Стюарт Ярнольд

ARDUINO
ДЛЯ НАЧИНАЮЩИХ

САМЫЙ ПРОСТОЙ
ПОШАГОВЫЙ
САМОУЧИТЕЛЬ

Стюарт Ярнольд

ARDUINO
ДЛЯ НАЧИНАЮЩИХ

САМЫЙ ПРОСТОЙ ПОШАГОВЫЙ САМОУЧИТЕЛЬ

Москва
2017

УДК 621.38
ББК 32.85
Я75

Arduino in easy steps
Stuart Yarnold

Ярнольд, Стюарт.

Я75 Arduino для начинающих : самый простой пощаговый самоучитель / Стюарт Ярнольд ; [пер. с англ. М. Райтман]. — Москва : Эксмо, 2017. — 256 с. — (Электроника для начинающих).

ISBN 978-5-699-98944-7

Самый простой и понятный самоучитель по Arduino для тех, кто делает первые шаги в работе с легендарными платами. В книге читатели найдут массу полезной информации и ценных рекомендаций, облегчающих процесс обучения. С этим самоучителем любой может освоить основы электротехники и научиться работать с компонентами и платами, а также программировать их, создавать скетчи и классные проекты легко и быстро!

УДК 621.38
ББК 32.85

ISBN 978-5-699-98944-7

Оглавление

1

Основные сведения об Arduino

Эта глава является введением в Arduino. Мы увидим, что представляет собой Arduino, что можно делать с ее помощью, а также ее преимущества по сравнению с конкурирующими платформами.

- Что такое Arduino?
- Почему именно Arduino?
- Какой тип платы Arduino следует выбрать?
- Что вы сможете делать с помощью Arduino?
- Основные принципы
- Аппаратное обеспечение
- Программное обеспечение
- Необходимые навыки для работы с Arduino

Что такое Arduino?

Arduino — это плата небольшого размера, содержащая 8-битный или 32-битный микроконтроллер, а также несколько других компонентов. Модели последнего поколения, такие как Uno, оборудованы USB-интерфейсом и рядом контактов для аналогового ввода и цифрового ввода и вывода.

История появления Arduino берет свое начало в Институте проектирования взаимодействий итальянского города Ивреа. В этом образовательном учреждении основное внимание уделяется взаимодействию с цифровыми устройствами и системами.

Суть концепции развития Arduino в упрощении создания интерактивных объектов или сред и возможности сделать их более доступными. С этой целью она и была разработана, чтобы быть недорогой и доступной, таким образом, предоставляя разработчикам-любителям, студентам и профессионалам возможность создавать устройства и проекты, которые взаимодействуют с окружающей средой с помощью датчиков и исполнительных устройств.

К стандартным примерам проектов Arduino относятся простые роботы, системы безопасности и датчики движения. Существует большое количество подобных примеров.

Arduino — это гораздо больше, чем просто аппаратное обеспечение. Микроконтроллер необходимо программировать. Для этого вы будете использовать интегрированную среду разработки (IDE), которая работает на персональных компьютерах. С ее помощью пользователи пишут программы (известные как *скетчи*), используя язык программирования C или C++.

Arduino — это современный эквивалент устаревших комплектов электроники из недавнего прошлого, которые были в продаже у таких компаний, как Radio Shack и Heath.

Микроконтроллер Arduino поставляется с загрузчиком, который значительно упрощает загрузку программ во Flash-память платы. Для сравнения: продукции конкурентов Arduino требуется использование внешнего программатора.

В соответствии с идеологией проекта Arduino процесс программирования упрощен настолько, насколько это возможно, и мы можем делать это с помощью персонального компьютера.

Arduino — идеальный инструмент для изучения основ электроники в целом, а также программирования.

Так как все разъемы на платах Arduino стандартных типов, мы можем расширить функциональность базовой комплектации Arduino посредством задействования встроенных плат, известных как *шилды*. Благодаря конструкции соединительных разъемов, их можно размещать друг на друге, что делает возможным создание больших и сложных проектов.

Пользователям доступен ряд плат Arduino. К ним относятся Uno, Duemilanove, Diecimila и Mega. Все они предназначены для использования с конкретными типами проектов, и поэтому поставляются с разными техническими характеристиками. Самой популярной из данных плат является Uno. Ее можно использовать для самого широкого круга проектов.

Почему именно Arduino?

Arduino — не единственный компьютер низкого ценового сегмента на рынке. Существует большое количество альтернативных вариантов, которые обеспечивают аналогичные возможности. К ним относятся:

- Raspberry Pi
- CubieBoard
- Gooseberry
- APC Rock
- OlinuXino
- Hackberry A10

В то время как в этой книге мы обсуждаем плату Arduino, вполне возможно, что вам мог бы подойти и другой тип компьютеров, принадлежащих к низкому ценовому диапазону.

Вам следует учесть, что Arduino обладает следующими реальными преимуществами.

Цена. Платы Arduino не такие дорогие, как другие платформы микроконтроллеров. Для многих людей это важный фактор.

Кроссплатформенность. Программное обеспечение Arduino работает в операционных системах Windows, macOS и Linux, в отличие от большинства других микроконтроллерных систем, которые ограничены поддержкой только системы Windows.

Простота и гибкость. Среда программирования Arduino достаточно проста для начинающих, однако она также обеспечивает гибкость, позволяя работать с более сложными проектами.

В качестве образовательного инструмента она очень уместно основывается на стандартной обрабатывающей программной среде с открытым исходным кодом, поэтому многие пользователи уже с ней знакомы.

Программное обеспечение с открытым исходным кодом. Программное обеспечение Arduino распространяется с открытым исходным кодом и находится в свободном доступе на ряде веб-сайтов. Для людей, не имеющих возможности писать свои собственные программы, или для тех, кто слишком занят, чтобы заниматься этим, в наличии имеется большое количество программ для бесплатного скачивания.

Открытое аппаратное обеспечение. Arduino работает на базе микроконтроллеров ATMEGA8 и ATMEGA168, схемы которых находятся в свободном доступе под лицензией Creative Commons. Таким образом, пользователи могут создать свою собственную версию модуля, чтобы улучшить его или повысить его потенциальные возможности.

На заметку

Arduino действительно обладает рядом преимуществ над конкурентами.

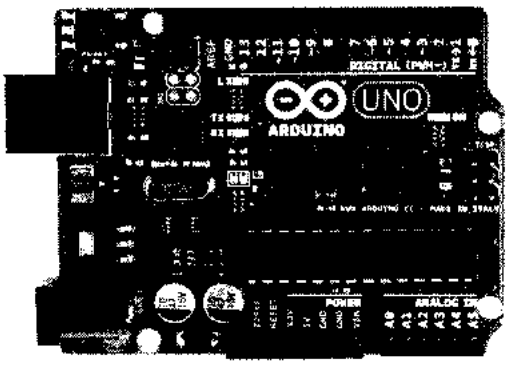
Какой тип платы Arduino следует выбрать?

Не все платы Arduino одинаковы — существует несколько типов, каждый из которых предназначен для самых разнообразных запросов. Поэтому, прежде чем расставаться с деньгами, следует рассмотреть все доступные варианты. Ниже мы рассмотрим три наиболее популярные платы.

Arduino Uno

Самая популярная плата Arduino из существующих на данный момент — Arduino Uno. Она обладает рядом функций, которые делают ее универсальной.

Arduino Uno — это стандарт, с которым сравнивают все остальные платы Arduino.



В качестве контроллера в нее встроен микропроцессор ATmega328, который может получать электропитание напрямую от USB, батареи или сетевого адаптера. Эта плата работает при напряжении 5 В.

Эта плата также содержит 14 контактов цифрового ввода-вывода, шесть из которых можно использовать в качестве выходов с широтно-импульсной модуляцией (ШИМ). Кроме того, она имеет шесть аналоговых входов плюс контакты **RX/TX** (последовательные данные).

Установленная память — 32 Кб Flash-памяти, 2 Кб статической оперативной памяти (SRAM) и 1 Кб электрически стираемого программируемого постоянного запоминающего устройства.

Различные комплектации Uno включают:

Leonardo — данная версия доступна как со штыревыми колодками, так и без них, и оборудована портом micro USB.

Ethernet — эта версия оборудована разъемом Ethernet RJ45 вместо USB-порта. Также она содержит картридер microSD.

Arduino Uno представляет собой идеальную плату для начинающих, так как обладает большим количеством параметров и возможностей по низкой цене.

К преимуществам Arduino Uno относятся:

- простота;
- низкая стоимость.
- функция Plug-and-play;
- множество доступных ресурсов, например учебники, примеры кода и т.д;
- большое количество дополнений, таких как шилды и библиотеки.

К недостаткам относятся:

- небольшое количество контактов ввода-вывода;
- низкий объем предоставленной памяти, что может налагать ограничения;
- 8-битный микроконтроллер.

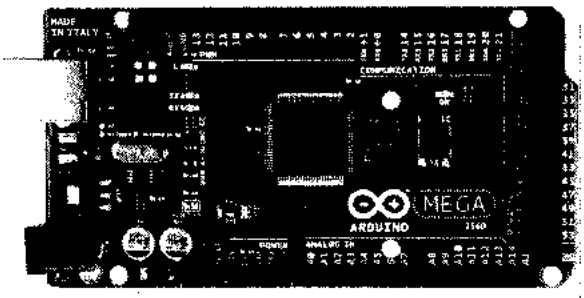
Arduino Mega 250

Эта плата — старший брат Uno и в значительной степени похожа на нее, за тем исключением, что она больше по размеру. Увеличение площади поверхности данной платы позволяет использовать 70 входных/выходных контактов (Uno имеет только 14). 16 из них являются аналоговыми входами, а оставшиеся 54 — цифровыми. 15 цифровых контактов могут работать с широтно-импульсной модуляцией (ШИМ). Также присутствуют четыре последовательных порта **RX/TX**.

Mega предназначена для более продвинутых пользователей.

Что касается контроллера, он представляет собой микропроцессор ATmega2560, который работает при напряжении 5 В. Arduino Mega оборудована в четыре раза большим объемом памяти, чем Uno. В частности, 256 Кб Flash-памяти, 8 Кб SRAM и 4 Кб EEPROM.

Различные варианты Mega включают:



Due — эта версия сконструирована на базе 32-битного микроконтроллера с ARM ядром. Будучи гораздо быстрее, чем Uno, она предназначена для использования с более требовательными приложениями. В связи с этим здесь также установлено больше памяти — 512 Кб Flash-памяти и 96 Кб SRAM. В отличие от других плат, Due работает при напряжении 3,3 В.

Если вы создаете проект, который будете использовать на смартфоне под управлением операционной системы Android, вам как раз подойдет плата Arduino ADK.

ADK — это плата, предназначенная для использования с мобильными устройствами на базе операционной системы Android.

К преимуществам Arduino Mega 2560 относятся:

- большое количество входных/выходных контактов;
- приличный объем памяти;
- большое количество доступных ресурсов, например учебники, примеры кода и т.д;
- предоставляются возможности для более масштабных проектов, чем плата Uno.

К недостаткам относятся:

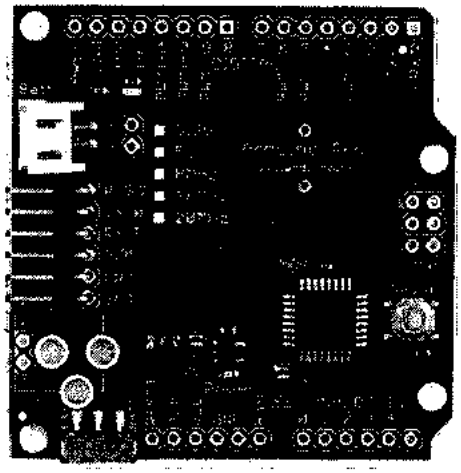
- более высокая цена по сравнению с Uno (примерно в два раза);
- доступно не так много ресурсов, как для Uno.

Arduino Pro

Как можно заключить из названия, Arduino Pro предназначена для профессионального использования. Эта плата построена на базе микроконтроллеров ATMEGA168 или ATMEGA328.

На заметку

Из всех плат Arduino в модели Pro реализовано наибольшее количество функций и возможностей применения.



Плата Arduino Pro поставляется как в исполнении с электропитанием при напряжении 3,3 В, так и при 5 В. Она оборудована 14 цифровыми входными/выходными контактами (шесть из которых можно использовать в качестве выходов ШИМ) и 6 аналоговыми входами.

Также она имеет отверстия для монтажа силового разъема, блок ICSP и штыревые колодки. К 6-контактному блоку можно подключить FTDI-кабель, чтобы обеспечить плате электропитание и интерфейс передачи данных через USB.

Arduino Pro предназначена для использования при полустационарной установке. На этой плате нет предустановленных колодок, что позволяет пользователю использовать все типы выводов и разъемов, необходимых для проекта, над которым он работает данный момент.

Внимание

Пайка соединений в Arduino Pro должна осуществляться вручную.

К преимуществам Arduino Pro относятся:

- хорошо подходит для использования во встраиваемых проектах;
- обеспечивает хорошую гибкость при разработке проектов;

- спаянные соединения приводят к повышению уровня надежности.

К недостаткам относятся:

- более высокая цена по сравнению с Uno;
- необходимо паять разъемы и соединения.

Что вы сможете делать с помощью Arduino?

Теперь, когда вы уже знаете, что такое Arduino и почему следует отдать предпочтение ей, а не конкурентам, а также какую версию использовать, вам, возможно, захочется рассмотреть возможности ее использования.

В первую очередь необходимо понять, что, поскольку эта плата поставляется с микроконтроллером, Arduino можно использовать в качестве «мозга» для практически любого проекта, связанного с электроникой. Это добавляет многоплановости простым электронным комплектам прошлых лет.

Несмотря на то, что вы можете выполнить множество операций с помощью базового комплекта Arduino, сложные проекты потребуют применения дополнительных компонентов.

Объединяя различные переключатели и датчики, Arduino может «чувствовать» свое окружение; речь идет о свете, звуке, температуре, движении, давлении и т. д. Получив эту информацию, Arduino может использовать ее для взаимодействия с окружающим миром с помощью электродвигателей и других механизмов. Случайный и довольно простой пример: вы можете использовать Arduino для автоматического управления освещением в доме.

К устройствам, которыми вы можете управлять с помощью Arduino, относятся переключатели, светодиодные индикаторы, электродвигатели, акустические системы, GPS-устройства, камеры, а также смартфоны и ТВ.

Arduino может работать как независимое устройство, так и в связке с компьютером, другими платами Arduino или электронными устройствами. В действительности, проявив немного изобретательности, вы можете подключиться к широкому спектру устройств и управлять ими при помощи Arduino.

Некоторые проекты Arduino предназначены просто для развлечения, например мигающий светодиодный индикаторный куб, робот, взбирающийся на дерево, лазерная арфа. Тем не менее большинство проектов имеют практическое применение. Примерами могут служить системы безопасности, автоматического полива растений и контроллеры гаражных ворот, управляемые с помощью смартфона. Arduino также представляет собой простую и недорогую платформу для создания прототипов, которая позволяет определять важность идей и воплощать их в реальность.

Плату Arduino используют архитекторы, дизайнеры и программисты-любители.

В то же время существуют приложения, с которыми плата Arduino по определению несовместима. Так как Arduino имеет ограниченный объем памяти и медленно обрабатывает информацию, эту плату нельзя использовать для проектов, требующих серьезной вычислительной мощности. Это исключает возможность обработки видео- и аудиоматериала, звукозаписи и вывода данных.

Также она обладает относительно высоким энергопотреблением, что ограничивает ее автономное использование с батареями.

Основные принципы

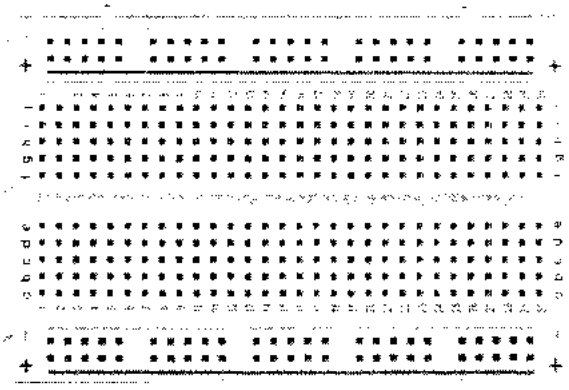
Прежде чем приступить к подробному изучению Arduino, мы познакомимся с основами. Это поможет решить, подходит вам Arduino или нет.

Аппаратное обеспечение

Проект Arduino состоит как из аппаратных, так и из программных элементов. Мы начнем с первых.

Макетная плата. В вашем комплекте Arduino находится пустая плата, известная как макетная плата, как показано ниже.

Вы будете использовать эту плату для создания цепей. Они состоят из электронных компонентов, таких как резисторы и конденсаторы. Цепи создаются путем вставки соединительных проводов компонентов в отверстия платы и затем соединения их при помощи перемычек. Все, что вам нужно для этого знать — это в какие отверстия их необходимо вставлять.



Обратите внимание, что нет необходимости паять компоненты на месте, что будет плюсом для тех, кто никогда раньше не занимался пайкой или просто не хочет этого делать.

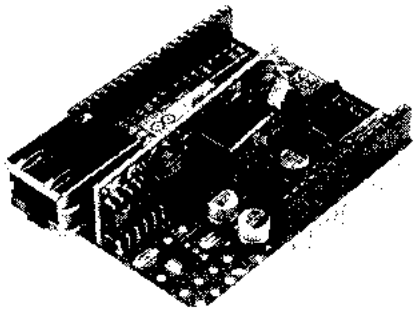
На заметку

При использовании Arduino навыки пайки не важны.

Макетные платы обладают двумя преимуществами: первое заключается в том, что они позволяют создавать цепи быстро и просто. Второе состоит в простоте перегруппировки компонентов с целью исправления ошибок или простого экспериментирования.

Компоненты. Ваш комплект Arduino включает большое количество компонентов, таких как резисторы, конденсаторы, переключатели, электродвигатели и т. д. Чтобы увидеть полный список, см. с. 105. Это «кирпичики», с помощью которых вы строите свои проекты.

Шилды. Для создания простых базовых проектов вам достаточно компонентов, которые поставляются с вашим комплектом. Как только вы изучите эти проекты, вы захотите создавать более сложные цепи.



Arduino Ethernet Shield

Если у вас есть такая возможность, то ничего не помешает создавать подобные проекты с помощью плат расширения. Если такой возможности нет или же вы просто не хотите утруждать себя, вы можете приобрести необходимые цепи в виде готовых печатных плат. Они известны

как шилды, и их можно свободно купить в специализированных магазинах и в интернете.

Если у вас нет ни времени, ни навыков для того, чтобы создавать цепи самостоятельно, вы можете приобрести готовые шилды.

Микроконтроллер. Сложив вместе различные составляющие элементы проекта, вам нужно будет передать Arduino команды, необходимые для того, чтобы проект работал, т. е. вы должны будете запрограммировать плату.

Та часть Arduino, которая отвечает за обработку (интерпретацию) команд (программы), называется микроконтроллером.

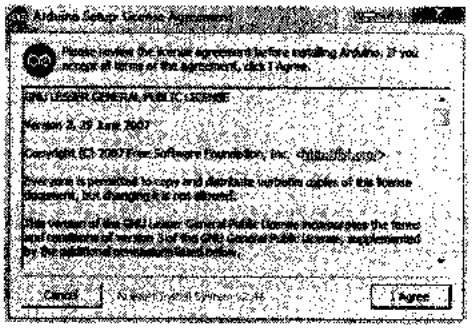
Однако прежде чем вы сможете начать программировать, сначала необходимо выполнить настройку программного обеспечения Arduino.

Программное обеспечение

Программное обеспечение не поставляется вместе с платой. Вам необходимо посетить веб-сайт Arduino по адресу **www.arduino.cc/en/main/software** и загрузить его оттуда. Это программное обеспечение известно как интегрированная среда разработки (IDE).

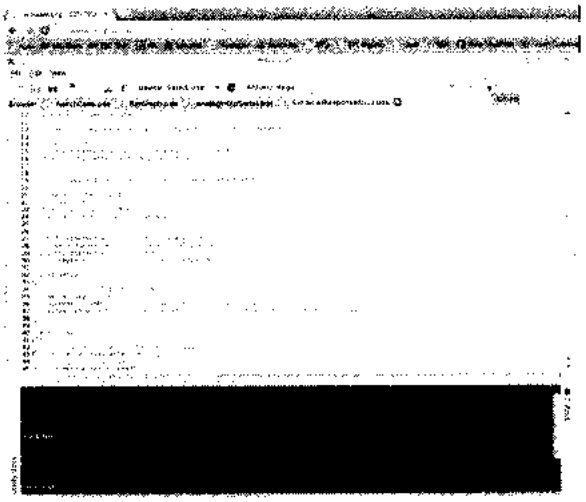
Чтобы программировать для своих проектов, вам необходимо использовать компьютер.

Существуют версии IDE для операционной системы Windows, macOS и Linux. Таким образом, вы сможете использовать Arduino в любой операционной системе, которая установлена на вашем компьютере.



Чтобы установить программное обеспечение, следуйте подсказкам мастера установки.

Программирование. После установки IDE вы сможете писать код, необходимый для вашего проекта. Это делается в IDE на компьютере, где есть возможность использовать окно для редактирования кода, как показано ниже. Программы, которые вы пишете, известны как *скетчи*, и ими можно управлять так же, как другими данными: сохранить на жесткий диск, опубликовать в интернете и т. д.



Вы можете приобрести книги «Программирование на C для начинающих» и «Программирование на C++ в примерах и задачах», чтобы научиться программировать на этих языках.

Когда скетч уже написан, подключите плату Arduino к компьютеру при помощи входящего в комплект USB-кабеля и загрузите код в микроконтроллер. На этом ваш проект закончен — остается всего лишь посмотреть, работает ли он.

Необходимые навыки

Система Arduino разрабатывалась для того, чтобы быть максимально понятной для пользователя. Но это не значит, что можно получить немедленный результат, лишь начав работать с системой. Для реализации проектов Arduino вам понадобится не только знание компонентов самой платы, но и другие навыки.

Одним из них является программирование — все проекты Arduino следует программировать, используя языки C и C++.

Другой навык — это электроника. Кроме самых простых, все проекты требуют создания внешних цепей. Это потребует знание электронных компонентов, проектирование цепей и сборку.

Для тех, кто не владеет необходимыми знаниями, интернет — прекрасный источник информации, где можно найти все, что можно сделать с помощью Arduino.

К другим навыкам, которые вам, вероятно, могут понадобиться, относится проектирование, механика, сборка и обработка данных. Очевидно, это в значительной мере зависит от масштаба ваших проектов.

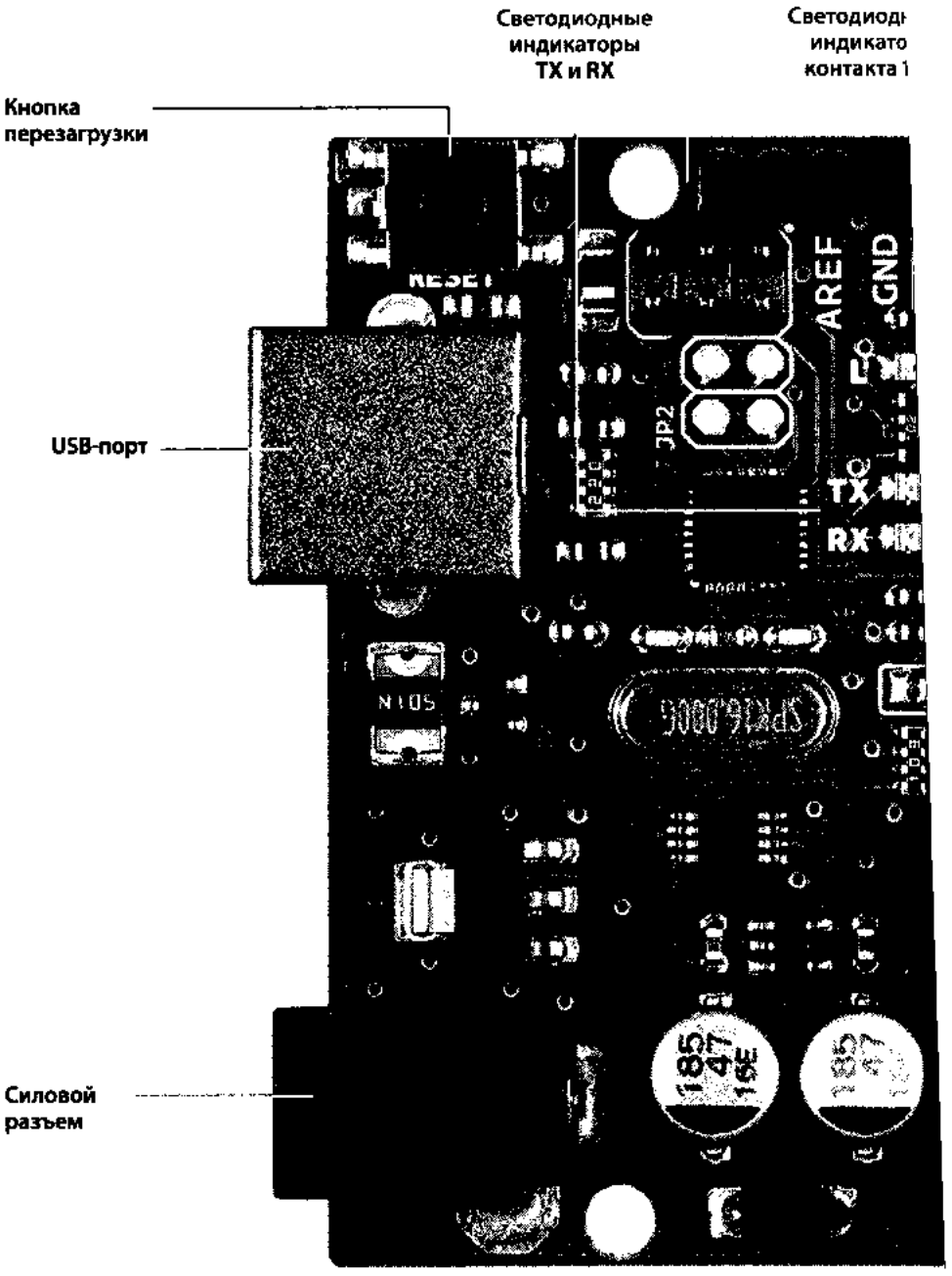
2

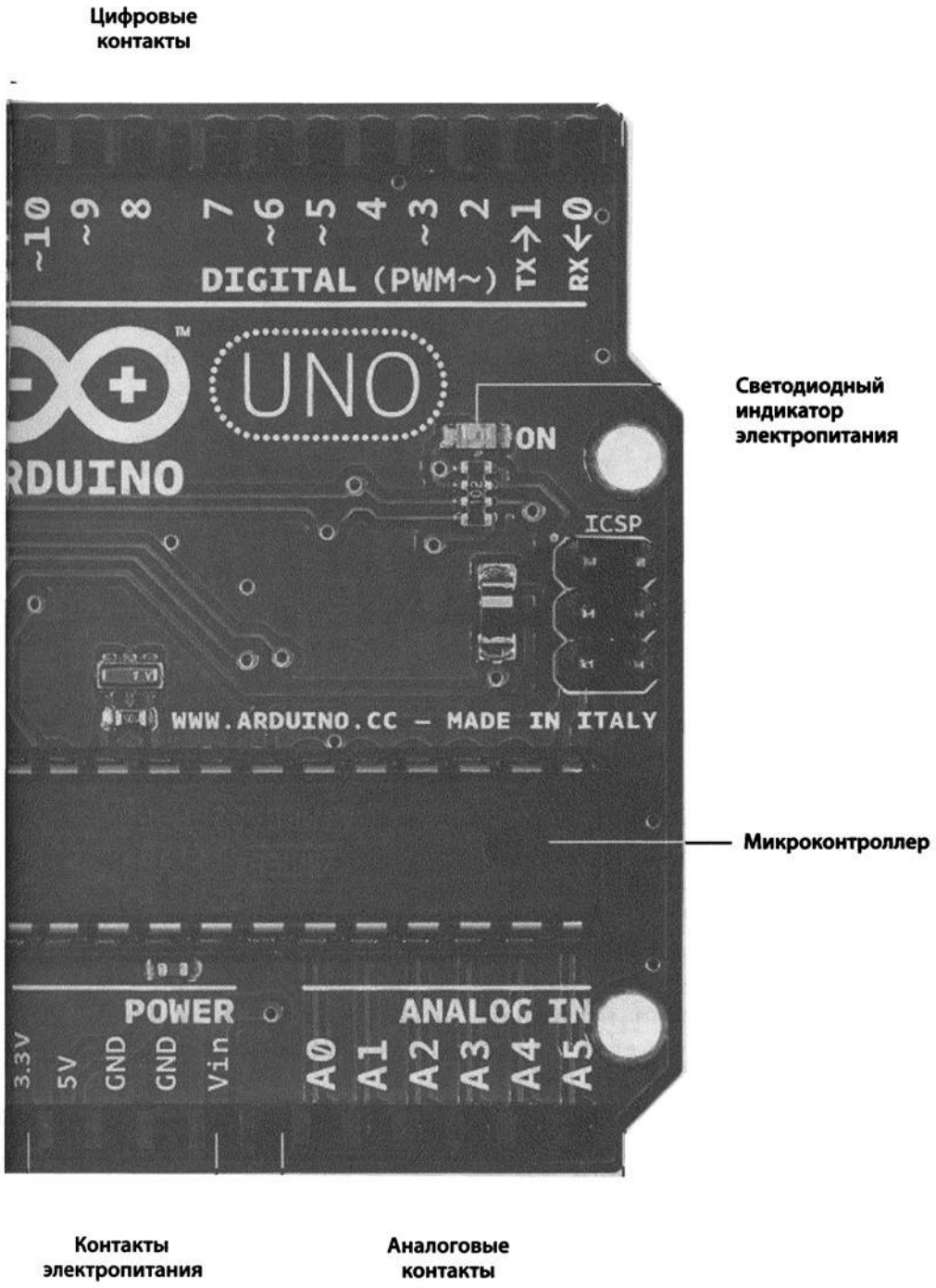
Набор инструментов Arduino

В этой главе мы взглянем на содержимое набора Arduino и объясним, для чего необходимы различные элементы.

- Плата Arduino Uno
- Важные элементы платы
- Макетные платы
- Перемычки
- Компоненты
- Основные платы

Плата Arduino Uno




Цифровые контакты
DIGITAL (PWM~)
TX→1
RX←0
UNO
ON
ICSP
WWW.ARDUINO.CC – MADE IN ITALY
Светодиодный индикатор электропитания
Микроконтроллер
POWER
ANALOG IN
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
Контакты электропитания
Аналоговые контакты

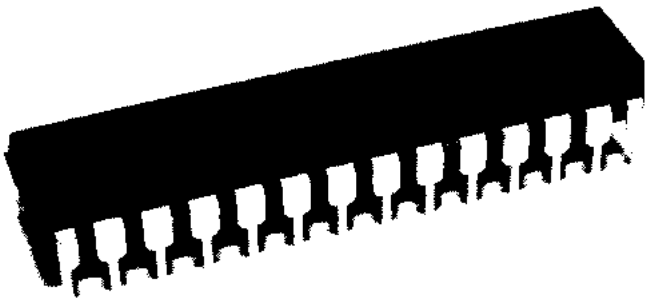
Важные компоненты платы

В первой главе мы описали Arduino с помощью общих терминов. Теперь давайте более подробно рассмотрим аппаратную часть. Начнем с самой платы Arduino.

