведена на рисунке 72.

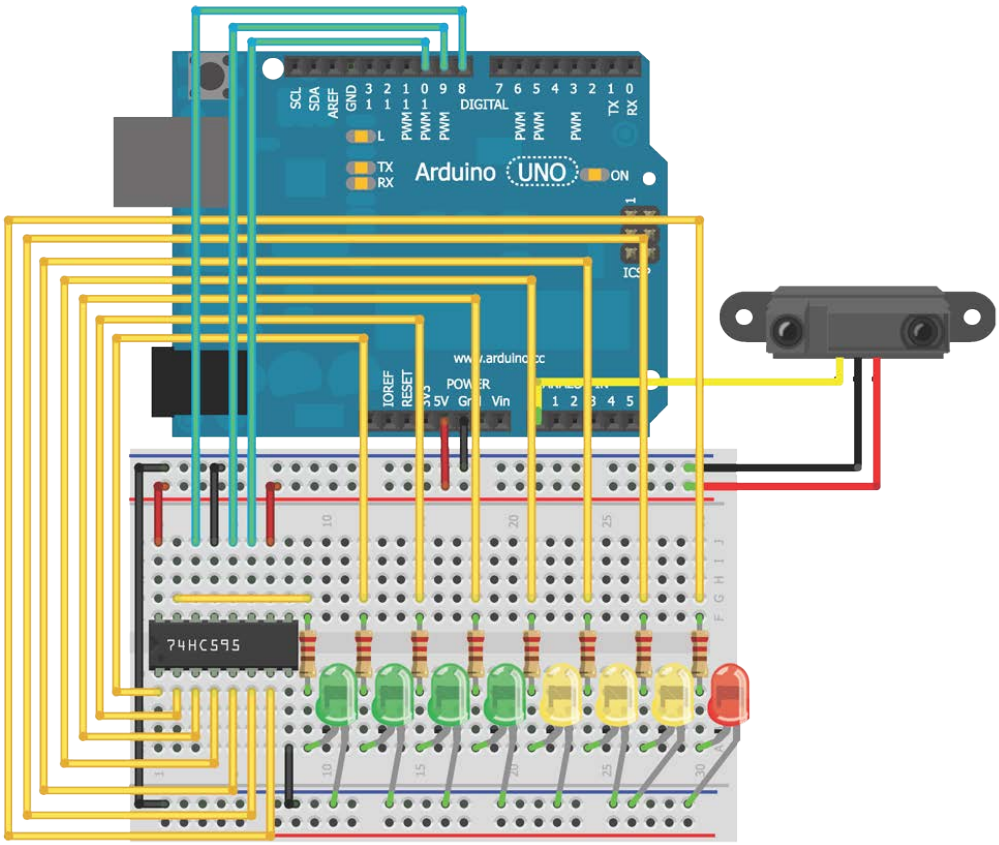

Рисунок 72. Схема устройства, реализующего эффект «гистограммы расстояния»

Зная принцип действия аналогового датчика и регистра сдвига, вы в состоянии самостоятельно выбрать пороги расстояния и соответствующие им комбинации включенных и выключенных светодиодов (рис. 73).

Из *главы 3* ясно, что диапазон используемых значений для ИК-датчика расстояния не должен превышать 10 бит (у меня максимальное значение оказалось равно 500, у вас оно возможно отличается). Лучше всего самостоятельно проверить дальность действия датчика и уточнить соответствующие значения. Все десятичные комбинации гистограммы хранятся в массиве из девяти элементов. Ограничиваем максимальное расстояние и приводим значения к диапазону от 0 до 8.

Листинг 32 иллюстрирует программную реализацию эффекта гистограммы.

Листинг 32. Гистограмма отображения расстояния — bargraph.ino

```
//Гистограмма расстояния

const int SER   =8;    //Последовательный выход сдвигового регистра DATA
const int LATCH =9;    //Контакт для подключения вывода Latch
const int CLK   =10;   //Контакт для подключения вывода Clock
const int DIST  =0;    //Контакт для подключения датчика расстояния

//Возможные значения светодиодов
int vals[9] = {0,1,3,7,15,31,63,127,255};

//Максимальное значение расстояния
int maxVal = 500;

//Минимальное значение расстояния
int minVal = 0;

void setup()
{
  //Установить контакты как выводы
  pinMode(SER, OUTPUT);
  pinMode(LATCH, OUTPUT);
  pinMode(CLK, OUTPUT);
}

void loop()
{
  int distance = analogRead(DIST);
  distance = map(distance, minVal, maxVal, 0, 8);
  distance = constrain(distance,0,8);

  digitalWrite(LATCH, LOW);                  //Latch – низкий. Старт отправки
  shiftOut(SER, CLK, MSBFIRST, vals[distance]); //Старший бит – первый
  digitalWrite(LATCH, HIGH);                 //Latch – высокий. Стоп отправки
  delay(10);                                 //Скорость анимации (задержка)
}
```

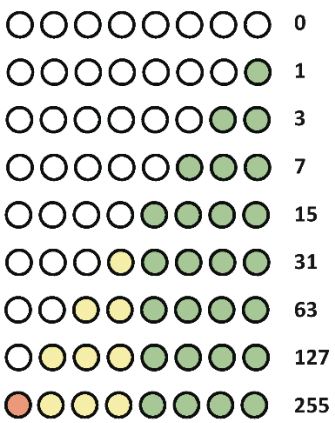

Рисунок 73. Комбинации включенных и выключенных светодиодов и соответствующие им десятичные значения

Загрузите программу на плату ***Arduino***, запустите на выполнение и перемещайте руку вперед-назад

перед датчиком расстояния. Вы должны увидеть, что гистограмма реагирует на движение руки. Если устройство работает неправильно, отрегулируйте значения `maxVal` и `minVal`, чтобы лучше соответствовать показаниям датчика расстояния. Для контроля значений, которые вы получаете на различных расстояниях, можно инициализировать последовательное соединение в заголовке `setup()` и вызвать функцию `Serial.println(DIST)` сразу после выполнения шага `analogRead(DIST)`.

> *ПРИМЕЧАНИЕ*
>
> *Для просмотра демонстрационного видеоклипа программы, формирующей гистограмму расстояния, зайдите по адресу https://www.exploringarduino.com/content/ch7/. Этот видеоклип доступен также на сайте издательства Wiley.*

Резюме

В этой главе вы узнали следующее:

- Как работают сдвиговые регистры.
- Чем отличается последовательная и параллельная передача данных.
- В чем различие между десятичной и двоичной формой представления данных.
- Как создать световую анимацию с помощью сдвигового регистра.

Часть III. Интерфейсы передачи данных

Глава 8. Интерфейсная шина I²C

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;
- USB-кабель B (для ***Uno***);
- красный светодиод;
- 3 желтых светодиода;
- 4 зеленых светодиода;
- 8 резисторов номиналом 220 Ом;
- 2 резистора номиналом 4,7 кОм;
- сдвиговый регистр SN74HC595N в DIP-корпусе;
- I2C датчик температуры TC74A0-5.0VAT;
- перемычки;
- макетная плата.

Электронные ресурсы к главе

На странице https://www.exploringarduino.com/content/ch8/ можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы https://www.wiley.com/en-ru/Exploring+Arduino%3A+Tools+and+Techniques+for+Engineering+Wizardry-p-9781118549360 в разделе «Downloads».

Что вы узнаете в этой главе

Вы уже знаете, как использовать аналоговые и цифровые вводы-выводы, но как общаться с более сложными устройствами? Плата ***Arduino*** способна раскрыть свои новые возможности, взаимодействуя через интерфейсы с множеством внешних модулей. Во многих интегральных схемах реализованы стандартные цифровые протоколы связи, упрощающие обмен данными между микроконтроллером и всевозможными периферийными устройствами. В этой главе рассмотрим шину I²C

Шина I²C обеспечивает устойчивую высокоскоростную двустороннюю связь между устройствами и требует минимального числа контактов ввода-вывода. К шине I²C подключено ведущее устройство (обычно микроконтроллер) и одно или несколько ведомых устройств, которые получают информацию от ведущего. Далее мы опишем протокол I²C и реализуем его, чтобы связаться с цифровым датчиком температуры, возвращающим результат измерений в градусах, а не в виде произвольного аналогового значения. Опираясь на знания, полученные в предыдущих главах, вы научитесь создавать более сложные проекты.

> ***ПРИМЕЧАНИЕ***
>
> *Вы можете шаг за шагом пройти данную главу, воспользовавшись демонстрационным видеоуроком, расположенным по адресу https://www.jeremyblum.com/2011/02/13/arduino-tutorial-7-i2c-and-processing/.[1] Этот видеоклип доступен и на сайте издательства Wiley.*

8.1. История создания протокола I²C

Чтобы уяснить, почему популярен тот или иной протокол связи, лучше всего посмотреть, как он развивался с течением времени. Протокол I²C был предложен фирмой Philips в начале 1980-х годов для обеспечения низкоскоростной связи между различными интегральными микросхемами. В 1990 году этот протокол был стандартизирован и другие компании начали его использовать, выпуская свои собственные совместимые чипы. Протокол часто называют «двухпроводным», поскольку связь осуществляется по двум шинам: линии синхронизации и линии передачи данных. Хотя не все двухпроводные протоколы, строго говоря, могут называться I²C (из-за неоплаты права на использование названия), но, как правило, их называют I²C -устройствами. Так же марку KLEENEX®

[1] На русском: http://wiki.amperka.ru/видеоуроки:7-i2c-и-processing

часто ставят даже на тех тканях, которые не производит эта фирма. Если в описании какого-то устройства сказано, что оно поддерживает «двухпроводной» протокол связи, можно быть уверенным, что оно будет работать так, как описано в этой главе.

8.2. Схема подключения устройств I²C

Подключение устройств по протоколу связи I²C иллюстрирует рисунок 74. От предыдущих способов цифровой передачи данных, рассмотренных в этой книге, I²C отличается тем, что несколько устройств используют одни и те же линии связи: шину синхронизации сигнала (SCL) и двунаправленную шину данных (SDA). Последняя служит для отправки данных от ведущего устройства к ведомым. Обратите внимание, что на каждой шине I²C требуется установка подтягивающих резисторов.

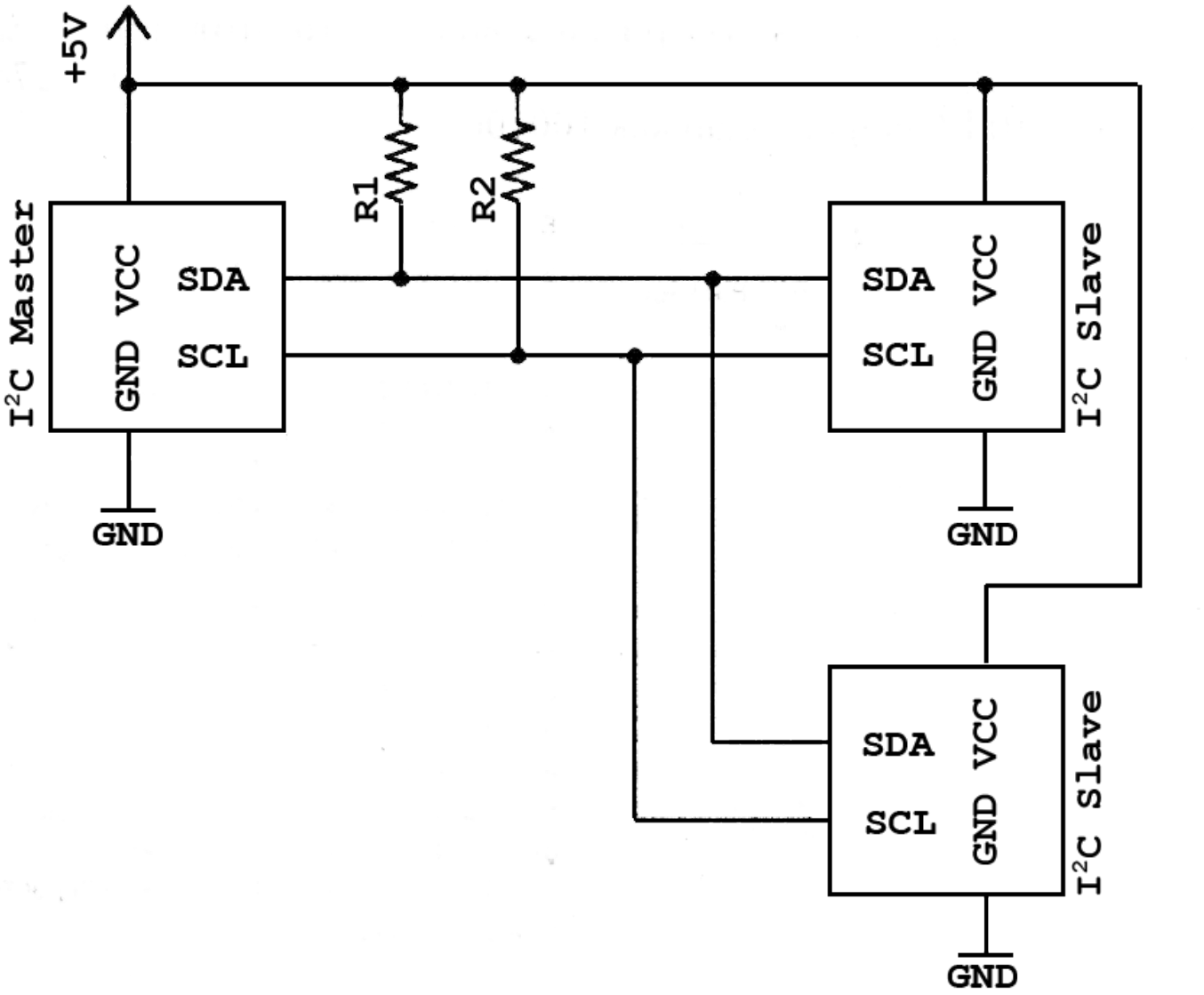

Рисунок 74. Схема подключения устройств I²C

8.2.1. Взаимодействие и идентификация устройств

Протокол I²C позволяет нескольким ведомым устройствам соединяться по одной шине с одним ведущим устройством. Далее роль ведущего устройства (мастера) будет играть плата ***Arduino***. Мастер шины отвечает за инициирование обмена. Ведомые устройства не могут инициировать обмен данных, они способны только отвечать на запросы, которые посылает ведущее устройство. Так как к одной линии подключено несколько ведомых устройств, принципиально важно, что устанавливать связь может только ведущее устройство. В противном случае, сразу несколько устройств могли одновременно пытаться отправить данные, что привело бы к искажениям.

Все команды и запросы, отправленные от мастера, принимаются всеми устройствами на шине. Каждое ведомое устройство имеет уникальный 7-битовый адрес (идентификационный номер, ID устройства). Когда ведущее устройство инициирует связь, оно передает идентификатор ведомого. Ведомые устройства реагируют на данные, передающиеся по шине, только тогда, когда они направлены в их

адрес.
Адрес каждого устройства на шине I²C должен быть уникальным. Некоторые устройства I²C имеют настраиваемые адреса, а другие — фиксированный адрес, заданный производителем. При подключении к одной шине нескольких I²C-устройств, необходимо чтобы у них были различные идентификаторы.
Л
Датчики температуры обычно допускают перепрограммирование I²C-адреса, поэтому на одной шине I²C может быть несколько таких датчиков. Далее мы рассмотрим датчик температуры TC74. Из рис. 75 видно, что этот датчик может иметь несколько разных адресов. В примерах данной главы у датчика TC74A0-5.0VAT (исполнение T0-220) I²C-адрес задан как 1001000.

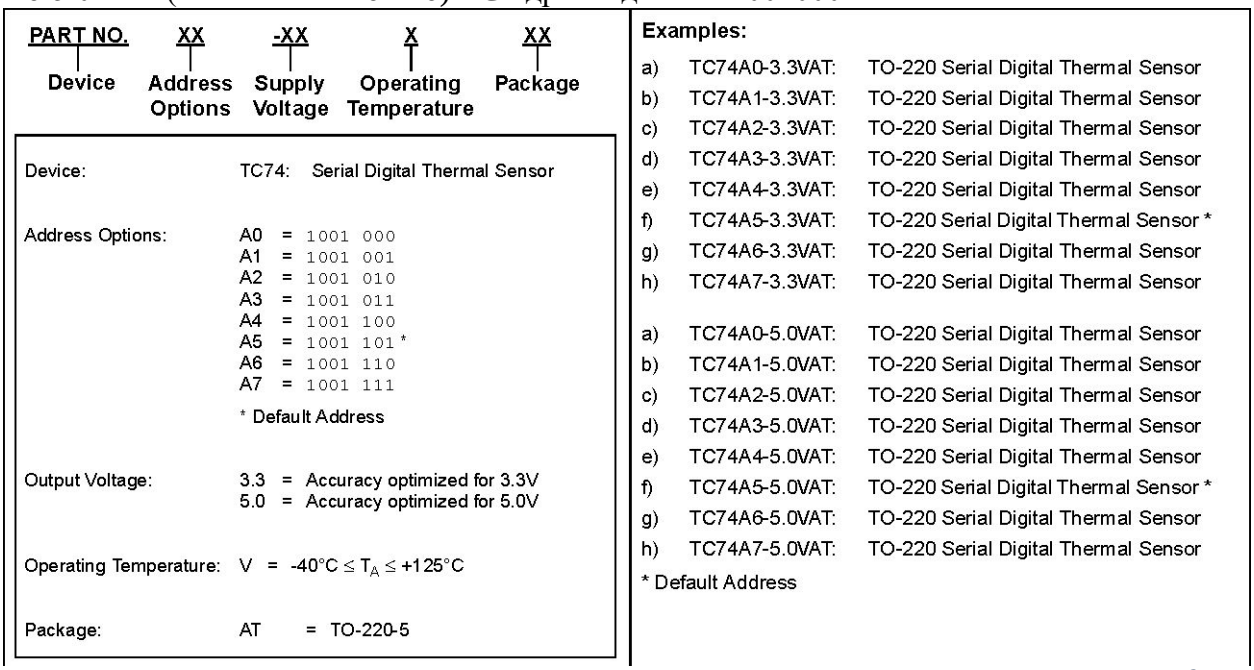

Рисунок 75. Фрагмент технического описания датчика TC74: расшифровка обозначения и варианты адресов I²C

Поскольку выпускаются датчики температуры с восьмью различными идентификационными номерами, к одной шине I²C можно подключить до восьми разных датчиков. Для написания программ для этих датчиков необходимо знать их ID, чтобы отправлять правильные команды.
В датчиках другого типа, например, AD7414 и AD7415 есть контакт (AS), который позволяет настроить адрес устройства I²C. Взгляните на данные датчика AD7414 из документации (Рисунок 76). Датчик AD7414 выпускается в четырех вариантах исполнения, с контактом AS и без него. Адрес устройства, снабженного контактом AS, зависит от состояния этого контакта: отключен, подключен к питанию или земле.

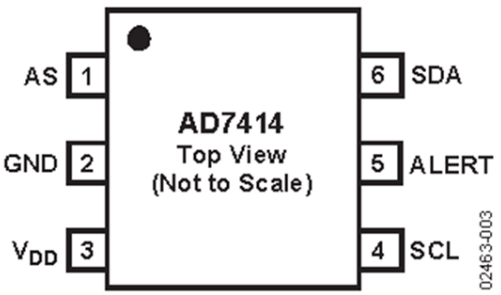

Figure 3. AD7414 Pin Configuration (SOT-23)

Table 4. I²C Address Selection

Part Number	AS Pin	I²C Address
AD7414-0	Float	1001 000
AD7414-0	GND	1001 001
AD7414-0	V_{DD}	1001 010
AD7414-1	Float	1001 100
AD7414-1	GND	1001 101
AD7414-1	V_{DD}	1001 110
AD7414-2	N/A	1001 011
AD7414-3	N/A	1001 111
AD7415-0	Float	1001 000
AD7415-0	GND	1001 001
AD7415-0	V_{DD}	1001 010
AD7415-1	Float	1001 100
AD7415-1	GND	1001 101
AD7415-1	V_{DD}	1001 110

Рисунок 76. Фрагмент технического описания датчика AD7414: цоколевка и варианты адресов I²C

8.2.2. Требования к оборудованию и подтягивающие резисторы

Из рис. 74 ясно, что типовая конфигурация шины I²C требует наличия подтягивающих резисторов на линиях синхронизации и передачи данных. Номинал резисторов зависит от ведомых устройств, данные можно посмотреть в документации на эти устройства. Мы рекомендуем взять резисторы номиналом 4,7 кОм, — это стандартное значение, которое указано во многих справочниках.

8.3. Связь с датчиком температуры I²C

Организация обмена данными с устройством I²C зависит от требований конкретного устройства. К счастью, наличие в ***Arduino*** библиотеки I²C освобождает от необходимости программирования операций синхронизации процесса обмена. Далее мы рассмотрим организацию обмена данными с I²C-датчиком температуры. Этот пример позволит в будущем легко осуществить обмен с другими устройствами I²C.

Основные шаги для управления любым I²C-устройством таковы:

- Мастер посылает стартовый бит.
- Мастер посылает 7-разрядный адрес ведомого устройства.
- Мастер устанавливает на шине данных «1» (чтение) или «0» (запись) в зависимости от того, хочет ли он отправить данные в ведомое устройство или получить данные от него.
- Ведомое устройство выставляет бит ACK (логический уровень низкий).
- В режиме записи, мастер передает один байт информации, ведомое устройство выдает бит ACK. В режиме чтения, мастер получает один байт и посылает бит ACK в ведомое после каждого байта.
- Когда связь завершена, мастер посылает стоп-бит.

8.3.1. Сборка схемы устройства

Чтобы убедиться, что наша программа будет работать, как ожидалось, можно выводить показания от датчика температуры в монитор последовательного порта. Это цифровой датчик, и он выдает показания в градусах. В отличие от датчиков температуры, описанных в предыдущих главах, вам не нужно беспокоиться о преобразовании аналогового значения к фактической температуре. Это очень удобно! Подключите датчик температуры к плате ***Arduino***, как показано на рис. 77.

Обратите внимание, что выводы SDA и SCL датчика подключены к контактам A4 и A5 платы. Напомним, что SDA и SCL — это линия передачи данных и линия синхронизации, соответственно. Контакты ***Arduino*** A4 и A5 мультиплексируются между аналого-цифровым преобразователем (АЦП) и аппаратным интерфейсом I²C. При инициализации библиотеки Wire эти контакты подключены к контроллеру I²C в ATmega, обеспечивая взаимодействие объекта `Wire` с I²C -устройствами. При этом использовать A4 и A5 как аналоговые входы нельзя, потому что они заняты обменом с устройствами I²C.

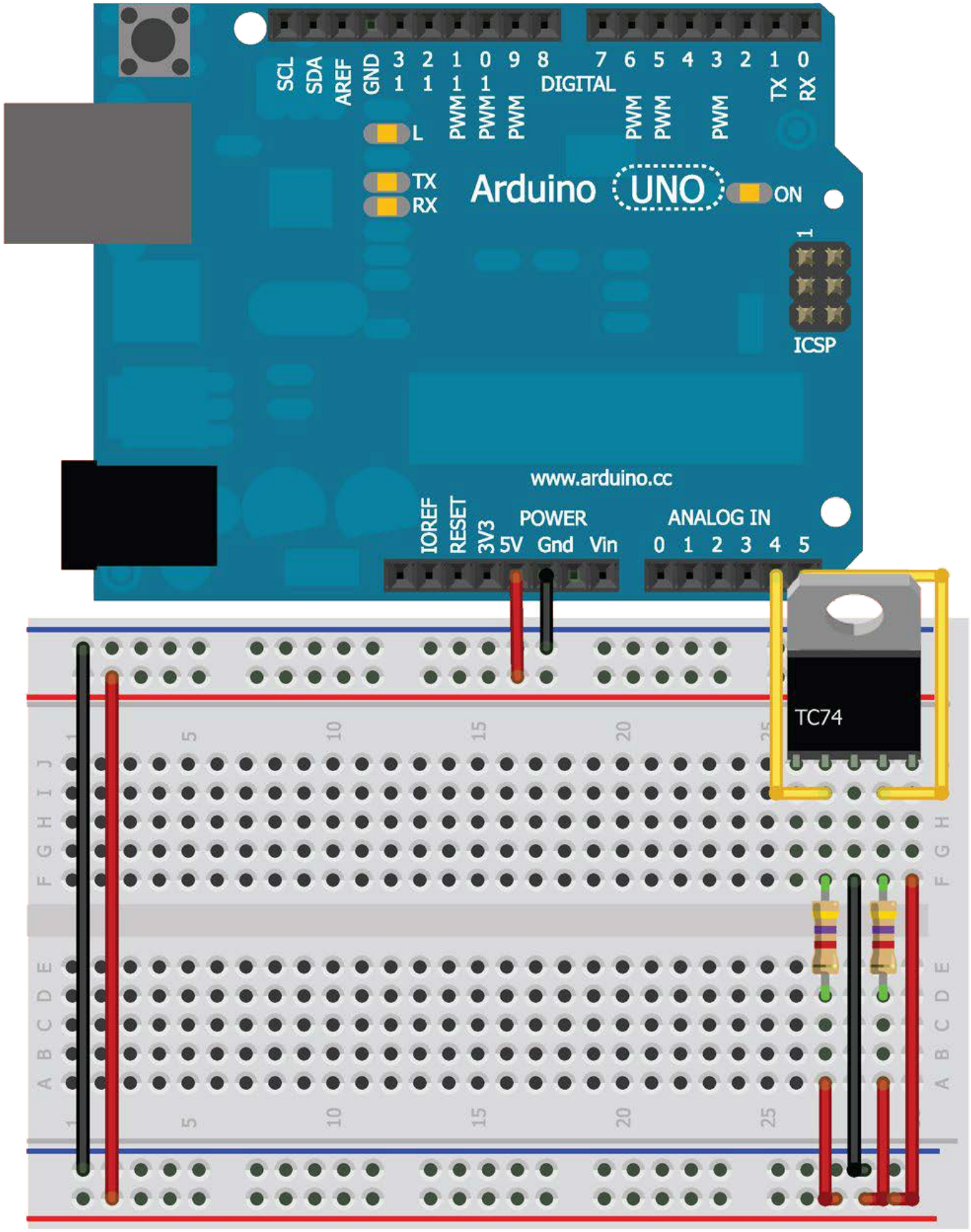

Рисунок 77. Схема подключения датчика температуры

8.3.2. Анализ технического описания датчика

Теперь нужно написать программу, которая определяет действия ***Arduino*** для получения данных от I^2C-датчика температуры. С помощью библиотеки Wire сделать это довольно легко. Чтобы не допустить ошибки, следует внимательно прочесть справочную информацию об алгоритме связи, который поддерживает именно этот чип. Давайте проанализируем протокол взаимодействия,

представленный в таблицах и на графиках, показанных на рис. 78 и 8.6.
На рис. 8.6 показано, как осуществлять чтение и запись данных для датчика TC74. Микросхема имеет два регистра: в первом хранится значение текущей температуры в градусах Цельсия, во втором — информация о конфигурации чипа (включая режим ожидания и режим передачи данных). Это ясно из табл. 4.1 на рис. 8.6. Нет необходимости вникать в нюансы конфигурации, требуется только получить значение температуры от устройства. В табл. 4.3 и 4.4 на рис. 8.6 показано, как хранится информация о температуре внутри 8-разрядного регистра.

Формат команды записи данных

S	Адрес	WR	ACK	Команда	ACK	Данные	ACK	P
	7 бит			8 бит		8 бит		

Адрес ведомого устройства

Байт команды: выбор устройства для записи данных

Байт данных: данные, отправляемые в устройство, выбранное байтом команды

Формат команды чтения данных

S	Адрес	WR	ACK	Команда	ACK	S	Адрес	RD	ACK	Данные	NACK	P
	7 бит			8 бит			7 бит			8 бит		

Адрес ведомого устройства

Байт команды: выбор устройства для чтения данных

Адрес ведомого устройства: повторяется в связи с изменением направления потока данных

Байт данных: чтение данных из устройства, выбранного байтом команды

Формат команды чтения данных

S	Адрес	RD	ACK	Данные	NACK	P
	7 бит			8 бит		

Байт данных: считывает данные из регистра, заданного последней передачей байта чтения или байта записи.

S = START Condition — Старт-бит
P = STOP Condition — Стоп-бит
Затененные ячейки = Посылки от ведомого устройства
WR — бит записи (передачи) данных
RD — бит чтения (приёма) данных
ACK — квитирующий бит

Рисунок 78. Протокол обмена датчика TC74

В секции Write Byte format на рис. 79 показано, как прочитать значение температуры из TC74:

- послать адрес устройства в режиме записи и выставить 0, чтобы указать, что нужно перейти в режим чтения из регистра данных;
- отправить на адрес устройства команду запроса (1 байт) на чтение информации от устройства;
- подождать прихода всех 8 битов информации о значении температуры.

Теперь становится понятно, как работать с подобными I^2C -устройствами. Если вы еще не все уяснили, поищите в Интернете примеры программ подключения ***Arduino*** к различным устройствам I^2C. Далее перейдем к написанию программы, которая выполняет три действия, описанные ранее.

4.0 REGISTER SET AND PROGRAMMER'S MODEL

TABLE 4-1: COMMAND BYTE DESCRIPTION (SMBUS/I²C READ_BYTE AND WRITE_BYTE)

Command	Code	Function
RTR	00h	Read Temperature (TEMP)
RWCR	01h	Read/Write Configuration (CONFIG)

TABLE 4-2: CONFIGURATION REGISTER (CONFIG); 8 BITS, READ/WRITE)

Bit	POR	Function	Type	Operation
D[7]	0	STANDBY Switch	Read/Write	1 = standby, 0 = normal
D[6]	0	Data Ready *	Read Only	1 = ready 0 = not ready
D[5]-D[0]	0	Reserved - Always returns zero when read	N/A	N/A

Note 1: *DATA_RDY bit RESET at power-up and SHDN enable.

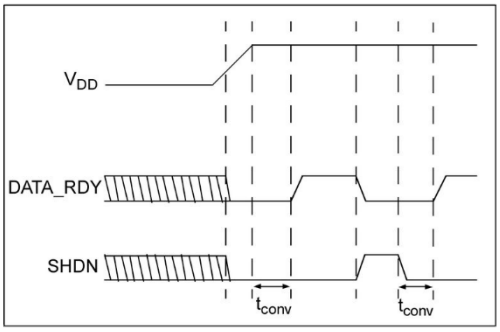

FIGURE 4-1: *DATA_RDY, SHDN Operation Logic Diagram.*

4.1 Temperature Register (TEMP), 8 Bits, READ ONLY

The binary value (2's complement format) in this register represents temperature of the onboard sensor following a conversion cycle. The registers are automatically updated in an alternating manner.

TABLE 4-3: TEMPERATURE REGISTER (TEMP)

D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
MSB	X	X	X	X	X	X	LSB

In temperature data registers, each unit value represents one degree (Celsius). The value is in 2's complement binary format such that a reading of 0000 0000b corresponds to 0°C. Examples of this temperature to binary value relationship are shown in Table 4-4.

TABLE 4-4: TEMPERATURE-TO-DIGITAL VALUE CONVERSION (TEMP)

Actual Temperature	Registered Temperature	Binary Hex
+130.00°C	+127°C	0111 1111
+127.00°C	+127°C	0111 1111
+126.50°C	+126°C	0111 1110
+25.25°C	+25°C	0001 1001
+0.50°C	0°C	0000 0000
+0.25°C	0°C	0000 0000
0.00°C	0°C	0000 0000
-0.25°C	-1°C	1111 1111
-0.50°C	-1°C	1111 1111
-0.75°C	-1°C	1111 1111
-1.00°C	-1°C	1111 1111
-25.00°C	-25°C	1110 0111
-25.25°C	-26°C	1110 0110
-54.75°C	-55°C	1100 1001
-55.00°C	-55°C	1100 1001
-65.00°C	-65°C	1011 1111

4.2 Register Set Summary

The TC74 register set is summarized in Table 4-5. All registers are 8 bits wide.

TABLE 4-5: TC74 REGISTER SET SUMMARY

Name	Description	POR State	Read	Write
TEMP	Internal Sensor Temperature (2's Complement)	0000 0000b (1)	√	N/A
CONFIG	CONFIG Register	0000 0000b	√	√

Note 1: The TEMP register will be immediately updated by the A/D converter after the DATA_RDY Bit goes high.

Рисунок 79. Страница из технического описания датчика TC74

8.3.3. Написание программы

Как уже упоминалось, в ***Arduino*** есть библиотека Wire для связи устройств по протоколу I²C. После подключения библиотеки можно читать данные из устройства и записывать данные в него. Загрузите

код из листинга 33, который иллюстрирует преимущества функций, встроенных в библиотеку `Wire`.

Листинг 33. Чтение данных с I²C-датчика температуры — readtemp.ino

```
//Чтение температуры из I²C-датчика
//и вывод значений в последовательный порт

//Подключение библиотеки Wire
#include <Wire.h>
int temp_address = 72; //Запись адреса 1001000

void setup()
{
  //Запуск последовательного порта на скорости 9600 бод
  Serial.begin(9600);

  //Создание объекта Wire
  Wire.begin();
}

void loop()
{
  // Отправка запроса
  // Выбор устройства отправкой адреса устройства
  Wire.beginTransmission(temp_address);
  // Установка бита asking в 0 для чтения
  Wire.write(0);
  //Отправка стоп-бита
  Wire.endTransmission();

  //Чтение температуры из устройства
  //Получить 1 байт по адресу устройства
  Wire.requestFrom(temp_address, 1);
  //Ожидание ответа
  while(Wire.available() == 0);
  //Чтение данных в переменную
  int c = Wire.read();

  //Перевод данных из шкалы Цельсия в шкалу Фаренгейта
  int f = round(c*9.0/5.0 +32.0);

  //Отправка значения в градусах Цельсия и Фаренгейта
  //в последовательный порт
  Serial.print(c);
  Serial.print("C ");
  Serial.print(f);
  Serial.println("F");

  delay(500);
}
```

Рассмотрим программу подробнее. Команда `Wire.beginTransmission()` начинает общение с ведомым устройством, отправляя адрес (уникальный идентификатор) устройства. Команда `Wire.write(0)` отправляет «0», указывая, что вы хотите читать из регистра температуры. Затем передаем стоп-бит, вызывая функцию `Wire.endTransmission()`, чтобы указать окончание записи

на устройство. Далее мастер получает информацию от ведомого устройства I²C. Команда `Wire.requestFrom` — мастер запрашивает получение одного байта данных из I²C -устройства. Команда `Wire.available()` будет блокировать выполнение остальной части кода, пока данные не станут доступны на линии I²C. Наконец, 8-разрядное значение считывается в переменную командой `Wire.read()`.