На с. 24–25 показана схема наиболее популярной платы Arduino Uno и ее главные компоненты:

Микроконтроллер

Эта интегральная схема (или ИС) является «мозгом» платы и обеспечивает ее вычислительную мощность. Различные модели Arduino используют разные версии микроконтроллера. Плата Arduino Uno построена на базе микроконтроллера Atmel ATmega328.



На заметку

На заметку. Разные платы Arduino обладают разными возможностями. В значительной степени это зависит от микроконтроллера.

Фактически это маленький компьютер, который может сам принимать решения. На программной плате — это контроллер, исполняющий команды, включающий несколько разных типов памяти, которые хранят данные и команды. Микроконтроллеры также поддерживают несколько разных способов отправки и получения данных.

Штыревые разъемы

Микроконтроллер подключается к ряду разъемов, которые известны как штыревые разъемы или контакты. Они расположены рядом с верхним и нижним

краями платы. Существуют контакты трех типов: цифровые, аналоговые контакты и электропитания.

Контакты необходимы для подачи электропитания на плату Arduino, а также для возможности быстро и просто подключать к плате дополнительные компоненты, такие как макетные платы и шилды.

Платы расширения (известные как шилды) также имеют штыревые разъемы и позволяют укладывать платы друг на друга при создании сложных проектов.

Цифровые контакты. Они располагаются вдоль верхней части платы. Контакты, пронумерованные от **0** до **13**, служат для получения и отправки цифровых сигналов. Контакты **0** и **1** также выступают в качестве последовательного порта, который используется для получения и отправки данных другим устройствам, например компьютеру через USB-интерфейс.

Обратите внимание, что под цифровым мы имеем в виду сигнал в одном из двух возможных состояний: включено или выключено. В случае Arduino Uno это означает напряжение равное 0 или 5 В.

Контакты аналоговых входов. В нижней части платы присутствует еще два ряда контактов. Шесть контактов, которые помечены как **A0–A5**, представляют собой аналоговые входы. Контакты **A4** и **A5** также можно использовать для получения и отправки данных другим устройствам.

Обратите внимание, что под аналоговым мы имеем в виду сигнал, который различается по силе между двумя фиксированными значениями. В случае Arduino Uno это означает напряжение, которое изменяется в диапазоне между 0 и 5 В.

Очень важно понимать различие между цифровым и аналоговым сигналом. Первый фиксирован и передает напряжение либо 0 В, либо 5 В. Напряжение аналогового сигнала различно и варьируются в диапазоне от 0 до 5 В.

Контакты аналоговых выходов. Эти контакты расположены в верхней части платы и помечены символом «тильда» (~). Они используют метод, известный

как широтно-импульсная модуляция (ШИМ), которая фактически имитирует аналоговый сигнал, используя цифровые контакты. Обратите внимание, что символ ~ отображен рядом с контактами **3**, **5**, **6**, **9**, **10** и **11** — все они могут обрабатывать ШИМ.

Сигнал ШИМ используется для управления устройствами, такими как электродвигатели, которым требуется аналоговый сигнал.

Контакты электропитания. В нижнем левом углу платы находятся восемь контактов электропитания. Они используются для обеспечения и распределения электропитания на необходимые компоненты.

Контакт **Vin** (силовой вход) может быть использован для подключения внешнего источника входного напряжения для платы Arduino или в качестве источника выходного напряжения для электропитания внешних компонентов.

Контакт **Pin2** — это контакт IOREF, который используется для того, чтобы сообщать рабочее напряжение платы Arduino любому шилду, присоединенному к плате.

Применяя контакт **Vin**, вы можете использовать плату Arduino для обеспечения электропитания внешней платы.

Два контакта **GND** позволяют замыкать цепи. Третий дополнительный контакт **GND** расположен около контакта **13** в верхней части платы. Все эти контакты связаны и используют одну и ту же линию заземления.

Контакт **5V** используется для обеспечения платы электропитанием с напряжением 5 В, а контакт **3,3V** — с напряжением 3,3 В.

USB-разъем

В верхнем левом углу платы находится USB-разъем. С помощью USB-кабеля вы можете подключить плату к компьютеру для загрузки в нее скетчей.

К этому разъему можно подключать USB-коннектор типа B. Он поставляется в комплекте. Это точно такой же кабель, который обычно используется для подключения устройств, вроде принтера или сканера, и вы сможете легко найти замену этому кабелю, если его потеряете.

Внешний силовой разъем

В нижней левой части платы находится внешний силовой разъем, который также называют разъемом или гнездом электропитания. Для электропитания платы вы можете использовать порт USB. Также вы можете использовать внешний источник электропитания, например адаптер преобразования переменного тока в постоянный или батарею.

Обратите внимание, что вы также можете обеспечить электропитанием вашу плату Arduino через USB-интерфейс.

Адаптер можно подключить к плате через 2,1 мм разъем к силовому разъему на плате. Провода от батареи подключаются к штыревым контактам электропитания **GND** и **Vin**. Если вы соберетесь делать это, следует быть очень внимательным, поскольку неверное подключение почти наверняка выведет плату из строя.

Плата Arduino может работать от внешнего источника электропитания напряжением от 6 до 20 В. Однако если подается менее 7 В, контакт **5V** может не обеспечивать нужное напряжение в 5 В и в итоге работа платы может оказаться нестабильной. Если же поступает свыше 12 В, регулятор напряжения может перегреться и повредить плату. Таким образом, рекомендуемый диапазон напряжения составляет от 7 до 12 В.

Внешний источник электропитания с напряжением свыше 12 В может повредить вашу плату.

Светодиодные индикаторы

Плата Arduino содержит четыре светодиодных индикатора. В правой части платы находится светодиодный индикатор, который загорается, когда на плату поступает электропитание.

Светодиодные индикаторы **TX** и **RX** соответственно сигнализируют о том, что плата передает или получает данные между Arduino и внешними устройствами либо с помощью последовательного порта, либо USB-порта.

Светодиодный индикатор L (который находится над светодиодным индикатором TX) подключается к цифровому контакту 13, и, когда он горит, это говорит о том, что плата работает корректно.

Кнопка перезагрузки

Как со многими компьютеризированными устройствами, часто все идет не по плану. По неизвестной причине рассматриваемая задача может перестать работать. Чтобы решить внезапно возникшие проблемы, на плате присутствует кнопка перезагрузки системы. Она находится в верхнем левом углу платы.

Нажав ее, вы перезагрузите текущий работающий скетч, а удержание данной кнопки полностью завершит его работу. Обратите внимание, что вы также можете выполнить эти действия из среды Arduino IDE на своем компьютере.

Макетные платы

Мы очень коротко рассмотрели макетные платы на с. 18. Здесь вы увидите, почему они играют такую важную роль в проектах Arduino.

Основным принципом Arduino является прототипирование, т. е. исследование идей, что хотелось бы делать быстро и недорого. Макетные платы обеспечивают такой способ.

Они делают это, предлагая основу для создания цепей прототипа, который является скорее временным, а не постоянным, из-за отсутствия пайки компонентов.

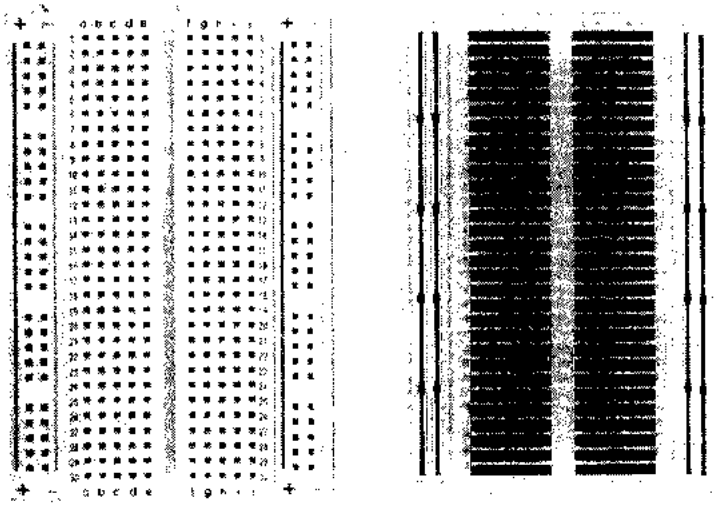
Макетные платы, не требующие пайки, подходят только для низкочастотных цепей. Также они не подходят для сложных цепей из-за количества требуемой проводки.

Таким образом, создав, протестировав цепь и подтвердив ее работоспособность, вы можете сделать ее постоянно действующей с помощью пайки. С другой стороны, если она не работает, просто вытащите компоненты из макетной платы и начните снова.

Клеммные колодки

Конструкция макетной платы фактически представляет собой пластиковый корпус с рядами отверстий на поверхности. Каждая колонка помечена буквой, например **a**, **b**, **c**, **d** и т. д. Вниз по плате идут пронумерованные ряды, а с другой стороны платы находятся проводящие контактные колодки, сделанные из меди.

Каждый ряд на плате помечен цифрой, а каждая колонка — буквой. Это помогает разобраться в случаях, если при построении цепи возникают сложности.

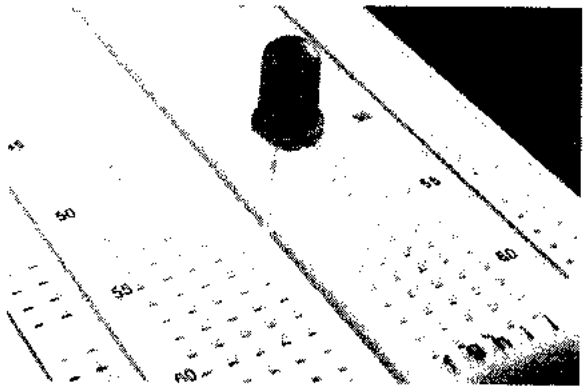


На рисунке выше: слева изображена верхняя часть макетной платы, изображающая ряды и колонки; справа можно видеть обратную сторону, где показаны контактные колодки.

Внутри каждого отверстия находится металлический зажим. Когда соединительный провод компонента вставляется в отверстие, зажим захватывает его, и удерживает на месте. Как только мы его вставили, с помощью электрического соединения компонент будет подключен ко всем другим компонентам, что находятся в этом ряду.

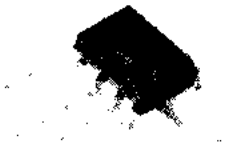
Канавка

Ближе к центру макетной платы есть углубление, известное как канавка. Она разделяет каждый ряд на два электрически независимых отдела.



Благодаря этому вы можете устанавливать компоненты на плату, не замыкая контакты, как можно видеть на рисунке выше.

Канавка также полезна для установки интегральных схем, требующих электрического разделения ножек с каждой стороны.



Шины электропитания

По каждой стороне макетной платы вертикально проходят шины электропитания. Они сделаны из медной ленты и идентичны контактным колодкам.

При создании цепей подача электропитания необходима на разных участках. Такой доступ предоставляют шины электропитания. Они помечены символами + и –, а также красно-синими или черными полосами, которые отображают положительное или отрицательное напряжение.

В макетных платах большого размера можно увидеть, что одна половина шин электропитания платы изолирована от другой, поэтому на каждой стороне есть верхняя и нижняя половина.

На заметку

Шины электропитания проходят по всей длине макетной платы, что делает электропитание доступным для всех элементов цепи.

Это полезно в таких ситуациях, когда для электропитания цепи необходимо подавать два разных напряжения. Тем не менее вам следует быть внимательными, так как легко подключить один из элементов к несоответствующей шине, и он может перегореть.

Перемычки

Перемычки — еще один важный компонент любого электронного проекта. Они представляют собой короткие куски изолированных проводов с зачищенными концами и используются для подключения компонентов на макетной плате.

Ваш комплект Arduino уже содержит несколько перемычек, но если вам понадобятся дополнительные, достать их будет несложно. Вы можете приобрести перемычки разного цвета и длины, упакованные в пакеты, или отрезать куски необходимой длины от мотка.

Совет

Макетные платы можно быстро заполнить компонентами, поэтому не всегда возможно разместить элементы там, где вы хотели бы сделать это. Перемычки обеспечивают необходимое решение.

Из этих двух вариантов мы предпочтем первый. Разные цвета полезны для определения целей, плюс готовые перемычки имеют жесткий наконечник, что делает подключение к макетной плате гораздо проще.

В то же время есть преимущества в том, чтобы нарезать перемычки самостоятельно. Во-первых, вы можете отрезать перемычку необходимой вам длины, что в значительной степени может помочь вам сохранять порядок в цепи. Во-вторых, этот вариант дешевле.

Компоненты

В комплекте Arduino вы обнаружите коробку, содержащую ряд электронных компонентов.

Строго говоря, нет необходимости в обязательном изучении того, что могут делать эти компоненты и как именно они это делают. Но не изучив их, вы никогда не сможете разрабатывать и создавать свои собственные проекты — вы сможете только имитировать их, копируя усилия других людей.

На заметку

В набор входит еще один тип перемычек с заранее определенной длиной и углом. Будучи неподвижно закрепленными, они лежат вдоль платы, помогая таким образом избавиться от ненужного скопления проводов.

Поскольку подобный подход не имеет большого смысла — вся суть Arduino заключается в инновации и экспериментах, — мы рекомендуем вам потратить время на изучение этих компонентов.

Совет

Если вы хотите создавать свои собственные проекты, вам необходимо знать основы электроники, а также изучить компоненты, благодаря которым все работает.

Этой теме посвящается 6 глава. Хотя в этой книге представлены только общие сведения, они дадут вам базовые знания о компонентах, используемых в электронных цепях.

Монтажная панель

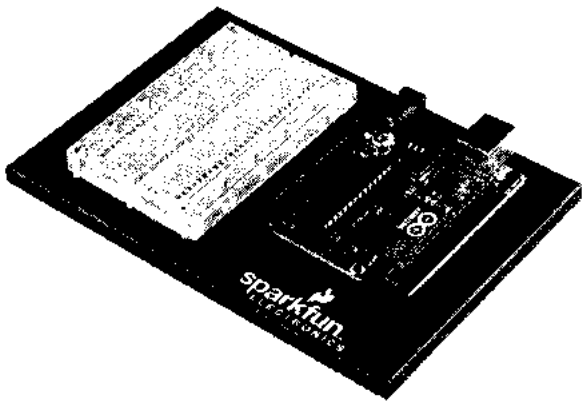
Многие проекты Arduino включают в себя как плату Arduino, так и макетную плату. Так как это две отдельные структуры, связанные только подключенными проводами, их необходимо установить на чем-либо, чтобы

они работали как одно устройство. Этого можно достичь с помощью монтажной или наборной панели.

В комплекте Arduino вы обнаружите специальную деревянную панель, которую можно собрать самостоятельно и которая поможет вам в осуществлении задуманного, как показано на рисунке справа. Инструкции по сборке находятся в руководстве по применению Arduino.

Плата Arduino крепится к монтажной панели с помощью гаек и болтов, которые идут в комплекте. Макетная плата обладает самоклеящимся покрытием, что позволяет ей крепиться рядом с платой Arduino.

Также следует упомянуть, что доступны монтажные панели лучшего качества. Они сделаны из акрила и имеют отверстия для крепления наиболее популярных плат Arduino. При стоимости несколько сотен рублей они будут удачной покупкой.



Выше приведен хороший пример. Панели такого типа имеют резиновое основание, которое предотвращает скольжение платы по рабочей поверхности.

Чтобы упростить жизнь, рекомендуем вам вкладывать деньги в монтажные панели хорошего качества.

3

Программное обеспечение Arduino

В этой главе мы рассмотрим программное обеспечение Arduino. Мы покажем, как можно выполнить его настройку, а затем объясним, как работают различные параметры.

- **Установка Arduino в операционной системе Windows**
- **Установка Arduino в операционной системе macOS**
- **Установка Arduino в операционной системе Linux**
- **Настройка Arduino**
- **Проверка работоспособности платы**
- **Среда Arduino**

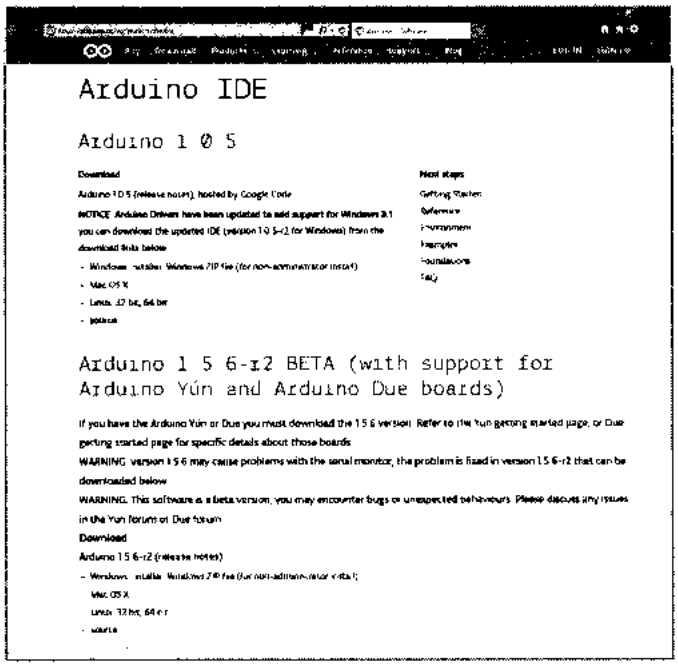
Установка Arduino в операционной системе Windows

Программное обеспечение не поставляется в комплекте с Arduino. Вам необходимо загрузить его на компьютер с веб-сайта Arduino.

На заметку

Здесь речь идет о системе Windows 8, но такие же инструкции применимы к Windows 8.1 и последующим версиям.

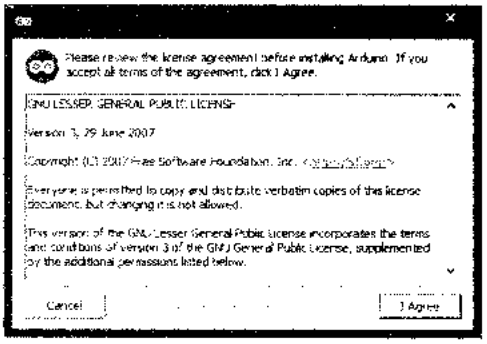
❶ Перейдите по адресу **arduino.cc/en/main/software**.



Совет

Вы увидите ссылки на загрузку нескольких версий Arduino. Вам нужна версия 1.0.5.

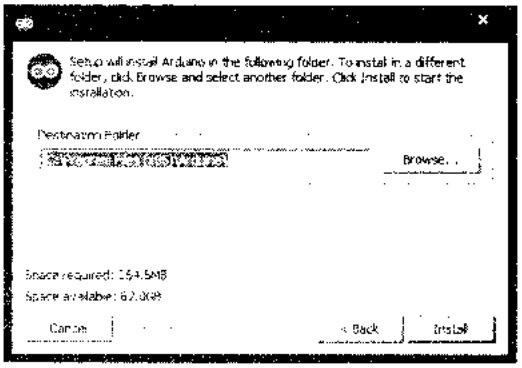
❷ **Под заголовком Arduino 1.0.5 щелкните мышью по ссылке Windows Installer. Начнется скачивание дистрибутива, по завершении которого автоматически начнется процедура установки.**



На заметку

Большинство аппаратных устройств поставляется с необходимым программным обеспечением на компакт-диске. В случае Arduino программное обеспечение необходимо загружать с веб-сайта проекта.

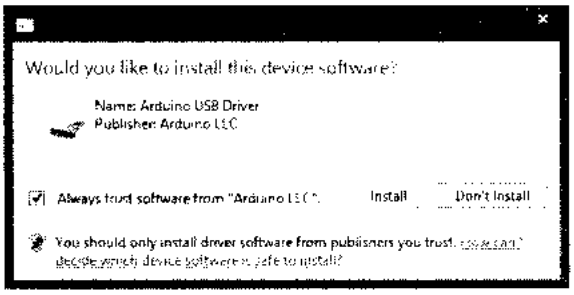
❸ **Нажмите кнопку I agree (Я согласен) в окне лицензионного соглашения.**



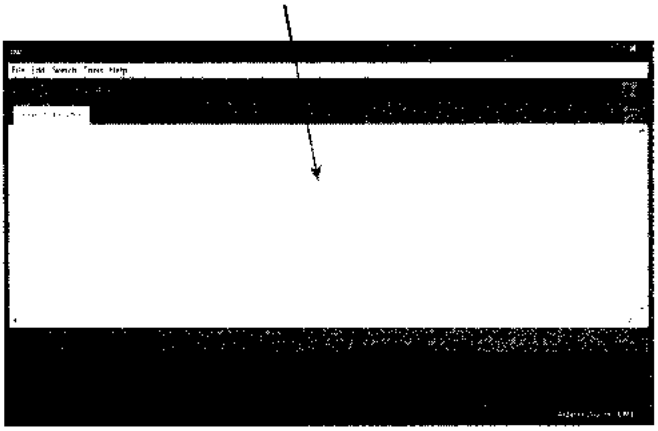
На заметку

Здесь мы видим установку Arduino в операционной системе Windows 8. Процесс установки аналогичен и для других версий операционной системы Windows.

4. По умолчанию среда Arduino будет установлена в каталог *C:\Program Files (x86)\Arduino*. Если вы хотите установить ее в другое местоположение, нажмите кнопку **Browse** (Просмотр) и выберите необходимую папку. Затем нажмите кнопку **Install** (Установить).

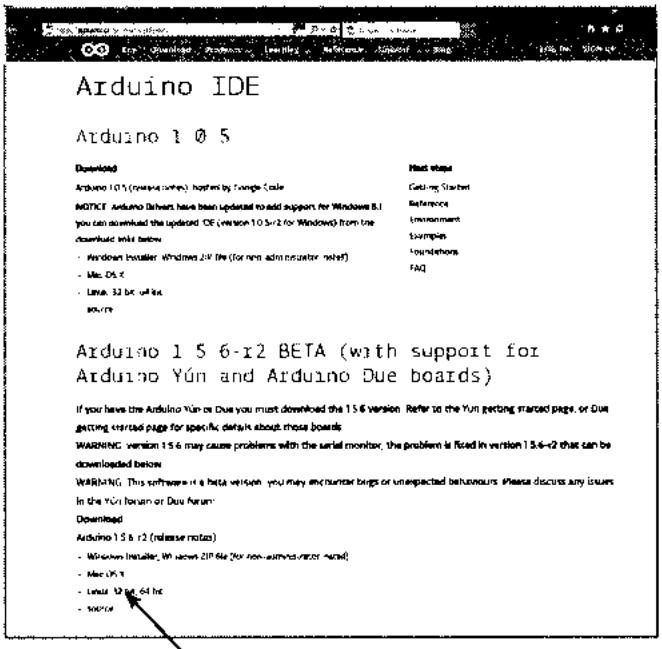


5. Когда появится окно с уведомлением об установке USB-драйвера, нажмите кнопку **Install** (Установить). После завершения установки вы увидите на рабочем столе значок Arduino. Щелкните по нему мышью, чтобы открыть среду Arduino IDE.

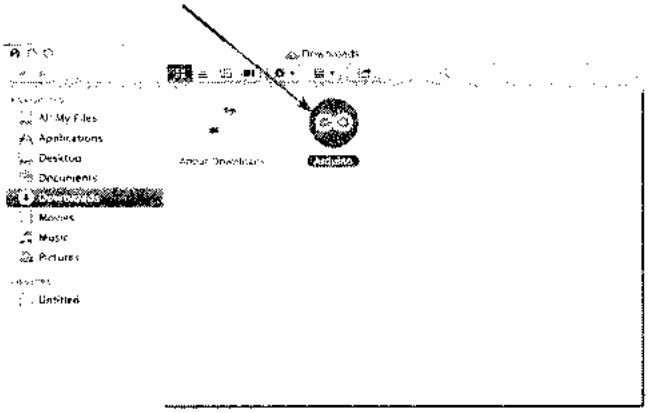


Установка Arduino в операционной системе macOS

❶ Перейдите по адресу **arduino.cc/en/main/software**.



❷ Под заголовком **Arduino 1.0.5** щелкните мышью по ссылке **macOS**, чтобы начать загрузку. По окончании перейдите в папку загрузок, где вы увидите значок Arduino.

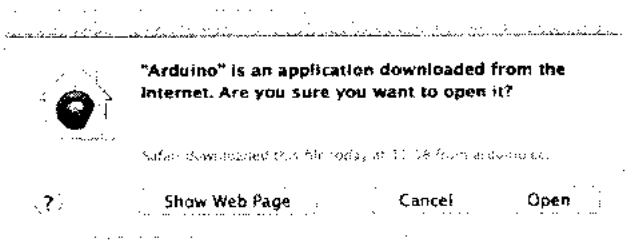


Приведенные инструкции демонстрирует установку Arduino в операционной системе OS X 10.8 Mountain Lion.

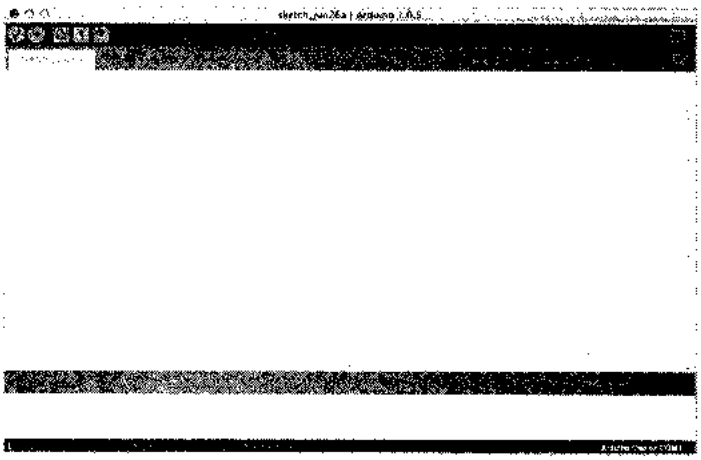
❸ Перетащите значок Arduino в папку Applications (Приложения). Кроме того, вы можете перетащить его на панель **Dock**, откуда можно будет легко получить доступ к приложению.

Панель **Dock** позволяет разместить значки часто используемых приложений для удобства запуска.

❹ Щелкните мышью по значку Arduino, чтобы открыть программу. Вы увидите уведомление о том, что Arduino является программой, загруженной из интернета, и запрос, уточняющий, действительно ли вы хотите ее открыть. В том, чтобы открыть ее, нет никакой угрозы, поэтому нажмите кнопку **Open** (Открыть).



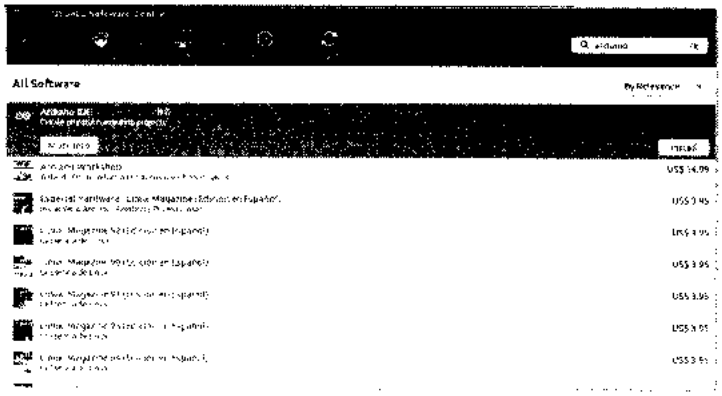
❺ Откроется интерфейс Arduino IDE. Теперь вы можете начинать писать код для программирования микроконтроллера.



Установка Arduino в операционной системе Linux

Установка Arduino в операционной системе Linux требует несколько иного подхода, чем в Windows и macOS. Существует много различных версий Linux, и следующий пример демонстрирует, как установить ее в Ubuntu, одной из самых популярных альтернативных операционных систем.

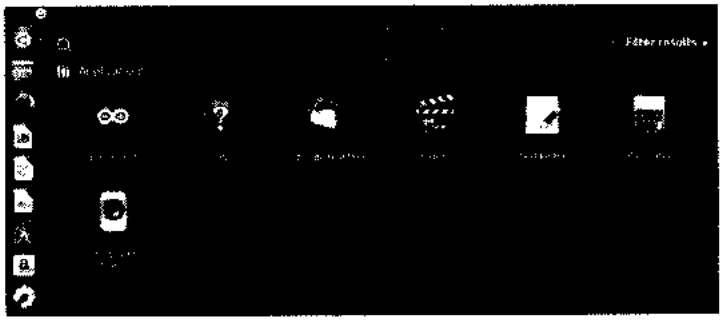
❶ На рабочем столе Ubuntu щелкните мышью по значку **Ubuntu Software Center**. Когда откроется центр приложений, введите в поле поиска слово **Arduino** в верхней правой части окна, как показано ниже.



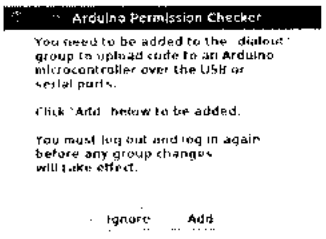
На заметку

К другим популярным версиям Linux относятся Debian, Mint и Fedora.

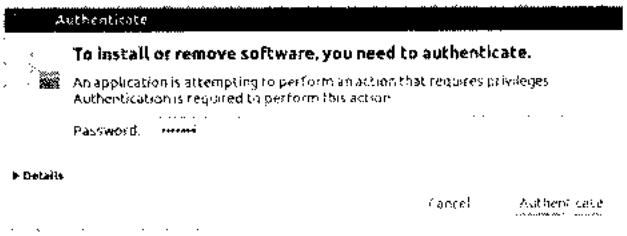
❷ Программа поиска автоматически обнаружит правильную версию Arduino. Когда вы увидите ее в списке результатов поиска, щелкните мышью по соответствующему пункту, чтобы отобразить кнопку **Install** (Установить) в правой части окна. Нажмите эту кнопку.



❸ Когда установка будет завершена, перейдите на рабочий стол и щелкните мышью по значку в верхнем левом углу «Выполнить поиск на вашем компьютере и в Интернете». Первым элементом в разделе **Applications** (Приложения) будет **Arduino**, как показано выше. Щелкните мышью по значку **Arduino**, чтобы открыть программу.

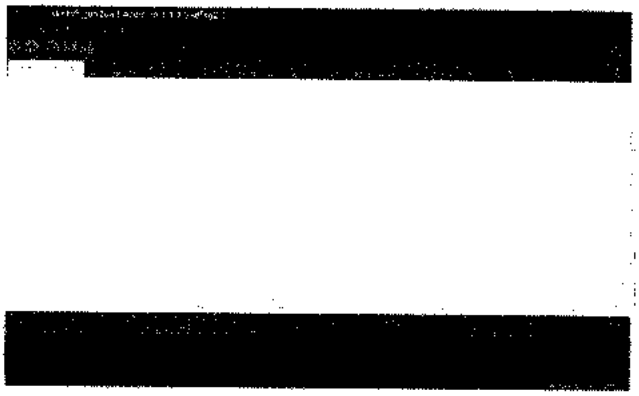


❹ Откроется окно с запросом на добавление в группу dialout, прежде чем вы сможете загружать код в микроконтроллер Arduino. Нажмите кнопку **Add** (Добавить).



Инструкции по установке, которые показаны здесь, предназначены для версии Linux Ubuntu.

5. Подтвердите указанное выше действие вводом своего пароля в Ubuntu в следующем окне, а затем нажмите кнопку **Aunthenticate** (Аутентифицировать).



6. На боковой панели в операционной системе Linux появится значок Arduino. Щелкните по нему мышью, чтобы открыть среду Arduino IDE, как показано выше.

Настройка Arduino

По сути, настройка Arduino сводится к подключению платы к компьютеру и последующей установке драйверов. При этом могут возникнуть проблемы, но, как правило, это происходит с компьютерами под управлением устаревших операционных систем.

Плата Arduino будет работать практически с любой операционной системой. Но чем старше система, тем сложнее будет выполнить настройку Arduino.

Первое, что вам нужно сделать — это подключить плату к компьютеру с помощью USB-кабеля, который можно найти в комплекте. Как только вы сделали

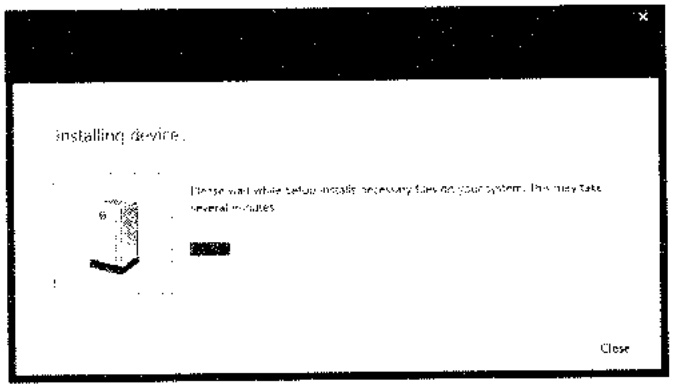
это, на плате должен загореться зеленый светодиодный индикатор, указывающий, что плата получает электропитание.

То, что происходит потом, зависит от вашей операционной системы.

Windows

Операционные системы Windows последних поколений хорошо взаимодействуют с Arduino. Если вы работаете с Windows 8 или 10, настройка пройдет без проблем. Операционные системы предыдущих поколений (Windows XP) и более ранние могут работать нестабильно.

Подключите плату к компьютеру, и спустя некоторое время появится окно **Device Setup** (Настройка устройства). Обратите внимание, что в этом примере мы настраиваем Arduino в операционной системе Windows 8.



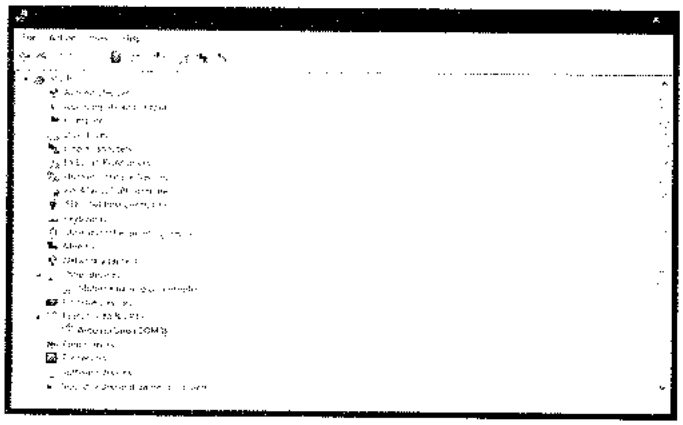
Внимание! На компьютерах под управлением операционной системы Windows Arduino всегда будет использовать порт COM3.

Операционная система Windows 8/10 автоматически обнаружит и установит драйвер Arduino — вам не придется предпринимать никаких действий.

Установка займет всего несколько минут, а затем окно закроется.

Чтобы проверить, корректно ли установлен драйвер, перейдите к компоненту **Device manager** (Диспетчер устройств) окна **Control Panel** (Панель управления). Щелкните мышью по треугольнику в строке **Ports** (Порты), и вы должны

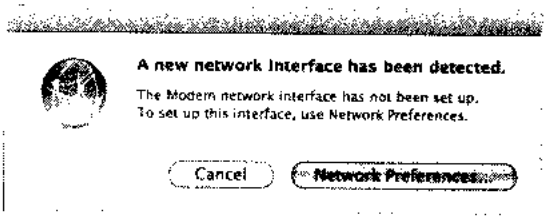
увидеть, что Arduino настроена на использование порта COM3, как показано ниже.



macOS

Процедура настройки Arduino для операционной системы macOS Lion, Mountain Lion, Leopard и Snow Leopard (и более поздних версий) представляет собой несложный процесс и не должна вызывать затруднений. С более ранними версиями могут возникнуть проблемы.

❶ Подключите плату к компьютеру с помощью USB-кабеля. Откроется диалоговое окно, как показано ниже.



Для других версий macOS (или же если у вас возникают проблемы с настройкой) ознакомьтесь с информацией по адресу **forum.arduino.cc**.

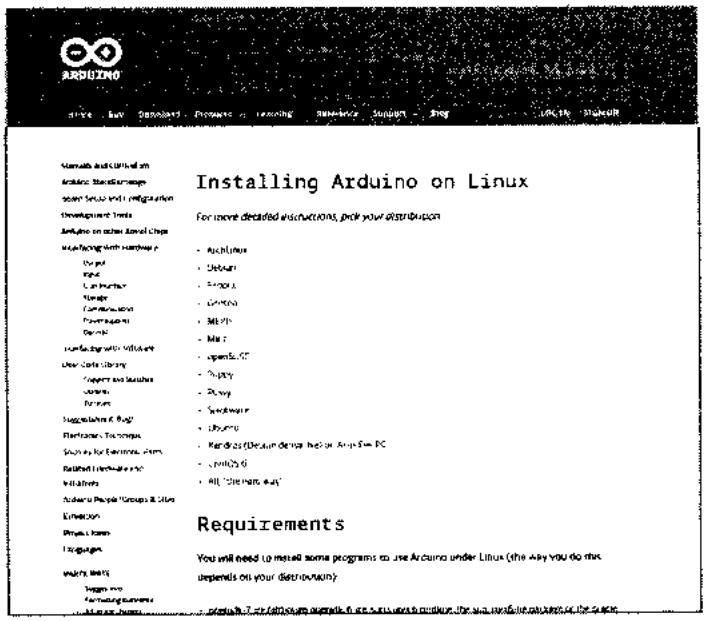
❷ Нажмите кнопку **Network Preferences** (Свойства сети) и в следующем окне кнопку **Apply** (Применить). Если вы взглянете на левую часть данного окна, то увидите, что пункт Arduino отображается с меткой «Не настроено». Не обращайте на это внимание, программное обеспечение Arduino корректно настроено и готово к использованию.

Сообщение «Не настроено» является ошибочным — игнорируйте его.

Linux

При использовании современной операционной системы Linux, такой как Ubuntu 14.04 и более поздних версий, плата Arduino не потребует какой-либо настройки.

Просто подключите плату с помощью USB-кабеля, запустите программное обеспечение Arduino, и можете приступать к работе.



Тем не менее с более ранними версиями могут возникнуть проблемы. Мы рекомендуем вам проконсультироваться на одном из множества веб-сайтов, посвященных таким проблемам.

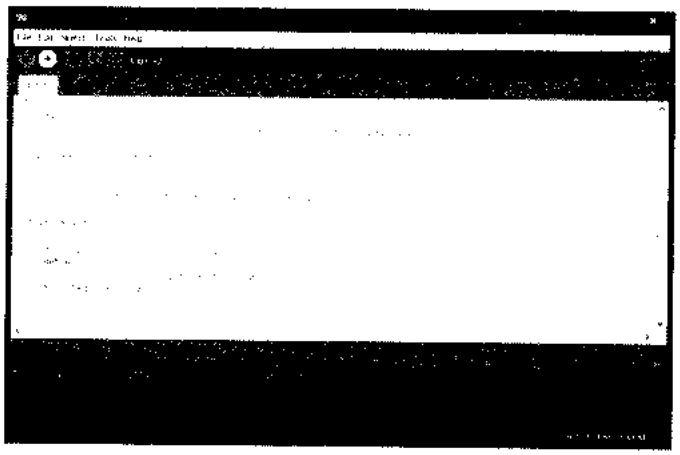
Если у вас возникли проблемы с настройкой Arduino в операционной системе Linux, мы рекомендуем вам ознакомиться с информацией по адресу **playground.arduino.cc/learning/linux**.

Например, перейдите по ссылке **playground.arduino.cc/learning/linux**. Здесь вы найдете подробные инструкции для установки Arduino во всех версиях Linux. Просто выберите свою версию из предложенного списка.

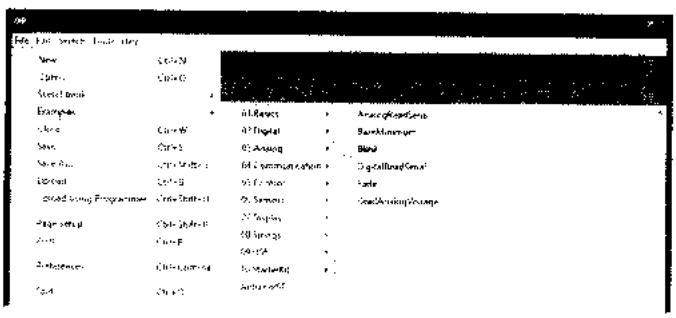
Проверка работоспособности

Выполнив настройку Arduino, прежде чем двигаться дальше, необходимо проверить, действительно ли плата Arduino обменивается данными с компьютером. Сделайте это следующим образом:

1. Подключите плату к компьютеру с помощью USB-кабеля.
2. Запустите программное обеспечение Arduino IDE на своем компьютере.
3. В меню **File** → **Examples** (Файл → Примеры) выберите пункт **Basics** → **Blink**.



4. Так вы загрузите скетч Blink в окно редактирования кода. Щелкните мышью по значку **Upload** (Загрузка) в верхнем левом углу.

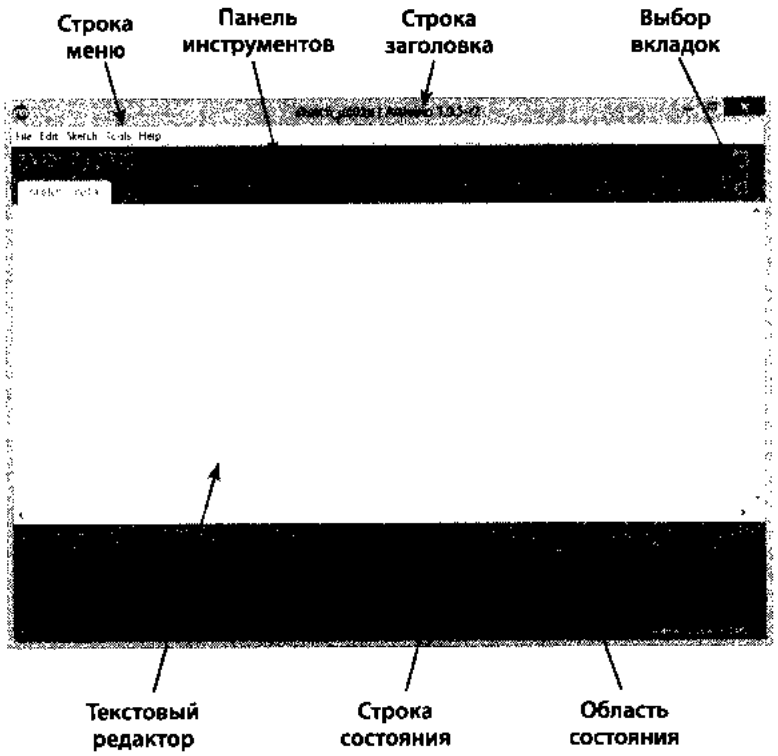


Вы увидите следующее: во-первых, сообщение о том, что загрузка завершена, отображающееся в нижней левой части экрана. И во-вторых, светодиодный индикатор контакта 13 на плате будет загораться и гаснуть. Все это говорит о том, что плата и компьютер обмениваются данными.

Загрузка скетча в микроконтроллер на плате Arduino и наблюдение возможных изменений в поведении платы продемонстрируют, что все работает должным образом.

Среда Arduino

Теперь на вашем компьютере установлена интегрированная среда разработки Arduino (IDE), которая настроена для обмена данными с платой Arduino. При помощи среды вы можете писать код, позволяющий программировать микроконтроллер. На языке Arduino эти программы называются *скетчами*.



В сущности, интерфейс Arduino IDE похож на текстовый процессор. Он включает четыре главных раздела: строку меню, панель инструментов, текстовый редактор и область состояния.

Arduino IDE написана на языке Java и основана на программном обеспечении с открытым исходным кодом, таком как Processing, avr-gcc и др.

Скетчи называются именно так, потому что они основаны на языке программирования Processing, который позволяет пользователям создавать программы так же быстро, как если бы они записывали идею в скетчбук — блокнот для набросков.

Строка заголовка

В верхней части находится строка заголовка, которая отображает название текущего скетча и версию IDE (Arduino 1.0.5-r2).

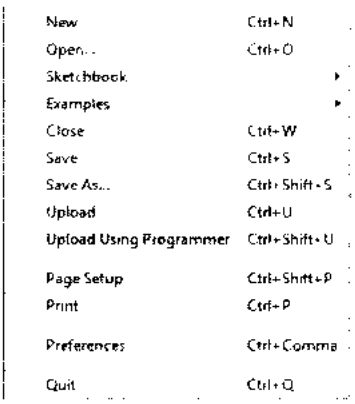
Строка меню

Ниже строки заголовка находится строка меню. Она содержит следующие меню.

Меню «Файл»

В меню **File** (Файл) присутствует ряд команд, от универсальных — **New** (Новый), **Open** (Открыть), **Close** (Закрыть) и т. д. — до команд, присущих только Arduino. Последние включают:

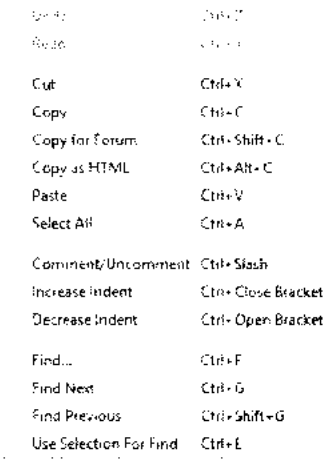
- **Sketchbook** (Папка со скетчами) — эта команда отображает список всех созданных вами скетчей и обеспечивает простой способ их открытия;



- **Examples** (Примеры) — здесь вы можете получить доступ к большому количеству предварительно написанных скетчей. Так как они написаны с использованием открытого исходного кода и бесплатны, вы можете свободно изменять их в своих собственных целях.

Меню «Правка»

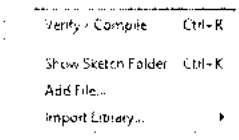
Многие команды в меню **Edit** (Правка) будут вам знакомы. К тем, которые вы, возможно, не знаете, относятся:



- **Copy for Forum** (Копировать для форума) — эта команда копирует код вашего скетча в буфер обмена в формате, совместимым с форумом Arduino;
- **Copy as HTML** (Копировать в HTML) позволяет копировать код скетча в буфер обмена в формате HTML, подходящем для размещения на веб-странице;
- **Comment/Uncomment** (Добавить/Удалить комментарий) — закомментированные блоки текста не загружаются на плату Arduino. Как правило, комментарии используются для того, чтобы напомнить важные детали или объяснить, как работает скетч.

Меню «Скетч»

Меню **Sketch** (Скетч) состоит из нескольких параметров, которые помогают управлять вашими скетчами:



- **Verify/Comply** (Проверить/Компилировать) — эта команда проверяет написанный вами код на наличие ошибок, а затем «компилирует» его в формат, понятный микроконтроллеру;
- **Upload** (Загрузка) — выбрав эту команду, вы загрузите код из текстового редактора на плату Arduino;
- **Upload Using Programmer** (Загрузить через программатор) — эта команда позволяет вам выгружать скетчи с помощью внешнего системного программатора (ISP); в настоящий момент рекомендуем вам игнорировать этот пункт;

На заметку

Существует несколько причин, по которым вы захотите использовать внешний программатор: более быстрое время загрузки, отсутствие последовательного соединения и увеличение памяти, доступной вашему скетчу.

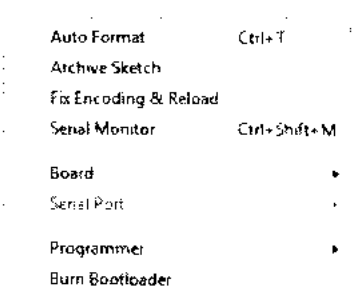
- **Import library** (Подключить библиотеку) позволяет добавить в скетч библиотеку (более подробно мы рассмотрим библиотеки в дальнейшем);
- **Add File** (Добавить файл) добавляет файл в скетч, который появляется в новой вкладке в окне скетча.

На заметку

Программное обеспечение Arduino автоматически сохраняет каждый скетч, который вы пишете, в папку скетча. Команда **Show Sketch Folder** (Показать папку скетча) открывает папку, содержащую текущий скетч.

Меню «Инструменты»

Меню **Tools** (Инструменты) предоставляет различные команды, которые могут оказаться полезными при работе с Arduino:



- **Auto Format** (АвтоФорматирование) выполняет автоформатирование кода скетча, чтобы его было проще читать;
- **Fix Encoding & Reload** (Исправить кодировку и перезагрузить) — исправляет ошибки кодировки, которые могут оказать негативное влияние на скетч;
- **Board** (Плата) — позволяет выбрать вашу плату из списка плат Arduino;
- **Serial Monitor** (Монитор порта) — отображает последовательные данные и полезна при отладке (см. главу 10).

Панель инструментов

Панель инструментов содержит значки наиболее часто используемых команд, чтобы у вас не было необходимости тратить время на их поиск. Ими являются:

Verify (Проверить), **Upload** (Загрузка), **New** (Новый), **Open** (Открыть) и **Save** (Сохранить).

Пункты **Increase Indent** (Увеличить отступ) и **Decrease Indent** (Уменьшить отступ) меню по сути являются командами форматирования, которые позволяют формировать свои скетчи таким образом, чтобы их код было легче читать.

Строка и область состояния

Строка состояния выводит сообщения о состоянии текущих операций. Также она служит индикатором прогресса, чтобы вы видели, как осуществляется загрузка. Область состояния используется для отображения сообщений об ошибке.

Текстовый редактор

Здесь вы вводите код, когда пишете скетчи. Работа редактора во многом похожа на работу текстового процессора. Щелчок правой клавишей мыши по тексту откроет меню редактирования, которое предлагает стандартные команды редактирования, такие как **Cut** (Вырезать), **Copy** (Копировать), **Paste** (Вставить) и т. д.

4

Шилды и библиотеки

Иногда у вас может возникнуть желание избежать лишней работы или вам может просто не хватать времени для того, чтобы самостоятельно разработать и создать монтажную плату. Решение состоит в том, чтобы использовать предварительно собранные платы, известные как шилды. По тем же причинам вы можете использовать заранее написанные разделы кода, известные как библиотеки.

- **Что такое шилд?**
- **Шилды с дисплеями**
- **Шилды для работы со звуком**
- **Шилды для прототипирования**
- **Шилды для видеоигр**
- **Шилды GPS**
- **Шилды электропитания**
- **Шилды для управления электродвигателями**
- **Шилды для передачи данных**
- **Шилды различного назначения**
- **Характеристики шилдов**
- **Библиотеки**

Что такое шилд?

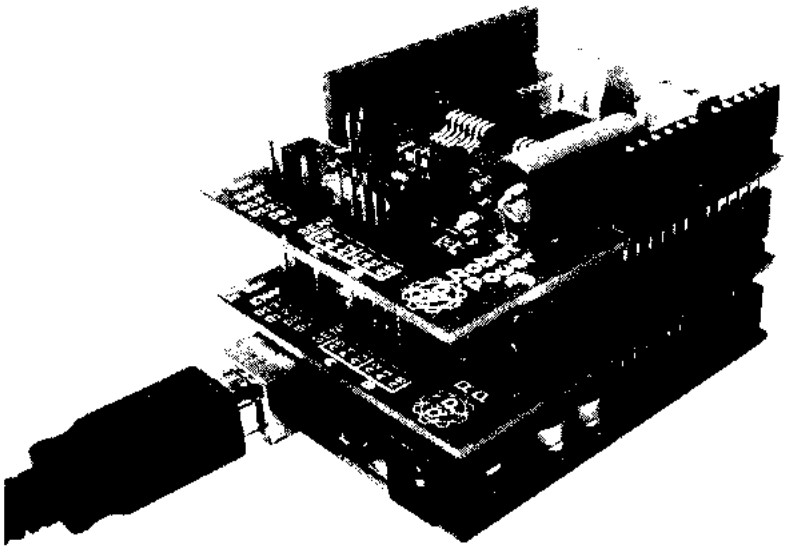
Шилд — это плата расширения, которая увеличивает возможности основной платы Arduino. Шилд просто подключается к штыревой колодке Arduino, создавая таким образом новый «ярус».

Большинство шилдов уже собраны и готовы к использованию, но некоторые поставляются в виде набора и потребуют выполнить сборку. Здесь следует отметить, что большое количество готовых шилдов поставляются без штыревых колодок — это означает, что другие шилды нельзя установить сверху. Решением в данном случае может стать использование наращиваемых клеммных колодок. Они продаются в виде комплектов, обычно содержащих 10-, 8- и 6-штыревые колодки.

Внимание

Прежде чем приобрести шилд, проверьте, содержит ли он в комплекте штыревые колодки. Если нет, купите также несколько клеммных колодок.

Устанавливая шилды друг на друга, мы можем добавлять их в проект, как показано на рисунке ниже. Так вы сможете создавать сложные проекты.



Большое преимуществом шилдов для новичков в Arduino заключается в том, что шилды позволяют встраивать готовые цепи в проекты без необходимости разрабатывать и конструировать их.

На заметку

Со многими шилдами также поставляется код, необходимый для запуска платы.

Подавляющее большинство плат Arduino обладают невысокой стоимостью, их цена обычно составляет от 200 до 3000 рублей. Некоторые из наиболее сложных стоят около 6000 рублей.

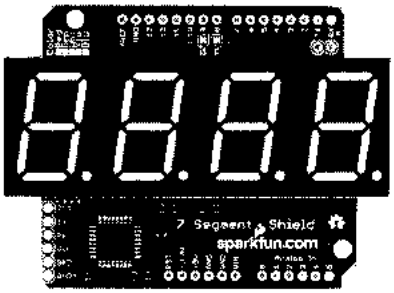
Доступно большое количество шилдов, спектр применения которых довольно широк. На нескольких следующих страницах мы рассмотрим наиболее популярные типы.

Шилды с дисплеями

Электронные устройства отображения информации являются повседневной частью нашей жизни — они имеют широкий диапазон применения. Для этой цели используются светодиодные индикаторы (LED) и жидкокристаллические дисплеи (LCD). Также распространены сенсорные TFT-экраны. Эти технологии доступны и в шилдах Arduino, как можно видеть ниже.

OpenSegment Shield

Этот шилд оборудован четырехзначным семисегментным дисплеем, каждый сегмент которого состоит из одного светодиодного индикатора. Также он оборудован микроконтроллером ATmega328, который позволяет управлять каждым сегментом по отдельности.



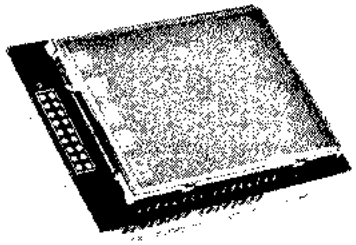
На заметку

Шилд OpenSegment доступен с разными цветами индикатора, включая красный, синий, желтый и зеленый.

Шилдом OpenSegment можно управлять тремя способами: через последовательный интерфейс TTL, последовательный интерфейс SPI или последовательный I2C-интерфейс.

Его функции включают в себя настраиваемую скорость передачи данных, устанавливаемый уровень яркости и индивидуальное управление сегментами каждой цифры. Дисплей позволяет отображать цифры, большое количество букв и несколько специальных символов.

TFT Touch Shield



Разработанный компанией Adafruit, этот шилд оборудован относительно большим цветным сенсорным дисплеем и встроенным слотом карт памяти microSD.

Размер экрана составляет 2,8 дюйма по диагонали, подсветка обеспечивается четырьмя светодиодными индикаторами, поддерживается разрешение 240 × 320 и множество различных оттенков цвета.

К преимуществам можно отнести резистивность дисплея, что позволяет ему распознавать касания в любом месте экрана.

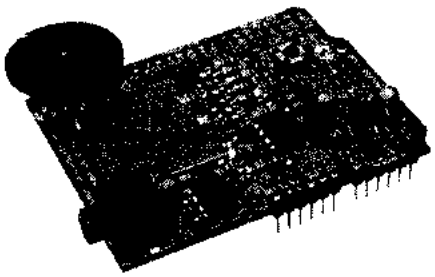
На заметку

Сенсорный TFT-экран также связан с библиотекой, которая позволяет определять координаты касаний по осям *x*, *y* и *z*.

Шилд поставляется в сборке и не потребует пайки. Просто подключите его и загрузите поставляемую библиотеку. На дисплей можно выводить пикселы, линии, прямоугольники, круги и текст.

Шилды для работы со звуком

Adafruit Wave Shield



Wave Shield — недорогая плата для воспроизведения аудиоданных при помощи Arduino. Она поддерживает воспроизведение звука с частотой до 22 кГц и 12-битные несжатые аудиофайлы любой длительности. Кроме того, шилд

оборудован цифро-аналоговым преобразователем (ЦАП), фильтром и операционным усилителем для вывода качественного звука.

SD-карта и динамик не входят в комплект поставки шилда Wave Shield, их необходимо приобрести отдельно.

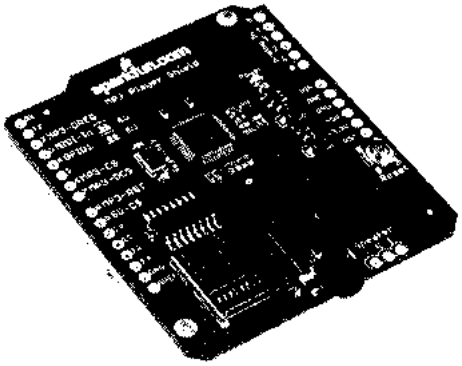
Этот шилд имеет необходимую библиотеку Arduino для быстрого использования. Скопируйте несжатые wave-файлы на SD-карту и подключите ее к шилду для воспроизведения.

Аудиоданные воспроизводятся асинхронно, поэтому Arduino может выполнять другие задачи во время воспроизведения. Громкость можно регулировать с помощью расположенного на плате регулировочной головки потенциометра.

Обратите внимание, что Wave Shield поставляется в виде комплекта и, чтобы собрать его, потребуется пайка.

MP3 Player Shield

Шилд MP3 Player Shield компании Sparkfun фактически превращает вашу Arduino в полноценный автономный MP3-плеер. В качестве бонуса он также декодирует аудиофайлы формата Ogg Vorbis, AAC, WMA и MIDI.



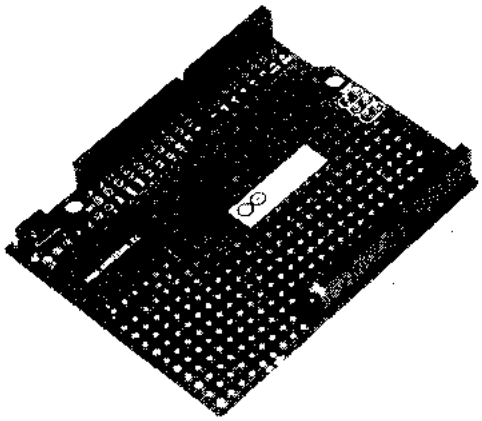
Также в комплект входит устройство чтения карт microSD, упрощающее загрузку файлов, и разъем TRS3,5 мм (mini jack), к которому можно подключить наушники или активные колонки.

Плата поставляется полностью собранной, за исключением штыревых колодок, которые необходимо будет припаять. Рекомендуется использовать 6- и 8-клеммные колодки с длинными ножками.

Шилды для прототипирования

Proto Shield Rev3

Proto Shield несколько отличается от обычных шилдов Arduino тем, что фактически не предоставляет каких-либо определенных функций. Вместо этого данный шилд представляет собой платформу, которая позволяет создавать цепи любого типа. Их можно создавать либо на постоянной основе (путем припаивания компонентов), или же временно, в экспериментальных целях.



На заметку

Шилд Proto Shield позволяет вам объединять пользовательские цепи и Arduino в один модуль.

Шилд Proto Shield не только обеспечивает возможность подключения всех входов и выходов Arduino, но и предоставляет достаточно места для монтажа компонентов — поверхностного и через отверстия.

Если вы приобрели данный шилд в виде комплекта, вы получаете не только саму плату, но и следующие компоненты:

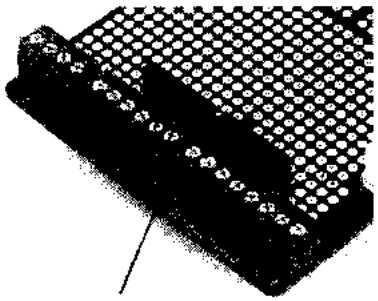
- 1 прямой линейный коннектор 40 × 1
- 1 прямой линейный коннектор 3 × 2
- 2 кнопочных выключателя платы
- 1 светодиодный индикатор красного цвета, 1 светодиодный индикатор желтого цвета и 1 светодиодный индикатор зеленого цвета
- 15 резисторов различного сопротивления

К преимуществам Proto Shield относится кнопка перезагрузки, большая рабочая область и разъем ICSP.

Шилд Proto Shield поставляется либо в сборке, либо в виде набора.

ProtoScrew Shield

PhotoScrew представляет собой шилд, который очень похож на Proto Shield. Разница заключается в том, что PhotoScrew оборудован клеммами с винтовым креплением, которые подключены к контактам. Поэтому данный шилд идеально подойдет тем, кому не нравится заниматься пайкой, или же тем, кому не хватает времени на сборку-разборку.



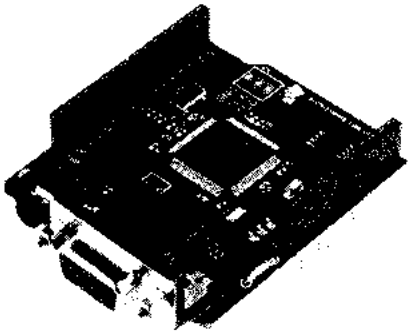
Данный шилд продается в виде комплекта и требует дальнейшей сборки.