Программа из листинга 33 также преобразует температуру по Цельсию в градусы Фаренгейта. Формулу такого преобразования можно найти в Интернете. В нашем примере результат округлен до целого числа.

Теперь запустите код листинга 8.1 на плате ***Arduino*** и откройте монитор последовательного порта. Вы должны увидеть вывод данных в последовательный порт, который выглядит примерно так, как на рис. 8.7.

Рис. 8.7. Отправка данных из I²C-датчика температуры в последовательный порт

8.4. Проект, объединяющий регистр сдвига, последовательный порт и шину I²C

Теперь у нас есть простая схема, получающая данные от I²C-устройства и выводящая результаты в последовательный порт, и можно сделать нечто более интересное. Подключив сдвиговый регистр (см. главу 7), а также Processing-приложение, визуализируем температуру на экране компьютера.

8.4.1. Создание системы мониторинга температуры

Сначала соберем схему устройства (рис. 8.8). По существу, нужно лишь добавить сдвиговый регистр к схеме, изображенной на рис. 8.4.

Рис. 8.8.I2C-датчик температуры с гистограммным индикатором на основе сдвигового регистра

нужно написать программу на Processing, которая будет отображать значение температуры на компьютере в удобном для чтения формате.

8.4.3. Написание программы на Processing

На данный момент плата ***Arduino*** уже передает необработанные данные на компьютер. Необходимо написать программу обработки, которая сможет интерпретировать их и отобразить на компьютере в удобном виде.

Так как обновлять текст нужно в режиме реального времени, необходимо сначала узнать, как загрузить шрифты в программу на Processing. Откройте Processing-приложение и создайте новый пустой проект. Сохраните файл, прежде чем продолжить. Затем через контекстное меню Tools -> Create Font вызовите окно, которое изображено на рис. 8.9.

Рис. 8.9. Загрузка шрифтов в Processing-приложение

Выберите свой любимый шрифт и размер (для этой программы я рекомендую размер около 200). После этого нажмите кнопку ОК. Шрифт будет автоматически установлен в папку data данного проекта.

Программа на Processing должна выполнить следующее:

♦ Сгенерировать графическое окно на компьютере для отображения данных температуры в градусах Цельсия и Фаренгейта.

♦ Прочитать входящие данные из последовательного порта, преобразовать их и сохранить значения в локальных переменных для отображения на компьютере.

♦ Постоянно обновлять экран при получении новых значений из последовательного порта.

Скопируйте код из листинга 8.3, задайте правильное наименование порта для вашего компьютера и имя выбранного шрифта. Подключите плату ***Arduino*** к компьютеру и нажмите на кнопку Выполнить. И наслаждайтесь волшебной картинкой!

Листинг 8.3. Программа на Processing для отображения данных температуры — display_.temp.pde

```
// Отображение температуры, получаемой с I2C-датчика import processing.serial.*;
Serial port;
String temp_c =
String temp_f =
```

```
String data = int index = 0;
PFont font;
void setup()
{
size(400, 400) ;
// Измените "COM9" на имя вашего последовательного порта port = new Serial(this, "COM9", 9600);
port.bufferUntil('.');
// Измените имя шрифта, выбранное вами font = loadFont("AgencyFB-Bold-200.vlw"); textFont(font,
200);
}
void draw()
{
background(0,0,0); fill (46, 209, 2); text(temp_c, 70, 175); fill(0, 102, 153); text(temp_f, 70, 370);
}
void serialEvent (Serial port)
{
data = port.readStringUntil('.');
data = data.substring(0, data.length() - 1);
// Ищем запятую - разделитель данных по Цельсию и Фаренгейту index = data.indexOf(",");
```



```
// Получить температуру в градусах Цельсия temp_c = data.substring(0, index);
// Получить температуру по Фаренгейту
temp_f = data.substring(index+1, data.length());
}
```

Как и в предыдущих примерах на Processing, программа начинается с импорта библиотеки serial и настройки последовательного порта. В секции setup о задается размер окна отображения, загружается сгенерированный шрифт и настраивается загрузка данных последовательного порта в буфер до получения символа точки. Функция draw о заполняет фон окна черным цветом и выводит значения температуры по Цельсию и по Фаренгейту двумя цветами. С помощью команды fill о вы сообщаете Processing о цвете (в значениях RGB) следующего элемента, который будет добавлен на экран. Функция serialEvent () вызывается при наступлении события bufferUntii о, она считывает содержимое буфера в строку, а затем разбивает его, учитывая расположение запятой. Два значения температуры хранятся в переменных, которые затем выводятся в окно приложения.

Результат выполнения программы показан на рис. 8.10.

Рис. 8.10. Отображение температуры на Processing

При изменении температуры датчика данные в окне Processing-приложения, а также светодиодная гистограмма должны обновиться.

ПРИМЕЧАНИЕ

Для просмотра демонстрационного видеофильма системы мониторинга температуры посетите страницу http://www.exploringarduino.com/content/ch8. Этот видеофильм доступен также на сайте издательства Wiley.

Резюме

В этой главе вы узнали следующее:

- Как организовать связь платы ***Arduino*** с несколькими I2C ведомыми устройствами (если они имеют разные адреса) по двухпроводному протоколу I2C.
- Как библиотека Wire облегчает связь с I2C-устройствами, подключенными к выводам A4 и A5 платы.
- Как объединить связь по протоколу I2C со сдвиговыми регистрами и обменом по последовательному порту для создания более сложных систем.
- Как генерировать шрифты для динамически обновляемых текстов в программе на Processing.
- Как отображать данные, полученные от I2C-устройств, подключенных к ***Arduino***, с помощью приложения на Processing.

ГЛАВА
Интерфейсная шина SPI
Список деталей
Для повторения примеров главы вам понадобятся следующие детали:
- плата ***Arduino Uno***;
- USB-кабель B (для ***Uno***);
- 1 красный светодиод;
- 1 желтый светодиод;
- 1 зеленый светодиод;
- 1 синий светодиод;
- 4 резистора номиналом 100 Ом;
- 2 резистора номиналом 4,7 кОм;
- динамик;
- цифровой SPI потенциометр MCP4231;
- перемычки;
- макетная плата.

Электронные ресурсы к главе
На странице http://www.exploringarduino.com/content/ch9 можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/exploringarduino в разделе Downloads.
Что вы узнаете в этой главе
Вы уже знакомы с двумя интерфейсами связи, используемыми платой ***Arduino***: шиной I2C и последовательной шиной UART. В этой главе вы узнаете о третьем интерфейсе цифровой связи, поддерживаемом аппаратными средствами ***Arduino***,— о последовательной шине периферийного интерфейса (или SPI).
В отличие от I2C, шина SPI имеет отдельные линии для отправки и получения данных, а также дополнительную линию для выбора ведомого устройства. Это требует

наличия дополнительных выводов, но устраняет проблему адресации ведомого устройства. SPI-интерфейс, по сравнению с I2C, проще в реализации и работает на более высокой скорости. Далее мы рассмотрим встроенную в ***Arduino*** IDE библиотеку SPI и аппаратные средства платы ***Arduino*** для подключения цифрового потенциометра. С помощью цифрового потенциометра будем регулировать яркость светодиода и громкость динамика, что позволит создать простое устройство, формирующее световые и звуковые эффекты.
ПРИМЕЧАНИЕ
Вы можете шаг за шагом посмотреть демонстрационный видеоурок к главе, расположенный по адресу http://www.jeremyblum.eom/2011/02/2011/02/20/arduino-tutorial-8-spi-interfaces1. Этот видеоурок также доступен на сайте издательства Wiley.
9.1. Общие сведения о протоколе SPI
Интерфейс SPI, разработанный компанией "Моторола", представляет собой полнодуплексный последовательный стандарт связи, который поддерживает одновременный двунаправленный обмен данными между ведущим устройством (мастером) и одним или несколькими подчиненными. Поскольку протокол SPI не имеет формального стандарта, работа различных устройств SPI может немного отличаться (например, различно число передаваемых в пакете битов или может отсутствовать линия выбора ведомого устройства). Далее рассмотрим общепринятые команды SPI, которые поддерживаются в ***Arduino*** IDE.
ВНИМАНИЕ!
Так как техническая реализация протокола SPI может быть разной, необходимо изучать техническое описание, прилагаемое к каждому устройству.
В зависимости от требований конкретного устройства существуют четыре основных способа реализации протокола SPI. SPI-устройства выступают при обмене в качестве подчиненных синхронных устройств, данные синхронизируются с тактовым сигналом (SCLK). Подчиненное устройство может воспринимать данные либо по положительному, либо по отрицательному фронту тактового сигнала (так называемая фаза синхронизации), а активное состояние SCLK по умолчанию

может быть высоким или низким уровнем (так называемая полярность синхронизации). В итоге получается, что обмен SPI в общей сложности можно настроить четырьмя способами (табл. 9.1).

Таблица 9.1. Режимы SPI в ***Arduino*** IDE

№	Режим SPI	Полярность синхронизации	Фаза синхронизации
1	Mode 0	LOW	По фронту синхросигнала
2	Mode 1	LOW	По спаду синхросигнала

' На русском: ЬЦр://нчккатрегка.ги/видеоуроки:8-интерфейсы-8р|.

Таблица 9.1 (окончание)

№	Режим SPI	Полярность синхронизации	Фаза синхронизации
3	Mode 2	HIGH	По спаду синхросигнала
4	Mode 3	HIGH	По фронту синхросигнала

9.2. Подключение устройств SPI

Систему обмена данными через SPI несложно настроить. Для связи между мастером и всеми подчиненными устройствами используется три вывода:

- ♦ последовательный сигнал синхронизации (SCLK);
- ♦ выход ведущего, вход ведомого (MOSI);
- ♦ вход ведущего, выход ведомого (MISO).

У каждого ведомого устройства также есть контакт выбора данного устройства (контакт SS). Следовательно, общее число портов ввода-вывода, необходимых на мастер-устройстве, всегда будет 3 +п, где п — число ведомых устройств. Пример SPI-системы с двумя ведомыми устройствами изображен на рис. 9.1.

GND

Рис. 9.1. Пример конфигурации SPI-устройств

9.2.1. Конфигурация интерфейса SPI

Любой интерфейс SPI содержит, как минимум, четыре линии передачи данных. Для каждого ведомого устройства добавляются дополнительные линии SS. Прежде чем отправлять или получать данные через SPI, нужно выяснить, что делают эти линии ввода-вывода и как они должны быть подключены (табл. 9.2).

Таблица 9.2. Описание линий ввода-вывода интерфейса SPI

Линии SPI	Описание
MOSI	Линия для отправки последовательных данных от ведущего устройства к ведомому
MISO	Линия для отправки последовательных данных от ведомого устройства к ведущему
SCLK	Линия синхронизации последовательных данных
SS	Линия выбора ведомого устройства, активный уровень — низкий

В отличие от интерфейса I2C, подтягивающие резисторы здесь не требуются, и протокол полностью двунаправленный. Итак, чтобы подключить устройство SPI к плате ***Arduino***, необходимо соединить его с контактами MOSI, MISO, SCLK и SS. После этого все готово к использованию ***Arduino*** библиотеки SPI.

Так как SPI не является универсальным стандартом, некоторые производители устройств SPI могут по-разному называть линии связи SPI. Линию выбора ведомого иногда называют CS, линию синхронизации — CLK; контакты MOSI и MISO ведомых устройств называют входом последовательных данных (SDI) и выходом последовательных данных (SDO) соответственно.

9.2.2. Протокол передачи данных SPI

Передача данных по SPI синхронизируется тактовым сигналом и зависит от состояния линий SS. Все команды, отправляемые мастером, проявляются на входах MOSI, MISO, SCLK всех ведомых устройств. Состояние контакта SS сообщает устройству, игнорировать эти данные или принимать. При написании программы следует учитывать, что при передаче данных только один контакт SS должен иметь низкий уровень.

Последовательность действий для связи с устройством SPI выглядит следующим образом:

1. Установить низкий уровень на линии SS устройства, с которым хотите установить связь.
2. Переключать на тактовой линии уровень сигнала вверх и вниз со скоростью, меньшей или

равной скорости передачи, поддерживаемой ведомым устройством.
3. На каждом такте отправлять 1 бит данных по линии MOSI или получать 1 бит данных по линии MISO.

4. Продолжать, пока передача (или прием) не закончится, и остановить переключения тактовой линии.
5. Установить на SS высокий уровень.
Обратите внимание, что на каждом такте данные должны быть отправлены (или получены). Например, далее в сценарии связи с цифровым потенциометром плата ***Arduino*** будет посылать данные, но ничего не получать от ведомого устройства.

9.3. Сравнение SPI и I2C

Многие виды устройств, в том числе акселерометры, цифровые потенциометры, дисплеи и т. п., доступны и в SPI- и в I2C-версиях. Что лучше выбрать? В табл. 9.3 перечислены некоторые преимущества устройств I2C и SPI. В конечном счете, выбор устройства зависит от конкретной ситу ации. Большинство начинающих считают, что работать с устройствами SPI легче, чем с устройствами I2C.

Таблица 9.3. Сравнение протоколов SPI и I2C

Преимущества SPI	Преимущества)2C
Может работать на более высокой скорости	Для организации обмена требуется только две линии
Легче программируется	Имеет аппаратную поддержку ***Arduino***
Не требует подтягивающих резисторов	
Имеет аппаратную поддержку ***Arduino***	

9.4. Подключение цифрового потенциометра SPI

Теперь пора применить полученные знания на практике. Рассмотрим устройство управления яркостью светодиодов с помощью цифрового потенциометра (кратко называемого digipot). В данном примере используем микросхему SPI цифрового потенциометра MCP4231 ЮЗЕ. Досту пно несколько вариантов данного чипа с различным значением сопротивления. Как и обычный потенциометр, цифровой имеет регулируемый вывод, который определяет сопротивление между двумя выводами микросхемы. Микросхема MCP4231 содержит два потенциометра на одном корпусе. Разрядность каждого из них составляет 7 бит, что определяет 128 значений в диапазоне от 0 до 10 кОм. Сначала с помощью цифрового потенциометра будем менять яркость свечения светодиода, а затем используем digipot для регулировки громкости динамика. Завершив эти два проекта, вы получите основу для реализации более сложных конструкций.

9.4.1. Техническое описание MCP4231

Прежде всего, следует изучить техническое описание микросхемы MCP4231, которое можно найти через поисковую систему Google. Ссылки на техническое опи-

сание для MCP4231 присутствуют на странице www.exploringarduiro.com/ content/ch9.
В техническом описании можно найти ответы на следующие вопросы:

- цоколевка микросхемы;
- какие выводы являются управляющими;
- как регулируется в данной микросхеме сопротивление потенциометра;
- какие команды SPI необходимы, чтобы управлять двумя потенциометрами.

Чтобы найти ответы на эти вопросы, на рис. 9.2-9.4 приведены некоторые важные фрагменты технического описания. Прежде всего, взгляните на цоколевку микросхемы MCP4231, изображенную на рис. 9.2.
При подготовке к работе с новым устройством необходимо сначала разобраться
с назначением контактов. Вот назначение выводов MCP4231:

- POA, POW и POB — выводы первого потенциометра;
- Pl A, Pl W и PIB — выводы второго потенциометра;
- VDD — вывод питания микросхемы 5 В;
- VSS — вывод подключения к земле;
- CS — контакт SS для интерфейса SPI, черта сверху означает, что активно1Й уровень низкий (О

В — чип выбран, 5 В — не выбран);

♦ SD1 и SDO — контакты последовательного ввода и вывода данных (соответствуют MOSI и MISO);

♦ SCK — линия синхронизации SPI;

♦ SHDN и WP — контакты для выключения и защиты от записи, соответственно. Для MCP4231 контакт WP не задействован и его можно игнорировать. Активный уровень на контакте SHDN низкий, как и на выводе CS. При низком уровне средний вывод потенциометра отключен. Чтобы потенциометр был всегда включен, необходимо соединить контакт SHDN непосредственно с шиной 5 В.

Далее необходимо узнать полное сопротивление потенциометра и сопротивление среднего вывода. Подобно обычному потенциометру, сопротивление между клеммами A и B в цифровом тоже постоянно. Средний вывод также имеет собственное

У

CS c 1 14 3 VDD

SCK C 2 13 3 SDO

SDIL3 12 □ SHDN

Vss c 4113WP

cmd r-lc -mn POR

PDIP, SOIC, TSSOP

Рис. 9.2. Цоколевка микросхемы MCP4231

сопротивление, и это нужно принимать во внимание. Обратимся к пятой странице технического описания (рис. 9.3).

Прежде всего, выясним полное сопротивление потенциометра, обозначаемое RAB. Доступны четыре варианта этого чипа, каждый с разным значением сопротивления (от 5 до 100 кОм). Далее используем вариант ЮЗ, сопротивление которого составляет примерно Ю кОм. Важно отметить, что цифровые потенциометры, как правило, имеют довольно большой разброс (из рис. 9.3 видно, что фактическое сопротивление может изменяться на ±20%). Также следует отметить, что собственное сопротивление среднего вывода потенциометра составляет от 75 до 160 0м. Это сопротивление нужно учитывать, особенно при управлении динамиком или светодиодом.

AC/DC CHARACTERISTICS (CONTINUED)

OC Characteristics Standard Operating Conditions (unless otherwise specified) Operating Temperature -40°C sTAs +125CC (extended) All parameters apply across the specified operating ranges unless noted. VD0 = +2.7V to 5.5V. 5 kfi. 10 kfi. 50 kfi. 100 kfi devices. Typical specifications represent values for V0D = 5.5V. TA = +25X.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Resistance (± 20%)	Rab	4.0	5	6.0	kfi	-502 devices (Note 1)
	8.0	10	12.0	kfi		-103 devices (Note 1)
	40.0	50	60.0	kfi		-503 devices (Note 1)
	80.0	100	120.0	kfi		-104 devices (Note 1)
Resolution	N	257			Taps	8-bit No Missing Codes
	129			Taps	7-bit	No Missing Codes
Step Resistance	R$	—	Rab 1 (256)	—	fi	8-bit Note 6
	—	Rab/ (128)	—	fi	7-bit	Note 6
Nominal Resistance Match	Irabo •	RABll i Rab	—	0.2	1.25	% MCP42X1 devices only
IRbwo-Rbw.I 1 Rbw	—	0.25	1.5	%	MCP42X2 devices only. Code = Full-Scale	
Wiper Resistance (Note 3. Note 4)	Rw	—	75	160	fi	VDD = 5.5 V, lw = 2.0 mA. code = OOh
	—	75	300	fi	VDD = 2.7 V. Iw = 2.0 mA. code = OOh	

Рис. 9.3. Фрагмент технического описания микросхемы MCP4231

Далее разберемся с командами для управления цифровым потенциометром. На MCP4231 необходимо отправить две команды. Первая определяет выбор нужного потенциометра, вторая устанавливает текущее значение сопротивления выбранного потенциометра. Формат команд приведен на рис. 9.4.

Из рис. 9.4 ясно, что существуют два вида команд: 8-разрядные и 16-разрядные. Первая команда

позволяет увеличить сопротивление потенциометра, вторая — установить произвольное значение сопротивления. Рассмотрим 16-битовую команду, обеспечивающую большую гибкость. По шине данных передается адрес ячейки памяти, код команды (чтение, запись, приращение или уменьшение) и значение данных.

Гпава 9. Интерфейсная шина SPI

193

8-битовая команда

Г

Г

_____A_____

Байт команды

_____л

*4

Л

16-битовая команда

A

Байт команды A Байт данных Л Л

(V >>

A D 3 A D 2 A D 1 A D 0 C 1 C 0 D 9 D 8

V г-1 —

Адрес A Биты

ячейки I данных

Биты команды

A	A D	A	A	C	C	D	D	D	D	D	D	D	D	D
D	D 0	D	D	1	0	9	8	7	6	5	4	3	2	1
3	2	1	0											

---у---л у—^----------•у--

Адрес A Биты

ячейки | данных

Биты

команды

Биты команды CC 1 O

0 0 = Запись данных

0 1 = INCR (увеличение)

1 0 = DECR (уменьшение) 1 1 = Read Data

(чтение данных)

Рис. 9.4. Формат команд MCP4231

В техническом описании приведены адреса регистров, связанных с каждым потенциометром. Регистр первого потенциометра расположен в ячейке памяти по адресу О, второго— по адресу 1. Зная это, можно отправить необходимые команды на установку значений для каждого потенциометра. Чтобы задать значение для первого потенциометра, первый байт будет содержать B00000000, а второй — величину сопротивления (0-128). Чтобы установить значение для второго потенциометра, первый байт будет равен B00010000, а второй— величине сопротивления. Как видно из рис. 9.4, первые 4 бита первого байта — это адрес регистра памяти, следующие 2 бита— код команды (00— для записи), следующие 2 бита— эго старшие биты величины сопротивления (должны быть равны нулю, потому что максимальное значение для этого потенциометра составляет 128).

Вот и все, что нужно знать для взаимодействия цифрового потенциометра с платой ***Arduino***. Теперь подключим его, чтобы управлять яркостью светодиодов.

9.4.2. Описание схемы устройства

Чтобы в полной мере проверить знания протокола SPI, возьмем две микросхемы MCP4231, что даст нам четыре управляемых потенциометра. Каждый из них подключен последовательно со своим светодиодом (красным, желтым, зеленым и синим) и регулирует его яркость. Задействованы только

две клеммы потенциометра. Один контакт каждого потенциометра подключен через резистор к шине 5 В, второй (средний вывод) — к аноду светодиода. Схема подключения одного светодиода изображена на рис. 9.5.

Катод светодиода подключен к земле. Когда сопротивление цифрового потенциометра минимально, ток течет от источника 5 В через резистор 100 Ом и средний вывод потенциометра (имеющий сопротивление ~ 75 Ом) и далее через светодиод. Когда сопротивление потенциометра максимально, ток течет через резистор 100 Ом, потенциометр (~ ЮкОм), а затем через светодиод. Даже при полностью выведенном потенциометре сопротивление участка цепи будет 175 Ом, что достаточно для ограничения тока через светодиод. При увеличении и уменьшении сопротивления цифрового потенциометра меняется ток через светодиод, а следовательно, его яркость. Этот метод регулирования яркости может оказаться очень полезным, если заняты все выводы ШИМ.

194

Часть III. Интерфейсы передачи данных

исс

Рис. 9.5. Схема подключения светодиода к потенциометру

Рис. 9.6. Схема подключения цифровых потенциометров

Гпава 9. Интерфейсная шина SPI

195

Теперь, учитывая цоколевку, подсоединяем цифровые потенциометры к шине SPI. На плате ***Arduino Uno*** контакт 13 — это SCK, контакт 12— MISO, контакт 11 — MOSI. Контакт 10 будем использовать как SS для одного чипа, а контакт 9— как SS для другого чипа. Схема подключения приведена на рис. 9.6. Помните, что каждую из микросхем нужно подключить к своей линии SCK, M1SO и MOST

Еще раз проверьте, что все правильно подключено, и перейдем к написанию программы управления яркостью светодиодов.

9.4.3. Написание программы

Чтобы проверить, что все подключено правильно и работает, напишем простую программу с использованием библиотеки SPI для управления яркостью четырех светодиодов.

Библиотека SP1 встроена в ***Arduino*** IDE, что сильно облегчает организацию обмена по протоколу SPI. Все, что остается программисту,— подключить библиотеку SPI и отправить данные в соответствии с протоколом SPI с помощью встроенных команд. Конечно, необходимо переключать флажки SS для выбора устройства. Чтобы отправить команду одному, из цифровых потенциометров на изменение яркости светодиода, необходимо выполнить следующее:

- ♦ установить на выводе SS требуемой микросхемы низкий уровень;
- ♦ отправить байт команды на выбранный потенциометр;
- ♦ отправить байт данных — значение для выбранного потенциометра;
- ♦ установить на выводе SS выбранной микросхемы высокий уровень.

Программа, приведенная в листинге 9.1, выполняет все описанные шаги: выбирает контакт SS, посылает байт выбора потенциометра и байт значения потенциометра по протоколу SPI. Функция spi.begin() инициализирует аппаратный интерфейс SPI на плате ***Arduino*** и после этого для передачи данных по шине SPI можно использовать команду SPI. transfer ().

Контакт выбора SS для первой микросхемы подключается к контакту 10 платы, для второй— к контакту 9. Функция settled () получает номер контакта SS, адрес регистра микросхемы и значение уровня потенциометра и передает данные в соответствующий чип. В цикле loop () яркость всех четырех светодиодов сначала увеличивается, а затем уменьшается. Загрузите программу на плату ***Arduino*** и увидите этот эффект.

ПРИМЕЧАНИЕ

Посмотреть видеоклип, демонстрирующий работу SPI цифрового потенциометра в качестве регулятора яркости, можно на странице http:// www.exploringarduino.com/ content/ch9. Этот видеофайл доступен также на сайте издательства Wiley.

Освоив простой пример, в следующем разделе создадим более сложное устройство, добавив звуковой эффект.

196

Часть III. Интерфейсы передачи данных

Листинг 9.1. Управление несколькими SPI цифровыми потенциометрами — SPlJed.ino

```
// Изменение яркости светодиодов не с помощью ШИМ,
//а регулировкой напряжения // Подключение библиотеки SPI #include <SPI.h>
// При подключении библиотеки SPI // по умолчанию используются контакты // 11 = MOSI, 12 = MISO, 13 = CLK
const   int SS1=10;    //
const   int SS2=9;     //
const   byte REG0=B00000000;       //
                  //
const   byte REG1=B00010000;       //
                  //
```
Контакт выбора Контакт выбора Команда записи (выбор первого Команда записи (выбор второго SS микросхемы 1 SS микросхемы 2 в регистр О потенциометра) в регистр 1 потенциометра)
```
void setup()
{
// Настройка контактов выбора SS на выход pinMode(SSI, OUTPUT); pinMode(SS2, OUTPUT);
// Инициализация аппаратного SPI SPI.begin();
}
// Подпрограмма выбора и отправки данных для каждого светодиода // Chip 1 (SS 10) регистр 0 - красный // Chip 1 (SS 10) регистр 1 - желтый // Chip 2 (SS 9) регистр 0 - зеленый // Chip 2 (SS 9) регистр 1 - синий
void setLed(int SS, int reg, int level)
{
digitalWrite(SS, LOW); // Установить SS в низкий уровень (выбор) SPI.transfer(reg);    // Отправка команды
SPI.transfer(level); // Отправка значения (0-128) digitalWrite(SS, HIGH); // Установить SS в высокий уровень
void loop()
{
for (int i=0; i<=128; i++)
{
setLed(SSl, REGO, i); setLed(SSl, REG1, i);
```
Гпава 9. Интерфейсная шина SPI
197
```
setLed(SS2, REGO, i); setLed(SS2, REG1, i); delay(10);
}
delay(300);
• (int i=128;    i>=0;   ; i-
setLed(SSI,    REGO,          i) ;
setLed(SSI,    REG1, i);
setLed(SS2,    REGO,          i) ;
setLed(SS2, delay(10);         REG1, i);
delay(300);
```
9.5. Создание световых и звуковых эффектов с помощью цифровых потенциометров SPI_____

Управление яркостью светодиодов— хороший пример для изучения протокола SPI, но менять яркость свечения можно и посредством ШИМ. Далее мы добавим в проект регулировку громкости звука, что невозможно реализовать с помощью ШИМ. Как упоминалось в главе 5, в ***Arduino*** IDE есть библиотека Топе, позволяющая генерировать на произвольном контакте платы ***Arduino*** меандр заданной частоты для воспроизведения звуков. Однако управлять громкостью звука прд этом нельзя. Затем мы собрали регулятор громкости, включив потенциометр последовательно с динамиком. Теперь используем цифровой потенциометр SPI для создания различных звуковых эффектов.

ПРИМЕЧАНИЕ

Данный проект может послужить основой для разработки многих гораздо более интересных устройств. Досконально разобравшись с описанной далее схемой и кодом программы, вы сможете творчески переработать эти идеи и создавать свои собственные оригинальные конструкции на основе

платы ***Arduino***.

9.5.1. Описание схемы устройства

Схема устройства похожа на предыдущую (см. рис. 9.6). Оставим три светодиода из четырех, а вместо последнего подключим динамик. Один вывод цифрового потенциометра, идущий к динамику, соединим через резистор с контактом платы ***Arduino***, который будет генерировать сигнал различной частоты. Сформированный меандр проходит через цифровой потенциометр, который меняет напряжение на динамике, а следовательно, громкость звучания. Измените предыдущую схему так, как показано на рис. 9.7.

198

Часть III. Интерфейсы передачи данных

Рис. 9.7. Схема управления светодиодами и громкостью динамика

Можете также поэкспериментировать с подключением аналоговых датчиков, чтобы световой и звуковой эффект зависел от освещения, движения и др.

9.5.2. Модификация программы

Внесем несколько простых изменений в нашу предыдущую программу для управления светодиодами (см. листинг 9.1). Добавьте переменную для контакта, подключенного к динамику, а также переменную для установки частоты сигнала, подаваемого на динамик. При желании внутри цикла loop о можно добавить операторы, увеличивающие частоту сигнала при каждой последующей итерации. Для установки громкости динамика подойдет та же самая функция setLedO, как и раньше, но ее название теперь вводит в заблуждение, так что рекомендуем его изменить. В листинге 9.2 она переименована в setRegO.

Гпава 9. Интерфейсная шина SPI

199

Листинг 9.2. Управление светодиодами и громкостью динамика с помощью SPI-потенциометров — LED_speaker.ino

```
// Изменение яркости светодиодов не с помощью ШИМ,
//а регулировкой входного напряжения // Подключение Arduino библиотеки SPI #include <SPI.h>
const int SPEAKER=8; // Вывод подключения динамика int freq = 100;
// При подключении библиотеки SPI //по умолчанию используются контакты // 11 = MOSI, 12 = MISO, 13 = CLK
const int SS1=10;
const int SS2=9;
const byte REG0=B00000000;
const byte REG1=B00010000;
// Контакт выбора SS микросхемы 1 // Контакт выбора SS микросхемы 2 // Команда записи в регистр 0 //(выбор первого потенциометра)
// Команда записи в регистр 1 //(выбор второго потенциометра)
void setup()
{
// Настройка выводов выбора SS на выход pinMode(SSI, OUTPUT); pinMode(SS2, OUTPUT);
// Инициализация аппаратного SPI SPI.begin();
// Подпрограмма выбора и отправки данных // Chip 1 (SS 10) регистр 0 - красный светодиод
// Chip 1 (SS 10) регистр 1 - желтый светодиод
// Chip 2 (SS 9) регистр 0 - зеленый светодиод
// Chip 2 (SS 9) регистр 1 - динамик
void setReg(int SS, int reg, int level)
{
digitalWrite(SS, LOW); SPI.transfer(reg) ;
SPI.transfer(level); digitalWrite(SS, HIGH);
}
// Установка SS в низкий уровень (выбор) // Отправка команды // Отправка значения (0-128)
// Установка SS в высокий уровень
void loop ()
{
```

```
tone(SPEAKER, freq);          // Частота звука
```
200
Часть III. Интерфейсы передачи данных
```
эг (int i=0;      i<=128; i++)
setReg(SSI,    REG0, i);
setReg(SSI,    REG1, i) ;
setReg(SS2,    REG0, i);
setReg(SS2,    REG1, i);
delay(10);
delay(300);
for (int i=128; i>=0; i—)
setReg(SSI,    REG0, i)
setReg(SSI,    REG1, i)
setReg(SS2,    REG0, i)
setReg(SS2,    REG1, i)
delay(10);
delay(300) ;
freq = freq+100;
if (freq > 2000) freq = 100;
```
Загрузите программу на плату ***Arduino*** и убедитесь, что меняется не только яркость светодиодов, но и громкость звука. На каждой итерации частота звука увеличивается на 100 Гц, пока не достигнет 2000 Гц. Громкость динамика регулирует тот же потенциометр, который управляет светодиодами.

И это всего лишь начало. Теперь у вас достаточно знаний, чтобы сделать действительно нечто впечатляющее. Вот несколько советов:

- можно управлять частотой и громкостью звука по сигналам от датчиков (например, инфракрасный датчик расстояния может менять частоту в зависимости от приближения к устройству и удаления от него);
- яркость светодиодов можно устанавливать в зависимости от внешних факторов, например от температуры;
- можно добавить кнопку, чтобы переключать громкость или частоту звука;
- можно сопоставить световые эффекты с проигрыванием музыки.

ПРИМЕЧАНИЕ

Посмотреть видеоклип, демонстрирующий работу SPI-устройства для создания световых и звуковых эффектов, можно на странице http:// www.exploringarduino.com/ content/ch9. Этот видеофайл доступен и на сайте издательства Wiley.

Гпава 9. Интерфейсная шина SPI

Резюме

201

В этой главе вы узнали следующее:

- Что согласно протоколу SPI для организации обмена требуются три общие линии (две линии данных и линия синхронизации) и по одной дополнительной линии выбора для каждого ведомого устройства.
- Что библиотека ***Arduino*** SPI позволяет облегчить коммуникации между платой ***Arduino*** и ведомыми устройствами.
- Что можно обмениваться данными с несколькими устройствами SPI, используя общие линии данных и синхронизации и раздельные линии SS выбора ведомого устройства.
- Как управлять SPI цифровыми потенциометрами с помощью библиотеки ***Arduino*** SPI.
- Как пользоваться техническими описаниями устройств.
- Как одновременно регулировать громкость и частоту звукового сигнала, используя библиотеку Топе и цифровой потенциометр SPI.

ГЛАВА

Взаимодействие с жидкокристаллическими дисплеями

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino Uno***;

- USB-кабель B (для ***Uno***);
- микрофон;
- цифровой SPI-потенциометр MCP4231;
- перемычки;
- макетная плата;
- динамик;
- две кнопки;
- вентилятор;
- 16x2-символьный ЖК-дисплей;
- 2 резистора номиналом 4,7 кОм;
- 2 резистора номиналом 10 кОм;
- 1 резистор номиналом 150 Ом;
- 1 потенциометр ЮкОм;
- датчик температуры TC74A0-5.0 VAT I2C;
- набор перемычек;
- макетная плата.

Электронные ресурсы к главе

На странице http://www.exploringarduino.com/content/chlO можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/exploringarduino в разделе Downloads.

Глава 10. Взаимодействие с жидкокристаллическими дисплеями

203

Что вы узнаете в этой главе

При проектировании автономных устройств стараются сделать так, чтобы и* работа не зависела от компьютера. До сих пор для отображения информации более сложной, чем включение индикатора, нам требовался внешний компьютер Добавив жидкокристаллический дисплей (LCD) на плату ***Arduino***, можно отображать сложную информацию (показания датчиков, отсчет временных промежутков, настройки и т. д.) непосредственно в устройстве без привязки к монитору компьютера.

В этой главе вы узнаете, как подключить LCD к плате ***Arduino***, как использовать библиотеку ***Arduino*** LiquidCrystal для вывода текста и произвольных пользовательских символов на экран жидкокристаллического дисплея. Изучив основы, мы, опираясь на опыт предыдущих глав, сможем создать простой прибор, измеряющий и показывающий температуру, а также включающий вентилятор для охлаждения. ЖК-дисплей будет отображать информацию о температуре, звуковой сигнал от динамика предупредит о повышении температуры, а вентилятор включится автоматически для охлаждения.

ПРИМЕЧАНИЕ

Видеоурок о работе с жидкокристаллическим дисплеем находится по адресу http://www.jeremyblum.com/2011/07/31/tutorial-13-for-arduino-liquid-crystal-displays .

Вы также можете найти этот видеофайл на сайте издательства Wiley.

10.1. Настройка жидкокристаллического дисплея

В устройствах, описанных в этой главе, мы применим жидкокристаллический дисплей. Подобные индикаторы часто встречаются и имеют множество модификаций. Наиболее распространены дисплеи, содержащие 16x2 символов, имеющие 16 (или 14, если нет подсветки) контактов в один ряд. Для рассмотренных далее примеров выбран 16-контактный ЖК-дисплей, на экране которого одновременно может отображаться 32 символа (16 столбцов и 2 строки).

Если у вашего дисплея нет штырьков для монтажа, необходимо сначала припаять их, чтобы можно было установить его на макетной плате. Если штыревые контакты припаяны (как на рис. 10.1), можно сразу закрепить индикатор на макетной плате.

Теперь нужно подключить дисплей, смонтированный на макетной плате, к ***Arduino***. Все параллельные ЖК-индикаторы имеют одинаковые выходы и их можно подключить в одном из двух вариантов: 4-контактном и 8-контактном. Для передачи информации служат четыре вывода, есть также контакты готовности данных, выбора режима команд или режима индикации, установки режимов чтения или записи данных. Назначение всех контактов приведено в табл. 10.1.

1 На русском: НЦр://\¥!ккатрегка.ги/видеоуроки:13-жк-дисплеи-1сс1.

Рис. 10.1. ЖК-дисплей с припаянными штыревыми контактами
Таблица 10.1. Контакты параллельного ЖК-дисплея

Контакт	Обозначение	Назначение
1	Vss	Заземление
2	Vdd	Питание +5 В
3	Vo	Регулировка контрастности (потенциометр)
4	Rs	Выбор режима (командный или отображения данных)
5	RW	Чтение/запись
6	En	Готовность данных
7	DO	Линия передачи данных 0 (не используется)
8	D1	Линия передачи данных 1 (не используется)
9	D2	Линия передачи данных 2 (не используется)
10	D3	Линия передачи данных 3 (не используется)
11	D4	Линия передачи данных 4
12	D5	Линия передачи данных 5
13	D6	Линия передачи данных 6
14	D7	Линия передачи данных 7
15	A	Анод подсветки
16	К	Катод подсветки

Назначение контактов следующее:

♦ Напряжение на контакте регулировки контрастности Vo определяет яркость дисплея, контакт подключается к среднему выводу потенциометра.

♦ Контакт выбора режима Rs переводит дисплей в режим приема команд или символов, данные, поступающие в дисплей, интерпретируются либо как данные, либо как символ.

♦ Контакт RW у нас всегда будет соединен с землей, это означает, что данные всегда передаются в дисплей, а не читаются из него.

♦ Сигнал на контакте EN сообщает, что данные готовы к приему или передаче.

♦ Контакты D4-D7 используются для передачи данных, а контакты D0-D3 остаются неподключенными.

Если ЖК-дисплей снабжен встроенной системой светодиодной подсветки с внутренним ограничивающим резистором, можно непосредственно подключить анод к +5 В, а катод к земле, если такого резистора нет, следует добавить токоограничивающий резистор в линию между анодом и катодом. Подробности необходимо уточнять в техническом описании конкретного дисплея.

В табл. Ю.2 приведен рекомендуемый порядок соединения контактов ЖК-дисплея и платы ***Arduino***. Можно подключить дисплей и к другим контактам ввода-вывода.

I I
s * с г т о
NI 90TVNV
и)Л роЭ AS! азмсм ‘
эхоцлрлеммл*
dSOI
щ

(ONri") oumpjy £
S й TViioia
o i r £ fr s 9 Z 86

Рис. 10.2. Подключение ЖК-дисплея к макетной плате и ***Arduino***

Таблица 10.2. Таблица соединения контактов

Контакты ЖК-дисплея	Контакты ***Arduino***

RS D2
EN D3
D4 D4
D5 D5
D6 D6
D7 D7

Подключите ЖК-дисплей к плате ***Arduino*** по схеме, приведенной на рис. Ю.2.

Теперь ЖК-дисплей готов к работе. Как только вы загрузите программу из следующего раздела на плату ***Arduino***, то сможете отображать текст на экране. С помощью потенциометра отрегулируйте контрастность символов.

10.2. Библиотека LiquidCrystal

Arduino IDE включает библиотеку LiquidCrystal, имеющую набор функций для взаимодействия с параллельными ЖК-дисплеями. Библиотека довольно обширна и реализует такие функции, как мигающий курсор, автоматическая прокрутка текста, создание пользовательских символов, изменение направления вывода текста. В этой главе мы рассмотрим только самые необходимые инструменты, чтобы понять, как взаимодействовать с дисплеем. Полное описание библиотечных функций и примеры, иллюстрирующие их использование, можно найти на сайте ***Arduino***: http://arduino.cc/en/Reference/LiquidCrystaI.

10.3. Вывод текста на дисплей

В нашем первом примере выведем на экран индикатора текст и число, меняющееся со временем. Это упражнение проиллюстрирует, как инициализировать дисплей, отображать текст и перемещать курсор.

Сначала необходимо подключить библиотеку LiquidCrystal:

```
#include <LiquidCrystal.h>
```

Затем нужно инициализировать объект LiquidCrystal:

```
LiquidCrystal LED (2,3,4,5,6,7);
```

Аргументы для инициализации объекта представляют собой контакты ***Arduino***, подключенные к выводам ЖК-дисплея в следующем порядке: RS, EN, D4, D5, D6, D7. Чтобы настроить конфигурацию ЖК-индикатора, в функции setup () необходимо вызвать библиотечную функцию begin o, ее аргументы— число столбцов и строк вашего ЖК-дисплея:

```
LED.begin (16, 2);
```

Гпава 10. Взаимодействие с жидкокристаллическими дисплеями
207

После этого можно вызывать библиотечные функции print () и setCursorO для вывода текста в определенном месте дисплея. Например, если вы хотите напечатать мое имя во второй строке, введите следующие команды:

```
LED.setCursor (0,1) ; LED.print ("Jeremy Blum");
```

Позиции на экране нумеруются, начиная с (0,0), что соответствует левому верхнему углу. Первый аргумент функции setCursorO определяет номер столбца, а второй — номер строки. По умолчанию курсор находится в позиции (0,0). При вызове функции print o без указания расположения курсора текст выводится, начиная с позиции в верхнем левом углу.

ВНИМАНИЕ!

Функция print () не проверяет длину строк и попытка напечатать последе ватель-ность символов, превышающую число знаков в строке, приведет к непредсказуемым последствиям. Всегда проверяйте, чтобы выводимый текст помещался на экране индикатора.

Теперь можно написать простую программу, которая отображает текст в первой строке и во втором столбце, а также выводит значение счетчика, которое уЕ;еличи-вается на единицу каждую секунду. В листинге 10.1 приведен текст данной программы. Загрузите ее на плату ***Arduino*** и убедитесь, что все работает правильно. Если изображение на дисплее отсутствует, отрегулируйте контрастность с помощью потенциометра.

Листинг 10.1. Вывод текста и значения счетчика на экран ЖК-дисплея — LCD_text.ino

```
// Текст и значение инкрементируемого счетчика на экране // Подключение библиотеки: linclude <LiquidCrystal.h>
// Начальное значение time=0 int time =0;
// Инициализация экземпляра библиотеки LiquidCrystal LiquidCrystal LED(2, 3, 4, 5, 6, 7);
```

```
void setup()
{
// Настройка экземпляра дисплея - число столбцов и строк: LED.begin(16, 2);
// Вывод текстового сообщения на экран дисплея LED.print("Jeremy's Display");
}
void loop()
{
// Установить курсор на вторую строку в первую позицию LED.setCursor(0,1);
```

```
// Вывод значения счетчика LED.print(time);
// Пауза 1 секунда delay(1000);
// Увеличение значения счетчика time++;
}
```
Программа должна быть вам понятна. В начале подключается библиотека LiquidCrystal. Переменной времени time присваивается нулевое начальное значение и затем оно увеличивается в цикле раз в секунду. Объект LiquidCrystal назван LED и ему присвоены значения контактов, соответствующие схеме соединения. Команда LED.begin(1 б, 2) задает размер дисплея: 16 столбцов и 2 строки. Поскольку первая строка не меняется, ее можно инициализировать в функции setup о командой LED.print(). Обратите внимание, что команда выполняется без предварительной установки курсора, потому что по умолчанию текст выводится, начиная с позиции (0,0). В цикле курсор всегда возвращается в позицию (0,1), поэтому при выводе значения переменной time каждый раз переписывается вторая строка. Экран обновляется раз в секунду и одновременно увеличивается счетчик времени.

10.4. Создание специальных символов и анимации

Что делать, если вы хотите отобразить информацию, которую нельзя выразить с помощью обычных символов? Например, требуется вывести греческую букву, знак градуса или индикатор загрузки. К счастью, библиотека LiquidCrystal поддерживает создание пользовательских символов, которые могут быть отображены на дисплее. В следующем примере мы воспользуемся этой возможностью, чтобы реализовать движущийся значок, перемещающийся по экрану. А после этого создадим знак градуса для отображения измеряемой температуры.

Создавать пользовательские символы просто. Если внимательно посмотреть на ЖК-дисплей, то видно, что каждое знакоместо состоит из матрицы 5x8 пикселов. Чтобы создать пользовательский символ, нужно определить значение каждой из этих точек и отправить соответствующую информацию на дисплей. Попробуем создать несколько символов, которые будут заполнять вторую строку дисплея анимированным прогресс-баром. Поскольку ширина каждого символа составляет 5 пикселов, для заполнения всей строки потребуются 5 пользовательских символов: первый с одним заполненным столбцом, второй с двумя столбцами и т. д.

В начале программы создаем для каждого символа массив байтов, в котором включенным пикселам будут соответствовать единицы, а выключенным — нули. Для первого символа (заполняющего 20% от ширины строки) массив выглядит следующим образом:

```
byte p20[8] = {
вюооо,
```

```
B10000,
вюооо,
вюооо,
вюооо,
вюооо,
вюооо,
вюооо,
};
```
Я назвал этот массив p20, чтобы показать, что этот символ заполняет 20% пространства. В функции setup () вызываем библиотечную функцию createChar(), чтобы назначить данному массиву

идентификатор ID пользовательского символа. Пользовательские символы имеют ID от 0 до 7, вы можете в общей сложности сформировать восемь таких символов. Определим массив p20 как пользовательский символ с ID=0:

```
LED.createChar (0, p20);
```

Для отображения пользовательского символа на дисплее поместите курсор в нужное место и выполните команду

```
LED.write ((byte) 0);
```

Добавьте остальные пользовательские символы и с помощью двух вложенных циклов в основном цикле программы 1oop() обновляйте прогресс-бар. Код данной программы приведен в листинге 10.2.

Листинг 10.2. Код прогресс-бара на экране дисплея — LCD_progress_bar.ino

```
// Прогресс-бар на Ж-дисплее // Подключение библиотеки LiquidCrystal #include <LiquidCrystal.h>
// Инициализация экземпляра библиотеки LiquidCrystal LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
// Создание массивов для символов прогресс-бара byte p20[8] = {
ВЮООО,
ВЮООО,
ВЮООО,
ВЮООО,
ВЮООО,
ВЮООО,
вюооо,
вюооо,
};
byte p40[8] = {
В1Ю00,
В1Ю00,
210
Часть III. Интерфейсы передачи данных
B11000,
B11000,
B11000,
B11000,
B11000,
B11000,
};
byte рбО[8] = {
B11100,
B11100,
B11100,
B11100,
B11100,
B11100,
B11100,
Biiioo,
};
byte p80[8] = {
B11110,
B11110,
B11110,
B11110,
B11110,
B11110,
B11110,
B11110,
};
byte p100[8] = { B11111,
```

```
B11111,
B11111,
B11111,
B11111,
B11111,
B11111,
B11111,
};
void setup()
{
// Настройка экземпляра дисплея - число столбцов и строк: LED.begin(16, 2);
// Вывод текста на Ж-дисплей LED.print("Jeremy's Display");
// Определение пользовательских символов LED.createChar(0, p20);
```
Гпава 10. Взаимодействие с жидкокристаллическими дисплеями
211
```
LED.createChar(1, p4 0) ; LED.createChar(2, рбО); LED.createChar(3, p80); LED.createChar(4, plOO);
}
void loopO
{
// Курсор в начало второй строки LED.setCursor(0,1);
// Очистка второй строки
// 16 пробелов
LED.print("     ");
// Перебор каждого символа на второй строке for (int i = 0; i<16; i++)
{
// Перебор каждого значения (p20 - plOO) for (int j=0; j<5; j++)
{
LED.setCursor(i, 1);   //        Столбец установки курсора
LED.write(j);   //        Вывод символа
delay(lOO);    //        Задержка
}
}
}
```
В начале каждого цикла во все 16 позиций второй строки выводится символ пробела (строка очищается). Внешний цикл for о перебирает все 16 знакомест строки. Во внутреннем цикле for о в каждое знакоместо с задержкой выводится увеличивающийся пользовательский символ прогресс-бара.

ПРИМЕЧАНИЕ

Для просмотра видеоклипа, демонстрирующего отображение прогресс-бара на ЖК-дисплее, посетите страницу http://www.exploringarduino.com/content/ch10. Этот видеоклип доступен также на сайте издательства Wiley.

10.5. Создание регулятора температуры

Давайте теперь заставим дисплей выполнять полезную функцию. Добавим 1"С-датчик температуры, рассмотренный в главе 8, вентилятор и динамик из главы 5. Дисплей будет показывать температуру и текущее состояние вентилятора. Когда станет жарко, динамик издаст предупреждающий звук и вентилятор включится. Когда снова станет прохладно, вентилятор перестанет работать. С помощью двух кнопок и фрагмента кода подавления дребезга из листинга 2.5 вы получите возможность увеличивать или уменьшать требуемую температуру.

212

Часть III. Интерфейсы передачи данных

10.5.1. Монтаж схемы устройства

Соединение деталей для этого устройства представляет собой комбинацию предыдущих схем. Маломощный вентилятор с двигателем постоянного тока подключается непосредственно к контакту платы ***Arduino***. При желании использовать мощный вентилятор, можно подключить его к внешнему источнику питания через транзистор (см. главу 4). Для этого подойдет схема, изображенная на рис.

4.1. ЖК-дисплей подключим так же, как в предыдущем примере (см. рис. 10.2).
Один из выводов кнопок подключен к шине питания, а другой — к контакту ***Arduino*** и через резистор 10 кОм на землю.
Один вывод динамика подключен к контакту ***Arduino***, а другой — через резистор 150 Ом к земле. Частота звука устанавливается программно.
1“С-датчик температуры подключен так же, как в главе 8. Монтажная схема устройства изображена на рис. 10.3. Изображение датчика температуры сделано частично прозрачным, чтобы можно было увидеть потенциометр позади него.
Глава 10. Взаимодействие с жидкокристаллическими дисплеями
213
10.5.2. Отображение данных на ЖК-дисплее
Для облегчения вывода информации на ЖК-дисплей заранее определим некоторые параметры. Решим, что температуру всегда будем отображать в виде двух цифр. Надписи "Current:" и "Set:" всегда будут неизменными, их можно вывести на экран один раз. Так как температура представлена в виде двух цифр, местоположение двух надписей "°C" тоже будет фиксированным. Текущее значение температуры будет отображаться в позиции (8,0) и обновляться при каждом цикле 1oop(), значение пороговой температуры размещено в позиции (8,1) и обновляется при нажатии кнопки. Значок включения вентилятора будет отображаться в нижней правой части дисплея в позиции (15,1) и обновляться, когда изменяется состояние.
При работе программы ЖК-дисплей будет выглядеть примерно так, как показано на рис. 10.4.
Рис. 10.4. Отображение данных на ЖК-дисплее
Знак градуса, символы включенного и выключенного вентилятора не входят в стандартный набор символов. Следовательно, в начале программы нужно сформировать соответствующие массивы, как показано в листинге 10.3.
Листинг 10.3. Массивы, определяющие пользовательские символы
// Пользовательский символ градуса byte degree[8] = {
B00110,
B01001,
B01001,
B00110,
B00000,
B00000,
B00000,
B00000,
};
// Символ "вентилятор включен" byte fan_on[8] = {
B00100,
B10101,
B01110,
Blllll,
B01110,
214
Часть III. Интерфейсы передачи данных
B10101,
вооюо,
воооо,
};
// Символ "вентилятор выключен" byte fan_off[8] = {
ВООЮО,
ВООЮО,
ВООЮО,
B11111,
ВООЮО,
ВООЮО,
ВООЮО,

booooo,
};
Вывод данных на экран ЖК-дисплея будет осуществлен в функции setup о. Размещаем курсор в нужной позиции и с помощью библиотечных функций print () и write () выводим надписи на экран (листинг 10.4).
Листинг 10.4. Вывод информации на экран ЖК-дисплея
// Создаем пользовательские символы LED.createChar(0, degree); lcd.createChar(1, fan_off); LED.createChar(2, fan_on);
// Отображаем статические надписи на экране Ж-дисплея LED.setCursor(0,0); LED.print("Current:"); LED.setCursor(10,0); LED.write((byte)0); LED.setCursor(11,0); LED.print("C"); LED.setCursor(0,1); LED.print("Set:"); LED.setCursor(10,1); LED.write((byte)0); LED.setCursor(11,1); LED.print("C"); LED.setCursor(15,1); LED.write(1);
В каждом цикле loop о обновляем текущее значение температуры и состояние значка включения вентилятора. Перед этим помещаем в нужное место курсор.
Глава 10. Взаимодействие с жидкокристаллическими дисплеями
215
10.5.3. Установка порогового значения температуры с помощью кнопок
В главе 2 мы использовали функцию предотвращения дребезга debounce(). Здесь немного изменим ее, чтобы устранить дребезг нескольких кнопок. Одна кнопка увеличивает пороговое значение температуры, другая уменьшает его. Необходимо определить переменные для хранения текущего и предыдущего состояний кнопок:
// Переменные, используемые при устранении дребезга кнопок boolean lastDownTernpButton = LOW; boolean currentDownTempButton = LOW; boolean lastUpTempButton = LOW; boolean currentUpTempButton = LOW;
Переделаем функцию устранения дребезга debounce () для случая нескольких кнопок. Добавим второй аргумент, задающий кнопку, для которой устраняем дребезг (листинг 10.5).
Листинг 10.5. Функция устранения дребезга двух кнопок
// Функция проверки дребезга для нескольких кнопок boolean debounce(boolean last, int pin)
{
boolean current = digitalRead(pin); if (last != current)
{
delay(5);
current = digitalRead(pin);
}
return current;
}
В основном цикле программы 1oop() проверяем состояние кнопок, устраняем дребезг, изменяем при нажатии кнопки значение переменной пороговой температуры set temp и обновляем значение на экране ЖК-дисплея (листинг 10.6).
Листинг 10.6. Программа установки пороговой температуры с помощью двух кнопок
// Устранение дребезга нескольких кнопок
currentDownTempButton = debounce (lastDownTernpButton, DOWN_BUTTQN-)7 currentUpTempButton = debounce(lastUpTempButton, UP_BUTTON);
// Уменьшить значение set temp /
if (lastDownTernpButton == LOW && currentDownTempButton == HIGH)
{
set_temp—;
}
216
Часть III. Интерфейсы передачи данных
// Увеличить значение set temp
else if (lastUpTempButton == LOW && currentUpTempButton == HIGH) {
set_temp++;
}
// Вывести значение set temp на ЖК-дисплей LED.setCursor(8,1); LED.print(set_temp);

// Изменить последние значения статуса кнопок lastDownTempButton = currentDownTempButton; lastUpTempButton = currentUpTempButton;
В листинге 10.6 запускаем функцию устранения дребезга debounce о для каждой кнопки, а затем изменяем переменную set_temp заданной пороговой температуры при нажатии одной из кнопок. Потом значение температуры обновляется на экране ЖК-дисплея.
10.5.4. Добавляем вентилятор и звуковое оповещение
Теперь добавим фрагмент кода для управления вентилятором и динамиком. Хотя ЖК-дисплей и так информирует нас обо всем, никогда не помешает дополнительное звуковое оповещение о событиях. Например, подача звукового сигнала перед включением вентилятора. В этом примере мы используем команды tone о в паре с задержкой delay о и notone о. Чтобы задать длительность звука, можно указать второй аргумент команды tone о. Добавив переменную состояния, можно устанавливать звуковое оповещение динамиком только один раз при превышении порогового значения температуры.
Фрагмент кода, приведенный в листинге 10.7, проверяет температуру и управляет динамиком, вентилятором и индикатором вентилятора на ЖК-дисплее.
Листинг 10.7, Выдача звукового оповещения при превышении пороговой температуры
// Стало жарко! if (c >= set_temp)
{
// Издать звук динамиком if (!one_time)
{
tone(SPEAKER, 400);
delay(500);
one time = true;
}

// Выключить динамик else {
noTone(SPEAKER);
}
// Включить вентилятор и знак на Ж-дисплее digitalWrite(FAN, HIGH); LED.setCursor(15,1); LED.write(2);
}
// Стало прохладнее else {
// Выключить динамик
// Сбросить состояние one_time в false
// Выключить вентилятор и значок на Ж-дисплее
noTone(SPEAKER);
one_time = false;
digitalWrite(FAN, LOW);
LED.setCursor(15,1) ;
LED.write(1);
}
Переменная one time позволяет выдать однократный, а не непрерывный звуковой сигнал. После того как динамик издает звук длительностью 500 мс частотой 400 Гц, переменная устанавливается в true и сбрасывается в false только тогда, когда температура падает обратно ниже заданного порога.
10.5.5. Итог рсего: полная программа
Теперь соберем части в единое целое. В начале программы необходимо подключить библиотеки, определить флажки и инициализировать переменные состояния. Полный текст программы приведен в листинге 10.8. Загрузите ее на плату ***Arduino*** и сравните результаты с видеоклипом, демонстрирующим систему в действии.
Листинг 10.8. Программа автоматического регулятора температуры — LCD_thermostat.ino
// Это программа автоматического регулятора температуры // Для вывода температуры используются 2 знака // Использует библиотеку Wire с установкой адреса #include <Wire.h>
#define TEMP_ADDR 72
// Подключение и инициализация библиотеки LiquidCrystal:
#include <LiquidCrystal.h>

```
LiquidCrystal LED(2, 3, 4, 5, 6, 7);
```

```
II Пользовательский символ градуса byte degree[8] = {
B00110,
B01001,
B01001,
B00110,
B00000,
B00000,
B00000,
B00000,
};
// Пользовательский символ "вентилятор включен" byte fan_on[8] = {
B00100,
B10101,
B01110,
Bllllll,
B01110,
B10101,
B00100,
B00000,
};
// Пользовательский символ "вентилятор выключен" byte fan_off[8] = {
B00100,
B00100,
B00100,          \
Bllllll,  \
B00100,
B00100,
B00100,
B00000,
};
// Выводы подключения динамика, кнопок, вентилятора const int SPEAKER =8;
const int DOWN_BUTTON =9; const int UP_BUTTON =10; const int FAN =11;
// Переменные для устранения дребезга кнопок boolean lastDownTempButton = LOW; boolean currentDownTempButton = LOW; boolean lastUpTempButton = LOW; boolean currentUpTempButton = LOW;
int set_temp =23;        // Значение граничной температуры
boolean one time = false; // Флаг звука динамика
```

```
void setup()
pinMode(FAN, OUT PUT);
// Создание объекта Wire (I2C-датчик температуры) Wire.begin();
// Настройки дисплея (число столбцов и строк) LED.begin(16, 2);
// Определить пользовательские символы LED.createChar(0, degree); lcd.createChar(1, fan_off); LED.createChar(2, fan_on);
// Вывод закрепленных сообщений на дисплее
LED.setCursor (0,0);
LED.print("Current:");
LED.setCursor(10,0);
LED.write((byte)0);
LED.setCursor(11,0);
LED.print("C");
```

```
LED.setCursor(0,1);
LED.print("Set:");
LED.setCursor (10,1);
LED.write((byte)0);
LED.setCursor(11,1);
LED.print("C");
LED.setCursor(15,1);
LED.write (1);
// Функция проверки на дребезг для нескольких кнопок boolean'debounce(boolean last, int pin)
{
boolean current = digitalRead(pin); if (l$st != current)
{
delay(5);
current = digitalRead(pin);
}
return current;
}
void loop()
{
// Получить значение от датчика температуры Wire.beginTransmission(TEMP_ADDR);
Wire.write(0);
Wire.endTransmission();
Wire.requestFrom(TEMP_ADDR, 1);
```

```
II Ожидаем передачу // Получить 1 байт while(Wire.available() == 0); int c = Wire.read();
// Установить курсор //и вывести текущее значение LED.setCursor(8,0); LED.print(c);
// Проверка на дребезг для двух кнопок
currentDownTempButton = debounce(lastDownTempButton, DOWN_BUTTON); currentUpTempButton =
debounce(lastUpTempButton, UP_BUTTON);
// Уменьшить пороговое значение температуры
if (lastDownTempButton== LOW && currentDownTempButton == HIGH)
{
set_temp—;
}
// Увеличить пороговое значение температуры
else if (lastUpTempButton== LOW && currentUpTempButton == HIGH)
{
set_temp++;
}
// Вывод порогового значения на экран LED.setCursor(8,1); LED.print(set_temp);
lastDownTempButton = currentDownTempButton; lastUpTempButton = currentUpTempButton;
// Очень жарко! if (c >= set_temp)
{
// Однократный звуковой сигнал на динамик if (!one_time)
{
tone(SPEAKER, 400); delay(500); one_time = true;
}
// Отключить вывод звука else {
noTone(SPEAKER);
}
// Включить вентилятор и вывести значок на дисплей digitalWrite(FAN, HIGH); LED.setCursor(15,1);
LED.write(2);
}
```

221
```
// Не жарко! else {
// Выключить динамик
// Сбросить состояние onetime в false
// Выключить вентилятор и значок на Ж-дисплее
noTone(SPEAKER);
one_time = false;
digitalWrite(FAN, LOW);
LED.setCursor(15,1) ;
LED.write(1);
}
}
```
Теперь, чтобы посмотреть температуру, необязательно подключать плату ***Arduino*** к компьютеру. Можно питать плату от батарейки или автономного источника питания и поместить ее в любом месте вашей комнаты.

ПРИМЕЧАНИЕ

Посмотреть видеоклип, демонстрирующий действие автономного терморегулятора, можно на странице http://www.exploringarduino.com/content/ch10. Этот видеофайл доступен также на сайте издательства Wiley.

10.6. Как усовершенствовать проект

Функциональные возможности описанной программы можно расширить. Вот несколько советов по улучшению проекта:

- Для увеличения мощности подключить вентилятор через транзистор.
- В зависимости от текущей температуры регулировать скорость вращения вентилятора с помощью широтно-импульсной модуляции.
- Добавить светодиоды для визуальной индикации предупреждений.
- Сделать звуковое оповещение в виде мелодии.
- Добавить фотодатчик и автоматически регулировать яркость подсветки дисплея в зависимости от освещенности в комнате, используя потенциометр SPI из главы 9.

Резюме

В этой главе вы узнали следующее:

- Как подключить ЖК-дисплей к плате ***Arduino*** по стандартной схеме.
- Как создавать собственные символы для ЖК-дисплея с помощью генерации произвольных растровых изображений.
- Как изменить функцию устранения дребезга кнопки для нескольких кнопок.
- Как объединить датчики, двигатели, кнопки и ЖК-дисплей в едином автономном устройстве.

ГЛАВА 11

Беспроводная связь с помощью радиомодулей XBee

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- 2 платы ***Arduino*** (рекомендуется ***Uno*** или Leonardo);
- USB-кабели для программирования плат ***Arduino***;
- источники питания для каждой платы ***Arduino***;
- адаптер SparkFun USB XBee Explorer;
- 2 радиомодуля XBee Series 1;
- 2 платы расширения XBee shield;
- кнопка;
- пьезозуммер;
- RGB-светодиод с общим катодом;
- 1 резистор номиналом 10 кОм;
- потенциометр 10 кОм;
- 1 резистор номиналом 150 Ом;
- 3 резистора номиналом 220 Ом;
- набор перемычек;

♦ 2 макетные платы.

Электронные ресурсы к главе

На странице http://www.exploringarduino.coin/content/chll можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/exploringarduino в разделе Downloads.

Что вы узнаете в этой главе

Настала пора отказаться от соединительных кабелей! При создании устройств на базе микроконтроллеров несложно реализовать беспроводную связь. Есть много

Гпава 11. Беспроводная связь с помощью радиомодулей XBee

223

способов для организации беспроводной связи, для плат ***Arduino*** проще всего применить радиомодули XBee, которые производит компания Digi. Модули XBee работают как беспроводной последовательный порт, что позволяет для связи с ними использовать чтение и отправку последовательных данных, а это мы уже изучили. В этой главе описана беспроводная связь только модулей XBee, но мы рассмотрим некоторые важные сведения, которые касаются беспроводной связи в целом.

Радиомодули XBee позволяют осуществить связь между ***Arduino*** и компьютером или между несколькими платами ***Arduino***. Далее мы расскажем об этом подробно.

ПРИМЕЧАНИЕ

Видеоурок по работе с радиомодулями XBee находится по адресу http://www. jeremyblum.com/2011/02/27/arduino-tutorial-9-wireless-communication/1. Этот видеофайл доступен также на сайте издательства Wiley.

11.1. Общие сведения о беспроводной связи XBee

Название говорит само за себя. Беспроводная связь позволяет связывать и общаться двум и более устройствам без проводов. Беспроводные передатчики передают данные через свободное пространство, излучая электромагнитные волны на определенной частоте. Существуют различные технологии передачи для разных диапазонов, чтобы предотвратить взаимные помехи. Правительственные учреждения, например Федеральная комиссия по связи (FCC) в США, регулируют эти вопросы, публикуя правила распределения частот. Модуль XBee передает данные на частоте 2,4 ГГц, на которой работают и многие другие устройства, например WiFi-маршрутизаторы. Это диапазон частот ISM (выделенный промышленным, научным и медицинским учреждениям), отведенный для нелицензионного использования беспроводной связи. Модули XBee соответствуют стандарту IEEE 802.15.4, который содержит перечень правил эксплуатации беспроводных персональных сетей (PAN). Модули XBee, как правило, соединяют согласно конфигурации PAN "точка-точка" или "точка-многоточка" (рис. 11.1). Схема "точка-точка" удобна, когда необходимо заменить проводную последовательную связь между двумя удаленными устройствами беспроводной. Конфигурация "точка-многоточка" часто используется для создания распределенных сетей датчиков.

0-0 0£гЮ

Точка — Точка Точка — Многоточка

Рис. 11.1. PAN-конфигурации

1 На русском: Мрг/Мккатрегка.ги/видеоурокиЭ-беспроводная-связь.

224

Часть III. Интерфейсы передачи данных

11.1.1. Радиомодули XBee

Радиомодули XBee могут обмениваться данными в последовательном режиме или с помощью интерфейса API. В последнем случае значения с цифровых или аналоговых выводов передаются напрямую. Далее мы рассмотрим работу модулей XBee в режиме простого последовательного обмена. Последовательные данные, передаваемые одним модулем, приходят в другой, и нА0борот. Это позволяет заменить проводной последовательный порт для связи между двумя платами ***Arduino*** (или для связи компьютера с платой ***Arduino***) радиомодулями XBee.