Шилды для видеоигр

Arduino можно использовать для такого популярного вида деятельности, как электронные игры.

Gameduino

Шилд Gameduino — это игровой адаптер для Arduino. Он оборудован разъемом VGA, который позволяет передавать видеоизображение на любой VGA-совместимый монитор, проектор или другое устройство вывода изображения. Звуковое сопровождение можно передавать с помощью разъема TRS3,5 мм (mini-jack).



К другим возможностям шилда относится вывод видеоизображения в разрешении 400×300 пикселов, 15-битная внутренняя обработка цвета и символьный фон с разрешением 512×512 пикселов. Шилд поставляется в полностью собранном виде.

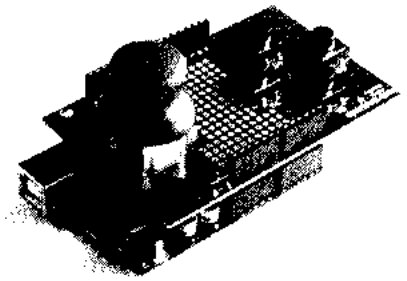
С помощью Gameduino вы можете как играть в игры, так и создавать их. Несмотря на то, что игровые характеристики ограничиваются 8 битами, данный шилд более мощный, чем 8-битные игровые консоли в 80-х годов, такие как Oric-1, Acorn Electron, ZX Spectrum, C64 и т. д.

Усовершенствованная версия этого шилда называется Gameduino 2. Она включает ряд улучшений, например встроенный сенсорный экран, превращающий шилд в игровой контроллер «все-в-одном».

На заметку

Помимо многих иных улучшений, шилд Gameduino 2 оборудован сенсорным дисплеем диагональю 4,3 дюйма, встроенным графическим процессором, акселерометром и слотом для карт памяти microSD.

Joistick Shield



В то время как Gameduino Shield позволяет играть в игры, с его помощью нельзя осуществлять управление в играх. Для решения этой проблемы используйте шилд Joistick компании Sparkfun.

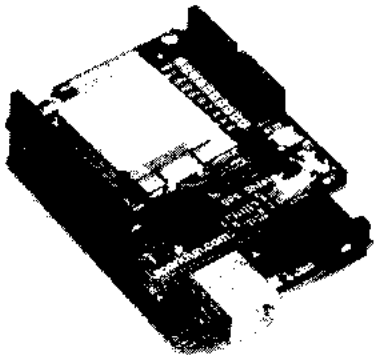
Эта портативная плата предоставляет все функции, которые можно найти в современном игровом контроллере. Вы получаете эргономичный джойстик со встроенной кнопкой-переключателем, а также четырьмя дополнительными кнопками, которые можно использовать для навигации и управления игровым процессом.

Шилд Joistick поставляется в виде набора, и поэтому потребует дополнительной сборки.

Шилды GPS

Сегодня существует множество приложений, которые требуют точного определения местоположения или возможности его отслеживания. Глобальная система позиционирования (GPS) обеспечивает такую возможность, и в связи с этим пользователю Arduino предоставляется ряд шилдов GPS:

GPS Shield Retail Kit



Этот шилд поставляется в разобранном виде и содержит в наборе GPS-приемник (другая версия этого шилда продается без приемника). По желанию можно использовать приемник другой модели, но для этого вам понадобится приобрести и установить соответствующие разъемы.

Этот шилд позволяет очень точно определить местоположение, и с его помощью вы можете создать ряд навигационных проектов, от контроля передвижения ваших детей до создания геолокационных сервисов. Переключатель DLINE/UART переключает вход/выход GPS-приемника между стандартными разъемами **TX/RX** на плате Arduino и любыми цифровыми контактами на плате Arduino. Обратите внимание, что для загрузки кода из среды Arduino переключатель должен быть установлен в режим DLINE.

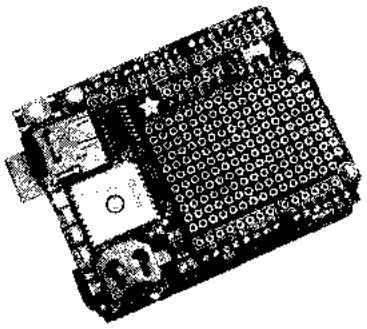
На заметку

Дополнительное преимущество шилда GPS заключается в том, что он способен получать точное время.

Ultimate GPS Logger Shield

Этот шилд можно использовать не только для спутниковой навигации, но также для протоколирования или сохранения данных GPS. Это осуществляется за счет встроенного слота для чтения SD-карт, который позволяет вам сохранять данные на SD-карту.

К другим преимуществам шилда относится большая область рабочей поверхности для прототипирования и возможность подключения внешней антенны. В последнем случае это прекрасная возможность для разработки проектов устройств, заключаемых в корпус.



Просто подключите внешнюю GPS-антенну к шилду, и модуль задействует ее автоматически.

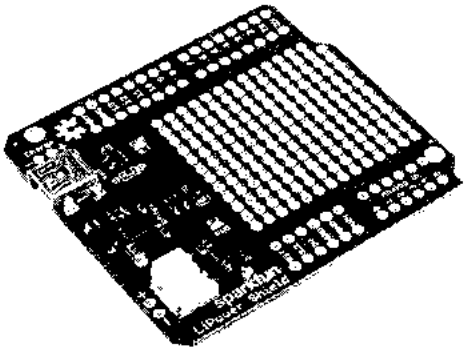
Совет

Возможность подключения SD-карты позволяет хранить гораздо больше данных, чем в лимитированной памяти на плате Arduino.

Шилды электропитания

LiPower Shield

Если вы хотите создать по-настоящему мобильное устройство на основе Arduino, вам необходимо обеспечить электропитание платы от батарей. Хотя вы можете использовать батареи типа AA и AAA, шилд LiPower предоставляет другой вариант — литиевые аккумуляторные батареи.

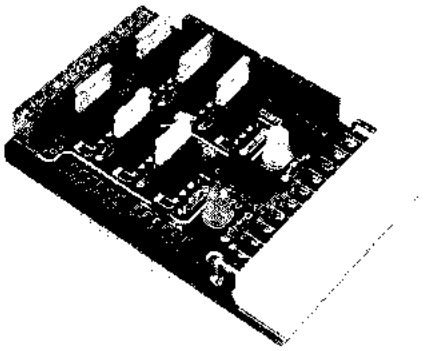


Проблема здесь состоит в том, что можно подключать только аккумуляторы с напряжением 3,7 В. Однако шилд LiPower позволяет поднять напряжение с 3,7 В до 5 В.

Кроме того, этот шилд может отслеживать состояние заряда батареи и отправлять оповещение, когда заряд снижается до определенного уровня.

Встроенная цепь зарядки настроена на 100 мА. На плате есть порт mini-USB. Он позволяет заряжать аккумулятор от USB-источника или подавать на стабилизированную линию 5 В.

Power Driver Shield



Плата Arduino не предоставляет электропитание высокой мощности, поэтому ее не получится использовать для проектов, требующих больших объемов электроэнергии. Шилд Power Driver призван решить данный вопрос.

На одной стороны платы находится разъем электропитания **ATX**. Он аналогичен разъемам, используемых в компьютерных блоках питания. Поэтому все, что вам понадобится, это взять одно из таких устройств и подключить его к шилду.

Из-за высоты шести транзисторов большой мощности вам придется использовать сдвоенные клеммные колодки, если вы хотите установить плату на шилд Power Driver.

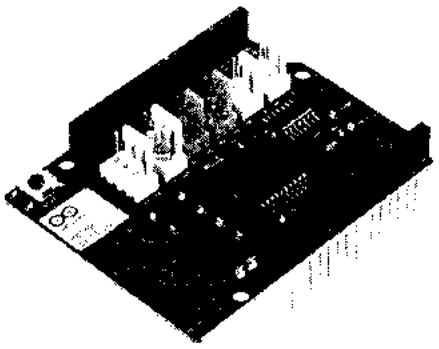
На другой стороне шилда находятся шесть выходных клемм широтно-импульсной модуляции (ШИМ), которые вы можете использовать, чтобы обеспечить питание вашего проекта током силой до 30 А.

Шилд Power Driver поставляется в разобранном виде и требует сборки.

Шилды для управления электродвигателями

Электродвигатели играют важную роль во многих проектах Arduino, в частности связанных с робототехникой. Существуют специальные шилды для управления электродвигателями. Мы рассмотрим две самые популярные модели.

Motor/Stepper/Servo Shield



Этот шилд обеспечивает практически все, что вам может понадобиться для управления электродвигателями. Он позволяет вам управлять не одним, а фактически несколькими электродвигателями — до двух 5-вольтовых электродвигателей с сервоприводами, двух шаговых электродвигателей или же четырех двунаправленных электродвигателей постоянного тока.

Подключение электродвигателей осуществляется довольно просто благодаря большим коннекторам клеммной платы (они также позволяют вам удаленно обеспечить электродвигатели электроэнергией).

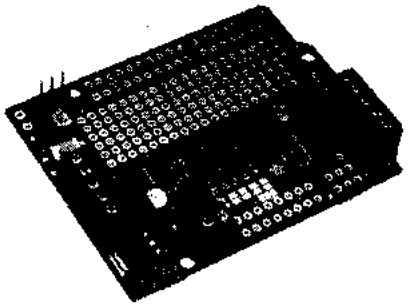
Этот шилд обеспечивает защиту от перегрева.

Шилд может обрабатывать ток силой до 1,2 А на канал и ограничен максимальной силой тока в 3 А. На плате установлен чип со специальным драйвером, который отвечает за управление вращением и скоростью электродвигателя. Для управления электродвигателем требуется всего два контакта.

Motor Shield R3

Шилд Motor Shield R3 менее функционален, чем описанный выше Motor/Stepper/Servo Shield, так как он может управлять либо двумя электродвигателями постоянного тока, либо одним шаговым электродвигателем.

Однако он может обрабатывать ток силой до 2 А на канал и ограничен максимальной силой тока в 4 А, позволяя таким образом устанавливать более мощные электродвигатели.



Плата имеет два канала с четырьмя контактами на канал, позволяя переключать направление вращения, управлять скоростью вращения и контролировать ток, подаваемый на электродвигатель.

Для работы шилд Motor Shield R3 должен быть подключен к внешнему источнику электропитания, например адаптеру с напряжением 7–12 В.

На заметку

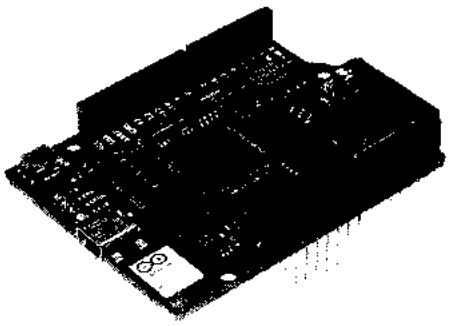
Шилд Motor Shield R3 совместим с модулями TinkerKit, что дает вам возможность подключить модуль TinkerKit к плате Arduino.

Шилды для передачи данных

Без передачи данных сегодня невозможен ни один стоящий проект. Ниже рассмотрены два наиболее популярных шилда для передачи данных.

Wi-Fi Shield

Шилд Wi-Fi позволяет подключать Arduino к интернету через беспроводную сеть стандарта 802.11. Таким образом, проекты, требующие подключения к интернету, становятся мобильными.



Шилд способен подключаться как к открытым сетям, так и к сетям с защитой WPA2 Personal или WEP.

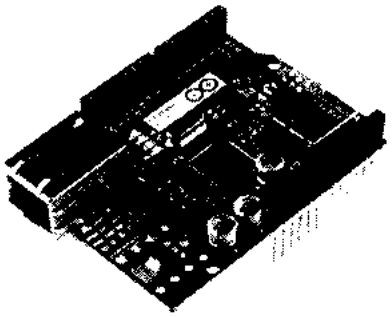
К преимуществам шилда относится встроенный слот карт памяти microSD, который можно использовать, например, для хранения файлов, передаваемых по сети, или результатов поиска.

Аббревиатура SSID расшифровывается как «идентификатор набора услуг». По сути, это идентификатор беспроводной сети.

Обратите внимание, что беспроводная сеть должна открыто транслировать свой идентификатор SSID, чтобы шилд мог к ней подключиться.

Ethernet Shield

Шилд Ethernet позволяет плате Arduino подключаться к интернету напрямую без помощи компьютера. Как и в случае с шилдом Wi-Fi, о котором говорилось выше, здесь есть встроенный слот для карт памяти microSD, который обеспечивает возможность хранения данных.



Этот шилд поддерживает до четырех одновременных подключений, которые соединяются с сетью через стандартный Ethernet-разъем RJ45.

Этот шилд также включает устройство управления с обратной связью, которое обеспечивает подобающую перезагрузку при включении электропитания. Кнопка перезагрузки также одновременно перезагружает плату Arduino.

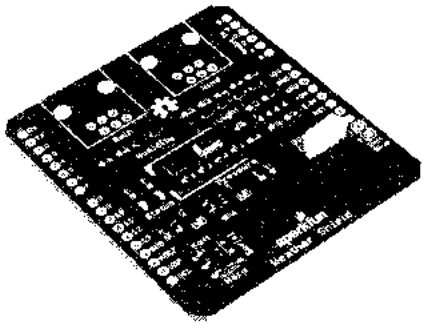
Шилд Ethernet подключается к плате Arduino с помощью длинных контактных колодок, установленных на плате. Благодаря этому остаются доступными штыревые контакты, и вы можете установить сверху другой шилд.

Шилды различного назначения

На с. 55–67 мы рассмотрели платы из наиболее популярных категорий. В то же время существует множество других шилдов, которые обеспечивают в равной степени полезные, хоть и редко востребованные функции. К ним относятся перечисленные ниже шилды.

Weather Shield

Шилд Weather представляет собой простую в использовании плату, оборудованную встроенными датчиками, предоставляющими доступ к барометрическому давлению, относительной влажности, освещенности и температуре.



Внимание

Шилд Weather имеет погрешности измерения температуры ± 3 °C, давления ± 50 Па и влажности ± 2%.

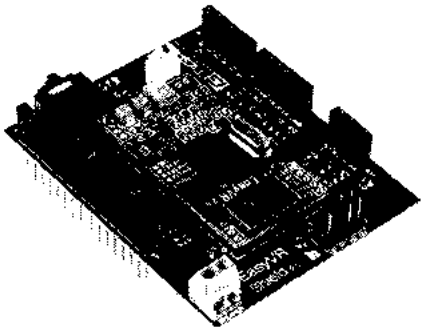
Также доступна возможность установки дополнительных датчиков, например скорости и направления ветра, дождя и GPS.

Шилд Weather может работать от источника тока напряжением от 3,3 до 16 В и имеет встроенные регуляторы напряжения и преобразователи сигнала. Обратите внимание, что колодки, коннекторы и дополнительные датчики нужно будет приобрести отдельно.

EasyVR Shield 2.0

В последнее время технология распознавания речи значительно улучшилась и вы можете добавить эту возможность в свои проекты Arduino с помощью шилда с технологией распознавания речи компании Sparkfun.

EasyVR2.0 — это универсальная плата для распознавания речи, разработанная с целью добавления многофункциональных, надежных, а также эффективных и экономичных возможностей для распознавания речи практически для любого приложения.



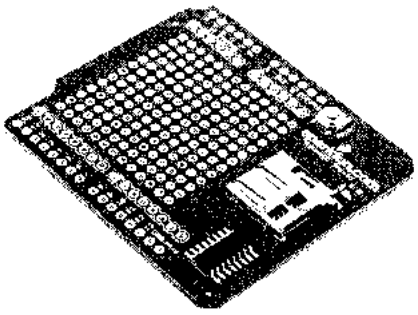
Этот шилд оборудован разъемом для подключения микрофона, аудиовыходом на колонки сопротивлением 8 Ом и аудиоразъемом TRS3,5 мм (mini-jack) для наушников. Также здесь присутствует программируемый светодиодный индикатор, предназначенный для обратной связи во время распознавания голоса.

На заметку

EasyVR Shield поддерживает 28 встроенных, не зависящих от диктора команд. Они доступны на английском, японском, итальянском, французском, немецком и испанском языках.

MicroSD Shield

Шилд MicroSD предназначен для использования в проектах, оперирующих большими объемами данных аудио-, видео-, графических и журналируемых данных. Arduino Uno (и другие платы Arduino) имеют ограниченный объем памяти, которого вряд ли будет достаточно для приложений такого типа.



Шилд MicroSD позволяет решить эту проблему, обеспечивая вашу плату Arduino функциональными возможностями подключения устройства внешней памяти.

Это такой же тип памяти, который используется в цифровых камерах и MP3-плеерах. Благодаря этому шилду вы можете добавить в ваши проекты гигабайты дискового пространства.

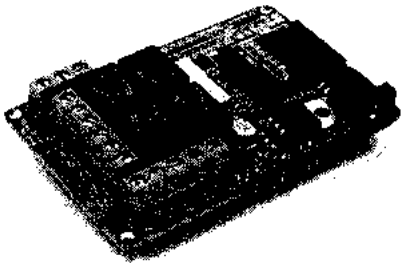
На заметку

Платы Arduino имеют ограниченный объем памяти. Если вам понадобится большой объем памяти, необходимо использовать такой шилд, как MicroSD.

Данный шилд оборудован разъемом для подключения для карты памяти microSD, LED-индикатором электропитания (красного цвета) и кнопкой перезагрузки. Также стоит отметить, что на шилде не предустановлены колодки.

Relay Shield

Шилд Relay представляет собой простую плату, которая обеспечивает возможность управления одним электромеханическим реле для коммутации нагрузки до 24 В постоянного тока или 40 В переменного тока при силе тока до 5 А.



Реле — это однополюсный переключатель, что означает, что его можно использовать как нормально разомкнутое или нормально замкнутое реле.

На заметку

При работе в качестве нормально разомкнутого реле шилд Relay способен выдерживать нагрузку до 10 А.

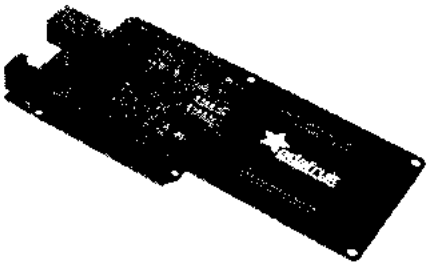
Это означает, что вы можете выбрать либо включение, либо выключение нагрузки при подаче высокого выходного сигнала с вашей платы Arduino.

Шилд получает электропитание напряжением 5 В напрямую от платы Arduino и совместим с большинством плат Arduino (не только с Uno), которые поддерживают напряжение в 5 В.

NFC Shield

Стандарт ближней радиосвязи (NFC) — это маломощная беспроводная линия связи с малым радиусом действия, по которой можно передавать малые

объемы данных между двумя соприкасающимися устройствами, оборудованными подходящим образом.



Эта технология все чаще используется в смартфонах и позволяет оплачивать покупки, просто дотронувшись телефоном до терминала, оборудованного чипом NFC.

Шилд NFC обеспечивает вашу плату Arduino функциональностью NFC. Он не только может считывать NFC-теги, но и осуществляет запись на них.

Также вы можете приобрести NFC-теги, чтобы использовать их со своей платой Arduino, оснащенной технологией NFC. Они имеют разную форму и размеры.

Также шилд может выступать в качестве NFC-тега для двунаправленной связи со смартфонами и планшетами, оснащенными технологией NFC.

Этот шилд поставляется в разобранном виде и поэтому требует пайки.

USB Host Shield

Шилд USB Host позволяет подключить USB-устройство к плате Arduino. Он содержит все цифровые и аналоговые схемы, необходимые для реализации максимальной скорости передачи данных через интерфейс USB2.0 с помощью вашей платы Arduino.

С помощью этого шилда вы можете подключать HID-устройства (мыши, клавиатуры и т. д.), игровые контроллеры, цифровые камеры, устройства хранения данных (карты памяти, жесткие диски и т. д.) и Bluetooth-адаптеры.

Шилд USB Host совместим с модулями TinkerKit.

Go-Between Shield

Этот шилд предназначен для тех ситуаций, когда вы не уверены в том, будет ли один шилд взаимодействовать с другим. Часто возникает ситуация, когда при создании проектов Arduino появляется проблема, связанная с тем, что двум шилдам необходим один и тот же контакт колодки одновременно.

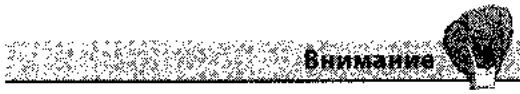
Шилд Go-Between может устранить проблемы несовместимости такого типа, буквально разместившись «между» двумя несовместимыми шилдами, и эффективно переназначить контакт(ы) по мере необходимости.

Характеристики шилдов

Теоретически не существует ограничений относительно количества шилдов, которые можно разместить друг на друге. Но на практике следует учитывать следующие моменты.

- **Размеры** — вы можете обнаружить, что некоторые шилды содержат компоненты большей высоты, чем контактные разъемы платы, в которые они подключаются, что делает подключение невозможным. Эту проблему можно решить, используя клеммные колодки.
- **Конфликт контактов** — убедитесь, что шилду не нужен контакт для передачи данных, который уже используется другим шилдом. Определить это может быть сложно, и вам, возможно, потребуется взглянуть на схематические планы производителя шилда. Обратите внимание, что совместное использование контактов **GND**, **5V**, **3V**, **RESET** и **IOREF** не вызывает конфликтов.
- **Блокировка** — некоторые шилды блокируют доступ к выходам и входам шилда, находящегося под ними, что делает весь процесс настройки бесполезным.
- **Электропитание** — некоторые шилды потребляют большое количество электроэнергии. Если у вас возникнет подобная ситуация, вы можете обнаружить, что плата Arduino не может обеспечить достаточный уровень мощности. Своими повышенными требованиями относительно электропитания известны шилды с жидкокристаллическим дисплеем (LCD) и шилды с возможностью беспроводной связи. У вас также могут возникнуть подобные проблемы с электропитанием при применении шилдов, работающих под напряжением 3 В.

  Простое и легкое решение этой проблемы заключается в использовании внешнего источника электропитания.



Всегда изолируйте шилды, использующие радиочастоты, размещая их на плате как можно дальше друг от друга.

- **Взаимные помехи** — шилды, использующие какой-либо из типов радиочастотной связи, могут вступать в конфликт друг с другом, особенно, если они используют одни и те же частоты. Для решения любых вопросов, связанных с этой проблемой, может быть достаточно разместить эти шилды на максимальном удалении друг от друга.

  Другой потенциальной проблемой являются электрические помехи. Причиной этого могут быть платы с электрическим шумом, такие как шилды для управления электродвигателями.

- **Программное обеспечение** — шилды различаются по своим требованиям к программному обеспечению. Объединение шилдов, использующих большие объемы кода, может вызвать проблемы за счет того, что потребуется больше памяти, чем обеспечивает плата Arduino; либо за счет инициирования конфликтов на уровне использования ресурсов.

Библиотеки

Фактически библиотека Arduino представляет собой модифицированный скетч, который можно легко использовать совместно с другими пользователями, а также для упрощения обновления кода.

Библиотеки позволяют быстро добавлять в скетч функции, таким образом повышая его функциональные возможности. Например, вы можете запрограммировать свою плату Arduino на использование определенного типа оборудования. Вместо того, чтобы писать необходимый код самостоятельно, вы просто можете импортировать его в свой скетч в виде уже существующего скетча.

Существует большое количество причин для создания библиотек. К ним относится упрощение использования, организация кода, повышение читабельности кода, а также логическая децентрализация.

В программном обеспечении Arduino уже может находиться необходимая библиотека; также вы можете загрузить нужный скетч из интернета. В интернете вы найдете документированный код огромного количества популярных проектов и функций многих пользователей Arduino. Вы также свободно можете интегрировать такой код в свои скетчи.

Когда вы станете опытным программистом, то сможете писать библиотеки самостоятельно. Сейчас мы рассмотрим некоторые библиотеки, которые поставляются вместе с платой Arduino Uno.

Вы найдете их, выбрав команду меню **Sketch → Import Library** (Скетч → Подключить библиотеку) в окне Arduino. Доступные библиотеки охватывают наиболее популярные категории проектов Arduino, и вы обязательно найдете здесь что-то, что будет полезно в одном из ваших собственных проектов.

- **EEPROM** — это сокращение фразы «Электрически Программируемое Постоянное Запоминающее Устройство». Этот тип энергонезависимой памяти используется в компьютерах и других электронных устройствах для хранения небольших объемов данных, которые необходимо сохранить при отключении электропитания. Библиотека позволяет вам выполнять запись и чтение с компонента EEPROM на плате Arduino.
- **Ethernet** — вы будете использовать эту библиотеку при подключении шилда Ethernet к плате Arduino. Она позволяет шилду подключаться к интернету в качестве либо сервера, либо клиента.
- **LiquidCrystal** — данная библиотека позволяет плате Arduino управлять жидкокристаллическими дисплеями (LCD). Библиотека основана на чипсете Hitachi HD44780 (или совместимом аналоге), который встречается в большинстве текстовых дисплеев.

Большинство библиотек Arduino, которые вы можете найти в интернете, распространяются с открытым исходным кодом. Это означает, что вы можете использовать их в собственных проектах.

- **GSM** — Глобальная система мобильной связи (GSM) представляет собой международную сотовую связь, которая доступна в Европе и других частях мира. Эта библиотека позволяет плате Arduino выполнять большинство операций, которые вы можете делать с помощью телефона, в котором присутствует GSM-модуль: совершать и принимать голосовые вызовы, отправлять и принимать SMS-сообщения и подключаться к интернету через протокол GPRS.

- **SD** — данная библиотека используется с шилдами, которые позволяют подключать карту памяти SD. Данные карты памяти широко используются в портативных устройствах, таких как смартфоны, цифровые камеры, GPS-навигаторы и т. д. Библиотека SD позволяет осуществлять как запись, так и чтение с SD-карт и поддерживает файловые системы FAT16 и FAT32.
- **Wi-Fi** — при использовании с платой Arduino шилда Wi-Fi эта библиотека позволяет плате Arduino подключаться к интернету. Она может функционировать либо как сервер, принимая входящие подключения, либо как клиент, осуществляя исходящие подключения. Библиотека поддерживает шифрование WEP и WPA2 Personal, но не WPA2 Enterprise.

Протоколы защиты данных (WEP) и защищенного доступа Wi-Fi II (WPA2) защищают сети посредством шифрования передаваемых данных.

- **Stepper** — эта библиотека позволяет вам управлять униполярными и биполярными шаговыми электродвигателями с помощью платы Arduino. Для использования этой библиотеки вам потребуется шаговый электродвигатель с аппаратным обеспечением для его управления.
- **Servo** — эта библиотека используется в сочетании с любительскими сервоприводами. Это электродвигатели, которые имеют встроенные механизмы и вал, а также точное управление. Библиотека Servo поддерживает до 12 электродвигателей на большинстве плат Arduino и до 48 — на Arduino Mega.
- **Firmata** –стандартный протокол связи, который позволяет управлять платой Arduino с помощью программного обеспечения на компьютере. Библиотеку Firmata также можно использовать для выборочной отправки и получения данных между устройством Arduino и программным обеспечением, работающим на вашем компьютере.
- **SPI** — последовательный периферийный интерфейс (SPI) — это интерфейсная шина, обычно используемая для отправки данных между микроконтроллерами и небольшими периферийными устройствами, такими как регистры сдвига, датчики и SD-карты. Когда речь идет про SPI-соединение, всегда существует ведущее устройство, которое управляет периферийными устройствами. Библиотека SPI позволяет связываться с SPI-устройствами с помощью платы Arduino в качестве ведущего устройства.

5

Инструменты и методы работы

Наличия одной лишь платы Arduino недостаточно. Также вам понадобятся инструменты, оборудование и знание методов работы, таких как пайка и проектирование.

- Макетные платы
- Пайка
- Адаптеры электропитания
- Оборудование для тестирования и диагностики
- Программное обеспечение для проектирования
- Электрические схемы

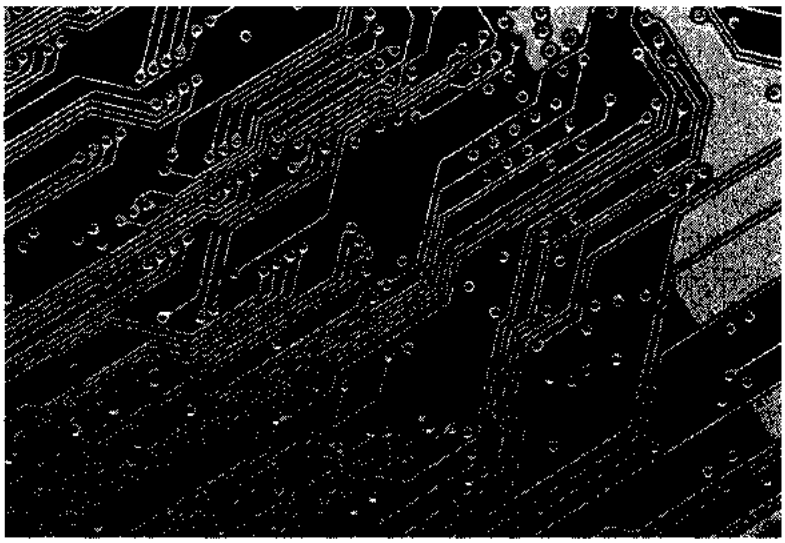
Печатные платы

На с. 30–33 мы рассмотрели беспаечные макетные платы и их использование для быстрой сборки и тестирования прототипных цепей. Их недостаток заключается в том, что компоненты легко извлечь случайно, так как они не припаяны.

Таким образом, как только сборка на макетной плате продемонстрировала свою работоспособность, самое время перенести компоненты на что-то, обладающее большим сроком службы. В данном случае у вас есть несколько вариантов.

Печатные платы

Первый вариант заключается в использовании печатной платы, иначе известной как PCB (Printed Circuit Board — печатная макетная плата). Они создаются на базе непроводящей основы, обычно стекловолокна, с проводящей медной фольгой, прикрепленной с одной или двух сторон. В процессе травления удаляется лишняя медь, оставляя токопроводящие медные дорожки, которые образуют цепи, как показано на рисунке ниже.



Просверленные в плате отверстия позволяют подключать компоненты к цепям, а затем припаивать их, создавая таким образом постоянное подключение.

Печатные платы могут быть односторонними (один слой меди), двухсторонними (два слоя меди) или многослойными. В последних стенки сквозных

отверстий имеют гальваническое покрытие, таким образом, соединяя слои. Усовершенствованные печатные платы обладают компонентами, встроенными непосредственно в основу самой платы.

При создании медных проводящих дорожек на печатной плате используются различные методы. К ним относится трафаретная печать, фотогравирование, фрезеровка печатных плат и гальванизация.

Сегодня серийно выпускаемые печатные платы можно обнаружить практически в любом электрическом и электронном оборудовании.