Все модули XBee имеют 20 контактов. В этой главе рассмотрим устройства XBee серии 1, работающие по стандарту 802.15.4. С их помощью можно реализовать PAN-конфигурации "точка-точка" и "точка-многоточка", но они не поддерживают стандарт ZigBee. У вас могут оказаться и модули других типов, но скорее всего это будут XBee S1 (рис. 11.2).

Следует помнить, что XBee модули серий 1 и 2 несовместимы друг с другом. Для начинающих я

рекомендую модули серии 1, т. к. они намного проще в настройке.
Есть различия и внутри каждой серии. Для большинства модулей выпускаются модификации Pro и поп-Рго, они полностью совместимы, но габариты, стоимость, энергопотребление и дальность действия первых больше (примерно 1 миля для Pro и 300 футов для поп-Рго1). Я рекомендую начинать с более дешевой версии модулей и переходить на Pro, если вам понадобится большая дальность.
Кроме того, существуют модули XBee для частот 2,4 ГГц и 900 МГц. Частота 900 МГц дает выигрыш в дальности действия, такой сигнал лучше проникает сквозь строительные конструкции, но она разрешена не во всех странах. Модули, работающие на частотах 2,4 ГГц и 900 МГц, несовместимы друг с другом.
1 1 миля (США)« 1,6 км; 1 фут « 30 см. — Ред.
Рис. 11.2. Модули XBee Series 1
Гпава 11. Беспроводная связь с помощью радиомодулей XBee
225
Наконец, XBee-модули поставляются с различными вариантами исполнения антенны: встроенными, внешними, с отдельными антенными модулями и внешними переходниками для антенн. Как правило, внешняя антенна обеспечивает лучшие характеристики, но занимает больше места.
В наших примерах будем использовать модули XBee серии 1, поп-Рго, с рабочей частотой 2,4 ГГц со встроенной антенной, работающие в режиме последовательного обмена данными. Маркировка и назначение контактов модуля приведены на рис. 11.3.
XBee®/XBee-PRO® RF Module Pin Numbers
(top sides shown - shields on bottom)
Pin Assignments for the XBee and XBee-PRO Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)
am
Name
| ' Direction
1 VCC - Power supply
2 DOUT Output UART Data Out
3 DIN /CONFIG Input UART Data In
4 D08* Output Digital Output 8
5 RESET Input Module Reset (reset pulse must be at least 200 ns)
6 PWMO/RSSI Output PWM Output 0/RX Signal Strength Indicator
7 PWM1 Output PWM Output 1
8 [reserved] Do not connect
9 DTR / SLEEP_RQ / DI8 Input Pin Sleep Control Line or Digital Input 8
10 GND 2 Ground
11 AD4/DI04 Either Analog Input 4 or Digital I/O 4
12 CTS / DI07 Either Clear-to-Send Flow Control or Digital I/O 7
13 ON /SLEEP Output Module Status Indicator
14 VREF Input Voltage Reference for A/D Inputs
15 Associate / AD5 / DIOS Either Associated Indicator, Analog Input 5 or Digital I/O 5
16 „ RTS/AD6/DI06 Either Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17 AD3/DI03 Either Analog Input 3 or Digital I/O 3
18 AD2/DI02 Either Analog Input 2 or Digital I/O 2
19 AD1/DI01 Either Analog Input 1 or Digital I/01
20 ADO/DIOO Either Analog Input 0 or Digital I/O 0
Рис. 11.3. Фрагмент технического описания модуля XBee Series 1: маркировка и назначение контактов
ВНИМАНИЕ!
Для питания модуля XBee необходимо напряжение 3,3 В, при подаче питания 5 В модуль выйдет из строя.
226
Часть III. Интерфейсы передачи данных
11.1.2. Платы расширения для XBee
Существуют специальные переходники — платы расширения (рис. 11.4), позволяющие легко

подключить модуль XBee к плате ***Arduino***. Есть несколько разновидностей XBee-переходников, обладающих одинаковыми функциями и незначительно отличающихся друг от друга.
Рассмотрим функции плат расширения XBee.
Arduino Wireless Shield Sparkfun Xbee ShieldCooking Hacks XBee Shield
Рис. 11.4. Внешний вид различных плат расширения для подключения модулей XBee
Стабилизатор 3,3 В
Большинство плат ***Arduino*** работает от источника 5 В, логические уровни также находятся в диапазоне от 0 (низкий уровень) до 5 В (высокий уровень). Напряжение питания модулей XBee равно 3,3 В, логические уровни тоже другие. Хотя у ***Arduino*** есть встроенный стабилизатор на 3,3 В, его ток недостаточен для питания XBee-модуля. Поэтому на большинстве XBee-переходников установлен линейный стабилизатор для питания модуля XBee.
Согласование логических уровней
Для реализации обмена UART соединяют контакты Tx и Rx платы ***Arduino*** и модуля XBee, однако при этом необходимо учитывать разный логический уровень для ***Arduino*** и XBee. Сигналы нужно привести к одному уровню (согласовать логические уровни). В различных моделях плат расширения это реализовано по-разному.
Светодиодные индикаторы
На большинстве плат расширения установлены два светодиодных индикатора:
- Associate — мигает, когда модуль работает в режиме последовательного обмена данными;
- RSSI — загорается на короткое время при получении данных.

Перемычка или переключатель выбора UART
Модули XBee общаются с ***Arduino*** через последовательный универсальный асинхронный приемопередатчик (UART) по контактам Rx и Tx. На многих платах ***Arduino*** (кроме Mega и Due) есть один доступный UART, который к тому же осуществляет USB-соединение с компьютером для программирования и отладки. На плате Leonardo тоже один UART (контакты Rx и Tx), но они не дуплексированы, т. к. программный интерфейс USB непосредственно встроен в микроконтроллер. В случае платы ***Arduino Uno*** возникает вопрос: как модуль XBee и компьютер подключить к одному разъему UART. Схема соединения контактов Rx и Tx (при подключении модуля XBee через плату расширения) приведена на рис. 11.5.
Модуль XBee
Рис. 11.5. Возможность возникновения коллизий в линиях UART
Обратите внимание на надпись "Коллизии" на рис. 11.5. Подумайте, что произойдет, если модуль XBee и ваш компьютер будут одновременно передавать данные на плату ***Arduino***. Как ***Arduino*** узнать, откуда приходят данные? Если данные будут передаваться одновременно, произойдет так называемая коллизия, и информация исказится. Значит, плата ***Arduino*** не сможет одновременно общаться с компьютером и модулем XBee по последовательному порту. Решить эту проблему можно двумя способами:
- отсоединять переходник XBee при программировании платы ***Arduino***;
- установить на переходнике XBee перемычку или переключатель для подключения к плате ***Arduino***.

Теперь при необходимости запрограммировать плату ***Arduino*** нужно либо отсоединить переходник XBee, либо установить в требуемое положение переключатель (перемычку).
Программная или аппаратная реализация UART
Для соединения платы ***Arduino*** с модулями XBee служит аппаратный UART-интерфейс (контакты 0 и 1 на плате ***Arduino***). Эти выводы также используются для

подключения к компьютеру по USB. Большинство переходников также осуществляет соединение между XBee и платой ***Arduino*** через аппаратный последовательный порт. Можно не отсоединять переходник XBee при программировании платы ***Arduino***, если воспользоваться библиотекой SoftwareSerial. Эта библиотека позволяет определить два произвольных цифровых контакта платы ***Arduino*** в качестве выводов Rx и Tx при соединении с переходником XBee. Но на переходнике XBee обязательно должен быть переключатель для выбора контактов платы ***Arduino*** в качестве Rx и Tx. На

переходнике Sparkfun XBee установлен такой переключатель, коммутирующий контакты 2 и 3 платы ***Arduino*** в качестве линий Rx и Tx. Если на вашем переходнике есть такой переключатель, можно организовать взаимодействие с радиомодулями XBee с помощью библиотеки SoftwareSerial.

11.2. Настройка модулей XBee

Прежде чем использовать модули XBee, их необходимо настроить для обмена информацией друг с другом. На заводе-изготовителе модули XBee по умолчанию настраивают на определенный канал и режим работы. При включении они сразу могут принимать или отправлять данные на другой совместимый модуль XBee. Иногда требуется внести изменения в стандартные настройки и далее вы узнаете, как это сделать.

11.2.1. Настройка с помощью USB-адаптера

Модули XBee можно запрограммировать так же, как и плату ***Arduino***, через интерфейс USB. Существуют два способа: с помощью платы ***Arduino*** и преобразователя USB (см. главу 6) или через специальный адаптер XBee USB. Для программирования модулей XBee я настоятельно рекомендую приобрести адаптер XBee USB, например, популярный SparkFun XBee USB Explorer (рис. 11.6).

Рис. 11.6. Адаптер SparkFun XBee USB Explorer

Первый вариант программирования (не рекомендуется)

Я не рекомендую программировать модули XBee с помощью платы ***Arduino Uno***, т. к. при неосторожности можно повредить плату. Если вы все же хотите запрограммировать модули XBee с использованием платы ***Arduino***, то возникнет проблема коллизии данных, которую мы упоминали ранее. Чтобы предотвратить коллизии, следует удалить с платы микросхему процессора ATmega, присоединить модуль к плате через переходник XBee и подключить ***Arduino*** к компьютеру USB-кабелем. После этого все команды с компьютера будут приходить на XBee.

ПРИМЕЧАНИЕ

Удалить можно только микросхему, вставленную в панельку. Изъять процессор с платы ***Uno*** SMD или другой с припаянным чипом ATmega невозможно.

Второй вариант программирования (рекомендуется)

Использовать SparkFun USB XBee Explorer очень просто: вставьте XBee-модуль в гнездо на адаптере, подключите SparkFun USB XBee Explorer к компьютеру с помощью USB-кабеля, и можно программировать. Адаптер SparkFun USE! XBee Explorer реализует USB-интерфейс с помощью тех же контроллеров FTD1, что и последние платы ***Arduino***. Далее в этой главе применим адаптер для установки беспроводной связи между компьютером и ***Arduino*** с подключенным модулем XBee.

11.2.2. Настройка модуля XBee и его подключение к компьютеру

Параметров настройки модулей XBee очень много и на описание всех понадобится отдельная книга. Здесь мы рассмотрим наиболее важные (рис. 11.7):

- ID (идентификатор PAN-сети) — все модули XBee, которые будут обмениваться данными друг с другом, должны быть отнесены к одной сети.
- MY (собственный адрес) — уникальный адрес идентификации каждого модуля XBee в пределах определенной персональной сети.
- DL (адрес назначения) — уникальный адрес модуля XBee, с которым вы хотите обмениваться данными.

Рис. 11.7. Параметры настройки сети XBee с конфигурацией "точка-точка"

- BD (скорость передачи данных) — скорость обмена данными для модулей XBee. Мы будем использовать значение по умолчанию (9600 бод).

Обратите внимание, что значения параметров MY и DL для каждого модуля XBee меняются местами, т. к. собственный адрес одного модуля является адресом назначения для другого, и нА0борот. Идентификатор сети PAN в наших примерах равен 1234, но можно выбрать другое четырехзначное шестнадцатеричное число. По умолчанию параметр BD равен 3. Значения скорости передачи в бодах связаны со значением параметра BD следующим образом:

- 0— 1200 бод;
- 1 — 2400 бод;
- 2 —4800 бод;

- 3 — 9600 бод (по умолчанию);
- 4—19 200 бод;
- 5 —38 400 бод;
- 6 —57 600 бод;
- 7— 115 200 бод.

Подключите модуль XBee к компьютеру с помощью одного из двух методов, описанных ранее. Далее следует определить последовательный порт, к которому подключен модуль. Способ определения порта был описан в главе 1. Запомните, к какому последовательному порту подключен модуль.

11.2.3. Настройка XBee с помощью Windows-приложения X-CTU

Теперь нужно запрограммировать модули XBee с параметрами, указанными на рис. 11.7. Если вы работаете в операционной системе Windows, то можете воспользоваться приложением X-CTU, предоставляющим удобный графический интерфейс. Если у вас нет компьютера с операционной системой Windows, сразу перейдите к следующему разделу, где описана настройка модулей XBee с помощью последовательного терминала в Linux или OS X.

Отыскать ссылку для скачивания самой последней версии X-CTU с сайта Digi можно через поиск Google. Ссылку на скачивание X-CTU вы можете найти также на странице http://www.expIoringarduino.com/content/chll. Выполните следующие действия:

1. Загрузите программу установки, установите X-CTU и запустите приложение. В результате появится окно, показанное на рис. 11.8. Слева будет выведен список доступных COM-портов.
2. Выберите COM-порт, к которому подключена ваш адаптер XBee Explorer и нажмите кнопку Test/Query (см. рис. 11.8). Если программируете новый модуль XBee, по умолчанию настроенный на скорость 9600 бод, должно появиться окно подтверждения текущей информации о конфигурации, считанной с модуля (рис. 11.9).

Гпава 11. Беспроводная связь с помощью радиомодулей XBee

231

Рис. 11.8. Главное окно программы X-CTU

Рис. 11.9. Окно X-CTU — подтверждение запроса

232

Часть III. Интерфейсы передачи данных

3. Перейдите к экрану настройки модема и нажмите кнопку Read, чтобы отобразить все доступные параметры настройки на вашем XBee-модуле. Результат должен выглядеть примерно так, как показано на рис. 11.10.

Рис. 11.10. Окно X-CTU — конфигурация модема

4. Теперь установите значения PAN ID, адрес источника, адреса отправителя и получателя. Можно задать множество других опций конфигурации, но мы в этой книге ограничиваемся только четырьмя перечисленными параметрами. Чтобы изменить значение настройки, просто щелкните по ней мышью. Установите следующие значения: ID —1234; DL — 1001; MY — 1000.
5. Нажмите кнопку Write в верхней части окна, чтобы записать эти значения в XBee-модуль. Измененные параметры будут выделены синим цветом (рис. 11.11).

Вы настроили свой первый модуль XBee! Теперь аккуратно извлеките один модуль из адаптера XBee Explorer и установите другой. Выполните описанные действия со вторым модулем XBee, поменяв значения параметров DL и MY, чтобы модули могли обмениваться данными между собой. На рис. 11.12 показана конфигурация для второго XBee-модуля.

Гпава 11. Беспроводная связь с помощью радиомодулей XBee

233

Рис. 11.11. Запись настроек в модуль XBee

Modem Parameter Profile Remote Configuration... Versions. . PC Settings j Range Test j Terminal Modem Configuration j Modem Parameter and Firmware Re8d I Write i Restore

Г Always Update Firmware

Parameter View Profile Versions

Clear Screen j j Save I Download new

Show Defaults Load j versions...

Modem. XBEE_____ Function Set

[XB24 3 (XBEE 802.15 4

8 d Networking & Security

Й (C)CH-Channel j~ S3 (1234) ID-PAN ID
0 (3) OH - Destination Address HIGH б (1000) DL- Destination Address Low a (1001)MY-16-bitSourceAddress a (13A200) SH - Serial Number HIGH a (4064E31D) SL- Serial Number Low & (0) MM-MAC Mode в (0) RR-XBee Retries a f0) RN* Random Delay Slots a (19) NT-Node Discover Time B (0) NO - Node Discover Options & (0) CE - Coordinator Enable a (i FFE) SC - Scan Channels j-~ a (4) SD - Scan Duration
;..6 (0) A1 - End Device Association
j-- a (0)A2-Coordinator Association a (0) At -Association Indication в (0) EE -AES Encryption Enable j - a KY-AES Encryption Key
ting settings
> I
Version____
'3 flOED 3
Рис. 11.12. Настройки для второго модуля XBee

Ваши модули XBee настроены и готовы для связи друг с другом. Назначив им идентификатор PAN ID, отличный от значения по умолчанию, вы уменьшаете вероятность помех другим сетям XBee. Теперь можно сразу перейти к разделу 11.3.

11.2.4. Настройка модулей XBee из последовательного терминала

Если у вас нет компьютера с операционной системой Windows, придется настраивать модули XBee через последовательный терминал. Этот процесс одинаков для Linux и Mae. Запустим системное приложение screen. Как и в главе 1, с помощью IDE ***Arduino*** выясним имя порта подключения для нашего адаптера XBee Explorer. Посмотреть имя можно в меню Сервис -> Последовательный порт.
Затем открываем терминал и выполняем следующие действия:

1. В терминале вводим команду, например

screen /dev/ttyUSB6 9600

(/dev/ttyusB6 замените на имя вашего порта подключения). При нажатии клавиши <Enter> инициируется соединение с адаптером и экран гаснет. После подключения при вводе команд они не будут отображаться на экране. Сначала я объясню процесс программирования, а потом приведу список команд. Чтобы запрограммировать радиомодуль XBee, необходимо выполнить следующие действия:

- установить модуль XBee в режим программирования;
- назначить ID PAN (ATID);
- задать адрес источника (ATMY);
- задать адрес получателя (ATDL);
- записать настройки в энергонезависимую память модуля XBee (ATWR).

В режиме программирования при долгой паузе (несколько секунд) возникает тайм-аут, поэтому постарайтесь вводить команды быстро. Помните, что при вводе команды они не видны на экране.

2. Введите команду +++ и ждите, не нажимайте <Enter>, пока терминал не ответит ок, это указывает, что модуль XBee вошел в режим программирования.
3. Введите команду atidi234 и нажмите <Enter>, значение идентификатора сети будет равно 1234.
4. Введите команду atmyiooo и нажмите <Enter>, значение адреса источника будет равно 1000.
5. Введите команду atdliooi и нажмите <Enter>, адрес назначения будет установлен 1001.
6. Введите команду atwr и нажмите <Enter>, это сохранит настройки в энергонезависимой памяти, содержимое которой не удаляется при отключении питания от модуля XBee.

7. Чтобы убедиться, что все настроено верно, введите одну из команд at id, atmy или atdl без значений и нажмите клавишу <Enter>, при этом на дисплей будет выведено текущее значение параметра.

ПРИМЕЧАНИЕ

Если вы вышли из режима программирования по тайм-ауту, можно повторно набрать +++ и продолжить с того места, где остановились.

После завершения всех предыдущих шагов, установите в адаптер другой модуль XBee. Проделайте шаги 2-7, но поменяйте значения для команд atmy и atdl так, чтобы модули XBee были настроены на

обмен данными друг с другом.
Теперь ваши модули XBee настроены и готовы к общению друг с другом! Если у вас возникли проблемы с настройкой, посмотрите видеоурок, упомянутый в начале этой главы, он продемонстрирует все этапы настройки.

11.3. Соединяемся с компьютером по беспроводной сети

Настроив модули XBee, пора начать их использование. Самое простое — организовать беспроводной обмен между компьютером и ***Arduino***. Запрограммировать плату ***Arduino*** непосредственно через соединение XBee невозможно, поэтому загружать и тестировать программы будем с помощью USB-интерфейса. А после загрузки готовой программы на ***Arduino*** USB-кабель можно отключить и общаться с ***Arduino*** по беспроводной связи XBee.

11.3.1. Автономное питание платы *Arduino*

Поскольку плата ***Arduino*** не будет подключена к компьютеру через USB, необходимо предусмотреть внешнее питание. Рассмотрим несколько вариантов решения этой проблемы.

Питание от USB с компьютера или сетевого адаптера

Если плата ***Arduino*** подключена через USB-кабель к компьютеру, то теряется смысл беспроводного соединения. Можно подключить плату к одному USB-порту компьютера, а модуль XBee будет общаться с платой USB XBee Explorer, подключенной к другому USB-порту компьютера. Способ подойдет для тестирования беспроводной связи, но бесполезен с точки зрения практического применения. В данном варианте необходимо убедиться, что правильно выбран последовательный порт для просмотра данных адаптера USB XBee Explorer в мониторе последовательного порта или в приложении Processing.

Можно использовать 5-вольтовый сетевой блок питания с USB-выходом. При этом отпадает привязка к компьютеру. Подобные адаптеры широко применяются для зарядки iPhone, Android-устройств, а также других планшетов и смартфонов. На рис. 11.13 показан стандартный сетевой USB-адаптер для американских розеток.

Рис. 11.13. Сетевой 5-вольтовый USB-адаптер

Питание от батареи

Можно питать плату ***Arduino*** от батареи, подключив ее к разъему питания или входу Vjn. С этих входов напряжение поступает на встроенный стабилизатор, который выдает напряжение 5 В для питания микропроцессора и других компонентов. На рис. 11.14 показана 9-вольтовая батарея со встроенным выключателем и разъемом питания.

Вместо 9-вольтовой батареи можно использовать несколько батареек AA с напряжением 1,5 В. Если соединить последовательно четыре таких батарейки, получится напряжение около 6 В. Минимальное падение напряжения на встроенном в ***Arduino*** стабилизаторе равно 1 В. При входном напряжении 5,5 В на шине питания будет

Рис. 11.14. Батарея 9 В

4,5 В. Можно рассчитывать, что плата будет работать и при напряжении 4,5 В, что приемлемо для платы на базе ATMega, хотя и ниже напряжения при питании от USB.

Сетевые источники питания

Еще один вариант питания для удаленной платы ***Arduino*** — стандартный сетевой адаптер. Он подключается к обычной розетке и имеет на другом конце разъем для соединения с ***Arduino***. При выборе подобного сетевого блока питания необходимо обратить внимание на три важных характеристики:

- размеры разъема (диаметр 2,1 мм с плюсовым центральным контактом (рис. 11.15));
- напряжение питания (желательно 7-12 В);
- максимальный ток (чем выше ток, тем больше устройств можно подключить к плате ***Arduino***).

Блок с током 1 А довольно распространен и обеспечивает более чем достаточную мощность для платы ***Arduino*** и некоторых дополнительных компонентов.

Теперь перейдем, наконец, к практическому примеру использования беспроводной связи. Так как модуль XBee настроен в режим простого последовательного обмена, можно начать с проектов из

главы 6. Необходимо выполнить следующие действия:

1. Загрузите программу, которая позволяет изменить цвет окна обработки с помощью потенциометра, подключенного к ***Arduino***. Сделайте это до установки переходника XBee на плату ***Arduino***, чтобы избежать коллизий UART, которые обсуждались ранее в этой главе. Если на переходнике есть перемычка или переключатель для выбора режима подключения/отключения XBee к UART, то сам переходник при программировании платы ***Arduino*** можно не извлекать. Уточните это в документации на ваш переходник XBee. Код программы, читающий данные потенциометра и передающий их на компьютер, приведен в листинге 11.1.

Листинг 11.1. Программа ***Arduino*** для отправки данных на компьютер — pot_to_processing/arduino_read_pot

```
// Отправка данных потенциометра на компьютер
const int POT=0; // Аналоговый вход для подключения потенциометра int val;       // Переменная для хранения данных потенциометра
```

Рис. 11.15. Разъем сетевого адаптера с плюсовым центральным контактом

11.3.2. Пример 1: беспроводное управление цветом окна на компьютере


```
void setup()
{
Serial.begin(9600); // Запуск последовательного порта
}
void loop ()
{
val = map(analogRead(POT), 0, 1023,
Serial.printIn(val); delay(50);
, 255); // Чтение и
// масштабирование данных // Отправка данных // Задержка
```

2. Отключите плату ***Arduino*** от компьютера и подсоедините к ней переходник вместе с модулем XBee. Подключите потенциометр к аналоговому входу 0, как показано на электрической схеме, изображенной на рис. 11.16.
3. Для питания платы ***Arduino*** используйте один из методов, описанных в предыдущем разделе. Я выбрал сетевой USB-адаптер питания.
4. Подключите XBee USB Explorer с другим запрограммированным модулем XBee к компьютеру с помощью кабеля USB. Если модули настроены правильно, Rx светодиод на переходнике USB XBee Explorer будет быстро мигать в момент получения данных.
5. Перед использованием приходящих данных в Processing-приложении откройте монитор последовательного порта в ***Arduino*** IDE. Выберите последовательный порт, к которому подключен ваш переходник USB XBee Explorer, и убедитесь, что данные поступают в компьютер (рис. 11.17).
6. После того как вы убедитесь, что данные поступают, закройте монитор последовательного порта и запустите программу на Processing для регулировки цвета окна. Проверьте, что последовательный порт выбран правильно. Текст программы на Processing приведен в листинге 11.2.

Листинг 11.2. Программа на Processing для чтения последовательных данных и установки цвета экрана — pot_to_processing/processing_display_color

```
// Программа на Processing для чтения переменной и изменения цвета экрана // Подключение и инициализация библиотеки serial import processing.serial.*;
Serial port;
float brightness =0; // Переменная для хранения значения потенциометра
void setup()
{
size(500,500); // Размер окна
port = new Serial(this, "COM3", 9600); // Инициализация
// последовательного порта
```



```
port.bufferUnt.il (' \п');          // Символ конца строки
```

```
}
void draw ()
{
background(O,0,brightness); // Перерисовать окно
}
void serialEvent (Serial port)
{
brightness = float(port.readStringUntil('\n')); // Получить переменную
}
```

Рис. 11.16. Схема соединения платой ***Arduino*** с потенциометром и переходником XBee

240

Часть III. Интерфейсы передачи данных

ffi COM 16

Рис. 11.17. Данные, переданные по беспроводному каналу в монитор последовательного порта

При запуске программы она должна работать так же, как это было при подсоединении платы ***Arduino*** непосредственно к компьютеру. Теперь вы можете ходить вокруг дома и офиса (если питаете ***Arduino*** от батареи) и менять цвет на экране компьютера.

11.3.3. Пример 2: управление RGB-светодиодом

В предыдущем примере мы убедились, что возможна беспроводная передача данных от ***Arduino*** к компьютеру. Теперь используем код из главы 6 для управления RGB-светодиодом, чтобы проверить, что можно отправлять данные без проводов от компьютера к плате ***Arduino***. Удостоверившись в успешном обмене данными между компьютером и ***Arduino*** по беспроводному соединению, вы сможете создавать множество интересных приложений (некоторые есть на веб-странице для этой главы).

Сначала загрузим программу (листинг 11.3) на плату ***Arduino***. Подобный пример был рассмотрен в главе 6. Принимаем строку значений RGB и устанавливаем цвет RGB -светодиода.

Листинг 11.3. Управление RGB-светодиодом через последовательный порт — processing_control_RGB/list_control

```
// Отправка многоразрядных значений // Константы выводов RGB-светодиода const int RED =11; const int GREEN =10; const int BLUE =9;
// Переменные значений выводов RGB int rval = 0; int gval = 0; int bval = 0;
```

Гпава 11. Беспроводная связь с помощью радиомодулей XBee

241

```
void setup()
{
Serial.begin(9600);    // Инициализация последовательного
// порта на скорости 9600 // Установить выводы на выход OUT
pinMode(RED, OUTPUT); pinMode(GREEN, OUTPUT); pinMode(BLUE, OUTPUT);
)
void loop()
{
// Пока в буфере есть данные while (Serial.available() > 0) {
rval = Serial.parselnt(); gval = Serial.parselnt(); bval = Serial.parselnt(); if (Serial.read() — '\n') {
// Первое число // Второе число // Третье число // Конец передачи
// Установить яркость R,G,B светодиода analogWrite(RED, rval); analogWrite(GREEN, gval); analogWrite(BLUE, bval);
)
}
```

Далее соединяем элементы, как вы делали в главе 6 (рис. 11.18). К плате ***Arduino*** присоединяем переходник XBee и модуль XBee.

Как и в предыдущем разделе, подключаем адаптер USB XBee Explorer к компьютеру и запускаем программу на Processing (листинг 11.4). Убедитесь, что файл hsv.jpg находится в папке data в каталоге программы. И не забудьте установить правильное имя последовательного порта.

Листинг 11.4. Программа на Processing для установки цвета RGB-светодиода — processing_control_rgb/ processing_control_RGB

```
import processing.serial.*;     // Подключение библиотеки serial
PImage img;
Serial port;
void setup()
{
size (640,256);
img = loadImageChsv.jpg");  // Загрузка фоновой картинки
port = new Serial(this, "COM9", 9600); // Открыть последовательный порт
```
242
Часть III. Интерфейсы передачи данных
```
void draw ()
{
background(0);          // Установка цвета фона
image(img,0,0);         // Картинка
}
void mousePressedO
{
color c = get(mouseX, mouseY); // Получить RGB-цвет по позиции курсора мыши String colors = int(red(c))+","tint(green(c))+","tint(blue(c))+"\n";
// Преобразовать в строку
print(colors);  // Вывод для отладки
port.write(colors);      // Отправка переменной в Arduino
ri<u.o321098  7 6 5 4
bi Й « б 1 1 1 1 DIGITAL
3 2 10

™ Arduino (UNO.) am ф
r-t
■

www.arduino.ee
c POWER      ANALOG IN 1
6 5V Gnd Vin 0 1 2 3 4 5
```
Рис. 11.18. Соединение ***Arduino*** и RGB-светодиода
Глава 11. Беспроводная связь с помощью радиомодулей XBee
243
При запуске программы появится экран выбора цвета как в главе 6. Выберите цвет. Данные будут переданы по беспроводной связи на плату ***Arduino***, и подключенный к ней светодиод должен светить выбранным цветом. Передаваемые значения можно контролировать в мониторе последовательного порта.
Теперь ясно, что с помощью модулей XBee можно обмениваться данными с компьютером в двух направлениях. В следующем разделе мы рассмотрим беспроводной обмен между двумя платами ***Arduino*** напрямую.
11.4. Беспроводной дверной звонок
Беспроводная связь между двумя платами ***Arduino*** имеет много практических применений. На основе нескольких плат ***Arduino*** можно построить сеть датчиков, передавать команды для радиоуправляемого автомобиля или осуществлять удаленный мониторинг. В этом примере с помощью двух плат ***Arduino***, снабженных модулями XBee, создадим беспроводной звонок для дома, квартиры или офиса. Плата ***Arduino***, встроенная в дверь, будет реагировать на нажатие наружной кнопки звонка. Когда кто-нибудь позвонит в дверь, другая плата ***Arduino*** включит световой или звуковой сигнал, чтобы сообщить, что к вам пришел посетитель.
ПРИМЕЧАНИЕ
Возможно, прежде чем создавать проект, вы захотите посмотреть видеоклип, демонстрирующий действие системы, доступный на странице http://www.exploringarduino.com/content/ch11.
11.4.1. Разработка системы

Система будет состоять из двух плат ***Arduino***. На каждой плате дополнительно установлен переходник с модулем XBee. Одну плату ***Arduino*** можно разместить за пределами дома или квартиры, рядом с кнопкой. Другая плата может находиться в любом месте внутри помещения, чтобы уведомить вас, что кто-то звонит в дверь. Дальность действия зависит от типа XBee-модулей, от количества стен между модулями и других факторов окружающей среды.

Просто подать звук — это скучно, наша плата ***Arduino*** будет мигать разноцветными огнями и издавать разнообразные звуки, чтобы привлечь внимание. Вы можете придумать собственные звуковые эффекты. Описанная в данном примере система будет кнопочной, но можно заменить кнопку инфракрасным датчиком, фотодатчиком или датчиком присутствия, которые автоматически определят, что к двери кто-то приближается.

Общая структура системы показана на рис. 11.19. Сверяясь с этим рисунком, будет проще разработать каждый из блоков.

Приемник MY= 1000 DL =1001

Передатчик MY= 1001 DL= 1000

Рис. 11.19. Структура системы беспроводного звонка

11.4.2. Оборудование для передатчика

Начнем с передатчика. Вам нужно подключить кнопку с резистором к цифровому входу платы ***Arduino*** с установленным на ней переходником с модулем XBee (рис. 11.20).

Тип платы ***Arduino*** не имеет принципиального значения, но важно отметить, что последовательное соединение на плате Leonardo будет работать по-другому, нежели на ***Uno***. На платах ***Uno*** и Micro один и тот же процессор управляет последовательным обменом и выполняет программу, а на Leonardo и Mega для этих целей

Рис. 11.20. Схема подключения передатчика беспроводного дверного замка

предусмотрены отдельные процессоры. Чтобы продемонстрировать эти различия, я выбрал для передатчика плату Leonardo. Схема для любой платы одна и та же. Программные различия рассмотрим далее.

Поскольку передатчик будет удален от компьютера, выберите один из описанных ранее вариантов автономного питания платы. В демонстрационном видеоуроке я питал плату ***Arduino*** от 9-вольтовой батареи. Возможно, вы захотите использовать сетевой адаптер.

СОВЕТ

Чтобы посетителям было удобно, рекомендую взять большую кнопку звонка, протянув провода через стену к плате ***Arduino***.

11.4.3. Оборудование для приемника

Далее соберем приемник, который будет оповещать нас о нажатии кнопки на передатчике. Он тоже состоит из платы ***Arduino*** с переходником и модулем XBee, RGB-светодиода, резисторов и небольшого пьезоизлучателя. Соберите схему по рис. 11.21. Обратите внимание, что в программе будет задействован только красный и зеленый цвет, поэтому подключение резистора к выводу B (синий) RGB-светодиода не требуется. Последовательно с пьезоэлементом можно установить потенциометр для управления громкостью звукового сигнала.

Рис. 11.21. Схема подключения приемника беспроводного дверного замка

Теперь необходимо выбрать тип платы ***Arduino*** и способ питания приемника. Я использовал плату ***Uno***, подключенную к сетевому USB-адаптеру. Подойдет также батарея или USB-кабель, соединенный с компьютером. Функциональность приемника можно расширить, добавив светодиоды или компьютерное Processing-приложение.

11.4.4. Программа для передатчика

Оборудование настроено, теперь необходимо написать программы для приемника и передатчика. Для реализации этой схемы связи есть много вариантов, далее будет описан только один из них.

Передатчик отправляет данные каждые 50 мс. Передается значение о, если кнопка отпущена, и 1, если нажата. Проверку на дребезг не проводим. Пьезоизлучатель будет издавать сигнал все время, пока

кнопка нажата.
Код программы зависит от того, какую плату ***Arduino*** вы выбрали. Как уже упоминалось, в случае ***Arduino Uno*** контакты Rx/Tx (0/1) выполняют функции как UART, так и USB. При программировании ***Uno*** или Mega необходимо удалить переходник XBee или задать на нем требуемое положение перемычек (переключателей).
При программировании платы Leonardo (или другой платы со встроенным USB-интерфейсом) отсоединять переходник XBee не нужно.
Код листинга 11.5 написан для ***Arduino*** Leonardo, если у вас плата ***Uno***, то замените В коде Seriall на Serial.
Листинг 11.5. Код передатчика для беспроводного дверного замка — doorbell/transmitting_arduino

```
// Код передатчика Arduino для беспроводного дверного замка const int BUTTON =12; // Вывод кнопки к контакту 12
void setup()
{
// Для платы Leonardo выводы Rx/Tx //не мультиплексированы с USB г // Код для Leonardo (Seriall = RX/TX)
// Для UNO измените Seriall на Serial Seriall.begin(9600) ;
}
void loop()
{
Seriall.println(digitalRead(BUTTON)); // Отправка статуса кнопки delay(50); // Небольшая задержка
}
```

В функции setup о последовательный порт подключается к модулю XBee и начинает работать со скоростью 9600 бод. Каждые 50 мс происходит опрос цифрового
Глава 11. Беспроводная связь с помощью радиомодулей XBee
247
входа и значение отправляется по беспроводному каналу. Команду digitalReadO можно разместить непосредственно внутри функции printino, поскольку выходное значение в другом месте в программе не используется.
11.4.5. Программа для приемника
Программа для приемника сложнее, чем для передатчика. Текст программы приемника, приведенный в листинге 11.6, написан для платы ***Arduino Uno*** U. В случае платы ***Arduino*** Leonardo замените в коде Serial на Seriaii.
В приемнике программа должна опрашивать последовательный порт для получения приходящих с передатчика данных о статусе кнопки, определять, нажата или отжата кнопка, выдавать световой и звуковой сигналы, при этом одновременно ожидая данные, приходящие в последовательный порт. Это усложняет программу, т. к. мы не можем использовать функцию delay о. При вызове функции delay о программа останавливается, пока не закончится задержка, что может привести к следующей проблеме: реакция приемника на сигнал передатчика не будет мгновенной и буфер может переполниться, т. к. передатчик будет посылать данные быстрее, чем их сможет обработать приемник.
Нам необходимо попеременно переключать красный и зеленый цвета RGB-светодиода и изменять частоту сигнала для пьезоэлемента. Реализовать задержку с помощью функции delay о мы не можем по причинам, указанным ранее. Вместо delay о применим функцию minis о, возвращающую количество миллисекунд с начала выполнения программы. Состояние светодиодов и частоту звука будем менять каждые 100 мс. Сохраняем текущее время и постоянно считываем значение minis о, пока оно не превысит предыдущее значение на ЮОмс. Когда это произойдет, меняем цвет светодиода и частоту звука. Кроме того, в цикле loop о непрерывно читаем значение статуса клавиши из последовательного порта и управляем светом и звуком.
В функции setup о инициализируем подключение по последовательному порту. Чтобы упростить работу программы, сохраняется текущее значение цвета светодиода, частоты звука и предыдущее значение, выдаваемое функцией minis о.
Полный текст программы для приемника приведен в листинге 11.6. Загрузите программу на плату ***Arduino***, не забыв перед этим переключить перемычки на переходнике XBee (или отсоединить его).
Листинг 11.6. Программа для приемника беспроводного дверного замка — doorbe!l/receiving_arduino

```
// Код приемника беспроводного дверного замка
const int RED =11; // Выход 11 - красный контакт RGB-светодиода const int GREEN =10; // Выход 10 - зеленый контакт RGB-светодиода const int SPEAKER =8; // Выход 8 подключения пьезоизлучателя char data; int onLED = GREEN;
```

```
int offLED = RED; int freq = 131;
unsigned long prev_time =0; // Таймер для переключения цвета светодиода
// и частоты звука
void setup()
{
Serial.begin(9600);
}
void loop()
{
// Для переключения звука и цвета светодиода
// прошло 100 мс?
if (millis() >= prev_time + 100)
{
// Переключение светодиода if (onLED == GREEN)
{
onLED = RED; offLED = GREEN;
}
else
(
onLED = GREEN; offLED = RED;
}
// Переключение частоты звука if (freq — 261)
{
freq = 131;
}
else
{
freq = 261;
}
// Корректировка времени для переключения // Текущее время становится предыдущим prev_time = millis ();
}
// Проверить наличие данных из последовательного порта if (Serial.available() > 0)
{
// Чтение байта данных data = Serial.read();
```

```
// Кнопка нажата - включаем звук и свет if (data == '1')
{
digitalWrite(onLED, HIGH); digitalWrite(offLED, LOW); tone(SPEAKER, freq) ;
}
// Кнопка отпущена - выключаем звук и свет else if (data == 'O')
{
digitalWrite(onLED, LOW); digitalWrite(offLED, LOW) ; noTone(SPEAKER);
}
}
}
```
Первый оператор `if()` в основном цикле программы loop о проверяет время, прошедшее с последнего момента установки переменной prev time. Если прошло более 100 мс, то значения переменных

текущего состояния цвета светодиода и частоты звука меняются, в результате получается чередование сигналов.

Второй оператор if () в цикле loop о проверяет наличие и значение входящих последовательных данных. Если приходит 0, свет и звук выключаются, если 1 — цвет и частота звука выставляются в соответствии со значениями переменных onLed, off Led, freq.

ПРИМЕЧАНИЕ

Посмотреть видеоурок, демонстрирующий работу беспроводного звонка, можно на странице http://www.exploringarduino.com/content/ch11. Этот видеофайл доступен также на сайте издательства Wiley.

Резюме_____

В этой главе вы узнали следующее:

- Что выпускается множество разновидностей модулей XBee.
- Что для использования радиомодулей XBee с большинством плат ***Arduino*** необходимо конвертировать логические уровни с 5 до 3,3 В.
- Как настроить модули XBee из программы X-CTU (для ОС Windows) или с помощью терминала (для операционных систем Linux и Mae).
- Как организовать автономное питание платы ***Arduino***.
- Как установить беспроводную связь между компьютером и платой ***Arduino*** с помощью модулей XBee.
- Как с помощью модулей XBee организовать беспроводную связь между двумя платами ***Arduino***.
- Как реализовать временную задержку с помощью функции minis ().

IV

Дополнительные темы и проекты

В этой части

Глава 12. Аппаратные прерывания и прерывания по таймеру Глава 13. Обмен данными с картами памяти SD Глава 14. Подключение ***Arduino*** к Интернету

ЧАСТЬ

| IV

Дополнительные темы и проекты

В этой части_____

Глава 12. Аппаратные прерывания и прерывания по таймеру Глава 13. Обмен данными с картами памяти SD Глава 14. Подключение ***Arduino*** к Интернету

ГЛАВА

12

Аппаратные прерывания и прерывания по таймеру

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino*** (рекомендуется ***Uno***);
- USB-кабель для программирования платы ***Arduino***;
- кнопка;
- пьезозуммер;
- RGB-светодиод с общим катодом;
- 1 резистор номиналом 10 кОм;
- 1 резистор номиналом 100 Ом;
- 1 резистор номиналом 150 Ом;
- 3 резистора номиналом 220 Ом;
- электролитический конденсатор ЮмкФ;
- микросхема 74HC14 (шесть инверторов с триггерами Шмитта);
- набор перемычек;
- 2 макетные платы.

Электронные ресурсы к главе

На странице http://www.exploringarduino.com/content/chl2 можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/exploringarduino в разделе Downloads.

Что вы узнаете в этой главе

Все предыдущие программы работали в синхронном режиме. В связи с этим возникали проблемы, например, выполнение команды delay () останавливает программу на некоторое время и не дает возможности ***Arduino*** осуществлять другие действия. В главе 1 мы создали программный таймер, использующий функцию minis o, что

254

Часть IV. Дополнительные темы и проекты

позволило избежать временного блокирования платы ***Arduino*** функцией delay o. Продолжим эту тему, добавив два таймера и аппаратные прерывания. Прерывания позволяют выполнять программу асинхронно, при наступлении определенного события (истечение временного интервала, изменение состояния входов и т. д.). Прерывания, как и следует из их названия, дают возможность остановить ход текущей программы ***Arduino***, выполнить код прерывания, а затем вернуться к прерванной задаче. Далее мы узнаем, как осуществить прерывания по времени и при изменении состояния контактов. На основе этих знаний мы построим систему аппаратных прерываний и напишем программу, использующую прерывания таймера.

ПРИМЕЧАНИЕ

Видеоурок по прерываниям и аппаратному устранению дребезга можно посмотреть на странице http://www.jeremyblum.com/2011/03/07/arduino-tutorial-10-interrupts-and-hardware-debouncing1. Найти данный видеофайл можно и на странице издательства Wiley.

12.1. Использование аппаратных прерываний

Аппаратные прерывания происходят при наступлении (или изменении) заданного состояния на входах-выходах. Аппаратное прерывание полезно, например, когда нужно изменить значение переменной, не проверяя непрерывно состояние кнопки. Ранее мы устраняли дребезг путем опроса состояния кнопки в цикле. Этот прием отлично работает, если время выполнения остальной части программы невелико. Но предположим, вы устраняете дребезг в цикле, выполнение которого занимает значительное время. Например, в основном цикле программы изменяется яркость

■2 * Команды начальной установки от

г

o < o >

Возникновение прерывания

>

1

Команда 1

1

Команда 2

Команда 3

1

Команда 4

i

Команда 5

Процедура обработки прерывания

Команды обработки прерывания

Рис. 12.1. Влияние внешнего прерывания на ход выполнения программы

1 На русском: Н«р:/Лу!кьатрегка.ги/видеоуроки:10-прерывания-и-аппаратная-стабилизация.

255

светодиода или скорость двигателя с помощью оператора for o с некоторой задержкой delay (). Возникает опасность пропустить нажатие кнопки, которое происходит в момент выполнения главной программы. Вот здесь и приходят на помощь прерывания. Определенные контакты на плате ***Arduino*** могут вызывать внешние аппаратные прерывания. Вы выполняете главную программу, и при возникновении внешнего прерывания запускается специальная процедура его обработки (рис. 12.1), причем прерывание может наступить в любом месте программы.

12.2. Что выбрать: опрос состояния в цикле или прерывания?

Аппаратные прерывания являются альтернативой опроса состояния входов в цикле loop о. Они не лучше и не хуже, всегда есть выбор между ними. При проектировании системы необходимо учитывать все факторы и выбрать вариант, наиболее подходящий для вашего приложения. Далее рассмотрим основные различия между опросом входов и прерываниями, чтобы понять, что лучше подойдет для конкретного проекта.

12.2.1. Программная реализация

Благодаря встроенному языку программирования ***Arduino*** программировать внешние прерывания сравнительно просто. Однако организовать опрос контактов в цикле еще проще, все, что требуется — это вызов команды digitaiRead (). Если нет безусловной необходимости в аппаратных прерываниях, то лучше их не применять, т. к. код программы усложнится.

12.2.2. Аппаратная реализация

С точки зрения аппаратной реализации между опросом контакта в цикле и прерыванием нет разницы, т. к. в обоих случаях считывается состояние входа. Тем не менее, при наличии дребезга (см. главу 2) возникает серьезная проблема: процедура обработки прерывания может быть вызвана несколько раз. Самое неприятное, что в процедуре обработки прерывания нельзя задействовать функцию программного устранения дребезга, потому что невозможен вызов функции delay(). Поэтому, если вы хотите использовать прерывание для входа с дребезгом, необходимо устранить дребезг аппаратно.

12.2.3. Многозадачность

Одна из причин использования прерываний — предоставление псевдомногозадачности. С помощью ***Arduino*** никогда нельзя обеспечить реальную многозадачность, т. к. на плате только один микроконтроллер, который за один такт может выполнить только одну команду. Однако, поскольку микроконтроллер выполняет команды очень быстро, можно прибегнуть к прерываниям, чтобы выполнять разные задачи почти одновременно. Например, с помощью прерывания можно в про-

цессе уменьшения яркости светодиодов реагировать на нажатие кнопки, которая регулирует скорость или цвет. В то время как при опросе контакта в цикле можно прочитать значение входа кнопки командой `digitalRead` () только один раз и имеющиеся в цикле loop () "медленные" операции снижают эффективность контроля входа кнопки.

12.2.4. Точность сбора данных

Для некоторых задач, где требуется быстрый сбор данных, прерывания являются необходимостью. Один из примеров — энкодер, смонтированный на двигателе постоянного тока, отправляющий в микроконтроллер импульс при определенном числе оборотов вала. Так осуществляется следящая обратная связь за скоростью вращения двигателя постоянного тока. Это позволяет динамически регулировать скорость в зависимости от нагрузки или отслеживать поворот вала двигателя. Но в такой системе исключительно важно быть уверенным, что плата ***Arduino*** получает каждый импульс. Поскольку импульсы очень короткие (намного короче импульса при нажатии кнопки), то при опросе в цикле loop о их можно пропустить. Если энкодер посылает импульс два раза за оборот вала двигателя, то из-за пропуска одного импульса измеренная частота вращения окажется вдвое меньше истинной. Поэтому в данном случае без аппаратного прерывания не обойтись.

12.2.5. Реализация аппаратного прерывания в ***Arduino***

В большинстве плат ***Arduino*** для аппаратных прерываний выделены специальные контакты. Прерывания обозначаются с помощью идентификационного номера, которому сопоставлен определенный контакт (табл. 12.1). Исключение составляет плата Due, у которой для прерываний можно задействовать любой контакт, и номер прерывания совпадает с номером контакта.

Таблица 12.1. Аппаратные прерывания на различных платах ***Arduino***

Плата	INTO	INT1	INT2	INT3	INT4	INT5
UNO, Ethernet		Pin2	Pin3	—	—	—	—
Меда2560	Pin2	Pin3	Pin21	Pin20	Pin19	Pin18
Leonardo	Pin3	Pin2	PinO	Pin1	—	—

Для вызова прерывания существует встроенная функция attachinterrupt (). Ее первый аргумент — это идентификатор прерывания (для плат из табл. 12.1) или номер контакта ***Arduino*** (для платы Due). Если на плате ***Arduino Uno*** вы хотите назначить прерывание физическому контакту 2, то первый аргумент функции attachinterrupt () будет 0. ***Arduino Uno*** (и другие платы на базе ATmega328) поддерживает всего два внешних прерывания, платы Mega и Leonardo больше (см. табл. 12.1).

Аппаратные прерывания осуществляют переход к выполнению определенных функций. Второй аргумент функции attach_interrupt () — имя выполняемой функции. Если вы хотите переключать состояние логической переменной при каждом прерывании, то функция обработки прерывания будет выглядеть так:

```
void toggleLed()
{
var = !var;
}
```

При вызове этой функции переменная var переключается в состояние, противоположное предыдущему, и происходит возврат в основную программу. Последний аргумент функции attachinterrupt () — режим запуска прерывания. Существуют режимы `LOW`, change, rising и falling (для платы Due также есть режим `HIGH`). Режимы change, rising и falling наиболее популярны, потому что прерывание выполняется только один раз при изменении состояния входа. Режимы `HIGH` и `LOW` встречаются реже, потому что будут вызывать процедуру обработки прерывания непрерывно, блокируя остальную часть программы.

12.3. Разработка и тестирование системы противодребезговой защиты для кнопки

Чтобы применить полученные знания на практике, построим схему управления RGB-светодиодом с помощью кнопки с аппаратной противодребезговой защитой. Пусть яркость одного из компонентов цвета RGB-светодиода меняется от максимума к минимуму и нА0борот. При нажатии кнопки цвет меняется на другой, в то время как при изменении яркости используется функция задержки delay O .

12.3.1. Создание схемы аппаратного устранения дребезга

Как вы узнали в главе 2, при нажатии кнопок уровень напряжения на входе многократно меняется от максимального до минимального значения. При наличии аппаратных прерываний это представляет большую проблему, потому что программа обработки прерывания будет вызвана несколько раз. К счастью, дребезг можно устранить аппаратно, получая на входе неискаженный сигнал.

Обычная кнопка подключена к цифровому выводу с помощью подтягивающего резистора. Наличие подтягивающего резистора (pull-up) дает следующий эффект: по умолчанию на входе высокий уровень, при нажатии на кнопку контакт ***Arduino*** соединяется с землей, и на вход платы поступает низкий уровень.

На рис. 12.2 изображена осциллограмма сигнала на входе при нажатии кнопки. Видно, что напряжение скачет несколько раз вверх и вниз, прежде чем устанавливается в низком состоянии.

При таком сигнале функция обработки прерывания будет вызвана три раза подряд. Чтобы предотвратить это, добавим в схему RC-цепочку. Подключим параллельно

кнопке конденсатор, а последовательно подсоединим резистор (рис. 12.3). Когда кнопка не нажата, конденсатор заряжается через резисторы R1 и R2. При нажатии на кнопку конденсатор начинает разряжаться и через некоторое время на выходе напряжение упадет до нуля. Если есть дребезг, сигнал "подпрыгивает" вверх и вниз в течение нескольких миллисекунд, но конденсатор подзаряжается через резистор и на выходе поддерживается высокий уровень напряжения. Благодаря этому уровень сигнала меняется с высокого на низкий только один раз в течение интервала времени, определяемого значениями резистора и конденсатора.

Тек JL Ш Ready M Роя 0,000$ TRIGGER
*
Type
[ЙЕЗ
Souice
«• т
Mode
[Normal!
Coupling
ш

CH1 2.00V M 500jus CHI \ 2.80V
8-Apc-1310:32 <10Hr
Кнопка с дребезгом
Рис. 12.2. Дребезг при нажатии обычной кнопки
Pin 2 (IntO)
VCC
Рис. 12.3. Схема устранения дребезга с помощью RC-цепочки
Гпава 12. Аппаратные прерывания и прерывания по таймеру
259
Резистор R2, включенный последовательно с кнопкой, уменьшает ток заряда и разряда конденсатора. Слишком большой ток может повредить кнопку. Добавление резистора 100 Ом увеличивает время разряда и обеспечивает безопасность компонентов. Но в результате спад импульса приобретает вид, изображенный на рис. 12.4.
Тек JL 63 Ready M Рок 0.000s TRIGGER
Type
Sow ce
si
Slope ЯИШЯВ
Mode
[Normal]
Coupling
m
CH1 2.00 V M 2.50ms CH1\2^0V
$~Apf-T310:38 <10Hz
Погашение дребезга с помощью RC-контура Рис. 12.4. Сигнал на выходе RC-цепочки, устраняющей дребезг
Благодаря RC-цепочке дребезг исчез, но перепад напряжения на выводе ***Arduino*** приобрел экспоненциальную форму. При опросе контакта прерывание происходит при переходе от низкого уровня к высокому и от высокого к низкому с определенной скоростью. Из-за сглаживания, вызванного конденсатором, момент прерывания определяется неточно. Крутизну спада сигнала можно увеличить с помощью триггера Шмитта. Интегральные схемы с триггером Шмитта обеспечивают резкий перепад сигнала на выходе, при превышении входным сигналом определенного порога. Выходной сигнал с триггера Шмитта можно непосредственно подавать на контакт ***Arduino***. В этой главе мы используем инвертирующий триггер Шмитта на микросхеме 74HC14. В корпусе этой микросхемы находятся шесть отдельных инвертирующих триггеров Шмитта, в наших примерах понадобится только один. Цоколевка микросхемы приведена на рис. 12.5.
Подсоединим выход RC-цепочки к входу триггера Шмитта, а сигнал с его выхода подадим на контакт платы ***Arduino*** (рис. 12.6).
Поскольку выход триггера инверсный, то и сигнал будет перевернут. При нажатии кнопки уровень на входе ***Arduino*** будет высокий. При написании программы необходимо помнить об этом. Окончательный вид выходного сигнала изображен на рис. 12.7, как видим, сигнал чистый, без дребезга. Такой сигнал вполне подходит для формирования аппаратного прерывания.
260
Часть IV. Дополнительные темы и проекты
VCC
Рис. 12.6. Окончательный вариант схемы подавления дребезга
12.3.2. Монтаж схемы
Мы разобрались, как работает аппаратное подавление дребезга для кнопки. Для экспериментов возьмем RGB-светодиод и одну кнопку. Соедините элементы согласно схеме, изображенной на рис. 12.8.
Глава 12. Аппаратные прерывания и прерывания по таймеру
261
Тек SL SI Ready M Pgs; 0.000s TRIGGER
Type
ш

Source
4- щи
Mode
[Normal]
Coupling
jjg
CH1 2.00V M 250ms CH1/2.80V
8-Apf-1310:33 <10Hz
Погашение дребезга с помощью RC-контура и триггера Шмитта Рис. 12.7. Сигнал на выходе схемы подавления дребезга с триггером Шмитта
S 3 м м PWM-* ~ 3> PWM- O Ql PWM ^
PWM <т‘
;~2; PWM 01
Рис. 12.8. Схема устройства подавления дребезга
• «

12.3.3. Программа обработки аппаратного прерывания
Пришло время написать простую программу для проверки работы устройства устранения дребезга и уяснения возможностей аппаратных прерываний ***Arduino***. Наиболее очевидное и полезное применение аппаратных прерываний — опрос внешних входов во время работы программы, в которой присутствуют задержки delay (). Есть много сценариев, в которых это необходимо, мы будем управлять яркостью светодиода, формируя ШИМ-сигнал с помощью функции anaiogwrite (). Устройство подает на один из трех выводов RGB-светодиода сигнал, варьирующий от 0 до 255 и обратно. Каждый раз при нажатии на кнопку меняется цвет свечения. Это невозможно сделать с помощью опроса состояния вывода в цикле, т. к. момент нажатия кнопки почти наверняка будет пропущен.
Сначала разберемся, что такое переменные volatile. Если переменная будет меняться при выполнении прерывания, ее следует объявить как volatile (нестабильная). Это необходимо для корректной обработки переменной компилятором. Пример объявления переменной как volatile:
volatile int selectedLED = 9;
Для вызова прерывания в ***Arduino*** предусмотрена функция attachinterrupt (), находящаяся в setupo. Параметры функции: идентификатор прерывания (или номер контакта для платы Due), имя функции обработки прерывания и режим запуска прерывания (`LOW`, change, rising или falling). В нашей программе объявлено прерывание 0 (вывод 2 на плате ***Uno***), которое при срабатывании rising (по фронту) запускает функцию swap ():
attachinterrupt(0, swap, RISING);
Осталось написать код функции обработки прерывания swap () и добавить его к основной программе. После подключения прерывания и написания кода функции обработки прерывания вы можете писать остальной текст программы. Каждый раз при вызове прерывания основная программа приостанавливается, выполняется функция обработки прерывания, затем ход основной программы продолжается с того места, где она была прервана. Поскольку прерывание останавливает основную программу, его обработка должна быть очень короткой и не содержать задержки delay o. Теперь все готово для написания программы управления яркостью RGB-светодиода и переключения цвета по нажатию кнопки. Полный текст программы приведен в листинге 12.1.
Листинг 12.1. Аппаратные прерывания, реализующие многозадачность — hw_multitask.ino
```
// Кнопка с аппаратной противодребезговой защитой,
// управляемая прерыванием // Контакт кнопки
const int BUTTON_INT =0; // Прерывание 0 (вьюод 2 для Uno)
const int RED=11; // Красный вывод RGB-светодиода контакт11
const int GREEN=10; // Зеленый вывод RGB-светодиода контакт 10
const int BLUE=9; // Синий вывод RGB-светодиода контакт 9
```

```
// Переменные volatile можно изменять внутри функции обработки прерывания volatile int selectedLED
```

```
= RED;
void setup()
{
pinMode (RED, OUTPUT); pinMode (GREEN, OUTPUT); pinMode (BLUE, OUTPUT) ;
// Режим прерывания RISING (переход с LOW на HIGH) attachlnterrupt(BUTTON_INT, swap, RISING);
}
void swap()
{
// Выключить текущий цвет analogWrite(selectedLED, 0);
// Новое значение для переменной selectedLED if (selectedLED == GREEN) selectedLED = RED; else
if (selectedLED == RED) selectedLED - BLUE; else if (selectedLED == BLUE) selectedLED = GREEN;
}
void loop()
{
for (int i = 0; i<256; i++)
{
analogWrite(selectedLED, i) ; delay(10);
}
for (int i = 255; i>= 0; i—)
{
analogWrite(selectedLED, i); delay(10);
}
```

Загрузите программу на плату ***Arduino***, вы увидите изменение яркости одного из цветов (R, G или B) RGB-светодиода от нуля до максимума и обратно. Каждый раз при нажатии на кнопку выбирается следующий цвет с той же яркостью, что и предыдущий.

ПРИМЕЧАНИЕ

Посмотреть видеоурок, демонстрирующий пример аппаратного прерывания ***Arduino*** для кнопки без дребезга, можно на странице http.7/www.exploringarduino.com/ content/ch12. Этот видеофайл доступен также на сайте издательства Wiley.

12.4. Прерывания по таймеру

Аппаратные прерывания — не единственный вид прерываний, который возможен для ***Arduino***. Существуют также прерывания по таймеру. В контроллере ATmega328 (установленном на ***Arduino Uno***) есть три аппаратных таймера. На самом деле ***Arduino*** по умолчанию использует эти таймеры для функции minis o, работы с delay () и для включения ШИМ при вызове analogWrite (). Хотя в языке программирования ***Arduino*** отсутствуют специальные конструкции для работы с таймерами, управляя таймерами вручную, можно генерировать произвольные ШИМ-сигналы на любом контакте и делать многое другое. Далее мы расскажем, как с помощью сторонних библиотек (библиотека TimerOne) управлять 16-раз-рядным таймером Timerl в ATmega328. Подобные библиотеки есть и для плат Leonardo, но здесь опишем только работу с ***Arduino Uno***.

ПРИМЕЧАНИЕ

Таймер Timerl служит для включения ШИМ-сигнала на выводах 9 и 10, поэтому при работе с библиотекой TimerOne выполнить функцию analogWrite () для этих контактов не удастся.

12.4.1. Общие сведения о прерываниях по таймеру

Так же, как секундомер в ваших часах, таймеры в ***Arduino*** начинают отсчет с нуля, увеличивая значение с каждым тактом кварцевого резонатора. Timerl — 16-разрядный таймер, следовательно, значения в нем меняются от 0 до 216 - 1 (или 65 535). Как только это число будет достигнуто, значения сбрасываются в 0 и отсчет начинается снова. Время, через которое таймер достигает максимального значения, зависит от делителя частоты. Поскольку тактовая частота равна 16 МГц, то при отсутствии делителя переполнение и сброс Timerl произойдет много раз в секунду. Библиотека TimerOne берет на себя все нюансы работы с таймером и позволяет установить любой интервал времени в микросекундах для срабатывания прерывания по таймеру.

12.4.2. Установка библиотеки

Для начала загрузите библиотеку TimerOne либо со страницы сайта Wiley для этой главы, либо

непосредственно со страницы https://code.google.com/p/arduino-timerone/downloads. Распакуйте архив в папку с названием TimerOne и скопируйте в папку библиотек ***Arduino***. Размещение этой папки по умолчанию отличается для разных операционных систем:

- Windows — Documents/***Arduino***/libraries;
- Mac — Documents/***Arduino***/libraries;
- Linux — /home/YOUR_USER_NAME/sketchbook/libraries.

Если в момент распаковки архива и копирования папки TimerOne среда ***Arduino*** IDE была открыта, перезапустите ее и убедитесь, что библиотека загрузилась. Теперь все готово для работы с Timerl на ***Arduino***.

Глава 12. Аппаратные прерывания и прерывания по таймеру

265

12.4.3. Одновременное выполнение двух задач

Важно понимать, что с помощью ***Arduino*** нельзя реализовать "истинную" многозадачность. Прерывания создают лишь видимость одновременного выполнения нескольких операций, позволяя переключаться между несколькими задачами чрезвычайно быстро. Использование библиотеки TimerOne позволит заставить мигать светодиод по таймеру, пока в цикле loop () выполняются другие задачи. В конце главы мы опишем проект, где в основном цикле loop о данные выводятся в последовательный порт с задержкой delay о, а по прерываниям таймера осуществляется управление светодиодом и пьезозуммером. Чтобы убедиться, что библиотека работает правильно, можно загрузить программу, приведенную в листинге 12.2, на плату ***Arduino Uno***. Светодиод, подключенный к контакту 13 платы ***Arduino***, будет мигать раз в секунду и управляться с помощью таймера. Если вставить какой-либо фрагмент кода в цикл loop о, то он будет выполняться одновременно.

Листинг 12.2. Прерывания по таймеру для проверки blink — timerl.ino

```
// Использование прерывания по таймеру #include <TimerOne.h> const int LED=13;
void setup()
{
pinMode(LED, OUTPUT);
Timerl.initialize(1000000); // Значение для переполнения 1000000 мкс
// (1 секунда)
Timerl. attachlnterrupt (blinky) ; // Выполнять blinky()при переполнении
}
void loop ()
{
// Вставьте сюда любой код
}
// Обработка прерываний по таймеру void blinky()
{
digitalWrite(LED, !digitalRead(LED)); // Переключить статус светодиода
}
```

В функции Timerl.initialize о задается временной интервал таймера в микросекундах. В этом примере установлено значение 1 с (1000000 мкс). Аргумент команды Timerl. attachlnterrupt () — имя функции, которая будет выполняться по истечении заданного интервала времени. Очевидно, что функция обработки займет меньше времени, чем интервал между прерываниями.

266

Часть IV. Дополнительные темы и проекты

12.5. Музыкальный инструмент на прерываниях

Чтобы закрепить знание аппаратных прерываний и прерываний по таймеру, создадим электронный музыкальный инструмент, который воспроизводит ноты последовательно в нескольких октавах. Ноту выбирают с помощью кнопки с аппаратной противодребезговой защитой. По прерываниям таймера одна и та же нота по порядку воспроизводится сначала в октаве 2, затем 3 и так до октавы 6, пока нажатием кнопки не будет выбрана следующая нота. В цикле loop () для отладки можно предусмотреть вывод текущей ноты и октавы через последовательный интерфейс на экран компьютера.

Если известна начальная частота, несложно вычислить частоты нот для каждой октавы. Например, рассмотрим ноту (До). Нота До большой октавы (С2) имеет частоту около 65 Гц. Чтобы получить

частоту для До малой октавы С3 (130 Гц) умножим частоту С2 на 2. Чтобы получить частоту для С4 (260 Гц), умножим частоту для С3 на 2. Таким образом, частоту ноты для каждой октавы моясно вычислить, умножая исходную частоту на степень числа 2. Зная это правило, будем увеличивать коэффициент в два раза при прерываниях по таймеру.

Переключать ноты будем так же, как в предыдущем примере с цветным светодиодом, — по нажатию кнопки. Задаем базовые частоты для каждой ноты и переключаем их каждый раз при нажатии кнопки.

12.5.1. Схема музыкального инструмента

Аппаратная часть проекта очень простая. К схеме предыдущего примера с RGB-светодиодом добавляем динамик, подключенный к контакту 12 ***Arduino*** через резистор 150 Ом. Я взял пьезоизлучатель, вы можете выбрать другой динамик. Монтажная схема показана на рис. 12.9.

12.5.2. Программа для музыкального инструмента

Программа для музыкального инструмента использует аппаратные прерывания и прерывания по таймеру, а также функцию tone о для управления динамиком и осуществляет передачу данных в последовательный порт. Загрузите код из листинга 12.3 на плату ***Arduino*** и нажимайте кнопку для перебора базовых частот. Посмотреть частоту, воспроизводимую в текущий момент, можно в мониторе последовательного порта.

Листинг 12.3. Код для музыкального инструмента — fun_with_sound.ino

```
// Использование аппаратных прерываний и прерываний по таймеру // Подключение библиотеки TimerOne #include <TimerOne.h>
//
const int BUTTON_INT =0; // Прерывание 0 (вывод 2 для Uno) const int SPEAKER=12;      // Вывод 12 для подключения динамика
```

Гпава 12. Аппаратные прерывания и прерывания по таймеру

267

Рис. 12.9. Графическая схема музыкального аппарата

```
// Базовые частоты для нот #define NOTE_C 65 tdefine NOTE_D 73 #define NOTE_E 82 #define NOTE_F 87 #define NOTE_G 98 #define NOTE_A 110 #define NOTE B 123
// Переменные volatile для изменения при обработке прерываний volatile int key = NOTE_C; volatile int octave_multiplier = 1;
void setup()
{
// Запуск последовательного порта Serial.begin(9600); pinMode (SPEAKER, OUTPUT);
// Запуск для кнопки аппаратного прерывания по RISING attachlnterrupt(BUTTON_INT, changeKey, RISING);
//Set up timer interrupt
Timerl.initialize(500000);      // Период 0,5 секунды
```

• &

268

Часть IV. Дополнительные темы и проекты

```
Timerl.attachlnterrupt(changePitch);
}
// Выполнять changePitch()
// при прерываниях по таймеру
void changeKeyO
{
octave_multiplier = 1; if (key == NOTE_C) key = NOTE_D; else if (key == NOTE_D) key = NOTE_E; else if (key == NOTE_E) key = NOTE_F; else if (key == NOTE_F) key = NOTE_G; else if (key == NOTE_G) key = NOTE_A; else if (key == NOTE_A) key = NOTE_B; else if (key == NOTE_B) key = NOTE_C;
// Обработка прерывания по таймеру void changePitch()
{
octave_multiplier = octave_multiplier * 2; if (octave_multiplier > 16) octave_multiplier = 1; tone(SPEAKER,key*octave_multiplier);
}
void loop()
```

```
{
Serial.print("Key: ");
Serial.print(key);
Serial.print(" Multiplier: ");
Serial.print(octave_multiplier);
Serial.print(" Frequency: ");
Serial.println(key*octave_multiplier); delay(100);
```

Значения частот для нот можно найти в Интернете. Базовые частоты определены для нот второй октавы. Обратите внимание, что переменные key и octave_ multiplier должны быть объявлены как volatile, потому что их значения меняются во время обработки прерываний. Функция changeKeyO вызывается каждый раз при срабатывании прерывания по нажатию кнопки и изменяет базовое значение пере-

менной key. Функция changePitch () вызывает tone () для установки частоты сигнала для динамика. Она запускается каждые полсекунды по прерыванию таймера. При каждом вызове она удваивает базовую частоту, пока не будет достигнуто ее 16-кратное увеличение. Затем процесс повторяется с начальной базовой частоты. В цикле loop о в последовательный монитор каждые 0,1 с выводятся значения переменных key, octave_muitipiier и частоты.