Как и следовало ожидать, если вы закажете выпуск отдельной печатной платы в какой-либо коммерческой фирме, это обойдется вам в довольно крупную сумму денег. Однако существует другой способ. Выполните поиск в интернете, и вы найдете большое число онлайн-сервисов по производству печатных плат профессионального качества по доступной цене. Один из подобных сервисов можно найти по адресу **www.expresspcb.com**.

Однако может случиться так, что цены, указанные на подобных онлайн-сервисах, все равно больше тех, которые вы готовы заплатить, или же печатная плата нужна вам быстрее, чем они могут ее доставить. В таком случае вы можете создать плату самостоятельно.

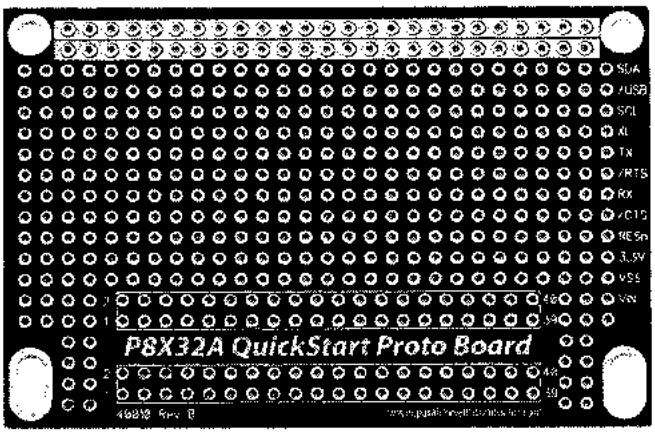
Процесс создания своей собственной печатной платы прекрасно описан на сайте **www.instructables.com/id/Making-A-Customized-Circuit-Board-Made-Easy**.

Для этого потребуется такое оборудование, как лазерный принтер, прозрачная пленка, медный протравливатель, проявитель для позитивного фоторезиста, пустая предварительно сенсибилизированная печатная плата, программное обеспечение CAD и еще много всего.

Это относительно простая процедура, освоить которую легко, и она поможет вам создать печатную плату с нужными характеристиками при действительно небольших затратах. В данном разделе не отведено место для объяснения того, как можно осуществить эту процедуру, но в интернете можно найти большое количество сайтов и видеороликов на YouTube, которые посвящены этому вопросу.

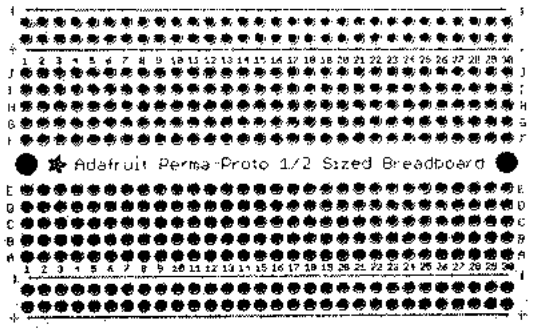
Протоплата

Протоплата — это макетная плата без металлических зажимов, которые удерживают компоненты на месте. Вместо этого вы припаиваете соединительные провода компонента с другой стороны платы, создавая таким образом постоянное соединение.



Протоплаты бывают различных форм и размеров, а также на одиночных, двойных и многослойных печатных платах. Вы можете приобрести протоплату такого же размера и компоновки, как ваша макетная плата Arduino.

Протоплата хорошего качества поставляется со сквозными отверстиями, для прочности изнутри покрытыми слоем металла. Чтобы устранить проблему окисления, что часто происходит с медным покрытием, в качестве покрытия здесь используется золото.



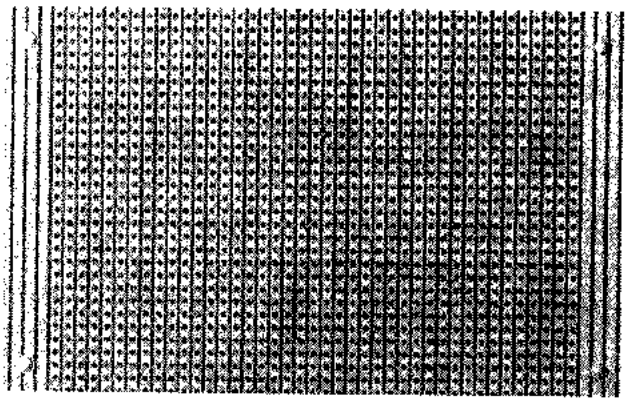
Протоплата типа Arduino

Как видно на изображении выше, протоплата имеет такую же пятидорожечную пластину и две шины электропитания с каждой стороны, как и макетная плата Arduino. Также идентична маркировка с каждой стороны.

Таким образом, копировать компоновку рабочей цепи с беспаечной макетной платы настолько просто, насколько возможно. Протоплаты также имеют монтажные отверстия, позволяющие надежно закрепить плату.

Макетная плата stripboard

Также известная как veroboard, макетная плата stripboard представляет собой плату, на которой расположены полоски меди, проходящие в одном направлении, как видно на рисунке ниже.



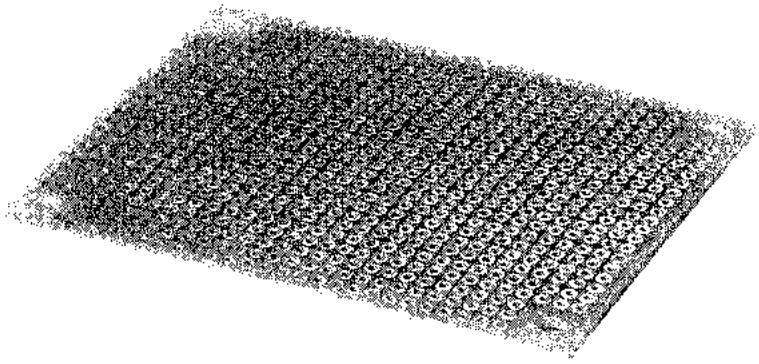
Через равные промежутки в плате просверлены отверстия, которые позволяют разместить вводы и выводы компонентов на плате, а затем выполнить их пайку с обратной стороны. В углах платы расположены крепежные отверстия.

После размещения всех компонентов на макетной плате создайте нужные вам цепи, при необходимости разрезая медные дорожки с обратной стороны платы при помощи ножа или скальпеля.

Перфорированная печатная плата

В перфорированной плате отверстия просверливаются предварительно по сетке через стандартные интервалы. Расстояние обычно составляет 0,1 дюйма. Каждое отверстие окружено круглой или квадратной медной накладкой.

В подобном типе платы пайка компонентов выполняется локально, но связь между компонентами можно осуществлять несколькими способами.



Один способ — это монтаж накруткой, при котором специальные разъемы с длинными ногами используются для удержания компонентов на месте. Эти ноги торчат с другой стороны платы, и цепи образуются за счет соединения контактов с помощью провода. Для создания соединения провод несколько раз оборачивается вокруг контакта, откуда и термин «накрутка».

Совет

Соединения с помощью метода накрутки наиболее надежны. Однако пайка представляет собой более простой и легкий способ, также обеспечивая хорошее и качественное соединение.

Обычно для накрутки используется специальное автоматизированное устройство, но можно это делать и с помощью ручного инструмента. При этом не только создается чрезвычайно надежное соединение, процесс накрутки также крепко удерживает разъемы компонента.

Однако в настоящее время такой метод используетсяредко. Большинство людей, использующих перфорированную карту, выполняют локальную пайку компонентов и соединяют их с помощью провода, подрезанного по заданному размеру.

Пайка

Как можно было увидеть на предыдущих страницах, вскоре вам придется овладеть техникой пайки, так как, пожалуй, единственным типом платы, где данный метод не используется, является беспаечная макетная плата. Быстросъемный тип соединения, использующийся в этих платах, может быть

быстрым и удобным, но он не обеспечивает надежной фиксации. Методом, обеспечивающим такую фиксацию, является пайка.

Что такое пайка?

Пайка — это способ соединения металлов. Его суть в расплавлении металла с низкой точкой плавления (припоя) на металле, с которым осуществляется соединение. Когда припой остывает и твердеет, он соединяет металлы. Для создания цепи соединяются два металла — ввод/вывод компонента и медная дорожка на плате.

Спаянные соединения можно довольно легко отпаять при необходимости.

Припаянное соединение можно сделать быстро и просто, и оно надежно прикрепит компонент к печатной плате с помощью прочного электропроводящего контакта. Еще одно преимущество пайки в том, что всегда есть возможность отпаять припаянное соединение путем его нагревания, если, скажем, по какой-либо причине появляется необходимость снятия компонента. По этим причинам пайка — наиболее распространенный способ добавления электронных компонентов в печатные платы.

Вам потребуется приобрести ряд предметов, прежде чем приступить к работе. Первый из них — паяльник.

Паяльники

Существует четыре типа паяльников:

- С постоянным температурным режимом
- С переменным температурным режимом
- Портативный
- Паяльная станция

Паяльники с постоянным температурным режимом самые простые. У вас в руках окажется паяльник со шнуром электропитания, губкой для чистки наконечника и железной подставкой, на которой стоит горячий паяльник, пока не используется.

Важная характеристика паяльника, на которую следует обратить внимание — его мощность (указываемая в ваттах). Существуют модели мощностью от 12

до 50 Вт. Чем выше мощность, тем горячее паяльник. Вы можете выбрать именно такой, который вам подойдет лучше всего.

При пайке соединения важно, чтобы припой плавился прежде, чем жар от паяльника распространится и повредит оборудование. Слишком низкая температура может принести вред, как и слишком высокая.

Так что хорошим решением в данной ситуации может быть паяльник со средним диапазоном мощности в 25 Вт или около того. Этот тип паяльника будет идеальным для того вида компонентов, который вы будете использовать в своих проектах Arduino.

Мы рекомендуем приобретать паяльники с пределом мощности, равным 25 Вт.

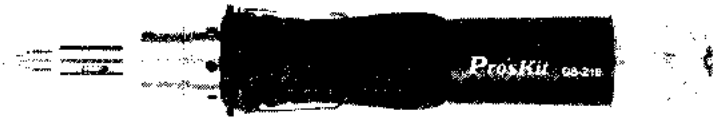
Паяльники с переменным режимом температуры во многом схожи с моделями с постоянным температурным режимом, кроме того факта, что их мощность можно регулировать. Как правило, такие паяльники обеспечивают мощность в диапазоне от 0 до 50 Вт.

Уровнем мощности, по которому характеризуется паяльник, будет самое высокое значение в ваттах, которое паяльник способен выдать. Если вы приобретаете паяльник такого типа, разумным будет приобрести инструмент с высоким уровнем мощности, поскольку он обеспечит лучший диапазон температур.

Преимущество паяльников с переменным температурным режимом состоит в том, что их можно использовать для широкого диапазона применений — например, для компонентов небольшого размера, при пайке которых используются низкие температуры (диоды и транзисторы), и компонентов, при пайке которых используются высокие температуры (резисторы и трансформаторы).

Портативные паяльники характеризуются отсутствием шнура электропитания; вместо этого они получают электропитание либо от аккумуляторной батареи, либо от газового баллона, находящегося внутри ручки.

Аккумуляторные модели используют стандартные батареи размера AAA, а газовые потребляют бутан.



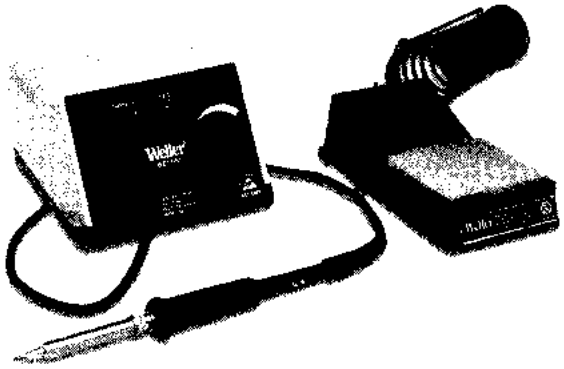
Преимуществом этих паяльников является то, что их можно использовать в тех местах, где нет электроэнергии, а также в ограниченном пространстве, где висящий кабель может быть неудобен.

Совет

Беспроводной паяльник, так же как и проводной, будет очень полезным приобретением.

Недостатки заключаются в том, что цена их довольно высока, а газовые модели потенциально опасны. Кроме того, срок службы от батареек очень недолог.

Паяльные станции позиционируются на первом месте в рейтинге паяльников. Они состоят из паяльника, док-станции для паяльника со встроенным отделением для чистки наконечника паяльника и блока питания.



Станции хорошего качества оборудованы такими функциями, как микропроцессорный блок управления температурой, цифровая индикация температуры и автоматическое выключение.

Основное преимущество паяльных станций такое же, как и у паяльников с режимом переменной температуры: способность обеспечивать различный диапазон рабочих температур. Помимо этого модели с цифровой индикацией температуры позволяют осуществлять чрезвычайно точный контроль температуры. Также станции обеспечивают другие функции, как упоминалось выше.

Совет

Не торопитесь покупать паяльную станцию, пока не знаете наверняка, что вам понадобятся функции, которые она обеспечивает.

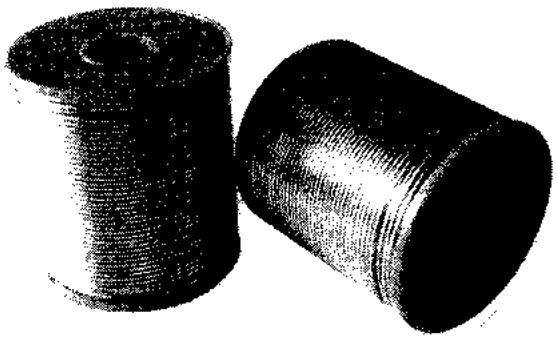
Также недостатком является цена — паяльные станции, относящиеся к профессиональному сегменту, весьма дороги.

Припой

Следующим пунктом в вашем списке покупок будет припой, который, как мы уже видели, представляет собой легкоплавкий металлический сплав, используемый для крепления компонентов на печатной плате.

Существует много различных типов припоя, но для использования на печатных платах используется два основных: свинцовый и бессвинцовый.

Припой продается в виде проволоки толщиной от 0,8 мм, которая идеально подходит для печатной платы, до гораздо более толстого варианта, который используется для соединения компонентов с большой нагрузкой, например колонок.



Совет

Припой с толщиной 0,8 мм идеально подходит для пайки компонентов на печатной плате.

В основном припой делается из олова и свинца — в пропорции примерно 60% олова и 40% свинца. Это так называемый свинцовый припой, и одним из его преимуществ является низкая температура плавления, позволяющая использовать не очень горячий паяльник: таким образом, у вас меньше шансов повредить компоненты. Недостаток в том, что свинец чрезвычайно ядовит, что может стать причиной отказа от его использования.

Альтернатива — бессвинцовый припой. Он состоит из нескольких материалов, таких как олово, медь, серебро, висмут, индий, цинк и сурьма. Однако, хоть этот вид припоя и безопасен в использовании, бессвинцовый припой имеет более высокую температуру плавления, чем свинец, и не создает настолько же надежное соединение.

Свинцовый припой потенциально опасен, и, поскольку бессвинцовый припой работает практически так же хорошо, мы рекомендуем вам использовать именно его.

И последнее, что вам необходимо знать о припое — это то, что для создания хорошего соединения вам нужен флюс для пайки. Он помогает устранить последствия окисления, что может предотвратить свободное растекание припоя.

Сегодня припой в основном продается с полой сердцевиной, содержащей флюс. При использовании такого припоя вы увидите, как выплавляется флюс. Однако дешевый припой может продаваться без флюса в сердцевине, и в этом случае вам придется купить банку флюса и применять его вручную.

Также флюс помогает распределять тепло от паяльника.

Советы по использованию паяльника

Рабочая часть любого паяльника — наконечник, и со временем он изнашивается. Поэтому он является заменяемым. Другая причина заключается в том, что существует широкий ассортимент наконечников для пайки, что позволяет использовать один паяльник для работ разного типа.

Форма наконечников варьируется от остроконечных конусных наконечников для пайки мелких близко расположенных компонентов до плоских наконечников в виде отвертки, подходящих для пайки крупных деталей.

Имея в наличии ассортимент наконечников для своего паяльника, вы сможете использовать его в различных ситуациях.

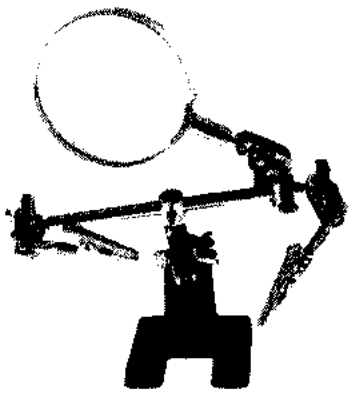
Аксессуары для паяльника

Очень полезный аксессуар для паяльника — держатель, или подставка, которая обеспечивает безопасное место для хранения горячего паяльника. Многие паяльники поставляются с держателем или подставкой, поэтому, когда вы покупаете его отдельно, убедитесь, что он обладает похожими характеристиками.

Также вам пригодится небольшая абразивная губка, с помощью которой при необходимости можно удалять излишки припоя с наконечника.

Губки, использующиеся на кухне, помогут вам содержать наконечник паяльника в чистоте.

Для тех, кто собирается много паять или же кому сложно держать в руках более одного компонента, несомненным преимуществом будут настольные тиски (также называемые третьей рукой).



Тиски удерживают печатную плату с помощью зажимов типа «крокодил», позволяя с удобством выполнять пайку. В некоторых моделях, подобных той, что показана на рисунке справа, также присутствует увеличительное стекло для более детального осмотра объекта во время работы.

Настольные тиски могут быть привинчены или закреплены с помощью болтов на рабочей поверхности, так что печатная плата может надежно удерживаться под любым углом.

Если вдруг вы ошибетесь и припаяете неправильный компонент к плате или припаяете в неположенное место, вам понадобится отпаять соединение путем повторного нагрева и удаления припоя. Сделать это без перегрева компонента довольно сложно. Для этой цели очень полезным окажется пневмоотсос припоя.

Это устройство представляет собой небольшой вакуумный насос, который отсасывает расплавленный припой с места соединения, оставляя его чистым, чтобы деталь можно было удалить.

Данные устройства могут быть немного неудобными в использовании, поскольку вам приходится управляться с ними с помощью одной руки, удерживая в то же время расплавленный припой другой рукой.

На заметку

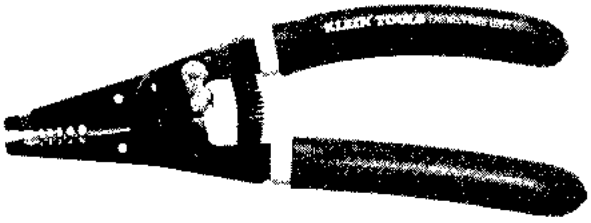
Еще одним инструментом, используемым для удаления припоя, служит отсос припоя. Он представляет собой полую резиновую грушу с насадкой, которая отсасывает припой, когда вы разжимаете руку с грушей.

Альтернативой отсосу припоя может служить капиллярный отсос. Это медный провод, который смотан в ленты и поставляется в мотках. Он функционирует, обеспечивая большую площадь поверхности для закрепления расплавленного припоя.

Просто удерживайте капиллярный отсос на месте соединения, которое необходимо отпаять, и нажимайте на него с помощью паяльника до тех пор, пока припой не расплавится и не будет поглощен фитилем. Затем оттяните фитиль, и припой отпадет. Вам может понадобиться повторить процедуру, чтобы удалить весь припой.

Другие инструменты

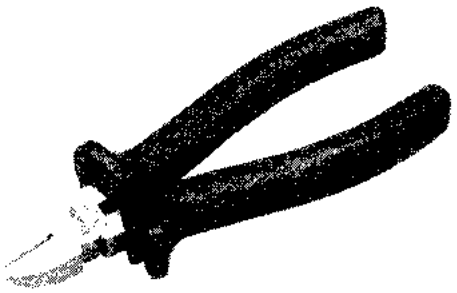
Помимо компонентов, которые мы уже упоминали, вам понадобится инструмент для снятия изоляции, который показан ниже.



Этот инструмент позволяет быстро и аккуратно снимать изоляцию с провода.

Также вам пригодятся острогубцы или щипцы. Они помогут при необходимости удерживать мелкие детали и управлять ими в ограниченных пространствах.

Наконец, рекомендуется приобрести кусачки. Они могут быть полезны при удалении остатков провода из паянного стыкового соединения.



Безопасность

Во время пайки возникают пары, которые потенциально могут представлять серьезную опасность для здоровья. При использовании припоя на основе свинца выделяются пары, которые могут привести к отравлению свинцом. Также ядовиты пары, выделяемые флюсом на основе канифоли.

Внимание

Не забудьте вымыть руки после того, как поработали со свинцовым припоем.

Для защиты от опасных паров припоя важно, чтобы вы работали в хорошо проветриваемом помещении. Кроме того, используйте вытяжной вентилятор для вентиляции испарений либо наружу, либо с помощью фильтра. Во многих паяльных станциях есть встроенный вентилятор, и это очень хорошая причина, чтобы потратить дополнительную сумму и приобрести подобный прибор.

Тем не менее, несмотря на все риски для здоровья, следует сказать, что нет абсолютно никакой опасности, если вы занимаетесь пайкой редко. Это проблема потенциально опасна только для тех, кто паяет часто.

Наиболее вероятно, что во время пайки вы можете повредить глаза. Паяльники могут «плеваться» маленькими кусочками припоя во всех направлениях, и хотя они очень маленькие для того, чтобы нанести серьезные повреждения, продолжительное воздействие может ухудшить ваше зрение. Так что, хотя это и может показаться чрезмерным, мы рекомендуем вам при пайке использовать защитные очки.

Антистатический браслет используется для заземления человека, который работает с электронным оборудованием. Это предотвращает накопление статического электричества в организме, приводящее к электрическому разряду, повреждающему компоненты.

Используя несколько иной подход, также можно рассмотреть возможность защиты ваших компонентов с помощью антистатического ремешка. Это предотвратит их повреждение электростатическим электричеством вашего тела.

Способы пайки

Когда вы действительно приступите к процессу пайки, вы обнаружите, что это не так просто, как выглядит на первый взгляд. Вероятно, первая проблема, с которой вы встретитесь — это достижение того, чтобы припой свободно растекался.

Если у вас возникают проблемы с расплавкой припоя, попробуйте использовать более тонкий припой.

Причиной проблемы почти наверняка будет окисление наконечника паяльника, что снижает его способность передавать тепло. Вы всегда сможете определить, что произошло окисление — вместо светло-серебристого цвета, каким

он должен быть, наконечник приобретет тускло-темный цвет. Очень сложно выполнить пайку, когда наконечник находится в таком состоянии.

1. Итак, прежде чем начать, проверьте наконечник. Если он грязный, несколько раз протрите его губкой.

2. Далее вам необходимо покрыть его оловом. Лужение — это процесс, который на какое-то время предотвращает повторное окисление наконечника. Делайте это посредством расплавления припоя на наконечник, пока он весь не покроется припоем и не приобретет светло-серебристый цвет. Затем сотрите излишки. Луженый наконечник подойдет для выполнения небольшого количества соединений, прежде чем его необходимо будет снова покрыть оловом.

Слева на рисунке показан плохо окисленный наконечник, а справа — тот же самый наконечник после того, как его покрыли оловом.

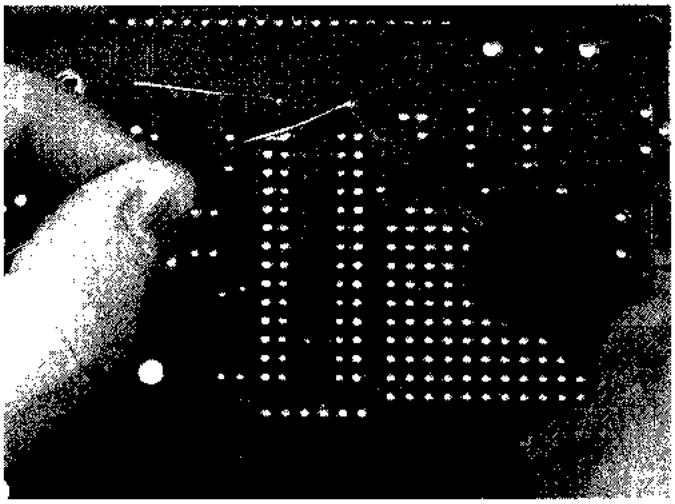
Некоторые паяльники продаются с наконечниками, имеющими специальное антиокислительное покрытие.

3. Вы готовы приступить к пайке. Прежде чем сделать это, запомните одно простое правило — припой всегда течет в сторону тепла. Поэтому нагрейте провод компонента и цепь, т. е. соединение, прежде чем применить припой. Не пытайтесь загрузить паяльник большим количеством припоя и затем перенести его на соединение — это не сработает.

4. Вставьте вводы/выводы компонента в отверстия печатной платы, переверните плату и загните провода так, чтобы компонент остался на месте.

Новые наконечники паяльников перед использованием обязательно следует покрывать оловом.

5. Теперь поместите наконечник горячего паяльника в место, где один из проводов касается поверхности печатной платы. Подержите его там несколько секунд, чтобы нагреть свинец и панель, а затем прикоснитесь к ним припоем; не касайтесь им наконечника паяльника. Припой расплавится и растечется по соединению. Когда вы увидите аккуратную красивую каплю блестящего припоя на месте соединения, можете поднимать паяльник — соединение выполнено.



Внимание

Будьте осторожны, чтобы не перегреть соединение — это может повредить компонент. Пайка соединения должна занимать не более нескольких секунд. Если необходимо, дайте соединению остыть и затем попробуйте снова.

6. Дайте соединению секунду или около того, чтобы остыть, и затем внимательно посмотрите на него. Капля припоя должна покрыть всю контактную площадку, не перетекая на другую пластину. При необходимости удалите припой с помощью отсоса и начните снова.

7. Наконец, используйте кусачки, чтобы отрезать лишний провод ввода/вывода.

При качественно выполненном соединении капля должна быть блестяще серебристой. Если у вас получился тусклый цвет, вероятно, вы выполнили так называемое холодное соединение. Оно может послужить причиной плохого электрического соединения между компонентом и платой и привести к различным проблемам.

Основная причина холодных соединений заключается в том, что вы не передаете достаточное количества тепла припою, так что припой не течет свободно. Если соединение действительно выглядит плохо, просто повторно нагрейте его, чтобы припой снова расплавился, и при необходимости добавьте еще немного припоя.

Адаптеры электропитания

Во многих ситуациях вы сможете обеспечивать электропитание для своей платы Arduino с помощью компьютера через порт USB. Однако если придется разместить плату вдали от компьютера, вам понадобится использовать адаптер электропитания.

Вопрос о том, *какой именно* адаптер электропитания следует использовать, весьма распространен на форумах Arduino. Ответ заключается в том, что для использования подойдет любой адаптер, отвечающий следующим критериям:

- Адаптер должен обеспечивать выход постоянного тока (DC).
- Адаптер должен быть оборудован штекером диаметром 2,1 мм, центральный контакт — положительный.
- Сила постоянного тока адаптера должна составлять по крайней мере 250 мА.
- Выходное напряжение должно быть в пределах 6–20 В. Но следует помнить, что если оно будет меньше 7 В, то 5 В контакт платы, возможно, не обеспечит нужное напряжение, что может привести к проблемам, связанным с нестабильной работой. Если же напряжение будет более 12 В, регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон напряжения составляет от 7 до 12 В.

Плата Arduino поддерживает широкий диапазон напряжений. Однако рекомендуется использовать напряжение в диапазоне от 7 до 12 В.

Как правило, подойдет любой адаптер электропитания со штекером диаметром 2,1 мм, центральным положительным контактом и выходным напряжением в пределах 6–20 В. Не возникнет проблем, если выходное напряжение адаптера постоянно, поскольку плата Arduino оборудована встроенным регулятором напряжения.

Обратите внимание, что мы также можем подавать электропитание на плату через силовые контакты (**Vin**).

Оборудование для тестирования и диагностики

Иногда плата просто не работает; иногда работает, но не так как надо или не очень хорошо; иногда вы просто не уверены, будет ли она работать.

В последнем случае вам необходимо все проверить, прежде чем двигаться дальше, а в других следует установить причину проблемы.

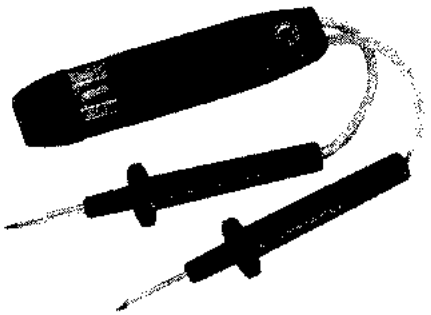
Чтобы помочь вам это сделать, существует большое количество устройств для тестирования и диагностики. Они варьируются от простых тестеров цепи, до усовершенствованного оборудования, такого как осциллографы и генераторы сигналов.

Тестеры цепи также можно использовать для проверки заземления оборудования.

Тестеры цепи

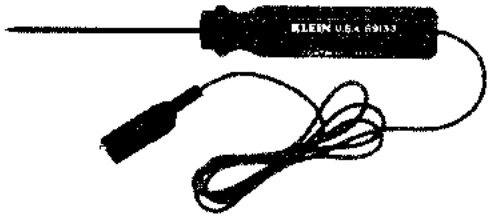
Это простое и недорогое устройство, которое используется для проверки наличия электричества в печатной плате. Тестер состоит из двух щупов, подключенных к неоновой лампе.

Чтобы использовать тестер, прикоснитесь с помощью одного щупа к проводу под напряжением, а другим щупом коснитесь заземления. Если неоновая лампа загорится, это подтверждает то, что электропитание на плате присутствует. Если не загорится — либо отсутствует электропитание, либо цепь неисправна.



Тестер для проверки на обрыв цепи

Еще одно простое, но, тем не менее, очень полезное устройство — тестер для проверки на обрыв цепи. Он состоит из индикатора (либо световой индикатор, либо зуммер) и источника электрической энергии, обычно аккумуляторной батареи. Каждый конец устройства выступает в качестве щупа.



Тестеры для проверки на обрыв цепи полезны для проверки переключателей.

Приступая к эксплуатации, сначала вы должны отключить электропитание цепи, затем присоединить зажим к одному концу цепи, чтобы выполнить тестирование, а щупом прикоснуться к другому концу. Если цепь рабочая, электричество из аккумуляторной батареи будет по ней проходить и активировать индикатор.

Мультиметр

Мультиметр представляет собой электронное измерительное устройство, обеспечивающее целый ряд измерительных функций в одном устройстве. К ним относится возможность измерения напряжения, тока и сопротивления.

Мультиметры могут быть аналоговыми, использующими микроамперметр, указатель которого перемещается по отградуированной шкале, или

цифровыми, отображающими измеряемое значение в цифрах. В настоящее время наиболее распространены цифровые мультиметры.

Мультиметры обычно доступны в виде портативных устройств, которые удобно использовать для поиска неисправностей в электрических цепях. Также существуют намного более продвинутые стендовые модели, обладающие более высокой точностью измерения и обеспечивающие более широкий диапазон измерительных функций.