ПРИМЕЧАНИЕ

Посмотреть видеоклип, демонстрирующий работу музыкального инструмента, можно на странице http://www.exploringarduino.com/content/ch12. Этот видеофайл доступен и на странице издательства Wiley.

Резюме

В этой главе вы узнали следующее:

- Как выбрать альтернативу: опрос входов в цикле или использование прерываний.
- Что реализация прерываний на различных платах ***Arduino*** неодинакова. На плате Due прерывания можно задать для любого контакта, на других платах для прерываний доступны только определенные контакты.
- Как создать противодребезговую защиту кнопок на аппаратном уровне с помощью RC-цепочки и триггера Шмитта.
- Что с помощью функций обработки прерываний можно асинхронно опрашивать входы ***Arduino***.
- Как с помощью сторонней библиотеки TimerOne организовать прерывания по таймеру.
- Как комбинировать прерывания таймера, аппаратные прерывания и опрос в одной программе, чтобы параллельно выполнять несколько задач.

ГЛАВА
13
Обмен данными с картами памяти SD

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino*** (рекомендуется ***Uno***);
- USB-кабель для программирования платы ***Arduino***;
- источник питания для платы ***Arduino*** (блок питания или аккумулятор);
- инфракрасный датчик расстояния;
- плата часов реального времени;
- плата расширения SD card shield;
- карта памяти SD;
- набор перемычек;
- макетная плата;
- компьютер с картридером.

Электронные ресурсы к главе

На странице http://www.expIoringarduino.com/conteiit/chl3 можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/expioringarduino в разделе Downloads.

Что вы узнаете в этой главе

Есть множество примеров устройств на основе плат ***Arduino*** для сбора данных о состоянии атмосферы с метеодатчиков, зондов, датчиков систем жизнеобеспечения зданий и т. п. Учитывая небольшой размер, минимальное энергопотребление и простоту взаимодействия с датчиками, платы ***Arduino*** удобны для построения регистраторов данных (устройств для записи и хранения информации в течение определенного периода времени). Такие регистраторы часто присутствуют в сетях сбора данных и для хранения накопленной информации требуется энергонезависимая память, например SD-карта. В этой главе вы узнаете, как связать ***Arduino*** с SD-картой.

Гпава 13. Обмен данными с картами памяти SD

271

Мы расскажем, как записывать данные в файл и как читать информацию с SD-карты. С помощью часов реального времени к данным мы добавим точные временные метки. Вы также узнаете о том, как отобразить эти данные на вашем компьютере.

ПРИМЕЧАНИЕ

Видеоурок по записи и хранению данных можно посмотреть на странице http://www.jeremyblum.com/2011/04/05/tutorial-11 -for-arduino-sd-cards-and-datalogging/1.

ПРИМЕЧАНИЕ

Видеоурок о регистрации данных местоположения с помощью GPS-приемника можно посмотреть на странице http://www.jeremyblum.com/2012/07/16/tutorial-15-for-arduino-gps-tracking/. Эти видеофайлы доступны и на странице издательства Wiley.

13.1. Подготовка к регистрации данных

Системы регистрации данных, как правило, предназначены для сбора информации, поступающей, например, с аналоговых датчиков. Неотъемлемая часть подобных систем — устройства памяти для хранения данных в течение длительного времени. Далее рассмотрим несколько способов использования SD-карты для записи данных с платы ***Arduino***. Кратко перечислим возможные применения регистраторов:

- метеостанция для мониторинга освещенности, температуры и влажности в течение длительного времени;
- GPS-трекер для отслеживания местоположения;
- система контроля за температурой компонентов персонального компьютера;
- регистратор для управления электрическим освещением в доме или офисе.

В конце главы мы разработаем систему протоколирования данных с инфракрасным датчиком расстояния, которая позволит создать журнал входа и выхода людей из помещения.

13.1.1. Форматирование данных с помощью CSV-файлов

Хранить данные на SD-карте будем в CSV-файлах. Файлы этого формата легко создать, считать и проанализировать в различных приложениях, что хорошо подходит для задач регистрации информации. Стандартный CSV-файл выглядит примерно так:

Date,Time,Valuel,Value2 2013-05-15,12:00,125,255 2013-05-15,12:30,100,200 2013-05-15,13:00,110,215

1 На русском: Шр://\У1к1.атрегка.ги/видеоуроки:11-8<1-карты-и-регистрация-данных.

272

Часть IV. Дополнительные темы и проекты

Массивы данных начинаются с новой строки, столбцы разделяются запятыми. Так как запятые служат разделителем, основным требованием для данных является отсутствие в них запятых. Кроме того, каждая строка должна, как правило, иметь одинаковое число элементов. Если приведенный файл открыть в программе Excel на компьютере, то он будет выглядеть, как показано в табл. 13.1.

Таблица 13.1. Просмотр CSV-файла в программе Excel

Date	Time	Valuel	Value2
2013-05-15	12:00	125	255
2013-05-15	12:30	100	200
2013-05-15	13:00	110	215

Поскольку CSV-файлы являются простыми текстовыми файлами, записывать в них данные можно с помощью знакомых нам команд print () и printin (). С помощью ***Arduino*** можно легко проанализировать CSV-файлы, считывая их по строкам и отделяя нужную информацию от разделителей.

13.1.2. Подготовка SD-карты для регистрации данных

Перед тем как начать запись данных с ***Arduino***, необходимо определиться с SD-картой. Вид SD-карты

зависит от SD-адаптера, который вы будете использовать. Существуют полноразмерные SD-карты и microSD. Большинство microSD-карт поставляется с адаптером, который позволяет подключить их к считывателю стандартных SD-карт. Для выполнения упражнений из этой главы вам потребуется картридер для компьютера (внешний или встроенный).

Большинство новых SD-карт заранее отформатировано и готово к использованию с ***Arduino***. Если на вашей карте была записана какая-нибудь информация, необходимо ее отформатировать в FAT16 или FAT32. Карты объемом до 2 Гбайт следует отформатировать в FAT16, карты с большим объемом памяти— в FAT32. В примерах этой главы мы применяем microSD-карту, отформатированную в FAT16. Обратите внимание, что при форматировании карты удаляется вся имеющаяся на ней информация, но при этом гарантируется, что карта готова для взаимодействия с ***Arduino***. Если у вас новая карта, следующие шаги можно пропустить и вернуться к ним только при возникновении проблем с доступом к карте из ***Arduino***.

Форматировать SD-карту в Windows легко:

1. Вставьте SD-карту в картридер. Откройте окно Computer (Мой компьютер) (рис. 13.1).
2. Щелкните правой кнопкой мыши по значку SD-карты (она может иметь другое имя) и выберите опцию Format (рис. 13.2). Появится окно с вариантами форматирования карты (рис. 13.3).
3. Выберите тип файловой системы (FAT для карт 2 Гбайт и меньше, FAT32 для карт большего объема), оставьте размер кластера по умолчанию и задайте метку

Гпава 13. Обмен данными с картами памяти SD
273
к ► Computer ►
ттшшш
Qrganue » System properties llnmstrrll or cheoge a program Map network drive Open Control Pane!
* s »*
Vi' Favorites ffi Desktop 4 Downloads y, Drepbox fe Google Drive ;.j Recent Places
^ Libraries * Documents Dropbcst ».! Google Drive ^ Music an: Pictures Й Videos
Hard Disk Drives (4)
Local Dick {Cc>
&’ iMmaxuMK £H_
85008free of Ш TE
Devices with Removable Storage {7) lyjg DVD RW Drive (£:)

Backup Drive (X:) (iniT.ririri,iiMi»ViiaiHwW[#n i-
Ш C8 free of 931GB
SD CARD (1+)
iS GB free of 1,86 SB

DVD P,W Drive (F:>
Removable Disk <fc)
*щфг
Removable Disk (G:)
Removable Disk DO
mm,*
Removable Disk (fc)
Homegroup
■M Computer &, Local Disk <C:(«» SD CARD (HO > a Backup Drive (X:)
Network ;*i PHOENIX
PHOENIX Workgroup: WORKGROUP Memory: BjOOGB
Processor; lntel(R) Core(TM) i7 CP...

Рис. 13.1. SD-карта в окне Computer

тома (я выбрал LOG, но вы можете выбрать любую). На рис. 13.3 изображена конфигурация форматирования для карты 2 Гбайт.

4. Нажмите кнопку Start, чтобы отформатировать карту памяти.

На компьютерах с ОС Мае процесс также прост:

1. Используйте команду Finder, чтобы найти и открыть приложение Disk Utility.
2. Нажмите на вкладку SDcard в левой панели и откройте вкладку Erase. Выберите пункт MS-DOS (FAT) для формата.
3. Нажмите кнопку Erase. Система отформатирует карту как FAT 16, независимо от ее емкости (в Mae нельзя отформатировать карту как FAT32).

В Linux можно отформатировать SD-карту из терминала. Большинство дистрибутивов Linux будет монтировать карту автоматически при ее вставке:

1. Вставьте карту, откроется всплывающее окно, отображающее карту.
2. Откройте терминал и введите команду df, чтобы получить список смонтированных устройств. Результат должен выглядеть так, как на рис. 13.4.

Последняя запись должна относиться к SD-карте. На моей системе она была смонтирована как /dev/mmcbikOpi, но у вас может отличаться.

Рис. 13.2. Выбор опции форматирования SD-карты

Рис. 13.3. Окно опций форматирования


```
’ jeremy<3>ubuntu: ~ jereroygubuntu:~$ df
Filesystem     IK-blocks      Used   Available      Use* Mounted on
/dev/loopO     9663011        3370261        6292750        35V/
udev   4057428        4       4057424        1% /dev
tmpfs  1626652        932     1625720        1% /run
jnone  5120   0       5120    6% /run/lock
jnone  4066624        152     4066472        1* /run/shm
jnone  102.400        44      102356         1% /run/user
j/dev/sda2     249954300      226830300      23124000       91% /host
j/dev/mmcblk0pl ijeremygubuntu:    1955424 -5 1  0        19SS424        0% /media/jer«
```

Рис. 13.4. Список смонтированных устройств

3. Прежде чем форматировать SD-карту, необходимо ее отмонтировать с помощью команды umount. Аргументом команды будет название вашей SD-карты (рис. 13.5).

```
ф:     JeremygBubuntu:
jeremygubuntu:~$ df
Filesystem     lK-blocks      Used   Available      Use* Mounted on
/dev/loope     9663011        3370261        6292750        35* /
;udev  4057428        4       4057424        1% /dev
:tmpf s162665?        932     1625720        1* /run
jnone  5120   0       5120    Q% /run/lock
jnone  4066624        1S2     4066472        IX /run/shm
jnone  102400         44      102356         1* /run/user
j/dev/sda2     249954300      226830300      23124000       91% /host
j/dev/mmcblkopl        1955424        0       1955424        0% /media/jeremy/0883-B99?
ijeremygubuntu:~$ umount /dev/mmcblkOpl |jeremygubuntu:~$ |
```

Рис. 13.5. Отмонтирование SD-карты в Linux

4. Форматируем SD-карту с помощью команды mkdosfs. Возможно, вам придется выполнить команду с правами привилегированного пользователя (с помощью команды sudo). Используем флаг -F, чтобы указать файловую систему FAT. Вы можете включить опции 16 или 32, чтобы выбрать FAT16 или FAT32. Отформа-

тировать карту, которая была смонтирована как /dev/mmcblkOpl, можно командой sudo mkdosfs -F 16 /dev/mmcblkOpl (рис. 13.6).

Ваша SD-карта отформатирована и готова к работе! Приступаем к взаимодействию с SD-картой через плату SD card shield.

O jeremy(®ubuntu: ■

```
jeremy@ubuntu:~$ df
Filesystem     IK-blocks      Used  Available      Use% Mounted on
/dev/loop©     9663011        3370261       6292750       35% /
judev  4057428        4       4057424       1% /dev
jtmpfs 1626652        932     1625720       1% /run
jnone  5120   0       5120    0% /run/lock
Inone  4066624        152     4066472       1% /run/shm
jnone  1O24O0         44      102356        1% /run/user
:/dev/sda2     249954300      226830300     23124000      91% /host
;/dev/mmcblkopl        1955424       0       1955424       0% /medta/jer<
jeremy@ubuntu:      umount /dev/mmcblk0pl
jeremy<aubuntu: -$ sudo mkdosfs -F 16 /dev/mmcblkOpl mkdosfs 3.0.13 (30 3un 2012) jeremy@ubuntu:~$
|
```

Рис. 13.6. Форматирование SD-карты в Linux

13.2. Взаимодействие *Arduino* с SD-картой

Для SD-карт, как и для радиомодулей XBee, которые мы рассмотрели в главе 77, требуется питание 3,3 В. Поэтому подключать SD-карту необходимо через переходник, который преобразует логические уровни и напряжение питания для SD-карты. Кроме того, связь с SD-картой возможна по интерфейсу SPI, описанному в главе 9. ***Arduino*** IDE поставляется с удобной библиотекой SD, реализующей функции низкого уровня и позволяющей легко читать и записывать файлы на SD-карту.

13.2.1. Платы расширения для SD-карт

Есть множество плат расширения (переходников) для подключения SD-карт к ***Arduino***. Все рассмотреть в книге невозможно, опишем некоторые из популярных, указав их достоинства и недостатки.

Общие черты всех переходников SD-карт:

♦ подключаются к плате ***Arduino*** по интерфейсу SPI, через 6 контактов ICSP платы ***Arduino*** или к контактам цифровых выводов (11, 12 и 13 на ***Uno*** или 50, 51 и 52 на Mega);

♦ имеют вход выбора (CS), который может быть контактом по умолчанию (53 на Mega, 10 на других платах ***Arduino***);

♦ подают питание 3,3 В на SD-карту и согласуют логические уровни.

Вот список наиболее распространенных плат расширения для SD-карт:

♦ Cooking Hacks Micro SD shield (http://www.exploringarduino.com/parts/cooking-hacks-SD-shield) (рис. 13.7) используется для иллюстрации примеров этой главы. Это самый маленький переходник из рассматриваемых нами, и его можно подключить либо к цифровым контактам (8-13 на ***Uno***), либо к 6-контактному ICSP на ***Arduino***. При подключении к контактам 8-13 вход выбора CS соединен с контактом 10. При подключении к разъему ICSP вход CS можно соединить с любым контактом ***Arduino***. Это полезно, если контакт 10 платы ***Arduino*** занят. Переходник поставляется с SD-картой 2 Гбайт.

Рис. 13.7. Плата расширения Cooking Hacks Micro SD shield

♦ Official ***Arduino*** Wireless SD shield (http://www.exploringarduino.com/parts/arduino-wireless-shield) (рис. 13.8)— это первая из нескольких "официальных" плат расширения ***Arduino*** с поддержкой SD-карт. Переходник содержит схему для подключения радиомодуля XBee и SD-карты памяти, что позволяет легко объединить примеры из настоящий главы и из главы 11. На этой плате расширения вход выбора SD-карты (CS) подключается к контакту 4 ***Arduino***. Вы должны назначить контакт 10 как выход, а также указать в библиотеке, что контакт 4 подключен к CS.

♦ Official ***Arduino*** Ethernet SD shield (http://www.exploringarduino.com/parts/arduino-ethernet-shield) (рис. 13.9) позволяет подключить плату ***Arduino*** к проводной сети. Переходник также реализует интерфейс SD-карты, хотя в основном он служит для хранения файлов, которые будут доступны через сеть. Оба контроллера (Ethernet и SD-карты) на этом переходнике являются SPI-устрой-

Рис. 13.8. Плата расширения ***Arduino*** Wireless SD shield

Рис. 13.9. Плата расширения ***Arduino*** Ethernet SD shield

ствами; вывод CS Ethernet подключается к контакту Ю платы ***Arduino***, а вывод CS SD-карты — к контакту 4.

♦ Official ***Arduino*** Wi-Fi SD shield (http://www.exploringarduino.com/parts/ arduino-wifi-shield) (рис. 13.10) также реализует подключение к сети, но через Wi-Fi. SD-карта, как и в Ethernet SD shield, служит для хранения файлов, доступных через сеть. Как и в Ethernet SD shield, Wi-Fi-контроллер для CS исполь-

зует контакт 10, а SD-карта— контакт 4. Необходимо следить, чтобы оба устройства не были включены одновременно; активной может быть только одна линия CS (низкий логический уровень).

♦ Adafruit data logging shield (http://www.exploringarduino.com/parts/ adafruit-data-logging-shield) (рис. 13.11) особенно хорошо подходит для экспериментов,

Рис. 13.10. Плата расширения ***Arduino*** Wi-Fi SD shield

Рис. 13.11. Плата расширения Adafruit data logging shield

которые мы рассмотрим далее в этой главе, потому что включает в себя часы реального времени (RTC) и интерфейс SD-карты. Вывод CS SD-карты на этом переходнике назначен по умолчанию (53 на Mega, 10— на других платах ***Arduino***), а чип часов реального времени подключен к шине I2C.

На плате расширения SparkFun MicroSD shield (http://www.exploringarduino.com/ parts/spark-fun-microSD-shield) (рис. 13.12) установлен только слот SD-карты. Тем не менее, предусмотрено монтажное поле для припаивания дополнительных компонентов. Вывод CS SD-карты подключен к контакту 8, его необходимо указать при использовании библиотеки SD с этим переходником.

Рис. 13.12. Плата расширения SparkFun MicroSD shield

13.2.2. SPI-интерфейс SD-карты

Как упоминалось ранее, связь платы ***Arduino*** с SD-картой осуществляется через SPI-интерфейс: контакты MOSI, MISO, SCLK (тактовые импульсы) и CS (выбор чипа). Для выполнения примеров этой главы мы будем использовать библиотеку SD. Это предполагает, что задействованы аппаратные SPI-контакты на плате ***Arduino***, а в качестве CS назначен или контакт по умолчанию или определенный пользователем. Библиотека SD для правильной работы требует установки контакта для CS по умолчанию в качестве выхода, даже если для CS задан другой вывод. В случае ***Uno*** это вывод 10, для платы Mega это вывод 53. Описанные далее примеры собраны на плате ***Arduino Uno*** с выводом CS, заданным по умолчанию.

13.2.3. Запись на SD-карту

Библиотека SD позволит создать несложный пример записи данных на SD-карту. Будем получать данные с датчиков и записывать на карту. Данные хранятся в фай-

```
if (! SD.begin (CS_j?in) )
{
Serial.printIn("Card Failure"); return;
}
Serial.println("Card Ready");
```


ле под названием log.csv, в дальнейшем его можно открыть на компьютере. Важно отметить, что при форматировании карты в FAT 16 имена файлов должны быть записаны в формате "8.3", т. е. расширение должно состоять из трех символов, а имя файла — не более чем из восьми символов.

Убедитесь, что переходник SD правильно подключен к плате ***Arduino*** и в него установлена SD-карта. Для Cooking Hacks Micro SD shield это будет выглядеть так, как на рис. 13.13 (используются контакты 8-13 ***Arduino*** и перемычка находится на правой стороне).

Рис. 13.13. Подключение платы расширения Cooking Hacks Micro SD shield

Для отладки программы будем контролировать статус нескольких функций библиотеки SD. Например, для установки связи с SD-картой необходимо вызвать следующую функцию (листинг 13.1):

Листинг 13.1. Функция инициализации SD-карты

Обратите внимание, что мы не просто инициализируем обмен с картой с помощью функции sd.begin (cs_pin), а получаем статус выполнения этой функции. При успешной инициализации программа выдает в последовательный порт сообщение об этом, в противном случае выводится сообщение о неуспехе и команда возврата останавливает дальнейшее выполнение программы.
При записи строки в файл подход аналогичный. Например, если вы хотите записать новую строку "hello" в файл, код будет выглядеть так, как в листинге 13.2.
Листинг 13.2. Функция записи информации на SD-карту

```
File dataFile = SD.open("log.csv", FILE_WRITE); if (dataFile)
{
dataFile.println("hello");
dataFile.close();        //.Данные не записываются,
// пока соединение не будет закрыто
}
else
{
Serial.println("Couldn't open log file");
}
```

В первой строке расположена команда создания нового файла (или открытие, если он уже существует) с названием log.csv. Если файл создан/открыт успешно, переменная dataFile получит значение true, и начнем процесс записи данных в файл. В противном случае сообщаем об ошибке в последовательный порт. Запись строки данных в файл осуществляет функция dataFile.println(); чтобы предотвратить добавление символа новой строки, вызывайте функцию dataFile.print o. Все данные направляются в буфер и записываются в файл только после выполнения команды dataFile. close ().
Теперь напишем простую программу, которая создает на SD-карте файл log.csv и каждые 5 секунд записывает в него через запятую метки и какое-либо сообщение (листинг 13.3). В каждой строке файла CSV будет записана временная метка (текущее значение функции minis o) и некоторый текст. Программа может показаться вам бесполезной, но на самом деле это важный пример, подготавливающий взаимодействие с реальными датчиками, чем мы займемся в дальнейших проектах.
Листинг 13.3. Тест записи данных на SD-карту — write_to_sd.ino

```
// Запись данных на SD-карту #include <SD.h>
// Подключение контактов //        MOSI =        pin      11
//        MISO =        pin      12
//        SCLK =        pin      13
```



```
// Подключение контакта выбора CS const int CS_PIN = 10;
// Контакт для питания SD-карты const int POW_PIN =8;
void setup()
{
Serial.begin(9600);
Serial.println("Initializing Card");
// Установить CS как выход pinMode(CS_PIN, OUTPUT);
// Для питания карты используется вывод 8, установить HIGH pinMode(POW_PIN, OUTPUT);
digitalWrite(POW_PIN, HIGH); if (!SD.begin(CS_PIN))
{
Serial.println("Card Failure"); return;
}
Serial.println("Card Ready");
}
void loop()
{
long timestamp = millisO;
String dataString = "Hello There!";
// Открыть файл и записать в него
```

```
File dataFile = SD.open("log.csv", FILE_WRITE);
if (dataFile)
{
dataFile.print(timestamp);
dataFile.print(",");
dataFile.println(dataString);
dataFile.close(); // Данные не записаны,
// пока соединение не закрыто!
// Вывод в монитор для отладки Serial.print(timestamp);
Serial.print(",");
Serial.println(dataString);
}
else
{
Serial.printIn("Couldn't open log file");
}
delay(5000);
```


Обратите внимание на несколько моментов, особенно если у вас такой же переходник, как у меня (Cooking Hacks Micro SD shield):

- CS можно установить на любом контакте; если это не контакт по умолчанию (10), то необходимо в setup() предусмотреть команду pinMode(lO, output), иначе библиотека SD не будет работать;
- питание на переходник подается через контакт 8, поэтому pow_pin должен быть установлен в качестве выхода и в setup о необходимо определить его значение как `HIGH`;
- при каждом проходе цикла loop о временная метка обновляется значением текущего времени, прошедшего с начала выполнения программы в миллисекундах. Переменная должна иметь тип long, потому что minis о возвращает число больше, чем 16 бит.

Файл открывается для записи и в него добавляются данные, разделенные запятыми. Мы также выводим эти данные в последовательный порт для отладочных целей. Если вы откроете терминал последовательного порта, то увидите вывод данных как на рис. 13.14.

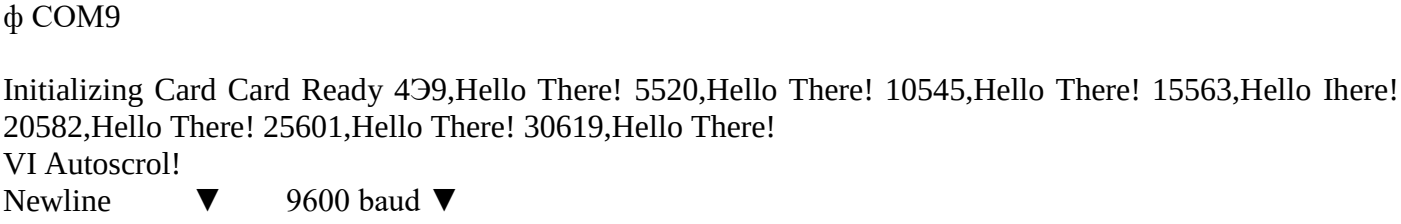

Рис. 13.14. Вывод данных в последовательный порт для отладки

A	B
1	10 Hello There!
2	5030 Hello There!
3	10049 Hello There!
4	15067 Hello There!
5	20085 Hello There!
6	25104 Hello There!
7	30123 Hello There!
8	35141 Hello There!
9	40160 Hello There!
10	45178 Hello There!
11	50197 Hello There!
12	55218 Hello There!
13	60239 Hello There!
14	65258 Hello There!

15 70277 Hello There!
16 75295 Hello There!
17 80313 Hello There!
Рис. 13.15. Просмотр записанных данных в программе электронных таблиц
При возникновении ошибок проверьте правильность подключения переходника, убедитесь, что SD-карта отформатирована и должным образом вставлена. Для проверки корректности записи данных вставьте SD-карту в компьютер и откройте файл в программе просмотра электронных таблиц (рис. 13.15). Обратите внимание,

что данные располагаются в таблице с учетом разделяющих запятых и символов перехода на новую строку.
13.2.4. Чтение с SD-карты
Теперь рассмотрим чтение информации с SD-карты. При регистрации данных это не нужно, но может оказаться полезным для установки параметров программы. Например, можно указать частоту регистрации данных.
Вставьте SD-карту в компьютер и создайте на ней новый текстовый файл с именем speed.txt. В этом файле просто введите время обновления в миллисекундах. На рис. 13.16 показано, что я задал время, равное 1000 мс (1 с).
После записи желаемой скорости обновления сохраните файл на SD-карте и вставьте карту в переходник, подключенный к ***Arduino***. Теперь изменим програм-
Рис. 13.16. Пример файла данных

му, предусмотрев чтение этого файла, извлечение информации и установку скорости обновления данных для регистрации.
Открыть файл для чтения позволяет та же команда SD.openO, но без указания второго параметра filejwrite. Поскольку класс File наследует свойства от класса потока (так же, как serial), можно использовать многие из полезных команд, например parseint (). Сказанное иллюстрирует листинг 13.4.
Листинг 13.4. Пример чтения информации с SD-карты

```
File commandFile = SD.open("speed.txt"); if (commandFile)
{
Serial.println("Reading Command File"); while(commandFile.available())
{
refresh_rate = commandFile.parseint() ;
}
Serial.print("Refresh Rate = ");
Serial.print(refresh_rate);
Serial.println("ms");
}
else
{
Serial.println("Could not read command file."); return;
}
```

Программа из листинга 13.4 открывает файл для чтения и считывает целые значения. Значение сохраняется в переменной частоты обновления, которую необходимо будет определить в начале программы. После чтения файл следует закрыть.
Теперь можно объединить листинги 13.3 и 13.4 и менять скорость записи, основываясь на содержимом файла speed.txt, как показано в листинге 13.5.
Листинг 13.5. Чтение и запись с SD-карты — sd_read_write.ino

```
// Чтение и запись SD-карты #include <SD.h>
// Подключение контактов //MOSI = pin 11 //MISO = pin 12 //SCLK = pin 13
// Подключение контакта выбора CS const int CS_PIN =10; const int POW_PIN =8;
// Скорость опроса по умолчанию int refresh rate = 5000;
```

287

```
void setup()
{
Serial.begin(9600) ;
Serial.println("Initializing Card");
// Установить CS как выход pinMode{CS_PIN, OUT PUT);
// Для питания карты используется контакт 8, установить HIGH pinMode(POW_PIN, OUTPUT); digitalWrite(POW_PIN, HIGH); if (!SD.begin(CS_PIN))
{
Serial.println("Card Failure"); return;
}
Serial.println("Card Ready");
// Чтение настроек (speed.txt)
File commandFile = SD.openCspeed.txt"); if (commandFile)
{
Serial.println("Reading Command File"); while(commandFile.available())
{
refresh_rate = commandFile.parseInt();
}
Serial.print("Refresh Rate = ");
Serial.print(refresh_rate);
Serial.printIn("ms") ;
commandFile.close(); // Закрыть файл настроек
}
else
{
Serial.println("Could not read command file."); return;
}
void loop()
{
long timeStamp = millisO;
String dataString = "Hello There!";
// Открыть файл и записать в него
File dataFile = SD.open("log.csv", FILE_WRITE);
if (dataFile)
{
dataFile.print(timeStamp); dataFile.print(","); dataFile.println(dataString); dataFile.close(); // Закрыть файл
```

288


```
// Вывод в последовательный порт для отладки Serial.print(timestamp) ;
Serial. print (", ") ;
Serial.println(dataString);
}
else
{
Serial.println("Couldn't open log file");
}
delay(refresh_rate);
```

После загрузки на плату и запуска программы данные будут записываться с частотой, указанной при настройке. За процессом можно наблюдать в мониторе последовательного порта (рис. 13.17).

Рис. 13.17. Регистрация данных на скорости Refresh Rate, указанной в настройках

13.3. Использование часов реального времени

Почти каждое приложение регистрации данных выиграет от использования часов реального времени. Наличие часов реального времени (RTC) в системе позволит вставлять временные метки измерений, поэтому легко можно отследить, когда произошло событие. В предыдущем разделе мы вызывали

функцию minis o, чтобы отследить время, прошедшее с начала включения платы ***Arduino***. Теперь задейству-
Гпава 13. Обмен данными с картами памяти SD
289
ем микросхему часов реального времени, позволяющую фиксировать текущее время регистрации данных на SD-карту.
13.3.1. Общие сведения о часах реального времени
Назначение часов реального времени ясно из названия. Вы устанавливаете время один раз, а часы продолжают очень точно отсчитывать время, даже с учетом високосных годов. В описанном далее примере выбрана популярная микросхема часов реального времени DS1307.
Микросхема часов реального времени DS1307
Часы реального времени DS1307 поддерживают связь с ***Arduino*** по протоколу l2C и подключаются к круглой батарейке, что обеспечивает ход часов в течение нескольких лет. К микросхеме подключается кварцевый резонатор, определяющий точность хронометража. Я выбрал плату расширения adafruit-DS 1307-breakout (http://www.expIoringarduino.com/parts/adafruit-DS1307-breakout), которая содержит микросхему DS1307, кварцевый резонатор, батарейку размером с монету, конденсатор развязки и I2C подтягивающие резисторы. Плату легко подключить к ***Arduino*** (рис. 13.18).
Рис. 13.18. Подключение платы расширения adafruit-DS 1307-breakout к ***Arduino***
Далее предполагается, что вы используете эту плату. Тем не менее, можно просто собрать схему из элементов на макетной плате и подключить непосредственно к ***Arduino***. Потребуется кварцевый резонатор на 32,768 кГц, подтягивающие резисторы номиналом 2,2 кОм и круглая батарейка 3,0 В размером с монету. Если вы решили собрать плату самостоятельно, можете приобрести эти компоненты и собрать их на макетной плате по схеме, приведенной на рис. 13.19.
290
Часть IV. Дополнительные темы и проекты
Рис. 13.19. Схема часов реального времени, собранная на макетной плате
Сторонняя библиотека ***Arduino*** RTClib
Как и в предыдущей главе, мы снова воспользуемся сторонней библиотекой для ***Arduino***. На этот раз для облегчения связи с микросхемой часов реального времени (RTC). Библиотека называется RTClib, первоначально она была разработана JeeLabs, затем обновлена Adafruit Indusrtries. Ссылку для загрузки библиотеки можно найти на веб-странице http://www.exploringarduino.coin/content/chl3. Скачайте библиотеку и распакуйте в папку libraries, как вы это делали в главе 12.
Работать с библиотекой просто. При первом выполнении кода нужно вызвать функцию rtc. adjust o для получения времени с компьютера и настройки часов. Далее RTC работают автономно и можно получать текущее время и дату посредством команды rtc. now (). В следующем примере мы будем использовать эту функцию для ведения журнала регистрации в режиме реального времени.
13.3.2. Использование часов реального времени
Теперь объединим SD-карту и часы реального времени, чтобы включить ведение журнала с помощью временных меток. Мы модифицируем программу, добавив запись показаний часов реального времени вместо значений, выдаваемых функцией
millis ().
Гпава 13. Обмен данными с картами памяти SD
291
Подключение модулей SD card shield и RTC
Подключим к ***Arduino*** модули SD card shield и RTC. Если вы используете платы расширения Cooking Hacks Micro SD shield и adafruit-DS 1307-breakout, то подключение будет выглядеть так, как на рис. 13.20.
Рис. 13.20. Плата ***Arduino*** с подключенными платами расширения Cooking Hacks Micro SD shield и adafruit-DS1307-breakout
Отметим, что последний контакт на RTC не связан с ***Arduino***; это меандр, генерируемый RTC, в нашем примере он не задействован. В программе следует подать на контакт A2 уровень `LOW` и на A3 уровень `HIGH` (+5 В), чтобы обеспечить питание RTC. Если вы собрали свой модуль RTC на макетной плате, то установка будет выглядеть немного по-другому.
Модификация программы для работы с RTC
Теперь нужно добавить функционал RTC в нашу предыдущую программу. Необходимо сделать

следующее:
- подключить библиотеку RTC;
- организовать питание модуля RTC (A2 — LOW, A3 — HIGH);

- инициализировать объект RTC;
- установить время RTC с помощью компьютера;
- записывать фактические временные метки в файл журнала.