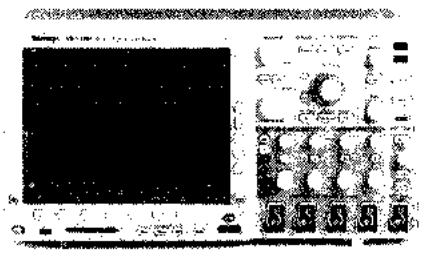
На заметку

Хороший мультиметр представляет собой единственное тестовое устройство, которое, скорее всего, понадобится большинству пользователей Arduino.

Для человека, увлекающегося электроникой, портативные устройства подойдут идеально, поскольку они обеспечивают практически все, что вам может понадобиться, по довольно доступной цене. Стендовые модели предоставляют функции, которые вам вряд ли понадобятся, и могут стоить сотни тысяч рублей.

Осциллограф

Осциллограф представляет собой улучшенный тип устройств для тестирования и диагностики, которые могут отображать электрический сигнал в волновой форме в любой точке цепи. Они также могут отображать прочие сигналы, такие как звук и вибрации, преобразуя их в напряжение.



Форму волны можно анализировать по таким свойствам, как амплитуда, частота, время нарастания, интервал, искажение и т. д.

Осциллографы представляют собой большие громоздкие устройства, которые подходят только для стендового использования. Также они имеют очень высокую цену и требуют, чтобы пользователь обладал обширными знаниями в области электроники. Любителям нет необходимости использовать устройства такого типа.

Программное обеспечение для проектирования

Много лет назад, до изобретения цифровых технологий, разработка макета печатной платы выполнялась с помощью ручки и бумаги. С тех пор подход не изменился; однако в настоящее время вы можете упростить этот процесс с помощью использования компьютерных программ.

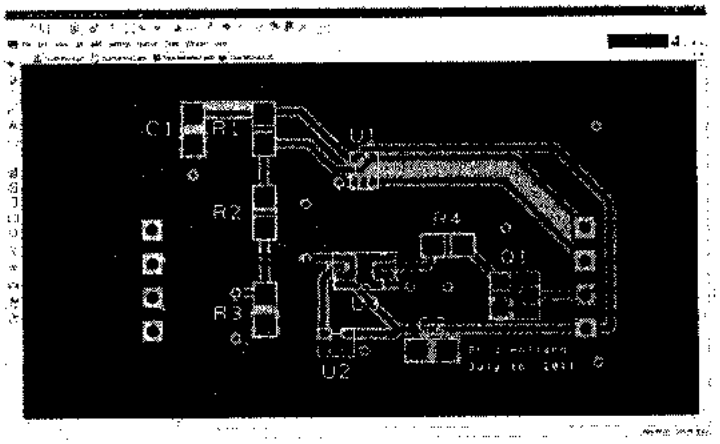
Некоторые из этих программ могут выступать также в качестве симулятора и в точности показать, какие операции выполняет цепь. Другими словами, вы сможете увидеть, будет ли то, что вы создали, делать то, что вы предполагали.

Определив, что цепь работает, вы сможете сохранить ее в виде файла, чтобы затем поделиться с другими пользователями или же чтобы отправить производителю печатных плат.

Гораздо проще использовать для проектирования цепей программное обеспечение, нежели ручку и бумагу.

Существует целый ряд программ, предназначенных для проектирования цепей, и многие из них бесплатны. Одна из популярных — программа Designspark, доступная для скачивания по адресу **www.rs-pmline.com/designspark/electronics/**. Интерфейс программы показан ниже.

Некоторые бесплатные программы позволят вам сохранить свой проект только в том случае, если вы закажете готовую печатную плату у производителя программы.



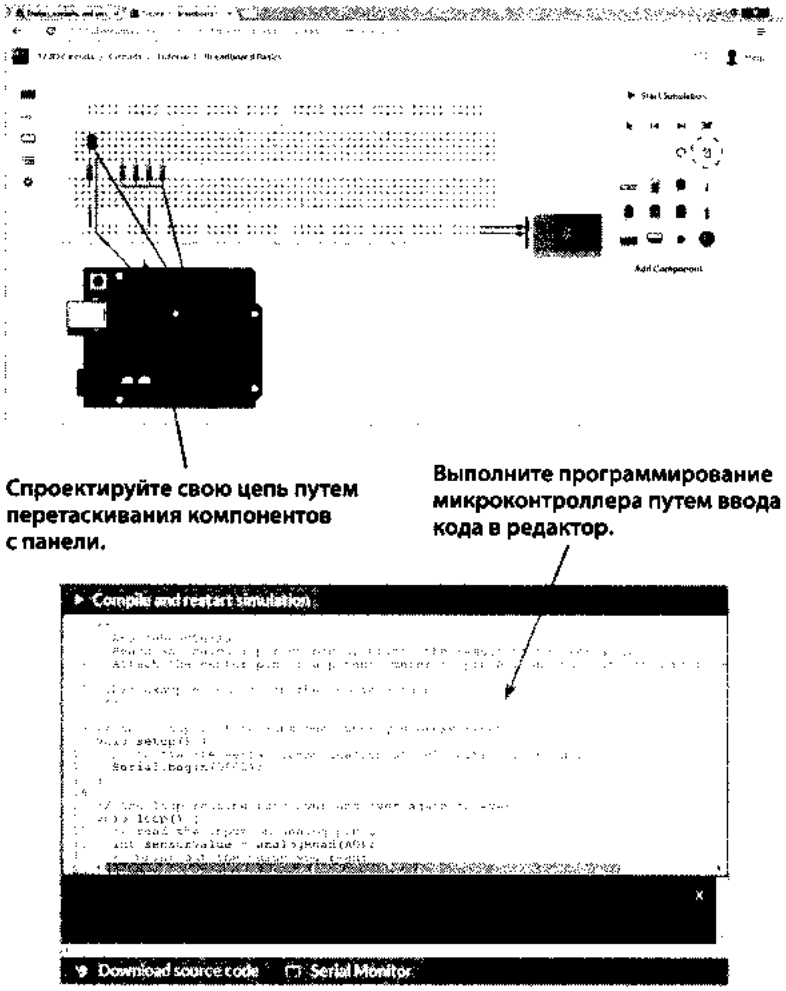
Другие известные программы:

- Fritzing (**www.fritzing.org/home**)
- Kicad (**www.kicad-pcb.org**)
- Expresspcb (**www.expresspcb.com**)
- Pad2pad (**www.pad2pad.com**)
- FreePCB (**www.freepcb.com**)
- CadSoft EAGLE (**www.cadsoftusa.com**)
- 123D Circuits (**www.123dapp.com/circuits**)

Все эти программы можно использовать бесплатно, хотя некоторые бесплатные версии имеют определенные ограничения.

123D Circuits — интересный вариант, так как эта программа позволяет разрабатывать свои цепи в режиме онлайн, без необходимости устанавливать на компьютер программное обеспечение. Кроме того, в ней доступен режим Arduino, позволяющий вам не только проектировать цепи Arduino, но и программировать микроконтроллер с помощью кода, а затем при помощи симулятора тестировать то, что получилось. Все это можно делать, не прикасаясь к физической плате Arduino.

123D Circuits представляет собой одну из самых популярных программ для проектирования цепей и настоятельно рекомендуется к использованию.



Специальный режим просмотра в программе 123D Circuits позволяет вам увидеть проект в том виде, каком он будет выглядеть на печатной плате.

Любые ошибки в коде будут выделены так же, как это происходило бы в редакторе кода на компьютере. Когда все будет функционировать так, как вам необходимо, просто скопируйте проект на физическую плату.

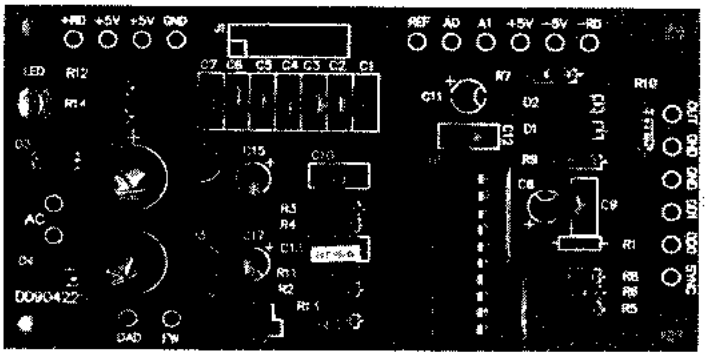
Электрические схемы

Цепь, состоящая всего лишь из нескольких компонентов, не нуждается в проектировании как таковом — она настолько проста, что вы сможете просто запомнить необходимый вам макет. Однако с более сложными цепями, которые используют большое количество компонентов, все будет несколько иначе — вы быстро растеряетесь.

Здесь в игру вступают электрические схемы. По сути, они выступают в качестве дорожной карты, отображая все компоненты цепи и четко показывая, как они связаны между собой.

На заметку

Когда вы приступаете к проектированию цепи, нарисуйте электрическую схему. Это может очень пригодиться на этапах планирования и конструирования.

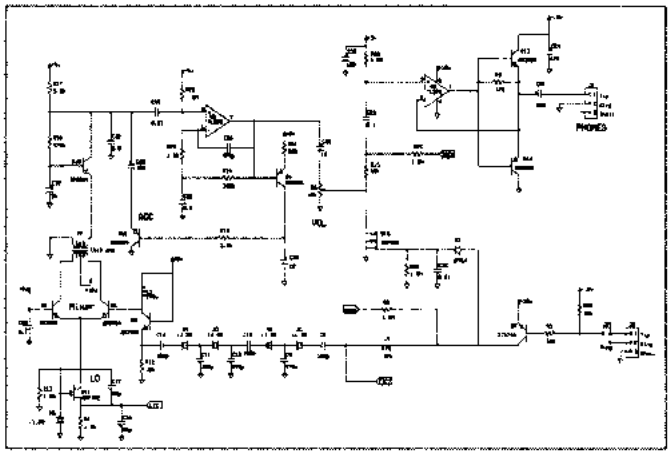


К примеру, возьмем цепь, показанную выше. Хотя она и относительно проста, было бы очень сложно спроектировать или собрать ее без некоторой документации — вероятно, было бы допущено много ошибок с подключением.

Однако с помощью электрической схемы вы сможете точно увидеть, что происходит. Это упрощает проектирование и конструирование цепи.

На заметку

Разновидностью электрической схемы является функциональная схема. Она используется при планировании и разработке больших цепей, где каждый блок представляет собой определенный участок цепи.



В электрической схеме каждый тип компонента представлен символом. Поэтому чтобы понимать, что именно вы видите (а также как собрать схему самостоятельно), вы должны знать, что эти символы означают.

Также, широко используемым компонентам, как правило, присваивается идентификатор. Это помогает различать их в цепи, где используется множество однотипных компонентов.

На заметку

Номинал некоторых компонентов также обозначается и на печатных платах.

Символы компонентов

В электронных цепях используется много различных типов компонентов. Каждый тип можно разделить на несколько категорий. На данный момент мы рассмотрим символы для наиболее часто используемых компонентов.

Резистор

Этот компонент можно обнаружить в большинстве цепей. Он препятствует течению электричества. Наиболее распространены фиксированные резисторы. Каждому резистору в цепи присваивается идентифицирующая метка, например R1, R2, R3 и т. д.

Резистор

Сопротивление (значение резистора) измеряется в омах (Ом).

Конденсатор

Конденсатор

Еще один широко используемый компонент — конденсатор — часто используется в сочетании с резистором, чтобы создать цепь синхронизации. Также его можно использовать в качестве фильтра, например чтобы блокировать сигналы постоянного тока, но пропускать сигналы переменного тока. Данное устройство помечается буквой C, т. е. C1, C2, C3 и т. д.

Емкость (значение конденсатора) измеряется в фарадах (Ф).

Диод

Диод

Диоды позволяют электричеству протекать только в одном направлении — от анода к катоду. Анод находится слева (треугольник), а катод (вертикальная линия) справа. Таким образом, ток течет слева направо. Устройство помечается обозначениями D1, D2, D3 и т. д.

Диоды (и другие полупроводники, такие как транзисторы) измеряются несколькими способами.

Светоизлучающий диод (светодиодный индикатор)

Светодиодный индикатор

Светодиодные индикаторы работают во многом так же, как и другие представители класса диодов. Однако они также преобразуют электричество в свет. Символ, обозначающих их, такой же, что и для обычного диода, но с двумя диагональными стрелками, указывающими в сторону от анода. Как и с диодами, светодиодные индикаторы маркируются обозначениями D1, D2, D3 и т. д.

Как и диоды, светодиодные индикаторы и транзисторы — это полупроводниковые устройства.

Транзистор

NPN-транзистор

Транзисторы способны от небольшого входного сигнала управлять значительным током в выходной цепи, что позволяет использовать их для усиления, генерирования, коммутации и преобразования электрических сигналов.

Вертикальная линия в верхней части символа представляет собой коллектор, вертикальная линия в нижней части — эмиттер, а горизонтальная линия слева — базу. Диагональная стрелка, указывающая вниз, обозначает NPN-транзистор. Если бы она указывала в обратном направлении, это был бы PNP-транзистор.

Транзисторы обозначаются метками Q1, Q2, Q3 и т. д.

Соединения

В цепи провода часто пересекаются — иногда они касаются друг друга, образуя таким образом соединение; иногда этого не происходит. В схематическом виде это отображается следующим образом.

Пересечение без контакта — два провода пересекаются, не касаясь друг друга. Существует два способа это представить. Оба они корректны, это просто вопрос предпочтения.

Пересекающиеся провода без контакта

Пересечение с контактом — два провода пересекаются и касаются друг друга, образуя таким образом соединение. Схема указывает на это точкой в месте соединения, как показано ниже.

Пересекающиеся провода с контактом

Заземление

Всем электронным цепям требуется обратный путь, чтобы электричество протекало в другую сторону. Такая функция обеспечивается контактом **GND** (Земля) цепи, и каждая цепь имеет по крайней мере один такой элемент.

Земля

Заземление — это также мера безопасности в высоковольтных цепях.

6

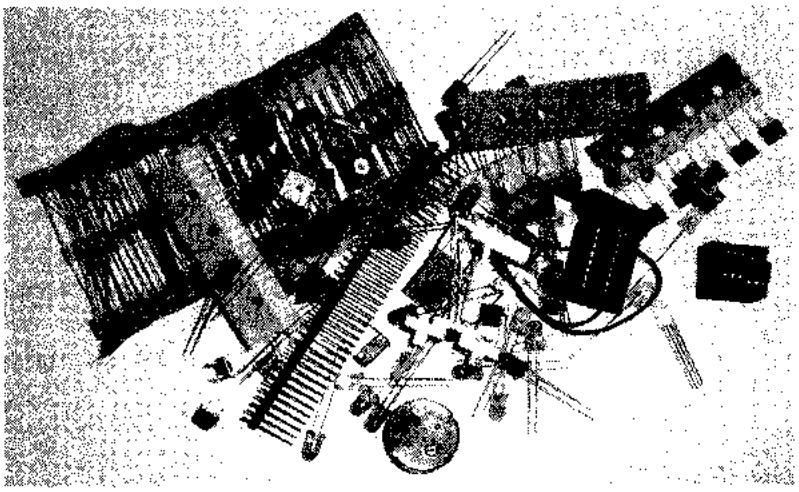
Электронные компоненты

Сама по себе плата Arduino не может выполнять большое количество операций. Чтобы создавать с ее помощью интересные и полезные проекты, вам понадобится расширить ее функциональность путем создания внешних электронных цепей. В этой главе мы рассмотрим электронные компоненты, которые поставляются в комплекте с Arduino, различные типы компонентов и объясним, какие функции они выполняют.

- **Состав набора Arduino**
- **Резисторы**
- **Цветовая маркировка резисторов**
- **Конденсаторы**
- **Катушки индуктивности**
- **Диоды**
- **Транзисторы**
- **Реле**
- **Трансформаторы**
- **Электродвигатели**
- **Интегральные схемы**
- **Датчики и приводы**

Состав набора Arduino

В комплекте с платой Arduino поставляется целый ряд электронных компонентов, как показано на рисунке ниже.



На заметку

Компоненты, поставляемые с вашей платой Arduino, необходимы для того, чтобы вы могли сразу приступить к работе. Со временем вам придется приобрести дополнительные компоненты.

Полный список состоит из следующих компонентов:

- 6 фоторезисторов
- 3 потенциометра
- 10 кнопок
- 1 датчик температуры
- 1 датчик наклона
- 1 буквенно-цифровой ЖК-дисплей
- 1 белый светодиодный индикатор
- 1 RGB светодиодный индикатор
- 8 красных светодиодных индикаторов
- 8 зеленых светодиодных индикаторов
- 8 желтых светодиодных индикаторов
- 3 синих светодиодных индикатора
- 1 6/9 В электродвигатель постоянного тока
- 1 сервопривод
- 1 пьезо-капсула
- 1 драйвер электродвигателя

- 2 оптопары
- 5 транзисторов
- 2 МОП-транзистора
- 5 конденсаторов 100 нФ
- 3 конденсатора 100 мкФ
- 5 конденсаторов 100 пФ
- 5 диодов
- 3 прозрачных фильтра
- 1 штыревая колодка
- 20 резисторов 220 Ом
- 5 резисторов 560 Ом
- 5 резисторов 1 кОм
- 5 резисторов 4,7 кОм
- 10 резисторов 10 кОм
- 5 резисторов 1 МОм
- 5 резисторов 10 МОм

Эти компоненты помогут вам незамедлительно начать постройку некоторых простых цепей. В этой главе мы рассмотрим некоторые компоненты, различные типы каждого компонента и увидим, какие функции они могут выполнять.

Резисторы

Из всех многочисленных типов электронных компонентов резисторы наиболее распространены, и их можно найти буквально на каждой плате, независимо от ее предназначения.

Цель этих устройств заключается в простом сопротивлении току в цепи. Эта способность измеряется в омах (Ом). Доступны различные номиналы резисторов, каждый из которых обеспечивает разный уровень сопротивления.

Существует три основных типа резисторов — постоянные, переменные и специализированные.

На заметку

Основная функция резистора заключается в ограничении течения тока в цепи. Это свойство можно использовать по-разному.

Постоянные резисторы

Этот тип резисторов наиболее распространен и используется в трех основных случаях.

- **Защита** — постоянные резисторы используются для ограничения тока, протекающего через другие компоненты, таким образом предотвращая их повреждение. Типичный пример — защита светодиодных индикаторов.
- **Разветвление** — постоянные резисторы используются для разделения напряжения между различными участками цепи.
- **Отсчет временного интервала/задержка** — при использовании в сочетании с конденсатором постоянный резистор создает цепь синхронизации или цепь задержки.

Переменные резисторы

Это резисторы, в которых уровень сопротивления можно изменить.

В обычном переменном резисторе сопротивление постоянно меняется в пределах заданного диапазона. Пример использования — регулятор громкости, а также управление низкими и высокими частотами в оборудовании класса Hi-Fi.

Еще один тип резисторов — потенциометры. Такие резисторы обладают элементом управления, который обычно настраивается однократно, чтобы установить нужное значение в цепи, и затем оно остается на том же уровне. Этот тип резисторов недоступен для настройки пользователю.

В электронных цепях, чтобы выполнить особое действие, можно объединять свойства двух или более различных компонентов. В тандеме конденсатор и резистор образуют цепь синхронизации.

Специализированные резисторы

К специализированным типам резисторов относится термисторы, сопротивление которых зависит от температуры. Их используют в цифровых термометрах, 3D-принтерах, тостерах, фенах и т. д.

Другим представителем данного типа резисторов служат фоторезисторы. Они похожи на термисторы за исключением того, что такие резисторы

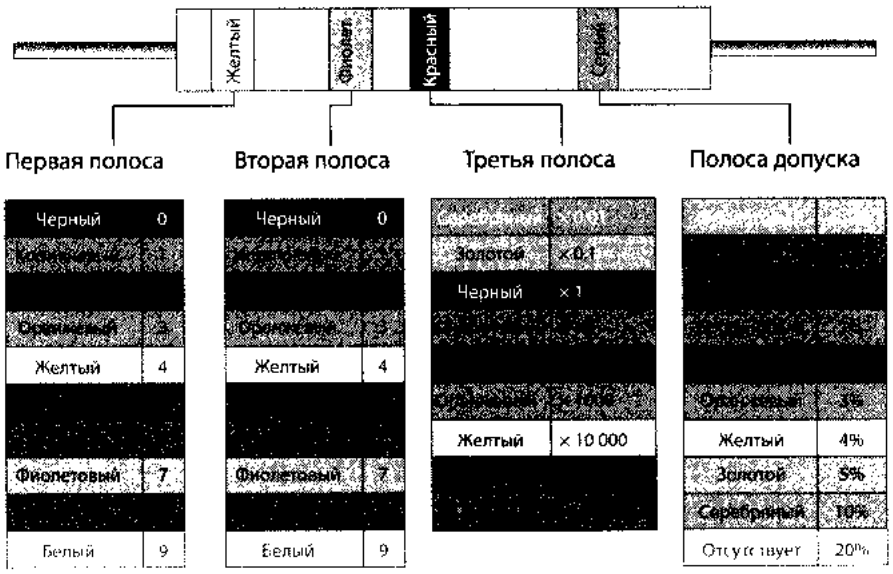
чувствительны не к температуре, но к свету. Примером их использования может служить элемент управления скоростью затвора в фотоаппаратах.

Цветовая маркировка резисторов

Многие резисторы обладают небольшими размерами, поэтому значение сопротивления не может быть указано на их корпусе. Для визуальной маркировки значения сопротивления используются цветные полосы.

Первая и вторая полосы представляют цифровое значение резистора, а третья полоса указывает умножитель частоты. Считывание цветных полос всегда происходит слева направо, начиная с той стороны, которая имеет полосу, находящуюся ближе к концу.

Цветная диаграмма снизу отображает различные используемые цвета и что они означают.



Большинство резисторов имеют очень маленький размер. Чтобы прочитать надписи на цветных полосах, вам, вероятно, понадобится увеличительное стекло.

Совет

Расшифровку цветового кода резисторов можно найти в интернете. Просто введите различные цвета, чтобы определить сопротивление.

Первая полоса слева обозначает число от 0 до 9. Каждому значению присваивается определенный цвет.

Вторая полоса также обозначает число от 0 до 9. Им также присваивается определенный цвет. Итак, желтая и фиолетовая полосы на резисторе выше обозначают значение, равное 47.

Третья полоса определяет множитель. В нашем примере, приведенном выше, полоса множителя красного цвета (×100), поэтому резистор имеет значение равное 47 × 100 = 4,700 Ом, или 4,7 кОм.

На заметку

Производитель не может присвоить резистору точное значение, поэтому указывается уровень допуска. Например, резистор 10 Ом с уровнем допуска 10% будет иметь значение сопротивления в диапазоне от 9 до 11 Ом.

Четвертая полоса обозначает уровень допуска. Это процентное значение указанного сопротивления. Коричневая полоса обозначает уровень допуска равный 1%, красная — 2%, оранжевая — 3% и так далее. Если полосы нет совсем, это указывает на уровень допуска равный 20%.

Конденсаторы

Конденсаторы представляют собой второй распространенный компонент в электронных цепях. Они бывают различных типов, форм и размеров.

По сути, эти устройства состоят из двух проводящих пластин, разделенных диэлектриком.

Когда подается электрический ток, между двумя пластинами накапливается заряд. Затем, отключив подачу тока, эту энергию можно сохранить в устройстве

в течение определенного периода или же пока внешнее событие не потребует освобождения этой энергии.

Если корпус конденсатора довольно велик, на нем можно будет отобразить его полные технические характеристики. Если же нет, вы увидите две или три цифры.

Если цифр только две, указано значение емкости в пикофарадах (пФ), например 50 будет означать емкость, равную 50 пФ.

Если присутствует три цифры, третье число — это множитель, например, 503 будет означать 50 × 3 — результирующее значение, равное 150 пФ.

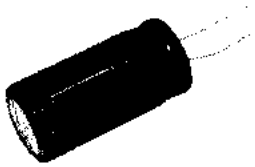
Единицей измерения емкости является фарад (Ф). В реальности это очень большая единица измерения, поэтому большинство конденсаторов имеют номиналы, измеряемые в микрофарадах (мкФ), нанофарадах (нФ) и пикофарадах (пФ).

Существует много различных типов конденсаторов, основными из которых являются пленочные, электролитические, керамические и переменные.

Пленочные конденсаторы — одни из самых распространенных и могут рассматриваться как универсальные конденсаторы, так как имеют невысокую цену и подходят для широкого спектра применений.

Электролитические конденсаторы используются в случаях, если в цепи требуется высокий уровень емкости. Эти устройства поляризованы, т. е. они имеют положительный и отрицательный полюса. Это означает, что они должны быть подключены правильно.

На заметку. Электролитические конденсаторы представляют собой устройства большого размера, напоминающие бочку.



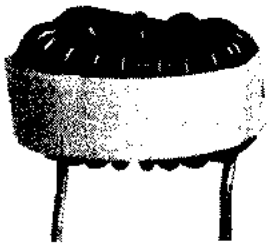
Керамические конденсаторы — еще один распространенный тип, эти конденсаторы имеют более высокую емкость относительно размера.

Переменные конденсаторы, также называемые подстроечными, обеспечивают регулируемый уровень емкости. Они используются для настройки цепей или в целях компенсации обратной связи.

Конденсаторы имеют множество применений. К ним относится фильтрация (удаление нежелательных сигналов), сглаживание (нивелирование колебаний в сигнале), удаление постоянной составляющей (конденсаторы пропускают только переменный ток), выравнивание напряжения (ограничение последствий скачков напряжения), импульсная мощность (конденсаторы эффективно выступают в роли небольшой аккумуляторной батареи), настройка цепей (например, конденсаторы переменной емкости используются для настройки радио) и зондирование (конденсаторы можно использовать для обнаружения изменений в окружающей среде).

Работа с электролитическими (поляризованными) конденсаторами может быть опасной. Если подключение выполнено некорректно, они могут самоуничтожиться, т. е. взорваться.

Катушки индуктивности



Катушка индуктивности — винтовая или спиральная катушка из свёрнутого изолированного проводника, обладающая значительной индуктивностью при относительно малой емкости и малом активном сопротивлении. При подаче напряжения, вокруг катушки появляется магнитное поле. В этом поле накапливается электрический ток. Другое свойство катушки индуктивности заключается в том, что она противодействует любому изменению величины тока, протекающего через нее.

Таким образом, индуктивность определяется как величина электродвижущей силы (ЭДС), или напряжение, созданное в компоненте (особенно катушках),

по сравнению со скоростью изменения электрического тока, который вырабатывает ЭДС или напряжение.

На заметку

Большое значение имеет форма сердечника. Тороидальные сердечники обеспечивают больше индуктивности для данного материала сердечника и количества витков, чем соленоидные (стержневые).

Единица измерения индуктивности — генри (Гн). Это большая единица измерения, и наиболее распространены микрогенри (мкГн), миллигенри (мГн) и наногенри (нГн).

Катушки индуктивности, как правило, классифицируются по материалу, используемому для производства сердечника.

Стандартные типы катушек индуктивности приведены ниже.

- **С воздушным сердечником** — у таких катушек совсем нет ядра, нехватка которого создает низкий уровень индуктивности. Область их применения обычно связана с высокой частотностью, например телевидение, радио, и т. д.
- **С железным сердечником** — такие катушки обеспечивают высокий уровень индуктивности и поэтому используются в цепях, требующих большого количества энергии.
- **С ферритовым сердечником** — феррит встречается в различных видах катушек. Такие катушки можно использовать в разных целях.
- **С железным порошком** — оксид железа обеспечивает высокий уровень индуктивности, который возникает на небольшом участке.

На заметку

Иногда катушки индуктивности классифицируются по механической конструкции, например тороидальные, керамические и пленочные.

К обычным областям применения катушек индуктивности относятся:

- **Радиодиапазон —** заглушает эти подавляющие высокочастотные сигналы, предотвращая таким образом их вмешательство в другие цепи.
- **Устройства резонансного контура**, такие как фильтры и гетеродины. В этой роли катушки индуктивности сочетаются с конденсаторами для получения резонансных частот.

- **Защита чувствительного оборудования** от разрушительных скачков напряжения и больших токов.

На заметку

Факторы, оказывающие влияние на индуктивность, включают количество витков, длину сердечника, площадь поперечного сечения сердечника и материал, используемый для создания сердечника.

Диоды

Диод представляет собой поляризованный компонент, который позволяет току двигаться через него только в одном направлении. Он имеет вход и выход: один из них образует анод, а другой — катод. Последний, как правило, отмечен серебряной полосой, как в примере, приведенном ниже.

На заметку

Диоды пропускают ток только в одном направлении.

Когда напряжение на аноде выше, чем на катоде, ток будет течь от анода. Однако если напряжение будет выше на катоде, диод не пропустит ток.

Такие устройства называются полупроводниками, и как таковые они не обладают специальным коэффициентом или значением, как, например, резисторы (сопротивление). Вместо этого полупроводники обладают различными параметрами, как, например, пиковое обратное напряжение в случае с диодами.

Существует множество типов диодов, наиболее распространены из которых следующие.

Светоизлучающие диоды (светодиодные индикаторы) изготавливаются из материала, который излучает свет в процессе, известном как электролюминесценция. Сегодня светодиоды можно обнаружить повсюду.

Стабилитроны, в отличие от обычных диодов не позволяющие току двигаться в обратном направлении, дают сделать это, но только на определенном уровне. Это свойство делает их идеальными регуляторами напряжения.

Фотодиоды имеют чувствительность к свету и способны преобразовывать его в ток. Они обладают широкой областью применения, в частности в потребительской электронике.

Выпрямители представляют собой массив диодов, расположенных таким образом, что они преобразуют переменный ток (AC) в постоянный (DC). Наиболее распространен мостовой выпрямитель, использующий четыре диода.

Существует множество способов использования диодов. Один из наиболее распространенных заключается в применении их в качестве надежного и недорогого источника освещения (светодиоды).

Диоды изготавливают из полупроводникового материала. Это вещество, которое обладает проводимостью между диэлектриком и металлом либо благодаря добавлению примесей, либо из-за воздействия температуры.

Оно является основой современной электроники, так как используется в интегральных схемах, диодах и транзисторах.

Другая сфера использования диодов относится к их способности преобразовывать переменный ток в постоянный. Эта способность используется в блоках электропитания для электронных устройств, таких как планшетные компьютеры.

Также велик потенциал светочувствительных свойств светодиодных индикаторов. К областям применения относятся экспонометры, детекторы рентгеновского излучения, приборы для считывания штрихкодов, инфракрасные пульты дистанционного управления, детекторы дыма и пламени и т. д.

Транзисторы

Транзисторы представляют собой устройства, изготавливаемые из полупроводникового материала и имеющие минимум три контакта. Принцип их работы заключается в том, что напряжение, применяемое к одной паре контактов, изменяет ток на другой паре контактов.

Тип транзистора указан с помощью кода, нанесенного на его корпус.

Биполярный транзистор

Полевой транзистор

Три контакта биполярного транзистора носят названия эмиттер, коллектор и база. Ток протекает между эмиттером и коллектором, тогда как база используется для управления устройством.