Кроме того, в код программы добавлен вывод в файл заголовка столбца при каждом перезапуске журнала. Таким образом, вы легко найдете в журнале, записанном в файл CSV, моменты перезапуска.

ВНИМАНИЕ!

Если после запуска программы вы заметите, что она через некоторое время останавливается, то причина может заключаться в нехватке оперативной памяти. Так происходит из-за строк, которые занимают большой объем оперативной памяти, это относится к командам вывода в последовательный порт Serial, print () и Serial. print In (). Проблему можно решить, удалив из программы указанные команды и поручив компилятору хранить строки не в оперативной памяти, а во флэш-памяти ***Arduino***. Для этого для строк используем обертку F(), например Serial, print In (F ("Hello")). Описанный метод реализован в листинге 13.6.

Обновленная программа (листинг 13.6) использует RTC в качестве таймера для регистрации данных. Программа перемещает большинство строк во флэш-память, чтобы предотвратить переполнение оперативной памяти.

Листинг 13.6. Чтение и запись данных на SD-карту с использованием RTC — sd_read_write rtc.ino

```
// Чтение и запись данных на SD-карту с использованием RTC
#include <SD.h> // Подключение библиотеки SD
#include <Wire.h> // Для работы с RTC
#include "RTClib.h" // Подключение библиотеки RTC
// Подключение устройств SPI и I2C с контактами по умолчанию
// SD-карта SPI контакты
// RTC - стандартные 1^C контакты
const int CS_PIN =10;
const int SD_POW_PIN =8;
const int RTC_POW_PIN =A3;
const int RTC_GND_PIN =A2;
// Скорость опроса по умолчанию 5 секунд int refresh_rate = 5000;
// Создание объекта RTC RTC_DS1307 RTC;
// Переменные для даты и времени
String year, month, day, hour, minute, second, time, date;
void setup()
{
Serial.begin(9600);
Serial.println(F("Initializing Card"));
// Настроить контакты CS и питания как выходы pinMode(CS_PIN, OUTPUT);
```



```
pinMode(SD_POW_PIN, OUTPUT); pinMode(RTC_POW_ PIN, OUTPUT); pinMode(RTC_GND_PIN, OUTPUT);
// Установка питания карты и RTC digitalWrite(SD_POW_PIN, HIGH); digitalWrite(RTC_POW_PIN/ HIGH); digitalWrite(RTC_GND_PIN, LOW);
// Инициализация Wire и RTC Wire.begin();
RTC.begin{};
// Если RTC не запущены, загрузить дату/время с компьютера if (! RTC.isrunning())
{
Serial.println(F("RTC is NOT running!"));
RTC. adjust (DateTime (_DATE , ___TIME__) ) ;
}
```

```
// Инициализация SD-карты if (!SD.begin(CS_PIN))
{
Serial.printIn(F("Card Failure")); return;
}
Serial.println(F("Card Ready"));
// Чтение конфигурационного файла (speed.txt)
File coinmandFile = SD.open("speed, txt") ; if (coinmandFile)
{
Serial.println(F("Reading Command File")); while(commandFile.available() )
{
refresh_rate = commandFile.parselnt();
}
Serial.print(F("Refresh Rate = "));
Serial.print(refresh_rate);
Serial.printIn(F("ms")); commandFile.close() ;
}
else
{
Serial.println(F("Could not read command file.")); return;
}
// Запись заголовка
File dataFile = SD.open("log.csv", FILE_WRITE); if (dataFile)
{
dataFile.println(F("\nNew Log Started!"));
```

```
dataFile.println(F("Date,Time,Phrase")); dataFile.close();
// Запись в последовательный порт Serial.printIn(F("\nNew Log Started!"));
Serial.println(F("Date,Time,Phrase"));
}
else
{
Serial.printIn(F("Couldn't open log file"));
}
}
void loop()
{
// Получить значение даты и времени и перевести в строковые значения
DateTime datetime = RTC.nowO;
year = String(datetime.year(), DEC);
month = String(datetime.month(), DEC);
day = String(datetime.day() , DEC);
hour = String(datetime.hour(), DEC);
minute = String(datetime.minute(), DEC);
second = String(datetime.second(), DEC);
// Собрать строку текущей даты и времени date = year + "/" + month + "/" + day; time = hour +       +
minute + ":" + second;
String dataString = "Hello There!";
// Открыть файл и записать значения
File dataFile = SD.open("log.csv", FILE_WRITE);
if (dataFile)
{
dataFile.print(date);          dataFile.print(F(","));          dataFile.print(time);          dataFile.print(F(","));
dataFile.println(dataString); dataFile.close();
// Вывод в последовательный порт для отладки Serial.print(date);
```

```
Serial.print(F(","));
Serial.print(time);
Serial.print(F(","));
Serial.println(dataString);
}
else
{
Serial.println(F("Couldn't open log file"));
}
delay(refresh_rate);
```

Гпава 13. Обмен данными с картами памяти SD

295

Библиотека RTC импортируется в код строкой #inciude "RTClib.h" и создается объект rtc_dsi307 rtc. RTC является I2C-устройством, поэтому необходимо подключение библиотеки Wire, с которой мы знакомы из главы 8. В секции setup о функция RTC.isrunning{) проверяет, запущена ли микросхема RTC. Если нет, то в микросхему записываются данные с часов компьютера, полученные при компиляции. После установки времени оно не сбрасывается, пока микросхема RTC подключена к батарее. В функции setup о в лог-файл записывается заголовок столбца, чтобы отслеживать моменты перезагрузки системы регистрации.

Во время цикла loop о инициализируем объект DataTime текущими значениями даты и времени из RTC. Из объекта DateTime извлекаем значения года, месяца, дня, часа, минуты, секунды, конвертируем их в строки и объединяем строки в общую строку для представления даты и времени. Эти данные записываются в лог-файл и выводятся в последовательный порт.

Через некоторое время извлечем карту памяти и прочитаем лог-файл на компьютере в программе просмотра электронных таблиц. Таблица должна выглядеть так, как на рис. 13.21.

A	B	C
1		
2	New Log Started!	
3	Date Time Phrase	
4	4/8/2013 23:24:05 Hello There!	
5	4/8/2013 23:24:07 Hello There!	
6	4/8/2013 23:24:09 Hello There!	
1	4/8/2013 23:24:11 Hello There!	
8	4/8/2013 23:24:14 Hello There!	
9	4/8/2013 23:24:16 Hello There!	
10	4/8/2013 23:24:18 Hello There!	
11	4/8/2013 23:24:20 Hello There!	
12	4/8/2013 23:24:22 Hello There!	
13	4/8/2013 23:24:24 Hello There!	
14	4/8/2013 23:24:26 Hello There!	
15	4/8/2013 23:24:28 Hello There!	
16	4/8/2013 23:24:30 Hello There!	
17	4/8/2013 23:24:32 Hello Therel	
18	4/8/2013 23:24:34 Hello There!	
19	4/8/2013 23:24:36 Hello There!	
20	4/8/2013 23:24:38 Hello There!	

Рис. 13.21. Содержимое лог-файла в программе просмотра электронных таблиц

13.4. Регистратор прохода через дверь

Теперь можно приступить к созданию входного регистратора для вашей комнаты. Отслеживать моменты прохода людей через дверь будем с помощью датчика расстояния. Регистратор будет фиксировать эти события и записывать в лог-файл на SD-карту для последующего просмотра на компьютере.

296

Часть IV. Дополнительные темы и проекты

13.4.1. Схема регистратора

Все, что нужно сделать, — это добавить аналоговый датчик расстояния к существующей схеме. Если вы используете те же модули, как и я, не потребуется даже макетная плата. Просто подключите соответствующие контакты к земле, питанию и аналоговому входу A0. Монтажная схема приведена на рис. 13.22.

Рис. 13.22. Монтажная схема регистратора

Для того чтобы система работала стабильно, необходимо установить ИК-датчик расстояния так, чтобы луч датчика шел горизонтально вдоль всей двери. Тогда при проходе через дверь человек обязательно окажется в зоне действия датчика. ИК-датчик расстояния и плату ***Arduino*** на время тестирования программы можно прикрепить к стене с помощью липкой ленты (рис. 13.23).

13.4.2. Программа для регистратора

Для нашего регистратора необязательно читать переменные с карты памяти, поэтому можно удалить данную часть кода. В программу необходимо добавить проверку

Гпава 13. Обмен данными с картами памяти SD

297

Рис. 13.23. Крепление блоков регистратора возле входной двери

показаний ИК-датчика расстояния и определение величины их изменения за время между опросами. Если изменение есть, можно предположить, что кто-то вошел или вышел из комнаты. Нужно выбрать порог изменения аналоговых показаний датчика для точного определения факта прохода через дверь. Я экспериментально определил, что для моей установки изменение значения аналогового сигнала более чем на 75 единиц является достаточным признаком для фиксации движения (у вас, возможно, будет по-другому). Чтобы надежно установить момент прохода, необходимо проверять датчик достаточно часто.

Я рекомендую записывать данные на SD-карту только в момент появления человека, а в остальное время обновлять журнал лишь с определенной периодичностью. При этом будет обеспечен хороший баланс между объемом хранимой информации и точностью работы. Плата ***Arduino*** опрашивает датчик расстояния через 50 мс (и добавляет единицу к текущему столбцу каждый раз при обнаружении движения). Если движение не обнаруживается, записываем ноль в текущий столбец через I секунду (в отличие от 50 мс).

В листинге 13.7 приведена программа входного регистратора, работающая согласно описанному алгоритму.

Листинг 13.7. Программа входного регистратора — entranceJogger.ino

```
// Программа входного регистратора tinclude <SD.h>     // Подключение библиотеки SD
#include <Wire.h> // Для работы с RTC #include "RTClib.h" // Подключение библиотеки RTC
```

298

Часть IV. Дополнительные темы и проекты

```
// Подключение устройств SPI и I2C с контактами по умолчанию
// SD-карта SPI контакты
// RTC - стандартные I2C контакты
const int CS_PIN =10;          // SS для переходника SD
const int SD_POW_PIN =8;  // Питание для SD
const int RTC_POW_PIN =A3; // Питание для платы RTC const int RTC_GND_PIN =A2; // Земля для платы RTC
const int IR_PIN =0; // ИК-датчик расстояния к аналоговому входу A0 // Создать объект RTC RTC_DS1307 RTC;
// Инициализация переменных для даты/времени
String year, month, day, hour, minute, second, time, date;
// Инициализация переменных для ИК-датчика расстояния
int raw = 0;
int raw_prev =0;
boolean active = false;
int update_time = 0;
void setup()
{
Serial.begin(9600);
```

```
Serial.println(reinitializing Card")) ;
// Настроить контакты CS и питания как выходы pinMode(CS_PIN, OUTPUT); pinMode(SD_POW_PIN,
OUTPUT); pinMode(RTC_POW_PIN, OUTPUT); pinMode(RTC_GND_PIN, OUTPUT);
// Установка питания карты и RTC digitalWrite(SD_POW_PIN, HIGH);
digitalWrite(RTC_POW_PIN, HIGH); digitalWrite(RTC_GND_PIN, LOW);
// Инициализация Wire и RTC Wire.beginO ;
RTC.begin();
// Если RTC не запущены, загрузить дату/время с компьютера if (! RTC.isrunning())
{
Serial.println(F("RTC is NOT running!"));
RTC. adj ust (DateTime (_DATE_, _TIME__)) ;
}
// Инициализация карты SD if (!SD.begin(CS_PIN))
{
Serial.printIn(F("Card Failure")); return;
}
```
Гпава 13. Обмен данными с картами памяти SD
299
```
Serial.printIn(F("Card Ready"));
// Запись заголовка
File dataFile = SD.open("log.csv", FILE_WRITE); if (dataFile)
{
dataFile.println(F("\nNew Log Started!")); dataFile.printIn(F("Date,Time,Raw, Active")); dataFile.close();
// Запись в последовательный порт для отладки Serial.println(F("\nNew Log Started!"));
Serial.println(F("Date,Time,Raw, Active"));
}
else
{
Serial.printIn(F("Couldn't open log file"));
}
void loop()
{
// Получить значение даты и времени и перевести в строковые значения
DateTime datetime = RTC.nowO;
year = String(datetime.year(), DEC);
month = String(datetime.month(), DEC);
day = String(datetime.day(), DEC);
hour = String(datetime.hour(), DEC);
minute = String(datetime.minute(), DEC);
second = String(datetime.second(), DEC) ;
// Собрать строку текущей даты и времени date = year + "/" + month + "/" + day; time = hour + ":" + minute
+ + second;
// Собрать данные движения raw = analogRead(IR_PIN);
// При изменении значения более чем на 75 между показаниями // фиксируем факт прохода через дверь,
if (abs(raw-raw_prev) > 75) active = true; else
active = false; rawjprev = raw;
// Открыть лог-файл и записать в него, if (active || update_time == 20)
{
File dataFile = SD.open("log.csv", FILE_WRITE); if (dataFile)
{
dataFile.print(date);
```
300
Часть IV. Дополнительные темы и проекты
```
dataFile.print(F; dataFile.print(time); dataFile.print(F; dataFile.print(raw); dataFile.print(F(","));
dataFile.println(active); dataFile.close() ;
```

```
// Вывод в последовательный порт для отладки Serial.print(date);
Serial.print(F(", ") ) ;
Serial.print(time);
Serial.print(F(", ")) ;
Serial.print(raw);
Serial.print(F (", ") );
Serial.println(active);
}
else
{
Serial.println(F("Couldn't open log file"));
}
update_time = 0;
}
delay(50); update_time++ ;
}
```

13.4.3. Анализ зарегистрированных данных

После загрузки программы на плату ***Arduino*** установите регистратор возле двери и дайте поработать некоторое время. Когда наберется достаточное количество данных, вставьте SD-карту в компьютер и откройте лог-файл CSV в программе просмотра электронных таблиц. Предполагая, что на карте записаны данные для одного дня, можно построить график активности движения от времени. Пока никто не проходит через дверь, значение остается равным нулю. Когда кто-нибудь входит или выходит из комнаты, значение подскакивает до единицы, и можно точно узнать, когда это случилось. Процедура построения графика зависит от выбранного графического приложения. Я создал онлайн-таблицу, которая будет рисовать график. Чтобы воспользоваться этим сервисом, вам необходимо иметь учетную запись Google. Посетите веб-страницу http://www.exploringarduino.com/coiitent/chl3 и перейдите по ссылке на таблицу для построения графика. Вам будет предложено создать новую таблицу в своем аккаунте Google Drive. В итоге вы получите график данных, приведенный на рис. 13.24.

Проход через дверь

Рис. 13.24. Данные регистратора входа, представленные в графическом виде

Резюме_____

В этой главе вы узнали следующее:

- Что данные удобно хранить в CSV-файлах, использующих строки и запятые в качестве разделителей.
- Как отформатировать SD-карту памяти в операционных системах Windows, Mac и Linux.
- Что есть множество плат расширения SD-карт для ***Arduino***, каждая со своими особенностями.
- Как использовать ***Arduino*** библиотеку SD для записи и чтения из файла на S D-карте.
- Как с помощью RTC вставить временные метки в данные регистратора.
- Как преодолеть ограничения оперативной памяти за счет хранения строк во флэш-памяти.
- Как обнаружить движение по изменению аналогового значения, полученного с ИК-датчика расстояния.
- Как построить на компьютере график данных от регистратора, используя программу просмотра электронных таблиц.

14

Подключение ***Arduino*** к Интернету

Список деталей

Для повторения примеров главы вам понадобятся следующие детали:

- плата ***Arduino*** (рекомендуется ***Uno***);
- USB-кабель для программирования платы ***Arduino***;
- плата расширения Ethernet shield;
- фоторезистор;
- датчик температуры TMP36;

- RGB-светодиод;
- 1 резистор номиналом 10 кОм;
- 1 резистор номиналом 150 Ом;
- 3 резистора номиналом 220 Ом;
- динамик или пьезозуммер;
- кабель Ethernet;
- доступ к проводному маршрутизатору;
- набор перемычек;
- макетная плата.

Электронные ресурсы к главе

На странице http://www.exploringardumo.com/coiitent/chl4 можно загрузить программный код, видеоуроки и другие материалы для данной главы. Кроме того, листинги примеров можно скачать со страницы www.wiley.com/go/exploringarduino в разделе Downloads.

Что вы узнаете в этой главе

Вот она, последняя глава. Запустим плату ***Arduino*** в виртуальное пространство, подключив ее к Интернету. Доступ к Интернету — очень сложная тема, можно написать целые тома книг о лучшем способе подключения ***Arduino*** к Интернету.

Гпава 14. Подключение ***Arduino*** к Интернету

3G3

В этой главе мы рассмотрим использование платы расширения ***Arduino*** Ethernet shield для создания веб-страницы и отправки данных в Сеть. Вы узнаете о топологии Сети, о создании интернет-сайта, об использовании сторонних сервисов регистрации имен для подключения ***Arduino*** к Всемирной паутине.

14.1. Всемирная паутина, ***Arduino*** и Вы

Объяснить в одной главе, как работает Всемирная паутина, — слишком амбициозная затея, поэтому проиллюстрируем связь нашей платы ***Arduino*** с Интернетом с помощью упрощенной схемы, изображенной на рис. 14.1.

o

c;

г

<

>

V.

J

Рис. 14.1. Упрощенная схема взаимодействия Интернета и локальной сети

Если вы работаете только в локальной сети, то можете связаться с платой ***Arduino*** через веб-браузер компьютера, находящегося в той же локальной сети. Через маршрутизатор можно получить доступ к вашей плате ***Arduino*** из любой точки мира (или, по крайней мере, отовсюду, где есть доступ к Интернету).

14.1.1. Сетевые термины

Прежде чем подключить ***Arduino*** к сети, рассмотрим термины, которые встретятся нам на протяжении этой главы.

IP-адрес

IP-адрес представляет собой уникальный адрес, который идентифицирует каждое устройство, подключенное к сети. При работе в локальной сети есть на самом деле

304

Часть IV. Дополнительные темы и проекты

два вида IP-адресов: IP-адреса внутренней сети и глобальный IP-адрес. Если в вашем доме или офисе установлен маршрутизатор (см. рис. 14.1), то каждое устройство в пределах локальной сети обладает локальным IP-адресом, который виден всем устройствам вашей сети. Маршрутизатор/модем имеет один глобальный IP-адрес, который виден всему Интернету. Если вы хотите установить связь через Интернет между компьютером в другом месте и устройством, подключенным к маршрутизатору, необходимо преобразование сетевых адресов (NAT).

MAC-адрес

MAC-адреса, в отличие от IP-адресов, уникальны в пределах всей Сети (на практике они часто таковыми не являются). MAC-адреса назначаются каждому сетевому физическому устройству и не меняются. Например, когда вы покупаете компьютер, уникальные MAC-адреса присвоены и внутреннему модулю Wi-Fi, и адаптеру Ethernet. Это позволяет по MAC-адресу идентифицировать физические устройства в Сети.

HTML

HTML, или язык гипертекстовой разметки, является языком разметки документов в Интернете. Для отображения веб-страницы с вашего ***Arduino*** напишем простой HTML-документ, который создает кнопки и ползунки для отправки данных.

HTTP

HTTP, или протокол передачи гипертекста, определяет протокол для связи через World Wide Web (Всемирную паутину) и используется в браузерах. HTTP задает информацию заголовка, которая передается в виде части сообщения. Этот заголовок определяет, какая веб-страница будет отображаться и подтверждает успешное получение данных.

GET/POST

GET и POST определяют два способа передачи информации на удаленный вебсервер. Если вам встретится URL, который выглядит как www.jeremyblum. com/?s=arduino, то это запрос GET. GET определяет ряд переменных, следующих за вопросительным знаком в URL. В данном случае передается переменная s со значением arduino. Когда страница получает этот URL, он идентифицирует переменную s, выполняет поиск и возвращает страницу результатов. POST очень похож, но информация не видна непосредственно в URL. Вместо этого, данные передаются в фоновом режиме. POST обычно используется, чтобы скрыть конфиденциальную информацию.

DHCP

DHCP, или протокол динамической конфигурации узла, подключает устройства к локальной сети за один шаг. Чаще всего при подключении к Wi-Fi или проводной сети у вас на устройстве не установлен вручную IP-адрес, по которому маршру-

тизатор может подключаться к вам. Как же маршрутизатор отправляет пакеты на сетевое устройство? При подключении к сети маршрутизатор инициирует запрос DHCP, что позволяет маршрутизатору динамически назначать вам доступный IP-адрес. В результате настройка сети оказывается гораздо проще, потому что не нужно заботиться о конфигурации сети для подключения к ней. Однако программа для ***Arduino*** усложняется, потому что следует выяснить, какой IP-адрес был присвоен.

DNS

Аббревиатура DNS расшифровывается как Domain Name System (система доменных имен). Каждый веб-сайт, который расположен на сервере в сети Интернет, имеет уникальный IP-адрес. При вводе в www.google.com DNS-сервер смотрит на список, который информирует его о IP-адресе, связанном с этим URL. Затем он передает IP-адрес обратно в браузер вашего компьютера, чтобы браузер мог общаться с сервером Google. DNS позволяет вводить понятные имена вместо запоминания IP-адреса всех ваших любимых веб-сайтов. Можно образно сравнить DNS для вебсайтов со списком контактов в телефоне для телефонных номеров.

Клиенты и серверы

В этой главе вы узнаете, как использовать ***Arduino*** с платой расширения Ethernet shield в качестве сервера и клиента. Все устройства, подключенные к Интернету, выступают в роли сервера или клиента (хотя на самом деле часто исполняют обе роли). Сервер выдает информацию по запросу от запрашивающего компьютера, поступающему по сети. Это может быть запрос веб-страницы, данных из информационной базы, электронной почты и многого другого. Клиент— это устройство, которое запрашивает данные и получает ответ. При подключении компьютера к Интернету веб-браузер вашего компьютера выступает в качестве клиента.

Подключение к сети платы ***Arduino***

Для всех примеров, приведенных в этой главе, необходимо использовать плату ***Arduino*** в паре с фирменным Ethemet-адаптером. Есть несколько версий этого адаптера, наши примеры протестированы на самой последней версии с контроллером WIZnet Ethernet. На более старых адаптерах установлен другой чип, что не гарантирует работу описанных примеров. Для экспериментов вы можете также взять плату ***Arduino*** Ethernet, которая содержит встроенный интерфейс Ethernet.

СОВЕТ
Я рекомендую использовать внешний Ethernet-адаптер, т. к. он работает устойчивее, чем плата ***Arduino*** Ethernet со встроенным интерфейсом.
Прикрепите Ethernet-адаптер к вашей плате ***Arduino*** и подключите порт Ethernet адаптера к свободному порту Ethernet на вашем домашнем маршрутизаторе с помощью кабеля Ethernet. Подойдет обычный Ethernet-кабель типа "crossover". Соедините плату ***Arduino*** кабелем USB с вашим компьютером для программирования.

Если маршрутизатор находится далеко от компьютера, используемого для программирования, сначала запрограммируйте плату, а затем подключайте ее к маршрутизатору. Тем не менее, в некоторых примерах потребуется вывод информации, необходимой для отладки, в монитор последовательного порта. Чтобы не быть привязанным к компьютеру, можно подключить ЖК-дисплей для отображения IP-адреса, который в рассмотренных примерах выводится в последовательный терминал. Здесь пригодятся сведения из главы 10.

14.2. Управление платой ***Arduino*** из Интернета

Сначала используем плату ***Arduino*** в качестве веб-сервера. С помощью некоторых HTML-форм и интегрированной библиотеки Ethernet настроим ***Arduino*** для автоматического подключения к Сети и выведем веб-страницу, с помощью которой получим доступ для управления некоторыми контактами платы ***Arduino***. Мы разместим кнопки на веб-интерфейсе для переключения цвета RGB-светодиода и управления частотой звука. Наша программа будет расширяемой, что позволит добавлять дополнительные устройства, чтобы более комфортно работать с ***Arduino***.

14.2.1. Настройка оборудования для управления вводом-выводом

Настроим оборудование для управления платой ***Arduino*** в качестве сервера для управления контактами платы из сети Интернет. Для этого примера возьмем RGB-светодиод и пьезозуммер или обычный динамик. Схема соединения элементов показана на рис. 14.2. Напомним, что контакты 4, 10, 11, 12 и 13 предназначены для связи платы ***Arduino*** с адаптером Ethernet. Мы подключаем RGB-светодиод к выводам 5, 6, 7, а динамик — к выводу 3 платы ***Arduino***.

14.2.2. Создание простой веб-страницы

Разработать веб-страницу можно независимо от ***Arduino***. На нашей веб-странице будут простые кнопки для переключения цвета RGB-светодиода и ползунок для регулировки частоты звука, воспроизводимого динамиком. Веб-страница будет отправлять команды из браузера на сервер по протоколу HTTP GET. При разработке дизайна страницы веб-сайт может не быть подключенным к серверу ***Arduino***, на этом этапе не нужно, чтобы взаимодействие со страницей вызывало какие-нибудь действия на плате ***Arduino***.
Откройте свой любимый текстовый редактор (я рекомендую Notepad++ в Windows, который осуществляет визуальное цветовое выделение кода) и создайте новый файл с расширением html. Название не имеет значения, например test.html. Это будет пустая страница, без тегов, которые обычно присутствуют на веб-страницах, например <body> и <header>. Но это не повлияет на отображение страницы в браузере. Поместите в HTML-файл код из листинга 14.1.

Рис. 14.2. Схема подключения к ***Arduino***-серверу RGB-светодиода и динамика
Листинг 14.1. HTML-страница с формой — server_form.html

```
<form action*" method='get'>
cinput type*'hidden' name='L' value*'7' /> cinput type*'submit' value*'Toggle Red' />
</form>
cform action*" method*'get'>
<input type*'hidden' name='L' value='6' />
<input type='submit' value='Toggle Green' />
</form>
<form action=" method='get'>
cinput type='hidden' name='L' value='5' /> cinput type='submit' value='Toggle Blue' /> c/form>
cform action*" method*'get'>
cinput type*'range' name*'S' min*'O' max*'1000' step*'100' value*'0'/> cinput type*'submit' value*'Set
```

Frequency' />
</form>
HTML-страница из листинга 14.1 содержит четыре элемента формы. Тег cform> задает начало формы, тег c/form> — конец формы. Каждый элемент формы с тегом

<input> определяет, какие данные будут переданы на сервер при отправке формы. В нашем примере переменная называется l и на сервер передается номер необходимого вывода RGB-светодиода. Пустой элемент action указывает, что страница перезагружается при отправке формы. Значение скрытых полей (input с типом hidden) будут отправлены на сервер при нажатии кнопки submit. Для выбора частоты мы добавляем новый HTML-элемент (input с типом range). Это позволяет сделать ползунок диапазона. Вы можете перемещать ползунок с шагом 100 для выбора частоты, которая будет передаваться в переменной с именем s. Старые браузеры не поддерживают этот элемент (слайдер) и он будет отображаться как поле ввода. Чтобы увидеть, как будет выглядеть страница, запустите ваш браузер (я рекомендую Google Chrome). В Chrome нужно нажать комбинацию клавиш <Ctrl>+<0> (в Windows) или <Cmd>+<0> (в OS X), чтобы вызвать диалоговое окно Открыть. Откройте созданный HTML-файл (рис. 14.3).
| Toggle Red j
I Toggle Green |
\ Toggle Blue]
- (Set Frequency]
Рис. 14.3. Вид нашей веб-страницы в браузере Chrome
При нажатии на любую из кнопок вы должны увидеть в адресной строке браузера переменные, передаваемые методом GET.
14.2.3. Написание программы для ***Arduino***-сервера
Теперь нам необходимо созданный HTML-код веб-страницы интегрировать в программу ***Arduino***-сервера, который будет обрабатывать подключение к сети, выдавать на запросы клиентов страницу и обрабатывать переменные, получаемые методом GET.
Подключение к сети и получение IP-адреса через DHCP
Благодаря механизму DHCP подключить к сети плату ***Arduino*** с Ethernet-адаптером очень просто. Перед обсуждением кода программы давайте посмотрим, как это будет происходить. Вначале нужно подключить библиотеки SPI и Ethernet, определить MAC-адрес вашего Ethemet-адаптера (он указан на этикетке) и создать экземпляр объекта EthernetServer. В функции setup () ВЫ начинаете сеанс Ethernet с MAC-адреса, который вы определили, и запускаете веб-сервер. При необходимости можно назначить статический IP-адрес при инициировании Ethernet, но если этот аргумент пуст, то ***Arduino*** будет получать IP-адрес через DCHP и выведет назначенный IP-адрес в последовательный терминал. Затем этот IP-адрес потребуется для подключения к ***Arduino*** и просмотра веб-страницы.

Ответ на клиентский запрос
В основном цикле loop o происходит несколько действий. Для обработки различных действий проверяются переменные состояния, чтобы отслеживать, что было выполнено и что еще должно произойти для успешной связи с клиентом. Плата ***Arduino*** постоянно будет ожидать клиентских подключений к серверу, например с вашего ноутбука. При подключении клиента ответ содержит две вещи: HTTP-заголовок ответа и тип форматирования веб-страницы, выдаваемой по запросу. Заголовок ответа, отправляемый в браузер, сообщает, какая информация будет отправлена на запрос. Когда вы пытаетесь посетить несуществующую страницу, получаете "страшный" ответ: "404 Responce". Заголовок 404 указывает браузеру, что сервер не может найти запрашиваемую страницу. Ответ "200 Response" указывает, что запрос был получен и что HTML-данные будут передаваться в браузер. Если вы хотите послать сообщение "200 Response" в браузер, а затем передать HTML-данные, то заголовок будет выглядеть так:
HTTP/1.1 200 OK Content-Type: text/html
После заголовка должно следовать содержимое вашей HTML-страницы. Программа также обрабатывает GET-запросы. Чтобы определить GET-запрос, нужно искать символ ? в URL. Если найден символ ?, программа ждет, пока не получит имя переменной. Команда для управления

светодиодом — это ь, команда для регулировки частоты динамика — s. Программа анализирует значения переменных и управляет сигналами на выводах ***Arduino***. После этого следует команда break для выхода из цикла подключенного клиента, и программа начинает ожидать новое подключение клиента, чтобы повторить весь процесс сначала.

Итоговая программа веб-сервера

Теперь, учитывая все сказанное ранее, мы можем написать код программы вебсервера ***Arduino***. Программа довольно нетривиальная, потому что содержит несколько переменных состояния, которые позволяют отслеживать взаимодействие между клиентом и сервером. Код, приведенный в листинге 14.2, управляет RGB-светодиодом и динамиком. В программу легко добавить функции с большим числом GET-переменных. Места, куда вы можете вставить дополнительную функциональность, указаны в комментариях листинга.

Листинг 14.2. Код веб-сервера — control_LED_speaker.ino

```
// ***Arduino*** веб-сервер
// Часть адаптированного кода под лицензией MIT License от // http://bildr.org/2011/06/arduino-ethernet-pin-control/ #include <Ethernet.h>
#include <SPI.h> const int BLUE =5; const int GREEN =6;
```



```
const int RED =7; const int SPEAKER =3;
// Для управления RGB-светодиодом и динамиком
// Если вы хотите использовать дополнительные устройства,
// добавляйте переменные сюда int freq = 0; int pin;
// MAC-адрес устройства
// Должен быть написан на этикетке вашего устройства // или можете использовать этот
byte шас[] = { 0x90, 0xA2, OxDA, 0x00, 0x4A, OxEO };
// Сервер на порту 80
EthernetServer server = EthernetServer(80); //Порт 80 boolean receiving = false; // Отслеживание данных GET
void setup()
{
Serial.begin(9600); pinMode(RED, OUTPUT); pinMode(GREEN, OUTPUT); pinMode(BLUE, OUTPUT);
// Соединение с DHCP if ((Ethernet.begin(mac))
{
Serial.printIn("Could not Configure Ethernet with DHCP."); return;
}
else
{
Serial.println("Ethernet Configured!");
}
// Запуск сервера server.begin();
Serial.print("Server Started.\nLocal IP: ");
Serial, println (Ethernet. locallPO );
}
void loop()
{
EthernetClient client = server.available(); if (client)
{
// Запрос HTTP заканчивается пустой строкой boolean currentLinelsBlank = true; boolean sentHeader = false;
```



```
while (client.connected())
{
if (client.available())
{
```

```
char c = client.read(); // Чтение из буфера
if(receiving && c == ' ') receiving = false; // Окончание
// передачи
if(c == '?') receiving = true; // Поиск символа ?
// Ожидание данных GET if(receiving)
{
// Данные управления светодиодом (переменная L) if (c == ' L')
{
Serial.print("Toggling Pin "); pin = client.parselnt();
Serial.println(pin);
digitalWrite(pin, !digitalRead(pin));
break;
}
// Команда управления динамиком (переменная S) else if (c == ’S')
{
Serial.print("Setting Frequency to "); freq = client.parselnt();
Serial.println(freq); if (freq == 0) noTone(SPEAKER); else
tone(SPEAKER, freq); break;
}
// Код для управления дополнительными устройствами // добавляем сюда
}
// Отправка заголовка ответа if(!sentHeader)
{
// Отправить стандартный заголовок HTTP ответа
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html\n");
// Кнопка переключения красного для RGB
client .println ("<form action^' method='get '>") ;
client.println("cinput type='hidden' name='L' value=,7' />");
client.println("cinput type='submit' value='Toggle Red' />");
client.println("</form>");
```



```
// Кнопка переключения зеленого для RGB
client.println("<form action^" method='get'>");
client.println("<input type='hidden' name='L' value='6' />");
client.println("cinput type='submit' value='Toggle Green' />");
client.printIn("</form>");
// Кнопка переключения синего для RGB
client.println("<form action=" method='get'>");
client.println("<input type='hidden' name='L' value='5' />");
client.println("cinput type='submit' value='Toggle Blue' />");
client.println("</form>");
// Ползунок для выбора частоты динамика
client.println("<form action=" method='get'>");
client.print("cinput type='range' name='S' min='0' max='1000'"
"step='100' value='0'/>");
client.println("<input type='submit' value='Set Freq.' />"); client.println("</form>");
// Добавить формы для управления // дополнительными устройствами sentHeader = true;
}
if (c == '\n' && currentLinelsBlank) break; if (c == '\n')
{
currentLinelsBlank = true;
}
else if (c != '\r')
```

```
{
currentLinelsBlank = false;
}
}
}
delay(5);        // Дать время веб-браузеру на получение данных
client.stop(); // Закрыть соединение
}
}
```

Приведенная программа выполняет все функции, описанные в предыдущих разделах. Измените MAC-адрес на значение, указанное на этикетке вашего Ethemet-адаптера. Если вы не найдете этот адрес, возьмите значение из листинга 14.2. Загрузите программу на плату ***Arduino*** и запустите монитор последовательного порта. Убедитесь, что плата ***Arduino*** подключена к сети и в маршрутизаторе включен DHCP. Через несколько секунд в мониторе последовательного порта появится сообщение о назначенном IP-адресе (рис. 14.4).