Такие транзисторы — ключевой компонент практически во всей современной электронике. Они не только обеспечивают две важных функции (переключение и усиление), но и очень дешевы в производстве. Кроме того, их размер может быть невероятно мал. Например, процессоры, используемые в компьютерах, могут содержать несколько миллиардов транзисторов.

Будучи изготовленными из полупроводникового материала, при неправильном подключении транзисторы могут легко выйти из строя. Чтобы определить правильность подключения, контакты транзистора обычно помечаются точкой (коллектор) и/или плоским контактом эмиттера.

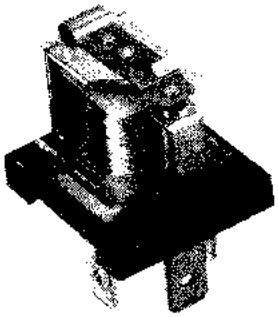
Также транзисторы можно классифицировать по функциям, например: коммутационные, силовые, высокочастотные и фототранзисторы.

Существует много типов транзисторов, но все их можно поместить в одну из двух основных групп.

- **Биполярные транзисторы** — биполярные транзисторы представляют собой устройства, управляемые током и имеющие три контакта: эмиттер, коллектор и база. Их часто используют в целях усиления.
- **Полевые транзисторы** — полевые транзисторы управляются с помощью напряжения и также имеют три контакта: исток, затвор, сток. Эти устройства часто используются в качестве цифровых коммутаторов.

В аналоговых цепях транзисторы применяются в качестве усилителей, тогда как в цифровых цепях они используются как переключатели. Это их два основных способа применения; однако их также можно использовать как буферы цепи и регуляторы.

Реле



Реле представляют собой переключатели, управление которыми осуществляется с помощью электричества. Они состоят из нескольких частей, таких как электромагнит, якорь и группа контактов. В более специализированных реле электромагнит заменяется на полупроводниковое устройство. В некоторых реле будет присутствовать всего одна группа контактов, тогда как в других их будет несколько (последние используются в случаях, когда несколько цепей должны находиться под воздействием одного сигнала).

В процессе работы электромагнит заряжается от применения тока и создает магнитное поле. Затем магнитное поле притягивает якорь, который, в свою очередь, перемещает контакт, таким образом замыкая цепь.

На заметку

Полезным свойством реле является то, что входная цепь полностью отделена от выходной цепи. По этой причине можно использовать реле для электрической изоляции цепей в целях защиты.

Основное применение реле заключается в том, чтобы брать ток малой величины и использовать его для включения или выключения гораздо более сильного тока. Эта способность реле невероятно полезна при использовании такими с устройствами, как датчики, которые производят токи слишком малой для электропитания цепи величины. Реле выступает в качестве интерфейса и защищает датчик от большого тока в цепи нагрузки.

В ситуациях, где необходимо переключать действительно большие токи, реле часто каскадные, т. е. небольшое реле переключает питание, необходимое для запуска большого реле, которое затем управляет током, необходимым для электропитания цепи.

К распространенным видам реле относится:

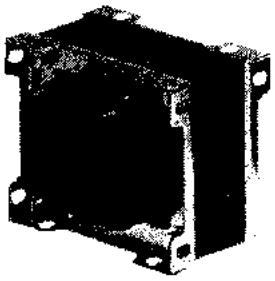
- **Электромеханические реле** состоят из электромагнита, якоря и группы контактов. Характерны два важных ограничения — медленная скорость и относительно короткий срок службы.
- **Герконовые реле** — хотя этот тип реле также представляет собой электромеханические устройства, благодаря своей конструкции они примерно в десять раз быстрее, чем электромеханические реле. Также они обладают более продолжительным сроком службы.
- **Полупроводниковые реле (SSR)** изготавливаются из полупроводникового материала и не имеют движущихся частей. В результате они работают гораздо быстрее и дольше, чем электромеханические реле.

Герконовые реле могут повреждаться вследствие искрения. Это происходит в случаях, когда дуга, возникающая между контактами, плавит часть поверхности контакта.

К типичной области применения относятся автомобили, где постоянный ток напряжением 12 В используется для управления сильноточной цепью, например в фарах, стартерах и моторизованной домашней технике, такой как кухонные миксеры.

Трансформаторы

Трансформаторы — устройства, которые могут изменять напряжение переменного тока (AC). Повышающий трансформатор увеличивает напряжение, тогда как понижающий трансформатор уменьшает его.



Эти устройства состоят из двух катушек изолированного провода, намотанного на сердечник. Входное напряжение применяется к одной катушке, называемой первичной, которая создает постоянно изменяющееся магнитное поле вокруг катушки. Это магнитное поле в свою очередь индуцирует переменный ток в выходной катушке, называемой вторичной). Оттуда он передается на цепь.

При производстве сердечников трансформаторов используется широкий диапазон материалов. К ним относятся ламинированная сталь, железо и воздух.

Напряжение на вторичной катушке зависит от количества витков в катушках. Например, если в первичной катушке один виток, а во вторичной — десять, напряжение на вторичной катушке будет в 10 раз больше, чем в первичной. Если в первичной катушке будет 10 витков, а во вторичной — один виток, напряжение во вторичной катушке будет в десять раз меньше, чем в первичной.

Некоторые трансформаторы содержат более одной вторичной катушки, что позволяет им подводить ток к нескольким цепям нагрузки.

Существует ряд трансформаторов, которые работают по принципам, перечисленным выше. К ним относятся:

- **Силовые трансформаторы** широко используются для преобразования сетевого напряжения в низкое напряжение, необходимое для электропитания электронных устройств.
- **Импульсные трансформаторы** создают электрические импульсы, которые используются в цифровой и телекоммуникационной аппаратуре.

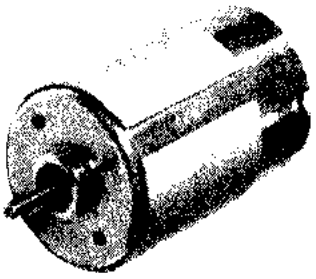
Трансформаторы подвержены потерям различных типов, которые снижают эффективность. Примерами могут служить потери на гистерезис и вихревые токи.

- **Радиочастотные трансформаторы** имеют много областей применения в высокочастотных устройствах, таких как УКВ, дециметровый диапазон радиоволн и микроволновые печи.
- **Аудиотрансформаторы** используются в аудиоцепях, например для фильтрации нежелательных помех звукового сигнала.

Примеры, приведенные выше, представляют собой лишь несколько способов использования трансформаторов. Существует также множество других.

Электродвигатели

Электродвигатели (электромоторы) широко распространены — вы можете найти их у себя дома, на работе, везде, где только можно представить. Эти устройства преобразуют электрическую энергию в энергию вращения.



Они состоят из нескольких частей, которые включают кожух, или корпус, ротор, статор, коллектор, вал, или мост, и щетки.

На заметку

Мощность электродвигателей измеряется в лошадиных силах, например ½ лошадиных силы.

Электродвигатели используют такие же электромеханические принципы, что и реле. В принципе, применение тока к входным контактам электродвигателя создает два магнитных поля, которые попеременно толкают и тянут вал. Это приводит к вращению вала.

Вращательное усилие, или крутящий момент, создаваемый электродвигателем, определяется тремя обстоятельствами: напряженностью магнитного статора (постоянный магнит, который окружает ротор), силой электрического тока на входных контактах и количеством витков на роторе (чем больше витков, тем быстрее вращается).

Электродвигатели можно разделить на три основных типа: переменного тока (AC), постоянного тока (DC) и универсальные. Электродвигателям переменного тока для запуска необходим переменный ток, электродвигателям постоянного тока — постоянный, а универсальные электродвигатели могут работать как на переменном, так и на постоянном токе.

Универсальные электродвигатели наиболее часто встречаются в мощных бытовых приборах, которые обычно используются в течение непродолжительного времени. К таким устройствам относятся кухонные комбайны, блендеры и пылесосы.

Электродвигатели переменного тока выпускаются в однофазной и трехфазной версиях. Однофазную версию электродвигателя переменного тока вы можете найти у себя дома, тогда как трехфазная версия применяется в промышленности.

На заметку

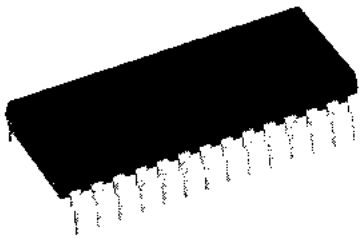
Скорость электродвигателя обычно обозначается в виде количества оборотов за минуту в режиме холостого хода.

Электродвигатели постоянного тока бывают нескольких типов. К ним относятся щеточные, бесщеточные и шаговые электродвигатели. Наиболее распространены шаговые электродвигатели.

Шаговым называется бесщеточный электродвигатель, который позволяет компьютеризированной системе управления «трансформировать» вращение электродвигателя. Он полезен тогда, когда необходимо переместить что-либо на определенное расстояние. Стандартным примером является робототехника.

Интегральные схемы

Интегральные схемы представляют собой основу современной микроэлектроники. Эти устройства изготавливаются из полупроводникового материала, такого как кремний. Обычно их называют чипами или микрочипами.



Интегральные схемы могут содержать буквально миллиарды компонентов, таких как диоды, транзисторы и резисторы. В результате они образуют весьма сложные цепи, которые тем не менее имеют очень небольшой размер. К другим преимуществам интегральных схем относятся:

- **вес** — интегральные схемы представляют собой устройства, имеющие небольшой вес;
- **цена** — создать интегральную схему гораздо дешевле, чем использовать стандартные компоненты, необходимые для обеспечения таких же функций;
- **надежность** — так как здесь нет паяных соединений, очень мало межкомпонентных соединений и используются низкие температуры, то интенсивность отказов весьма низка;
- **электропитание** — поскольку компоненты имеют небольшой размер, интегральные схемы потребляют очень мало энергии.

Большинство интегральных схем заключены в пластиковый корпус с большим количеством выступающих ног (контактов) для подключения к цепи. Эти

контакты имеют двухрядное расположение выводов (DIP, dual in-line package) и пронумерованы в направлении против часовой стрелки.

На заметку

Две наиболее распространенных области применения интегральных схем — это цифровые часы и калькулятор.

Интегральные схемы с двухрядным расположением выводов можно подключать в разъем интегральной схемы. Это исключает необходимость пайки, которая может их повредить. Также это позволяет легко заменять интегральные схемы, к примеру, во время ремонта.

На заметку

Интегральные схемы можно использовать в качестве чипа для поверхностного монтажа с 8, 14 или 16 контактами.

Интегральные схемы высокой мощности могут генерировать большое количество тепла, поэтому они имеют металлический тег, который может быть подключен к радиатору для отвода тепла.

Так как они содержат полные цепи, практически все интегральные схемы предназначены для выполнения определенной функции. Аналоговые интегральные схемы используются в усилителях, радиоприемниках, компараторах и регуляторах напряжения и т. д.

Цифровые интегральные схемы в основном используются в компьютерах. Стандартные функции включают в себя таймеры, счетчики, микросхемы калькуляторов и памяти, микроконтроллеры и т. д. На вашей плате Arduino микроконтроллером является цифровая интегральная схема.

Датчики и приводы

Вероятно, наиболее простым проектом Arduino является мигающий светодиодный индикатор. Такой проект хорош в качестве введения в предмет и только — он не выполняет никакой полезной функции, в чем заключается вся суть Arduino.

Чтобы обеспечить возможность вашим проектам Arduino действительно что-то делать, как правило, необходимо ввести в процесс электронные датчики и приводы.

Датчики

Датчики представляют собой устройства, реагирующие на физические изменения в среде, например температуры, движения, света и т. д., и либо генерирующие электрический сигнал (пассивные датчики), либо изменяющие свойство, как, к примеру, уровень сопротивления устройства (активные датчики). Эти сигналы пропорциональны степени изменения и обеспечивают ввод в цепь.

Датчики могут быть либо аналоговыми, либо цифровыми.

Чтобы функционировать, большинство активных датчиков требуют наличия внешнего источника электропитания, так называемого сигнала возбуждения. Пассивным датчикам, с другой стороны, не нужен источник электропитания, так как они генерируют электрический сигнал в ответ на внешние воздействия.

Существует множество различных типов датчиков, которые можно использовать для обнаружения широкого спектра характеристик. Далее перечислены некоторые из них: ускорение, свет, цвет, радиация, звук, температура, влажность, касание, жидкое состояние и расстояние.

Приводы

В цепи сигнал, снимаемый с датчика, обрабатывается и затем отправляется на внешнее устройство для выполнения определенной функции. Это устройство известно как привод и преобразует электрическую или магнитную энергию в другой вид энергии, например движение, свет, тепло и т. д.

Любое устройство, которое питается от одного типа энергии и производит на выходе другой тип энергии, можно классифицировать как привод. Электрический двигатель — один из наиболее известных типов привода.

Датчики, чаще всего используемые с Arduino

Доступны датчики, которые могут обнаруживать практически каждое свойство, известное человеку. Это означает, что область исследования ваших проектов ограничена только вашей фантазией и изобретательностью.

Для начала, однако, вы, вероятно, ограничитесь более простыми проектами, которые, как правило, будут включать перечисленные далее датчики.

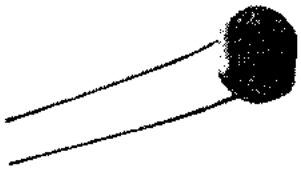
Датчики давления

Резистивный датчик давления

Существуют разные типы датчиков давления, используемых в проектах Arduino. Одним из наиболее распространенных является резистивный датчик давления. Он используется для определения физического давления, такого как толкание, сжимание и вес.

В эксплуатации сопротивление этого датчика пропорционально давлению, которое к нему применяется. Чем выше давление, тем ниже сопротивление.

Датчики освещенности

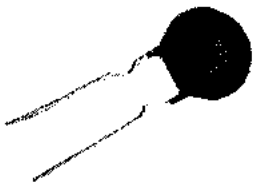


Фоторезистор

Типы датчиков освещенности включают фотоэлементы, фоторезисторы, фотодиоды и датчики приближения и освещенности. Фотоэлементы поставляются в виде больших панелей, которые не подходят для проектов Arduino.

Но существуют другие, подходящие типы датчиков. Благодаря мгновенному отклику фотодиоды используются в качестве переключателей в цифровых цепях, например пультах дистанционного управления. Датчики приближения реагируют на инфракрасное излучение и часто используются в робототехнике, в то время как фоторезисторы используются для измерения и реагирования на уровень освещенности.

Датчики температуры



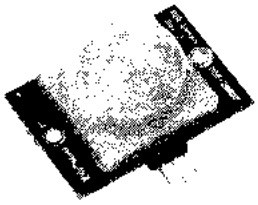
Термистор

Как и в случае с другими датчиками, существуют различные типы датчиков температуры. Двумя из наиболее распространенных типов этих датчиков являются термисторы и аналоговые датчики.

В случае с термисторами изменения температуры влияют на сопротивление устройства. Обычно термисторы применяются в цифровых термостатах. Аналоговые датчики температуры представляют собой полупроводниковые устройства и используются тогда, когда необходима точность измерения. Они также надежны и дешевы.

Датчики движения

Наиболее часто используемым типом датчиков движения, когда речь идет про Arduino, является пассивный инфракрасный датчик. Эти устройства обнаруживают изменения в инфракрасном излучении. Они компактны, недороги и просты в использовании. Чаще всего такой тип датчиков используется в домашних системах безопасности.



Пассивный инфракрасный датчик

При работе с микроволновыми и ультразвуковыми волнами используется другой тип датчиков движения. Также такие датчики используются в видеокамерах для обнаружения движения в поле зрения с помощью соответствующего программного обеспечения. Благодаря этим датчикам запись видео может начинаться после обнаружения движения.

7

Цепи

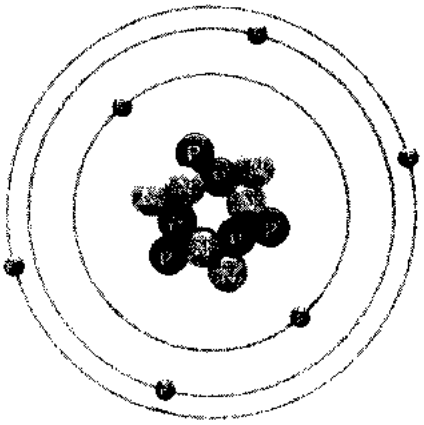
Эта глава представляет собой учебное пособие по теме «электрические цепи». Вы узнаете важные принципы, такие как закон Ома, различные типы тока и цепей, как использовать резисторы и конденсаторы и многое другое.

- **Концепции электричества**
- **Падения напряжения**
- **Мощность**
- **Последовательные и параллельные цепи**
- **Последовательные цепи и закон Ома**
- **Параллельные цепи и закон Ома**
- **Сопротивление в цепях**
- **Емкость в цепях**
- **Переменный и прямой ток**

Концепции электричества

Что такое электричество?

Все объекты состоят из мельчайших строительных блоков, известных как атомы. Каждый атом, в свою очередь, состоит из трех небольших компонентов — частиц: протонов, нейтронов и электронов.



Электроны содержат отрицательный заряд, протоны — положительный, а нейтроны — нейтральный, не имея ни положительного, ни отрицательного заряда.

На заметку

Чем больше положительных протонов или отрицательных электронов присутствует в атоме, тем сильнее притяжение и, таким образом, тем сильнее поток тока. Это притяжение известно как «заряд».

Протоны и нейтроны тесно взаимосвязаны в ядре атома и обладают высокой устойчивостью к смещению. Электроны не связаны и вращаются вокруг ядра на значительном расстоянии. В результате они гораздо менее устойчивы к смещению, и их легко выбить с орбиты.

Атомы содержат равное количество протонов и электронов, поэтому они сбалансированы и стабильны. Однако если атом теряет электрон, то он имеет больше протонов, чем электронов, и является положительно заряженным. Атом, который получает дополнительные электроны, с другой стороны, имеет больше отрицательных частиц, и таким образом имеет отрицательный заряд. Если рядом также находятся атомы, заряд которых положителен или отрицателен, они начнут взаимодействовать друг с другом.

Это происходит в виде потока электронов, передающегося от одного атома к другому. Один электрон оставляет свой атом и присоединяется к соседу, оставляя исходный атом без электрона. Тот, в свою очередь, притягивает электрон от другого атома. Этот поток электронов от отрицательного к положительному называется током и известен нам как электричество.

Поток электронов (электричество) всегда идет от отрицательного к положительному заряду.

Проводники и изоляторы

Степень движения электронов в атоме зависит от материала. С некоторыми типами материалов, такими как металлы, электроны в атомах настолько слабо связаны с ядром, что для их перемещения требуется очень мало усилий.

В других материалах, таких как стекло, электроны в атоме связаны с ядром гораздо более тесно, и поэтому, чтобы заставить их двигаться, необходимо затратить гораздо больше усилий.

Степень движения электронов в материале называется электрической проводимостью. Это определяется двумя основными факторами: типом атома в материале и способом связи атомов между собой.

Материалы, в которых электроны могут перемешаться легко, называются проводниками. Материалы, в которых электроны тесно связаны (свободных электронов мало или нет совсем), называются изоляторами. Хорошие проводники — металлы (наилучший — серебро), графит и вода. Хорошими изоляторами служат стекло, резина, масло, керамика, дерево и воздух.

Электропроводимость некоторых материалов изменяется при нагреве или охлаждении. Например, при нагреве до очень высокой температуры стекло становится хорошим проводником.

Цепи

Создание потока электронов — это одно дело; использование его с практической точки зрения — совсем другое. Первое, что необходимо сделать — это найти способ направить поток туда, куда нам нужно, т. е. управлять потоком.

Это осуществляется за счет создания непрерывного физического пути для электронов, поэтому у них нет выбора кроме того, чтобы двигаться туда, куда их направили. Мы добиваемся этого путем создания цепей.

Изготовленные из проводящего материала, обычно меди, который помогает потоку электронов, цепи образуют замкнутый контур, в который помещаются различные компоненты, чтобы изменять свойства электрического тока, такие как сила, частота, полярность и т. д.

Это показано на рисунке ниже.

Важно помнить о том, что, если разорвать цепь, поток электронов прервется. Не имеет значения, где именно будет разрыв.

На данном примере можно увидеть, что, когда существует непрерывный путь между источником электронов и местом, куда электроны направляются, поток электронов обязан пройти по этому маршруту. Однако если в цепи имеется разрыв, электроны не смогут по ней протекать

Напряжение

Однако необходимость непрерывного пути — не единственное условие того, чтобы поток электронов мог существовать. Также необходимо инициировать поток электронов, т. е. запустить его и поддержать его течение после того, как движение началось. Другими словами, должна быть какая-то сила или давление, чтобы физически заставить электроны двигаться по цепи.

Мы добиваемся этого с помощью аккумуляторных батарей и электрических генераторов. В случае с аккумуляторами данные устройства состоят из анода, катода и электролита. Во всех батареях анод является источником, а катод пунктом назначения.

Химическая реакция в аккумуляторной батарее создает накопление электронов на аноде. Это приводит к электрической разнице, или неравномерности, между анодом и катодом. Чтобы восстановить баланс, электроны на аноде попробуют добраться до катода, но не смогут этого сделать, потому что их останавливает электролит. Таким образом, создается сила, или энергия.

Поскольку эту энергию нельзя освободить, она известна как потенциальная энергия, или более широко известное название — напряжение. Однако если между катодом и анодом подключить провод, образуется цепь, и энергия, таким образом, освобождается. Потечет ток, и это будет продолжаться до тех пор, пока аккумулятор будет производить напряжение.

Хотя, если прервать цепь в любой точке, движение тока немедленно прекратится — тока не будет ни на одном участке цепи.

Если цепь разорвана, напряжение на батарее также будет присутствовать между двумя концами разрыва. Конец, подключенный к положительно заряженной стороне батареи (катоду), также будет положительно заряженным, а конец, подключенный к отрицательно заряженной стороне (аноду), будет отрицательно заряжен. Это явление известно как полярность.

Сила потока электронов и, следовательно, сила тока пропорциональна напряжению, применяемому к цепи. Подключите последовательно две батареи 1,5 В, и напряжение в цепи будет равно 3 В, а ток будет в два раза выше, чем от одной батареи.

Сопротивление

Когда электроны текут по проводнику, они встречают естественное сопротивление, которое препятствует продвижению, что ослабляет поток. Это сопротивление присутствует в каждом типе материала. Одни материалы обладают гораздо большим сопротивлением, чем другие.

Это обусловлено столкновениями между перемещающимися электронами и другими элементами, известными как ионы.

Уровень сопротивления, предлагаемый проводником, определяется несколькими факторами.

- **Атомная структура проводника** — в некоторых материалах, например металлах, электроны могут двигаться гораздо более свободно, чем в других, к примеру, в стекле. Эти материалы оказывают гораздо меньше сопротивления прохождению тока.
- **Длина проводника** — сопротивление длинного провода больше, чем короткого, потому что электроны более часто сталкиваются с ионами.

- **Толщина проводника** — сопротивление тонкого провода больше, чем толстого, потому что тонкий провод содержит меньше электронов, чтобы переносить ток.

Сопротивление представляет собой показатель того, насколько объект препятствует прохождению электронов. Единица измерения электрического сопротивления — ом (Ом).

Феномен сопротивления имеет два нежелательных последствия.

- **Нагрев.** Когда двигающиеся электроны сталкиваются с другими элементами в проводнике, результирующая энергия выделяется в виде тепла. Это может привести к повреждению компонентов, а также быть опасным для пользователей. Это добавляет еще одну переменную величину, которую следует учитывать при проектировании цепей.
- **Потеря энергии.** Поскольку энергия дорогая, потерю энергии в виду тепла необходимо свести к минимуму. Так же как с необходимостью бороться с нагревом, потеря энергии является фактором, который может повлиять на проектирование цепей.

Хотя все не так уж и плохо. Сопротивление определенных значений можно намеренно ввести в цепи в виде *резисторов*. Эти компоненты играют очень важную роль в электронике, поскольку они позволяют с точностью управлять прохождением тока в цепи.

Энергия, потраченная движущимися электронами в процессе преодоления сопротивления в материале, называется нагревом.

Падение напряжения

Движению тока противодействуют не только резисторы. По сути, все электронные компоненты обладают собственным сопротивлением к нему. Чтобы преодолеть это сопротивление, электрический ток должен потреблять энергию по всему компоненту.

Это потребление энергии приводит к уменьшению напряжения, что описывается термином «падение напряжения».

Падение напряжения — это количество напряжения, которое используется, когда электроны проходят через сопротивление. Используется все напряжение в цепи, т. е. сумма всех падений напряжения будет равна напряжению источника.

В качестве примера выступает простая цепь, изображенная выше. Она включает в себя батарею 9 В, лампу и переключатель. При открытом выключателе, напряжение на клеммах батареи составит 9 В.

Когда вы замыкаете выключатель и таким образом замыкаете цепь, лампа загорается. Если бы сейчас вы измерили напряжение батареи, то увидели бы, что оно упало приблизительно до 7,5 В — разница составила бы 1,5 В.

Это падение в размере 1,5 В и является падением напряжения и вызвано количеством энергии, которую должна потратить батарея, чтобы преодолеть сопротивление лампы. Все компоненты в цепи, даже провода, представляют собой определенный уровень сопротивления и, таким образом, могут послужить причиной падения напряжения.

Измерение падения напряжения производится путем измерения напряжения на входе и на выходе устройства. Разница между этими двумя значениями называется падением напряжения.

Многие приборы работают стабильно, используя разный диапазон напряжения. В то же время другие приборы очень чувствительны к напряжению электропитания. Это означает, что следует учитывать все потенциальные падения напряжения во время проектирования цепей и напряжение электропитания должно быть скорректировано соответствующим образом.

Мощность

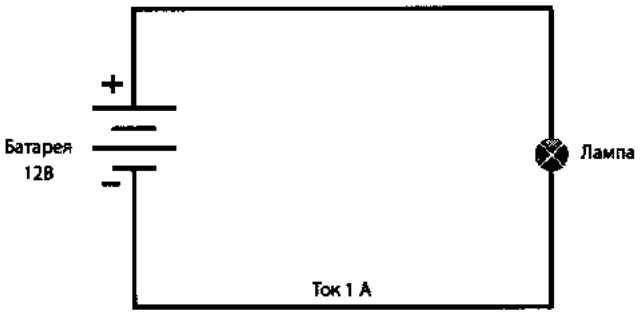
Сопротивление, напряжение и ток — не единственные активные величины в цепях; есть еще одна, которая называется мощностью. Это, по сути, измерение объема работы, которое можно выполнить за конкретный период.

В электрической цепи это задача как напряжения, так и тока в этой цепи. Поэтому она равна току, умноженному на напряжение. Соответственно, мощность вычисляется с помощью следующей формулы, где P — мощность, I — ток, а V — напряжение:

$$P = I \times V$$

Единица измерения мощности — ватт (Вт).

Единицу измерения электрической мощности, ватт, можно рассматривать как аналог механической мощности, выраженной в лошадиных силах. 1 лошадиная сила равна 746 Вт.



Следует четко понимать, что мощность представляет собой комбинацию напряжения и тока в цепи. Если вдуматься: напряжение — это конкретная работа на единицу заряда, тогда как ток — это скорость, с которой электрический заряд движется по проводнику.

Следовательно, поскольку напряжение аналогично работе, выполненной при поднятии веса с учетом сопротивления гравитационной тяги, а ток аналогичен скорости, с которой вес поднимают, вместе напряжение и ток представляют собой мощность.

Если в цепи нет напряжения или тока, там также отсутствует и мощность.

В простой цепи, показанной на рисунке выше, у нас есть батарея с напряжением 12 В и силой 1 А. Формула $P = I \times V$ сообщает нам, что лампа выделяет 12 Вт мощности в виде тепла и света.

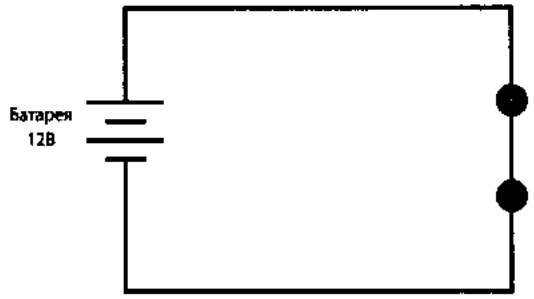
Если цепь в каком-то месте разорвана, движение тока прерывается, и мощность будет равна нулю. Аналогично, если напряжение по какой-то причине пропадает, мощность будет отсутствовать.

Последовательные и параллельные цепи

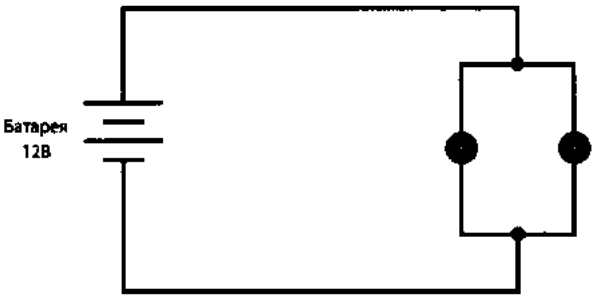
До сих пор в этой главе мы продемонстрировали вам несколько простых цепей. Они бывают двух типов: последовательные и параллельные. Сейчас мы объясним разницу между ними и покажем некоторые характерные области применения для каждого из этих типов.

В последовательной цепи ток в равной степени распределяется между компонентами. Падение напряжения на каждом компоненте будет отличаться в зависимости от сопротивления каждого компонента, но у них будет общее напряжение электропитания. Мы складываем все сопротивления, чтобы они равнялись общему сопротивлению.

В последовательной цепи все компоненты связаны непрерывно, поэтому для течения электричества остается только один путь. Это показано на схеме выше, которая состоит из батареи и двух ламп.



Параллельная цепь, показанная ниже, содержит такие же компоненты, но они подключены по-другому.



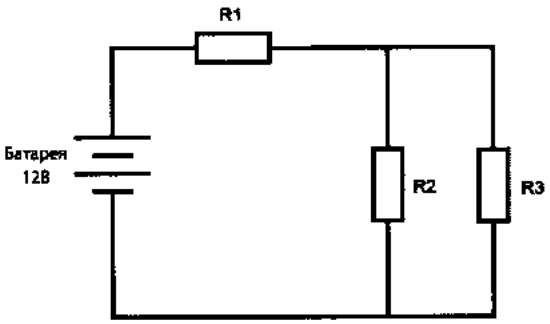
В параллельной цепи все компоненты в равной степени делят напряжение, токи в каждой ветви складываются в суммарный ток, а сопротивление становится равным суммарному сопротивлению цепи.

Вместо непрерывного подключения, когда ток течет через один компонент, чтобы попасть в другой, каждая лампа имеет свою собственную, не зависимую от другой ветвь. Эффект этого может быть двойственным.