В случае, показанном на рис. 14.4, ***Arduino*** был назначен локальный IP-адрес 192.168.0.9. В вашей сети почти наверняка будет другой адрес. Зная IP-адрес, можно использовать его для доступа к веб-интерфейсу.

Рис. 14.4. Вывод в последовательный порт полученного по DHCP IP-адреса

14.3. Управление платой ***Arduino*** по сети

Теперь, когда код сервера работает, и ваша плата ***Arduino*** подключена к сети с действительным IP-адресом, вы можете с помощью браузера получить к ней доступ и осуществлять управление. Сначала сделаем это по локальной сети, а затем воспользуемся переадресацией портов на вашем роутере для доступа к плате ***Arduino*** из внешней сети.

14.3.1. Управление платой ***Arduino*** по локальной сети

Чтобы убедиться, что веб-интерфейс работает должным образом, проверьте, что ваш компьютер подключен к той же сети, что и плата ***Arduino*** (через Wi-Fi или Ethernet). Откройте ваш браузер и введите в адресную строку IP-адрес из предыдущего раздела. В результате должна открыться HTML-страница, созданная ранее. Попробуйте нажать кнопки для включения/выключения цветов RGB-светодиода. Переместите ползунок и нажмите кнопку установки частоты динамика. Плата ***Arduino*** должна отреагировать на ваши действия. Если открыть монитор последовательного порта ***Arduino*** IDE, вы увидите вывод в него отладочной информации. Обратите внимание, что переменные передаются на сервер ***Arduino*** методом GET через адресную строку браузера (рис. 14.5).

Если управление платой ***Arduino*** по локальной сети проходит успешно, можно перейти к следующему разделу, чтобы управлять платой ***Arduino*** из любой точки мира.

Рис. 14.5. Управление платой ***Arduino*** через веб-интерфейс и вывод отладочной информации в последовательный порт

ПРИМЕЧАНИЕ

Видеоурок, демонстрирующий управление платой ***Arduino*** по локальной сети, можно посмотреть на странице http://www.exploringarduino.com/content/ch14. Этот видеофайл доступен также на сайте издательства Wiley.

14.3.2. Организация доступа к плате ***Arduino*** из внешней сети

В предыдущем разделе мы научились управлять платой ***Arduino*** из любой точки локальной сети, поскольку IP-адрес платы находился за маршрутизатором. Если требуется связать плату ***Arduino*** с компьютерами за пределами вашей локальной сети, следует воспользоваться передовыми технологиями, позволяющими получить доступ к устройству через ваш маршрутизатор из внешней сети.

Чтобы сделать это, нужно осуществить три действия:

1. Зарезервировать локальный адрес DHCP, используемый платой ***Arduino***.

2. Перенаправить внешний порт на маршрутизаторе к внутреннему порту платы ***Arduino***.
3. Подключить маршрутизатор к службе DNS.

ВНИМАНИЕ!

Последовательность действий зависит от типа маршрутизатора. Я буду опускать подробности, т. к. предполагаю наличие у вас некоторых знаний по администрированию маршрутизатора. Подробные указания для конкретного маршрутизатора можно отыскать в Интернете. Если это ваш первый опыт администрирования маршрутизатора, то можно нарушить настройки домашней сети. Некоторые маршрутизаторы даже не поддерживают все функции, необходимые для включения переадресации портов и динамического обновления DNS. Если вы совсем не знакомы с сетевым администрированием, ограничьтесь доступом к плате ***Arduino*** из локальной сети.

Вход в панель администрирования маршрутизатора

Сначала необходимо войти в панель администрирования маршрутизатора. URL-адресом панели администрирования является IP-адрес шлюза для вашей сети. Почти во всех конфигурациях домашних сетей это первые три значения локального IP-адреса платы ***Arduino***, разделенные точками и дополненные единицей. Если, например, IP-адрес вашей платы ***Arduino*** равен 192.168.0.9, то ваш адрес шлюза, скорее всего (но не обязательно), будет 192.168.0.1. Попробуйте ввести этот адрес в адресную строку браузера, чтобы увидеть окно входа в систему. Введите учетные данные для входа к странице администрирования маршрутизатора (это не данные для беспроводного соединения). Если вы никогда не меняли значения, заданные по умолчанию, то можете найти их в руководстве по настройке маршрутизатора.

Если этот IP-адрес не работает, придется назначить его вручную. В Windows можно вызвать командную строку и ввести в нее команду ipconfig. Используйте адреса шлюза по умолчанию для активного сетевого подключения. В ОС Mac выберите в меню Системные настройки пункт Сеть, нажмите кнопку Дополнительно, перейдите на вкладку TCP/IP и используйте значение адреса маршрутизатора. Если вы находитесь в Linux, откройте терминал, наберите route -n и задайте в качестве адреса маршрутизатора последний Gateway, отличный от нуля.

Резервирование IP-адреса для ***Arduino*** в DHCP

После того как вы окажетесь в консоли администратора маршрутизатора, необходимо зарезервировать адрес DHCP. При этом гарантируется, что каждый раз при подключении к маршрутизатору устройства с определенным MAC-адресом ему будет присвоен одинаковый локальный IP-адрес. При резервировании IP-адреса DHCP никогда не отдаст этот адрес клиентам с MAC-адресом, отличным от указанного, даже если клиент с зарезервированным адресом в настоящее время не подключен к маршрутизатору. Резервируя в DHCP IP-адрес платы ***Arduino***, вы всегда будете иметь возможность направить трафик к нему на следующем шаге.

Как только вы найдете эту опцию, зарезервируйте текущий IP-адрес платы ***Arduino***, назначив ее MAC-адрес, установленный в предыдущей программе. Для применения настройки необходима перезагрузка маршрутизатора. Убедитесь в правильности настройки, перезагрузив маршрутизатор и посмотрев, что ***Arduino*** получает тот же IP-адрес после восстановления.

ПРИМЕЧАНИЕ

Аналогичного эффекта можно добиться, назначив плате ***Arduino*** статический IP (не используя DHCP) в программе. Узнать, как это сделать, вы можете на странице http://arduino.cc/en/Reference/EthernetIPAddress.

Перенаправление порта 80 на плату ***Arduino***

Теперь, когда у вас есть неизменный локальный IP-адрес для платы ***Arduino***, необходимо перенаправить входящий интернет-трафик на внутренний IP-адрес. Port Forwarding — это прослушивание трафика на определенном порту маршрутизатора и перенаправление его на определенный внутренний IP-адрес. Порт 80 по умолчанию предназначен для HTTP-связи. Найдите пункт Port Forwarding в панели администрирования маршрутизатора и назначьте перенаправление внешнего порта 80 на внутренний порт 80, на IP-адрес вашей платы ***Arduino***. Если маршрутизатор определяет диапазон для портов, просто выберите опцию 80-80. Теперь весь трафик к маршрутизатору через порт 80 будет идти на вашу плату ***Arduino***.

Обновление динамического DNS

Последний шаг— получение доступа к маршрутизатору из сети Интернет. Возможно, у вас задан

статический глобальный IP-адрес. Это бывает довольно редко для домашних интернет-соединений, но все же иногда встречается. Если вы знаете статический IP, то можете получить доступ к этому IP-адресу из любой точки мира и направить трафик на плату ***Arduino***. Можно даже приобрести доменное имя и настроить серверы DNS на ваш 1P-адрес.

Но может оказаться, что у вас динамический глобальный IP-адрес. Провайдер, скорее всего, меняет ваш IP-адрес один раз в несколько дней или недель. Таким образом, даже если удастся выяснить ваш глобальный IP-адрес на сегодняшний день и получить доступ к ***Arduino*** через этот IP-адрес, он может перестать работать завтра. Способ решения этой проблемы — использование динамических служб IP. Эти службы запускают небольшую программу на маршрутизаторе, которая периодически проверяет ваш глобальный IP-адрес и сообщает его на удаленный веб-сервер. Веб-сервер обновляет ваш поддомен (например, myarduino.dyndns.org), и вы всегда попадаете на глобальный IP-адрес маршрутизатора, даже если он изменяется. DynDNS — это сервис, программное обеспечение которого встроено в большинство современных маршрутизаторов. Поищите на странице администрирования маршрутизатора, какой сервис DynDNS он поддерживает. Некоторые из них бесплатны, некоторые взимают ежегодную плату. Вы можете следовать инструкциям по установке в панели администрирования маршрутизатора, чтобы создать учетную запись с одной из этих служб и подключить его к маршрутизатору. В итоге вы сможете получить доступ к ***Arduino*** удаленно, даже с динамически меняющимся

глобальным IP-адресом. Если ваш маршрутизатор не поддерживает услуги DynDNS, имейте в виду, что некоторые сервисы предлагают услуги DynDNS клиентам, которые будут работать на компьютерах в вашей сети, а не на маршрутизаторе.

Как только будет определен ваш публичный IP-адрес (или динамически обновляемый URL), его можно указать в вашем браузере и подключение к ***Arduino*** должно заработать. Дайте этот адрес другу, чтобы проверить функционирование удаленно.

14.4. Отправка данных в реальном времени в графические сервисы

В предыдущем разделе вы узнали, как превратить плату ***Arduino*** в веб-сервер, который предоставляет интерфейс для управления контактами платы по локальной сети или через Интернет. Существует еще одна распространенная причина для подключения ***Arduino*** к Интернету— создание сети датчиков. Поскольку датчики обычно только передают информацию, а не принимают команды, ***Arduino*** будет инициализировать запрос к известному ресурсу в Интернете (через онлайн-сервис графиков) и вам не придется заботиться о переадресации IP-адресов, резервировании IP-адреса и т. п.

В этом разделе используется веб-ресурс Xively (ранее Cosm) для создания графиков данных, получаемых с платы ***Arduino***.

14.4.1. Создание потока данных на Xively

В рассматриваемом примере мы с помощью веб-сервиса Xively через Интернет будем строить графики данных, получаемых платой ***Arduino*** от датчиков. При подключении к сайту Xively вы будете освобождены от большей части нетворческой работы, которую приходится делать для отображения данных в Интернете.

Создание учетной записи Xively

Для начала, посетите страницу http://www.xively.com и создайте бесплатный аккаунт. Перейдите по ссылке, содержащейся в электронном письме с подтверждением, и вы сможете войти на сайт.

Создание потока данных

Как только аккаунт будет создан, нажмите кнопку Develop в верхней части страницы, чтобы создать канал. Затем нажмите кнопку Add Device. В окне (рис. 14.6) вам будет предложено назвать свой канал и добавить его описание. Вы также можете сделать ваш канал публичным или приватным.

После ввода необходимых данных нажмите кнопку Add Device. Появится страница с сообщением о созданном новом потоке. Оставьте страницу открытой, т. к. эта информация пригодится нам в дальнейшем при настройке программы для ***Arduino***.

DEVELOP MANAGE
DEVELOPER CENTER

< > Add Device
The Xivef-y Developer Workbench w® help you ю get your devices, sppticetems and services ttlfctfig «o each other through X/ve!y. The hrst step fs to create a development device. Begin fey providing some has* information.
Device Name
My ***Arduino***
Device Description
This is a super awesome ***Arduino*** Project that I'm making with the netp of Jeremy Shim s "Exploring ArcMno" 8ook?
Privacy
Private Device
v№!®f ЛЯ №yt ta«wb3ei> -fatvl iKut.veif thafspi device's
® Public Device
Viyj ro voire 4 Jr-v.-ss' s tteia under :n" CCO 10 <Mlv*r#*i license.
✓ Add Device
Cancel
Рис. 14.6. Добавление устройства в Xively

Установка библиотек Xively и HTTPCIient

Xively предоставляет для ***Arduino*** удобную библиотеку, облегчающую доступ через Интернет. Библиотеки Xively и HttpClient связаны друг с другом, поэтому их необходимо скачать с сайта GitHub. Посетите следующие две ссылки: https:// github.com/xively/xively-***Arduino*** и https://github.com/amcewen/HttpClient и нажмите кнопку Download ZIP для скачивания архивов. Ссылки для скачивания библиотек можно найти на веб-странице http://www.exploringarduino.com/content/chl4.

Скачав файлы, выполните следующие действия:

1. Распакуйте файлы и переименуйте папки библиотек так, чтобы они не содержали символа тире (GitHub добавляет тире к именам папок автоматически). Я рекомендую переименовать папку HttpClient-master в HttpClient, а папку Xively-***Arduino***-master в xively.
2. Переместите эти папки в каталог библиотек ***Arduino***, как было описано в главе 12.
3. Откройте ***Arduino*** IDE (если программа была открыта при копировании библиотек, необходима перезагрузка компьютера) и перейдите в меню Файл -> При-

меры. В списке должны появиться папки HttpClient и xively. Значит, библиотеки были успешно установлены.

Для первого эксперимента с Xively рассмотрим пример публикации в Сети состояния одного аналогового датчика. В ***Arduino*** IDE откроем пример DatastreamUpload из папки xively. Так как мы будем модифицировать код примера, создадим новый проект, используя в меню Файл опцию Сохранить как. В примере идет передача аналогового значения датчика, подсоединенного к контакту 2:

// Analog pin which we're monitoring (0 and 1 are used by the // Ethernet shield) int sensorPin = 2;

В следующем разделе подключим аналоговый датчик к плате ***Arduino*** с установленным адаптером Ethernet.

Подключение к плате ***Arduino***

Подсоединим датчик к аналоговому контакту 2 платы ***Arduino***. Пример иллюстрирует чтение аналогового входа 2 и отправку данных в ваш аккаунт на Xively. Возьмем фоторезистор и резистор номиналом 10 кОм и подключим их к аналоговому входу 2 в качестве делителя напряжения (рис. 14.7), как вы это делали в главе 3. Затем подключим плату ***Arduino*** к компьютеру и к Сети.

Рис. 14.7. Подключение фоторезистора к плате ***Arduino*** с установленным адаптером Ethernet

Настройка Xively и выполнение программы

Вы уже установили соответствующие библиотеки и открыли проект с примером. Его нужно настроить, скомпилировать и запустить на вашей плате ***Arduino***. Сначала настроим программу в соответствии с параметрами устройства в нашем аккаунте Xively. Вам нужно изменить значение трех параметров:

MAC-адрес вашего адаптера Ethernet, xively api key и Feed id. MAC-адрес будет таким же, как и в предыдущих примерах, xively api key и Feed id можно найти в вашем аккаунте Xively в разделе API Keys (рис. 14.8).

API Keys----------

Auto-generated My ***Arduino*** device key for feed 1242622121

qkjXS1oUKqbCG-hqh3fw4Wls<JvOSAKx4ZXZYSWhGUWdxczGg permi«ion* READ.UPDATE,CREATE.DELETE

private access?

+ Add Key

Рис. 14.8. Параметры Xiveli API key и Feed ID

В этом разделе вы найдете Feed id (первый параметр) и xively api key (второй параметр) для вставки в код программы. Далее приведены строки, которые нужно обновить соответствующими значениями.

Замените MAC-адрес:

```
// MAC-адрес вашего адаптера Ethernet
byte mac[] = { OxDE, OxAD, OxBE, OxEF, OxFE, OxED };
```

Замените xively api Key своим значением:

```
// Ваш Xively API key
char xivelyKey[] = "YOUR_XIVELY_API_KEY";
```

Замените Feed id (15552 в примере) своим значением:

```
// Feed ID
XivelyFeed feed(15552, datastreams, 1 /* номер потока */);
```

В листинге 14.3 приведен полный код программы.

Листинг 14.3. Загрузка потока данных в Xively — xively.ino

```
#include <SPI.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>
// MAC-адрес Ethernet-адаптера
byte mac[] = { 0x90, 0xA2, OxDA, 0x00, 0x4A, OxEO };
```



```
// Ваш Xively API key
char xivelyKey[] = "qkjXSloUKqbCG-hqh3fw4WIsdvOSAKx4ZXZYSWhGUWdxczOg";
// Аналоговый контакт для подключения датчика int sensorPin = 2;
// Строка для идентификаторов потока char sensorId[] = "sensor_reading";
XivelyDatastream datastreams[] = {
XivelyDatastream(sensorId, strlen(sensorld), DATASTREAM_FLOAT),
};
// Feed ID
XivelyFeed feed(1242622121, datastreams, 1 /* number of datastreams */); EthernetClient client;
XivelyClient xivelyclient(client);
void setup()
{
// Запуск последовательного порта Serial.begin(9600);
Serial.printIn("Starting single datastream upload to Xively...");
Serial.println();
while (Ethernet.begin(mac) !== 1)
{
Serial.printIn("Error getting IP address via DHCP, trying again..."); delay(15000);
}
}
void loop<)
{
int sensorValue = analogRead(sensorPin); datastreams[0].setFloat(sensorValue);
Serial.print("Read sensor value ");
```

```
Serial.println(datastreams[0].getFloat());
Serial.println("Uploading it to Xively"); int ret = xivelyclient.put(feed, xivelyKey);
Serial.print("xivelyclient.put returned ");
Serial.println(ret);
Serial.println(); delay(15000);
}
```

Загрузите программу в плату ***Arduino***, и все будет готово к передаче данных. При первом подключении ***Arduino*** сервер Xively автоматически добавляет выдачу данных на веб-странице, открытой вами ранее. В программе вы создаете объект, который содержит всю информацию о вашем потоке. Данные представлены в виде массива datastreams [ ], содержащего имя дат-

чика и тип значений (в данном случае float). Объект xivelyFeed содержит идентификатор потока, информацию о потоке данных и количестве потоков данных, которые находятся в массиве.

Отображение данных на веб-странице

Данные будут передаваться сразу после загрузки программы в плату ***Arduino***. Если вы не видите возвращенный статус "200" в мониторе последовательного порта, то, вероятно, ошиблись при копировании параметров xively api key или Feed id. Проверьте эти значения и повторите попытку. Если все работает, вернитесь на сайт Xively, поток данных sensor_reading теперь должен автоматически обновляться каждые 15 с. Нажмите на ссылку sensorjreading, и вы увидите график значений, поступающих от фоторезистора. Через некоторое время график может выглядеть примерно так, как показано на рис. 14.9. Вот и все, что нужно сделать. Плата ***Arduino*** будет продолжать отправлять данные и обновлять информацию на сервере Xively.

Channels
updatesfew seconds tgo
Request Log
II Pause
sensor_reading
$ IK* & COM9
679.00
:; - urj&stv v « fey* seconds «50
00: JO
Loci
Ilf A.
Tags
Read sensor value 728.00 Cploading it to Xively xivelycllent.put returned 200
Reed sen3or value 679.00 Oploading it to Xively xivelycllent.put returned 200
!/] AutoscroS
▼ 9600baud *
Description
200 PUT feed 20:39:40 2
roc put feed 20 зозз zl*
1
200 put feed 20 30.0? f
200 put feed 20.33.5i z
200 PUT feed 20 33:34 2
API Keys
Auto-generated My ***Arduino*** device key fof feed 1242622121
OtcjXS1oUK<j&CO-nqh3Jw4WlsdvOSAKx42X2VSWt!<;uwoxc!Oj perml»»ien«
0EAD.tfP0ATE.CfiEA7E.DEt.gTE
accде*
-i" Add Key
Triggers

Рис. 14.9. Отображение значений фоторезистора на сайте Xively

14.4.2. Добавление компонентов в поток

Один датчик, отображающий поток в Xively, это здорово, но может вы хотите добавить больше датчиков? К счастью, это сделать очень легко. Помимо фоторезистора, установим на плате ***Arduino*** датчик температуры. Можно добавить любой другой датчик, включая цифровые датчики I2C и SPI.
323
Глава 14. Подключение ***Arduino*** к Интернету_____
Добавление аналогового датчика температуры
Добавим простой аналоговый датчик температуры TMP36 (см. главу 3), подсоединив его к аналоговому контакту 3, как показано на рис. 14.10.
Рис. 14.10. Добавление аналогового датчика температуры TMP36
Добавление показаний датчика в поток
Теперь нужно вставить данные от нового датчика в поток данных, отправляемых на сервер Xively. По сути, в коде программы нужно просто прибавить дополнительный поток данных везде, где указан первый поток данных. Вы также можете переименовать потоки с идентификаторами datastream на более понятные, например light_reading и temp_reading. Код листинга 14.4 похож на предыдущий, но теперь у нас два потока данных. Параметры xively api key, Feed id и MAC-адрес такие же, как в предыдущей программе.
Листинг 14.4. Загрузка потока данных в Xively от нескольких датчиков— xively2.ino
#include <SPI.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>
324
Часть IV. Дополнительные темы и проекты
II MAC-адрес вашего адаптера Ethernet
byte mac[] = { 0x90, 0xA2, OxDA, 0x00, 0x4A, OxEO };
// Your Xively key to let you upload data
char xivelyKey[] = "qkjXSloUKqbCG-hqh3fw4WIsdvOSAKx4ZXZYSWhGUWdxczOg";
// Аналоговые датчики подключения устройств (0 и 1 уже задействованы // адаптером Ethernet)
int lightPin =2; // Датчик температуры int tempPin =3; // Фоторезистор
// Строки для идентификаторов потока char lightld[] = "light_reading"; char templd[] = "temp_reading";
XivelyDatastream datastreams[] = {
XivelyDatastream(lightId, strlen(lightld), DATASTREAM_FLOAT), XivelyDatastream(templd, strlen(templd), DATASTREAMJFLOAT),
};
// Feed ID
XivelyFeed feed(1242622121, datastreams, 2 /* number of datastreams */); EthernetClient client;
XivelyClient xivelyclient(client);
void setup()
{
// Запуск последовательного порта Serial.begin(9600) ;
Serial.println("Starting double datastream upload to Xively...");
Serial.println();
while (Ethernet.begin(mac) != 1)
{
Serial.println("Error getting IP address via DHCP, trying again..."); delay(15000);
}
}
void loop()
{
int lightValue = analogRead(lightPin); datastreams[0].setFloat(lightValue);
Serial.print("Read light value ");
Serial.println(datastreams[0].getFloat()); int tempValue = analogRead(tempPin); datastreams[1].setFloat(tempValue);
Serial.print("Read temp value ");
Serial.println(datastreams[1].getFloat());

Serial.println("Uploading it to Xively");
Глава 14. Подключение ***Arduino*** к Интернету
325
int ret = xivelyclient.put(feed, xivelyKey);
Serial.print("xivelyclient.put returned ");
Serial.println(ret);
`Serial.println();` delay(15000);
}
Прежде всего, заметим, что предыдущие ссылки на датчик (фоторезистор) были обновлены. Теперь, когда информация поступает от двух датчиков, их необходимо различать. Поток данных tempid[] добавлен в объект datastreams [ ]. Определение объекта xivelyFeed было обновлено, чтобы показать, что у нас не один, а два потока данных. В цикле loop о строки кода вывода данных фоторезистора продублированы для датчика температуры. Обратите внимание, что поток для фоторезистора представлен в объекте datastreams первым, и на него ссылка задана как datastreams [0]. Поток для датчика температуры в объекте datastreams представлен вторым, и ссылка на него datastreams [1]. При запуске программы на плате ***Arduino*** веб-интерфейс автоматически обновляет отображение потоков данных. Возможно, вы захотите удалить старый поток данных sensor_reading, т. к. он заменен потоком 1 ight_reading. После нескольких минут работы ***Arduino*** графики должны выглядеть примерно так, как показано на рис. 14.11.
Итак, из платы ***Arduino*** вы успешно создали как веб-сервер, так и клиент удаленной веб-службы. Попробуйте теперь добавить цифровые датчики и визуальную обратную связь, чтобы сделать вашу систему по-настоящему интерактивной.
326
Часть IV. Дополнительные темы и проекты
Резюме_____
В этой главе вы узнали следующее:
- Каков смысл параметров IP, DHCP, DNS, MAC и др.
- В чем состоят различия между клиентами и серверами.
- Как создать HTML-код для формы управления платой ***Arduino*** через Интернет.
- Как запустить веб-сервер на вашей плате ***Arduino***.
- Как управлять контактами ввода-вывода на ***Arduino*** через Интернет.
- Как подключить ***Arduino*** к серверу Xively для построения графиков.
- Как отображать онлайн-данные, поступающие от нескольких датчиков.

ПРИЛОЖЕНИЕ
Документация на микроконтроллер ATmega и схема платы ***Arduino***
Основа всех плат ***Arduino*** — микроконтроллеры Atmel. Здесь мы не будем рассматривать особенности микроконтроллеров для всех плат ***Arduino***, но приведем технический паспорт контроллера ATmega, чтобы получить представление, как он работает. Кроме того, мы опишем принципиальную схему платы ***Arduino Uno***, чтобы лучше понять, как она функционирует.
Знакомство с технической документацией_____
Один из самых важных навыков для инженера — умение читать техническое описание. Практически любой электронный компонент имеет соответствующий технический паспорт, который содержит технические характеристики, рекомендации по использованию и другую необходимую информацию.
Анализ технического описания
Рассмотрим, например, техническое описание для микроконтроллера ATmega 328P. Напомним, что микросхема ATmega 328 используется в ***Arduino Uno*** и во многих других платах. Найти технический паспорт бывает нелегко. Я рекомендую ввести в Google запрос "ATmega 328P datasheet" и выбрать из перечня ссылок PDF-файл на сайте Atmel. Техническое описание микроконтроллеров, применяемых в ***Arduino***, можно найти на сайте http://www.arduino.cc в разделах технических описаний плат. Начнем с рассмотрения первой страницы технического паспорта (рис. П.1). В большинстве случае, на первой странице говорится об особенностях данного микроконтроллера.
Даже беглый взгляд на первую страницу дает много информации о микроконтроллере. Мы видим, что микроконтроллер имеет 32 Кбайт программируемой флэш-памяти, может быть перепрограммирован примерно 10 000 раз, напряжение питания составляет от 1,8 до 5,5 В (в случае ***Arduino*** 5 В). Здесь также можно найти сведения о числе входов-выходов, поддержке специальных интерфейсов (SPI,

I2C), разрядности аналого-цифрового преобразователя (АЦП).
328
Приложение
Features
• HIGH Performance, Low Power AVR® 8-Bit Microcontroller
• Advanced RISC Architecture
- 131 Powerful Instructions - Most Single Clock Cycle Execution
- 32 x 8 General Purpose Working Registers
- Fully Static Operation
- Up to 20 MIPS Throughput at 20 MHz
- On-chip 2-cycle Multiplier
• HIGH Endurance Non-volatile Memory Segments
- 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
- 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
- 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
- Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
- Data retention: 20 years at 85 C/100 years at 25 C'1*
- Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program True Read-While-Write Operation
- Programming Lock for Software Security
• Peripheral Features
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator
- Six PWM Channels
- 8-channel 10-bit ADC in TQFP and QFN/MLF package
Temperature Measurement
- 6-channel 10-bit ADC in PDIP Package
Temperature Measurement
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (Philips l2C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change
• Special Microcontroller Features
- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated Oscillator
- External and Internal Interrupt Sources
- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
• I/O and Packages
- 23 Programmable I/O Lines
- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
• Operating Voltage:
- 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
• Temperature Range:
- -40°C to 85’C
• Speed Grade:
- 0-20 MHz @1.8-5.5V
• Low Power Consumption at 1 MHz, 1.8V, 25 C for ATmega48PA/88PA/168PA/328P:
- Active Mode: 0.2 rnA
- Power-down Mode: 0.1 pA
- Power-save Mode: 0.75 pA (Including 32 kHz RTC)

8-bit AY*
Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash
ATmega48PA AT mega88PA ATmega168PA ATmega328P
Rev. 8161D-AVR-10.,09
Рис. П.1. Первая страница технического описания микроконтроллера ATmega 328P
Документация на микроконтроллер АТтеда и схема платы ***Arduino***
329
На самом деле техническое описание насчитывает сотни страниц, я хочу отметить только наиболее важные темы, на которые следует обратить внимание. Техническое описание, как правило, снабжено PDF-закладками, обеспечивающими быстрый поиск нужной информации.
Особый интерес представляет информация о портах ввода-вывода, таймерах, различных аппаратных интерфейсах. В качестве примера рассмотрим рисунок 13-1 из раздела I/0-Ports технического описания, который приведен здесь на рис. П.2. Схемы, подобные этой, присутствуют во всем техническом описании и позволяют глубже понять принцип действия платы ***Arduino***. На приведенной схеме видно, что все контакты ввода-вывода снабжены защитными диодами, предотвращающими появление на выводе отрицательного напряжения. Важную роль играет и конденсатор Cpi,,, определяющий интервалы времени фронта и спада.
Рис. П.2. Фрагмент технического описания ATmega 328P: схема контакта ввода-вывода
Цоколевка микросхемы ATmega 328P
Все технические описания содержат цоколевку, которая четко иллюстрирует функции каждого вывода. Выводы микроконтроллеров могут выполнять несколько функций, поэтому изучение цоколевки помогает уяснить назначение того или иного контакта. Рассмотрим цоколевку ATmega 328P (рис. П.3). Изучив цоколевку микроконтроллера ATmega, легче будет понять схему ***Arduino Uno***, которую мы рассмотрим в следующем разделе.
Обратите внимание, что расположение выводов на рис. П.3 и на самом микроконтроллере идентично. Полукруг (метка) в верхней части соответствует аналогичному полукругу на корпусе микросхемы и позволяет определить местоположение вывода 1.
Возможно, вы заметили аббревиатуры, не встречавшиеся ранее. Рассмотрим некоторые из них:
- ♦ VCC — источник питания микросхемы (в случае ***Arduino*** 5 В);
- ♦ AVCC — отдельное напряжение питания для АЦП (в случае ***Arduino*** 5 В);

Рис. П.3. Цоколевка микроконтроллера ATmega 328P
Рис. П.4. Схема подключения выводов ATmega 328p к пронумерованным контактам ***Arduino***
Документация на микроконтроллер АТтеда и схема платы ***Arduino***
331
- ♦ AREF — вывод для подключения опорного напряжения для АЦП (< 5 В);
- ♦ GND — заземление.

Остальные выводы контроллера— вводы-выводы общего назначения. Каждый из них сопоставляется с уникальным номером контакта в программном обеспечении ***Arduino***, так что вам не придется помнить букву и номер порта. Названия в скобках представляют собой запись альтернативных функций для каждого вывода. Например, выводы PD0 и PD1 также являются контактами универсального синхронноасинхронного приемопередатчика Rx и Tx соответственно. Выводы PB6 и PB7 служат для подключения внешнего кварцевого резонатора. На плате ***Arduino Uno*** к ним подключен кварцевый резонатор 16 МГц и вы не можете их использовать в качестве контактов ввода-вывода. Если у вас возникли проблемы с расшифровкой названий, дополнительную информацию можно отыскать в тех разделах технического описания, где приведена терминология. На сайте ***Arduino*** в разделе http://arduino.cc/en/Hacking/PinMappingl68 можно найти схему подключения выводов ATmega к пронумерованным контактам платы ***Arduino*** (рис. П.4).
Принципиальная схема ***Arduino***_____
Один из лучших способов изучения проектирования — анализ схем существующих продуктов, например, ***Arduino***. На рис. П.5 изображена принципиальная схема платы ***Arduino Uno***.
Можете ли вы, глядя на рис. П.5, установить соответствие элементов схемы и реальной платы ***Arduino***? Начните с микроконтроллера (MCU) (элемент ZU4 на схеме) и проследите, как связаны выводы ATmega 328p с контактами разъема. Нетрудно убедиться, что выводы ATmega соответствуют контактам, доступным в интегрированной среде разработки (IDE) ***Arduino***. Ранее мы отмечали, что выводы PDO и PD1 подключены к контактам USART Rx и Tx. По схеме ***Arduino*** можно увидеть, что

эти выводы подключены к соответствующим контактам контроллера 16U2 (конвертер USB в последовательный порт). Вы также знаете, что к контакту 13 ***Arduino*** подключен светодиод. Из схемы видно, что контакт 13 подключен к выводу PB5 на ATmega. Но где же светодиод? Обозначив провода специальными метками, можно указать связь между ними в разных частях схемы без соединения линиями. На первый взгляд это может показаться непонятным. Внимательно посмотрев на вывод PB5, можно заметить, что провод, идущий из MCU, помечен как SCK, а в верхней части схемы есть провод, тоже помеченный SCK и подключенный к светодиоду. Такие обозначения соединительных проводов встречаются на большинстве принципиальных схем. Уяснив общий принцип, продолжайте анализировать схему ***Arduino***, пока не поймете назначение всех соединений.

Рис. П.5. Схема платы ***Arduino Uno*** Rev3

Предметный указатель