- В последовательной цепи ток распределяется между лампами. Поэтому чем больше ламп, тем тусклее будет светить каждая из них. В параллельной цепи, однако, каждая лампа получает полный ток, и, в результате они светят ярче, чем в последовательной цепи. Не имеет значения, сколько имеется ламп в наличии, все они будут светить ярко.
- В последовательной цепи, если одна лампа перегорает, цепь разрывается, и поэтому протекание тока останавливается. В параллельной цепи этого не происходит, потому что независимо от того, какая лампа перегорела, по-прежнему есть свободный путь, проходящий через другие лампы. Уровень освещенности может быть вдвое меньше, но, тем не менее, он будет.

Однако в реальном мире очень немного таких простых цепей. Большинство из них, по сути, являются комбинированными цепями, содержащими элементы, как последовательных, так параллельных цепей, подобно приведенному ниже примеру.

В комбинированной цепи источник электропитания, устройства управления или защиты обычно находятся в последовательной цепи; потребители электроэнергии — в параллельной.



В этой цепи ток течет от батареи через резистор R1, затем разделяется и в равном объеме проходит через резисторы R2 и R3, а затем воссоединяется, прежде чем пройти через другую сторону батареи.

Практическое использование последовательных цепей

Последовательные цепи чаще всего имеют простую конструкцию с относительно небольшим количеством компонентов. Одним из самых известных примеров является гирлянда, где при перегорании одной лампочки гаснут все огни.

Другим является фонарик, где есть батарейка, переключатель, провода и лампочка, и все они подключены последовательно.

Практическое использование параллельных цепей

Параллельные цепи встречаются везде. Цепь освещения в вашем доме является параллельной, так же как и уличное освещение. Элементы управления высоким, средним и низким уровнем нагрева в электронагревателе устанавливают

два элемента параллельно для получения полной мощности, один элемент для того, чтобы получить половину мощности, и два элемента последовательно для малой мощности.

Каждый раз при добавлении компонента в параллельную цепь из источника извлекается больше тока. Таким образом, мы можем перегрузить цепь, требуя больше тока, чем нужно для безопасного функционирования цепи.

Батареи, подключенные параллельно, производят больше тока. Типичным применением этого является система зажигания судового двигателя.

Последовательные цепи и закон Ома

Мы узнали, что поток электронов вдоль проводящего пути называется током. Также мы увидели, что сила, необходимая, чтобы заставить электроны двигаться, называется напряжением. Более того, движение тока зависит не только от напряжения в цепи, но также и от сопротивления в этой цепи.

Таким образом, налицо существование фундаментального соотношения между тремя этими величинами. В частности, для данного сопротивления ток прямо пропорционален напряжению. Другими словами, если вы повысите напряжение в цепи, сопротивление которой фиксировано, ток будет увеличиваться. Если вы уменьшите напряжение, то и ток будет уменьшаться.

Ключевым моментом в понимании принципа работы токов является знание закона Ома.

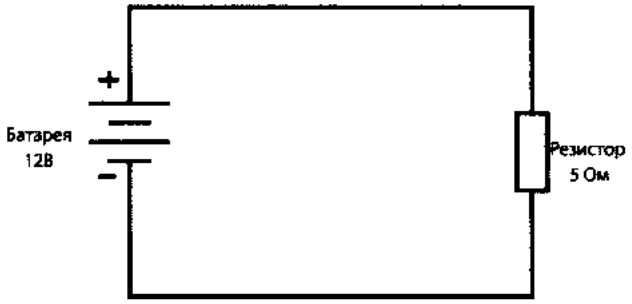
Это взаимоотношение известно как закон Ома и выражается в виде следующей математической формулы:

$$V = I \times R$$

Из этой формулы мы видим, что напряжение равно току, умноженному на сопротивление. Таким образом, если вам известен ток и сопротивление в цепи, вы можете использовать две эти величины, чтобы рассчитать напряжение. Более того, немного изменив формулу, мы можем рассчитать ток и сопротивление:

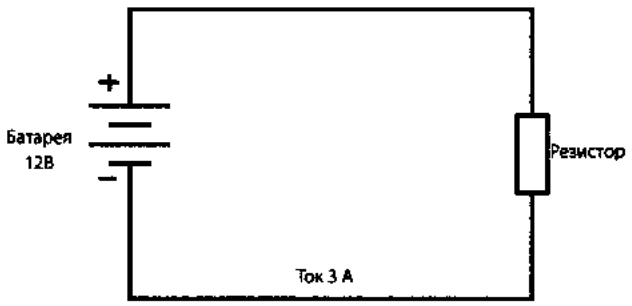
$$I = \frac{V}{R} \qquad R = \frac{V}{I}$$

Если мы разделим напряжение на сопротивление, мы получим ток. А разделив напряжение на ток, мы получим сопротивление. Давайте посмотрим, как это работает на практике.



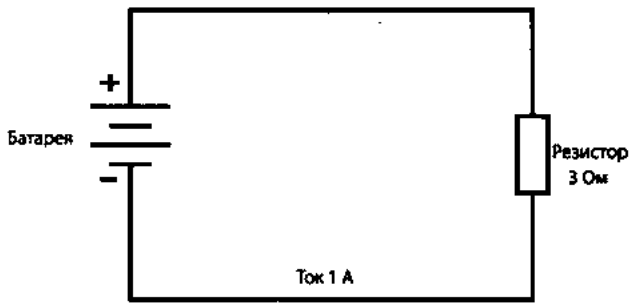
На схеме, представленной выше, ток вычисляется следующим образом:

$$I = \frac{V}{R} = \frac{12\ В}{5\ Ом} = 2{,}5\ А$$



В данной цепи мы знаем, что напряжение составляет 12 В, а сила тока — 3 А. Значение резистора вычисляется по формуле:

$$R = \frac{V}{I} = \frac{12\ В}{3\ А} = 4\ Ом$$



И наконец, для вычисления напряжения с известным сопротивлением и током:

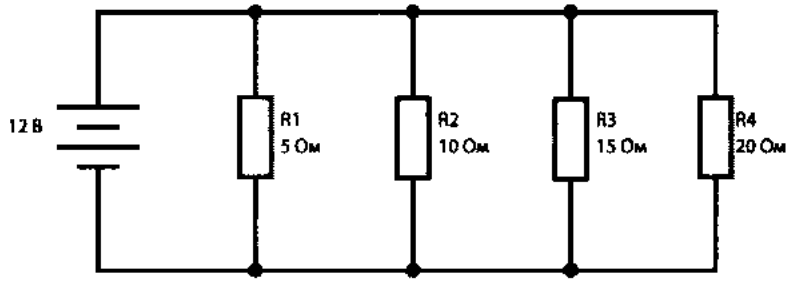
$$V = I \times R = 1\ А \times 3\ Ом = 3\ В$$

Закон Ома — очень важное понятие, когда речь идет о цепях. Мы будем постоянно им пользоваться. Вам следует знать эти три формулы.

Параллельные цепи и закон Ома

Уравнения закона Ома также можно применять и к параллельным цепям. Однако в связи с различным способом расположения компонентов в этих цепях уравнение следует изменить.

Изменение необходимо только при расчете значений сопротивления — ток и напряжение вычисляются так же, как мы делаем это в последовательных цепях — $I = V/R$ и $V = I \times R$.



На заметку. Напряжение на резисторах, подключенных параллельно, одинаково для каждого резистора.

Цепь, изображенная выше, содержит четыре резистора, подключенные параллельно. Если бы они были подключены последовательно, мы просто суммировали бы значения, чтобы получить общее значение. В цепи с параллельным подключением это не сработает.

$$R_т = \frac{1}{\frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} + \frac{1}{R4}}$$

Чтобы получить общее сопротивление, мы должны использовать обращенную формулу, показанную выше справа. Это работает следующим образом:

$$R_т = \frac{1}{\frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} + \frac{1}{R4}} = \frac{1}{\frac{1}{5} + \frac{1}{10} + \frac{1}{15} + \frac{1}{20}}$$

$$\frac{1}{0{,}2 + 0{,}1 + 0{,}06 + 0{,}05} = \frac{1}{0{,}416} = 2{,}4 \text{ Ом}$$

Таким образом, четыре подключенные последовательно резистора в сумме дадут 50 Ом. Те же четыре резистора, подключенные параллельно, дадут общее сопротивление 2,4 Ом. Это можно объяснить следующим образом.

При вычислении общего сопротивления в параллельной цепи вы не можете просто суммировать значения сопротивлений, как вы сделали бы в последовательной цепи.

Предположим, что у нас есть цепь с двумя резисторами с сопротивлением 4 Ом, подключенными в параллельные ветви. Эта цепь предлагает два равных пути для потока электронов, где половина заряда проходит через резистор в одной ветви, а другая половина — через резистор в другой ветви.

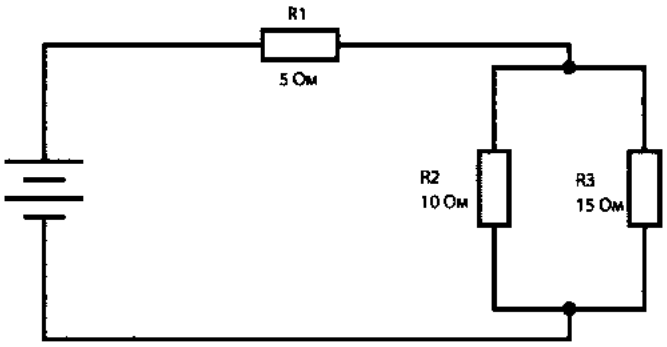
Таким образом, только половина полного заряда в цепи встретит сопротивление 4 Ом каждой ветви. Следовательно, пока аккумуляторная батарея производит заряд, два резистора с сопротивлением 4 Ом в параллельном соединении эквивалентны резистору с сопротивлением 2 Ом в цепи.

Используя формулу выше, любое количество резисторов в параллельной цепи можно эффективно свести к одному резистору. Как только мы будет знать это одно значение, мы сможем вычислить напряжение и ток, как описано на с. 136–138.

Когда мы имеем дело с комбинированными цепями, включающими в себя как параллельные, так и последовательные элементы, все становится гораздо сложнее. Хитрость заключается в том, чтобы изолировать параллельные

элементы (их может быть несколько) и вычислить общее сопротивление каждого с помощью формулы на с. 139. Затем повторить действие, пока в результате не останется всего один элемент. Теперь все, что вам необходимо сделать — это добавить его сопротивление к последовательным сопротивлениям, чтобы получить общее сопротивление в цепи.

Например:



В этой цепи резисторы R2 и R3 параллельны. Первый шаг заключается в вычислении общего сопротивления двух резисторов.

Как только вы вычислили общее сопротивление в параллельной цепи, вы можете вычислить напряжение и ток в последовательной цепи.

$$\frac{1}{\frac{1}{10}+\frac{1}{10}} = \frac{1}{0{,}1+0{,}06} = \frac{1}{0{,}16} = 6{,}2 \text{ Ом}$$

Фактически резисторы R2 и R3 образуют один резистор с сопротивлением 6,2 Ом. Так как он последовательно подключен к резистору R1, просто добавьте его сопротивление, получив общее сопротивление цепи в размере 11,2 Ом.

Сопротивление в цепях

Мы узнали, что представляют собой резисторы, изучили различные типы резисторов, как они работают, и как вычислить их сопротивление по цветовому коду.

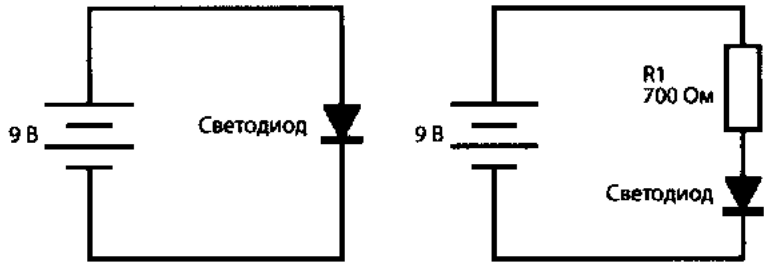
А теперь рассмотрим некоторые распространенные способы использования резисторов в цепях.

Защита

Все электронные компоненты имеют определенные допуски. Если указанные допуски превышены, т. е. какой-либо элемент подвергается воздействию чрезмерного тока, элемент может повредиться, что приведет к возможному отказу цепи. В этой ситуации поможет использование резисторов.

При подключении резистора не следует беспокоиться насчет полярности. Ток проходит через него в любом направлении.

В цепи, изображенной слева внизу, у нас есть светодиодный индикатор, подключенный напрямую к батарее 9 В. Сила тока будет гораздо выше той, на которую рассчитан светодиодный индикатор, и он сразу же перегорит.



Чтобы не допустить этого, мы помещаем резистор 700 Ом перед светодиодным индикатором, как показано на схеме справа. Это ограничит силу тока до уровня, который подходит светодиодный индикатору — как правило, от 10 до 20 мА.

Чтобы вычислить необходимое сопротивление, мы используем закон Ома. Следовательно, мы должны знать как напряжение, так и силу тока в цепи.

Помните, что закон Ома работает с основными единицами, т. е. омами, вольтами, амперами. В нашем примере 10 мА необходимо преобразовать в 0,01 А.

На светодиодных индикаторах происходит падение напряжения на 2 В (эта информация указана в характеристиках светодиодных индикаторов), таким

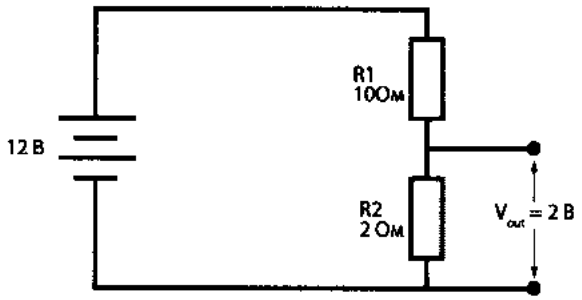
образом общее напряжение будет равно 9 – 2 = 7 В. Нам необходим ток силой около 10 мА (также указано в характеристиках светодиодных индикаторов), поэтому уравнение будет выглядеть следующим образом:

$$R = \frac{V}{I} = R = \frac{7}{0{,}01} = 700 \text{ Ом}$$

Делители напряжения

Делитель напряжения — это цепь, в которой два резистора используются для преобразования высокого напряжение в низкое.

Если два резистора равны по значению сопротивления, в таком случае, выходное напряжение равно половине напряжения на входе. Это истина, независимая от сопротивления резисторов.



Принцип работы делителей напряжения очень прост. На схеме, изображенной выше, ток течет из батареи к резистору R1. В резисторе R1 10 В падают, оставляя 2 В на резистор R2. Так как выход берется с резистора R2, напряжение составит 2 В.

$$V_{out} = V_{in} \times \frac{R2}{R1 + R2}$$

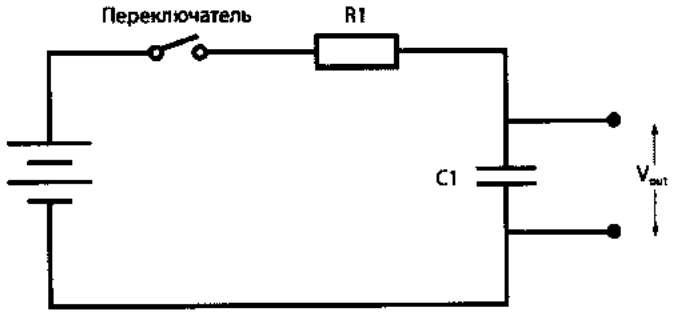
Если вы измените сопротивление резистора R1 на 5 Ом, а резистора R2 — на 7 Ом, вывод будет равен 7 В. Вывод любой цепи делителя напряжения можно установить с помощью математической формулы, приведенной выше.

Уравнение гласит, что напряжение на выводе прямо пропорционально напряжению на входе и соотношению резисторов R1 и R2.

Цепь синхронизации/задержки

При использовании в сочетании с конденсатором резистор образует цепь синхронизации/задержки.

В цепи синхронизации сопротивление резистора определяет скорость заряда конденсатора и, следовательно, длительность задержки.



Когда переключатель замкнут, ток течет через резистор и начинает заряжать конденсатор. Пока конденсатор заряжается, он позволяет току протекать в цепь нагрузки (V_{out}). Когда он полностью заряжен, движение тока в цепь загрузки прекращается.

Время между началом и остановкой движения тока называется «задержка», период которой можно задать с помощью значений резистора и конденсатора. Его можно использовать для запуска и остановки внешних событий.

Емкость конденсатора в цепях

Как и в случае с резисторами, конденсаторы используются практически во всех электрических цепях. Во многом эти элементы хранят электрический заряд так же, как это делают аккумуляторные батареи, и это свойство можно использовать по разному. Выше мы увидели, как их можно использовать в сочетании с резисторами для создания цепи синхронизации; теперь мы рассмотрим еще несколько применений для этого универсального компонента.

Фильтрация

Все конденсаторы имеют собственное внутреннее сопротивление течению тока, известное как емкостное сопротивление. В отличие от постоянных резисторов, где сопротивление не меняется, сопротивление конденсатора изменяется в зависимости от частоты применяемого сигнала.

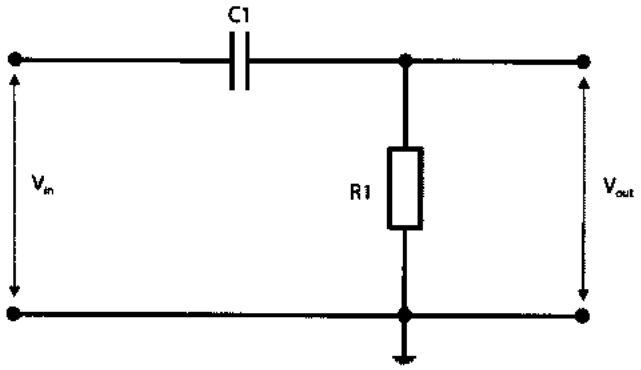
Емкостное сопротивление — это внутреннее сопротивление конденсатора течению тока.

Когда частота растет, сопротивление падает, и через конденсатор проходит больше тока. Когда частота падает, сопротивление растет, ограничивая тем самым прохождение тока. Фактически конденсатор действует как переменный резистор, сопротивление меняется, когда меняется частота используемого сигнала.

Следовательно, конденсатор (в сочетании с резистором) можно использовать для передачи или блокирования определенных диапазонов частот. Выступая в данном качестве, он носит название фильтр. Высокочастотные фильтры пропускают высокие частоты, низкочастотные — низкие, а полосовые — частоты в определенном диапазоне, или спектре.

По сути, фильтр — это цепь, которая будет отфильтровывать нежелательные частоты в электрическом сигнале. Будут пропущены только те частоты, которые предусмотрены.

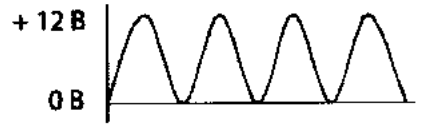
Высокочастотный фильтр показан на схеме ниже.



В фильтре такого типа ток проходит сначала через конденсатор, тогда как в низкочастотном фильтре ток сначала проходит через резистор.

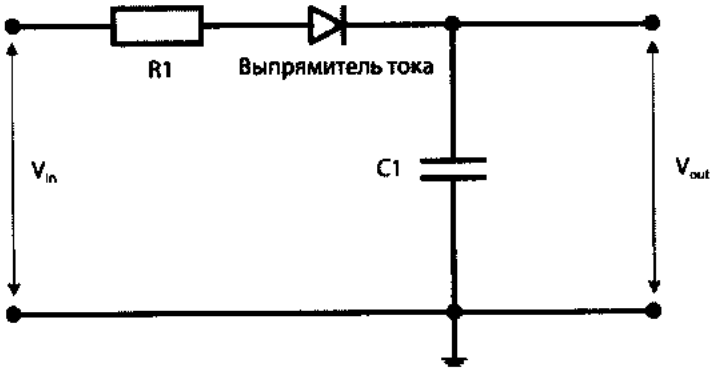
Сглаживание

Также очень важно то, что конденсаторы можно использовать в цепях электропитания, где переменный ток преобразуется в постоянный. Как правило, эти цепи используют диодный выпрямитель, который в основном отсекает отрицательную (нижнюю) половину сигнала переменного тока. Однако он по-прежнему оставляет положительную (верхнюю) половину, которая будет значительно изменяться от нуля до максимального значения.

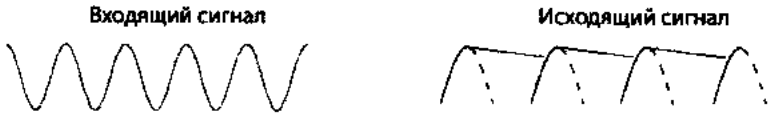


В сглаживающих цепях обычно используются электролитические конденсаторы.

Форма колебаний сигнала, приведенная выше, показывает выпрямленный сигнал переменного тока, напряжение которого непрерывно колеблется между 0 и 12 В, что совершенно не подходит для цепи, которой требуется ток с неизменным напряжением. Решением этого вопроса может выступать размещение сглаживающего конденсатора на выпрямленном выводе, как показано на схеме ниже.



C1 — это сглаживающий конденсатор. Когда напряжение на выводе из выпрямителя растет, конденсатор заряжается. Когда напряжение на выводе выпрямителя падает, конденсатор разряжается, тем самым поддерживая течение тока. Форма колебаний входного и выходного сигнала показаны ниже.



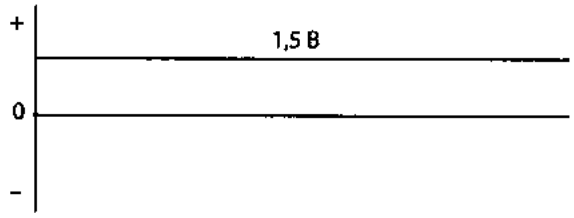
Чем выше амплитуда колебаний тока и, следовательно, больше волны, тем большего размера необходим конденсатор.

Переменный и постоянный ток

Существует два типа тока — постоянный (DC, direct current) и переменный (AC, alternating current). Оба имеют как преимущества, так и недостатки, которые делают их более пригодными для одних целей и менее пригодными для других.

Постоянный ток

При постоянном токе поток электронов постоянен и течет в одном направлении. Поэтому он и называется постоянным. Графически это выглядит следующим образом.



Постоянный ток используется в большинстве цифровых цепей, так как он идеально подходит для устройств, где требуется малая мощность и напряжение

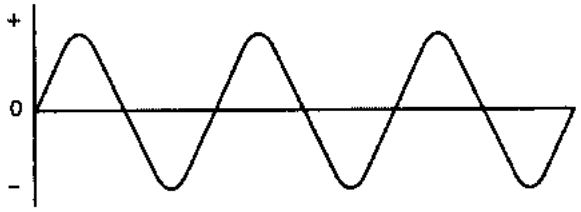
с малой амплитудой. Он используется в большинстве электроприборов, компьютеров, игровых контроллеров и т. д.

Постоянный ток можно получить несколькими способами. К ним относятся солнечные батареи, выпрямители и аккумуляторные батареи. Переменный ток можно получить с помощью генераторов и альтернаторов.

Переменный ток

При переменном токе поток электронов не постоянен и не течет в одном направлении, как при постоянном токе. Наоборот, он периодически меняет направление, двигаясь от положительного заряда к отрицательному, потом снова к положительному и т. д.

Форма колебаний сигнала переменного тока показана ниже.



В России переменный ток меняет направление 50 раз в секунду, т. е. частота составляет 50 Гц.

Существует большое количество форм колебаний сигнала переменного тока. Наиболее распространена синусоида, показанная выше. Также распространена квадратная волна, используемая в цифровых цепях.

Переменный ток больше подходит для областей применения, где требуется большой уровень мощности и большой радиус действия, как, например, промышленные источники электропитания. Получить переменный ток гораздо проще, чем постоянный. Во многих приборах с низким энергопотреблением, таких как компьютеры и бытовая техника, переменный ток преобразуется в постоянный с помощью цепи внутри прибора или адаптера электропитания.

8

Программирование Arduino

Возможность программирования — неотъемлемая составляющая при создании проектов с помощью платы Arduino. В этой главе мы рассмотрим, как это делается, взглянем на наиболее распространенные элементы и команды, используемые в написании программного кода. К ним относятся функции, циклы, переменные, условные инструкции, константы и многое другое.

- Концепции программирования
- Комментарии
- Функции
- Переменные
- Типы данных
- Инструкции
- Арифметика и логика
- Массивы
- Побитовые операторы
- Интерфейсы ввода/вывода
- Время
- Другие полезные функции
- Структура скетча

Концепции программирования

Сам по себе микроконтроллер на плате Arduino не может делать абсолютно ничего — ему необходимо давать указания. Эти указания называются скетчами, а процесс написания скетчей известен как программирование.

Существует большое количество языков программирования, наиболее популярны из которых Java, Python, Ruby, PHP, C и C++. Упрощенной версией C и C++ является язык, который используется для программирования микроконтроллеров Arduino.

Если вы уже знаете язык C, вам будет проще продолжить знакомство с материалом. Однако если он вам еще не известен, не следует переживать, так как для написания скетчей необходимо знать относительно небольшое количество команд. Их можно сгруппировать в четыре категории.

- Команды управления, которые определяют выражения, выполняют арифметические действия и т. д.
- Команды моментального перехода, благодаря которым программа незамедлительно переходит к другой части программы.
- Команды перехода, оценивающие состояние и выполняющие переход, если условие истинно.
- Команды цикла, которые повторяют фрагмент кода определенное количество раз.

Большинство языков программирования содержат относительно небольшое количество команд. Выполнение сложных операций возможно благодаря способности компьютера комбинировать и повторять инструкции миллионы раз в секунду.

В этой главе объясняется, что представляют собой эти команды и как использовать их для программирования вашей платы Arduino.

Прежде всего, важно понимать, что Arduino будет выполнять в точности то, что ей предписано делать — ни больше ни меньше. Она понимает только то, что находится в скетче — она не имеет никакого представления о намерениях программиста!

Более того, код скетча должен быть сформулирован с помощью специального синтаксиса, чтобы микроконтроллер Arduino правильно его интерпретировал.

Среди программистов популярна поговорка «Что посеешь, то пожнешь».

Программа, встроенная в интегрированную среду разработки Arduino, известная как компилятор, считывает определенные формулировки текста, написанные в редакторе, и преобразует их в язык микроконтроллера, или код.

Затем он загружается в микроконтроллер, который запускает скетч, построчно выполняя код.

Комментарии

Комментарии указываются в коде человеком, который пишет скетч, и представляют собой напоминания с целью упростить работу другим людям. Их делают для того, чтобы объяснить, что выполняет скетч или определенный фрагмент его кода или же то, как он работает.

Комментарии игнорируются в процессе компиляции, и, следовательно, они никак не влияют на скетч. Комментарии могут быть однострочными и многострочными.

Многострочные или блочные комментарии заключаются в символы `/*` и `*/`. Они обозначают начало и окончание комментария или комментариев.

Комментарии не относятся к исполняемому коду и игнорируются компилятором.

Например:

```
/*
  Blink
  Циклически включает светодиодный индикатор на одну
  секунду, затем отключает на одну секунду. Этот
  образец кода не защищен авторским правом.
*/
```

Комментарий сообщает читателю, что Blink — это название скетча, объясняет, что данный скетч делает, и сообщает, что код не защищен авторским правом.

Однострочные комментарии начинаются с символов `//`.

Например:

```
// Светодиодный индикатор подключен к цифровому
контакту 13.
int ledPin = 13;
```

Все, что указано после символов `//` в любом однострочном комментарии, игнорируется компилятором. В данном примере текст «`Светодиодный индикатор подключен к цифровому контакту 13.`» представляет собой комментарий и сообщает читателю о том, к какому контакту подключен светодиодный индикатор.

Вторая строка, «`int ledPin = 13;`», представляет собой код программы и будет выполняться процессором.

Комментарии наглядны в редакторе Arduino, поскольку выделены серым цветом.

Функции

Функции представляют собой набор инструкций, которые можно использовать в любом месте в скетче. Они выполняют конкретную задачу, и очень часто эта задача повторяется.

Вместо того чтобы постоянно писать определенный фрагмент кода, вы можете вызвать функцию, чтобы скетч выполнил задачу столько раз, сколько потребуется.

На заметку. Существует соглашение о присвоении имен функциям, имеющим в названии более одного слова. Поскольку в имени не может быть пробелов, каждое новое слово пишется с прописной буквы, чтобы сделать его читабельным, например `pinMode`.

Функции могут быть либо встроенными (входящими в состав языка Arduino), либо же созданными пользователями.

Фигурные скобки используются для обозначения начала и конца функции. Весь код функции должна быть заключена в скобки, как показано ниже:

```
void setup() {
  pinMode(led, INPUT);
}
```

Функция setup()

Функция, показанная на примере выше, носит имя `setup()` и является неотъемлемой частью любого скетча Arduino, даже если фактически не используется, т. е. внутри фигурных скобок код не вводится.

Это первая функция, которую считывает из скетча процессор, и она содержит все инструкции, необходимые для настройки платы Arduino. Среди прочего, к ним относятся переменные инициализации, режимы контактов и библиотеки, а также установка различных начальных значений.

Функция выполняется только один раз — после каждого включения электропитания или перезагрузки платы Arduino.

Функция loop()

Также для запуска скетча важно, чтобы чтение функции `loop()` происходило после функции `setup()`. Как правило, эта функция содержит большую часть кода скетча, и здесь происходит управление возможностями платы Arduino.

Самыми важными функциями Arduino являются `setup()` и `loop()`. Без наличия какой-либо из этих функций скетч попросту не запустится.

```
void loop() {
  digitalWrite(13, HIGH);
```

Имя функции расшифровывается как «цикл», потому что она запускается систематически, т. е. работает в цикле до тех пор, пока что-то ее не остановит. Действуя так, она позволяет скетчу изменяться и реагировать на события, такие как входные значения с датчиков.

Переменные

Переменные позволяют присваивать имена и хранить числовые значения, чтобы их можно было далее использовать в скетче. Например, данные с датчика или значение, используемое в вычислениях.

Как следует из названия, переменные можно изменять в любой момент, в отличие от констант, значение которых всегда остается одним и тем же.

Объявление и инициализация переменных

Прежде чем их можно будет использовать, все переменные необходимо объявить. Объявление означает определение типа и при необходимости установку начального значения. Последнее действие известно как инициализация переменной.

Переменная фактически представляет собой метку, присваиваемую фрагменту данных. Она обеспечивает простой метод хранения, изменения и доступа к данным.

Переменную необходимо объявить всего лишь один раз. Однако, даже сделав это, значение, присвоенное переменной, всегда можно изменить с помощью вычислений и назначений разных типов.

Например:

```
int inputVariable = 0;          // Объявляет переменную
                                   и присваивает ей
                                   значение равное 0
inputVariable = analogRead(2);  // Задает переменной
                                   значение
                                   аналогового
                                   контакта 2
```

Первая строка кода объявляет, что `inputVariable` представляет собой `int` (целое число) и что оно имеет начальное значение, равное 0. Вторая строка указывает, что переменная считывает значение входного напряжения на контакте **A2**.

Как только переменная объявлена, ее используют, присваивая переменной значение, которое хотят сохранить с ее помощью. Это делается с помощью оператора присваивания, указываемого в виде символа =. Оператор присваивания

сообщает скетчу о необходимости поместить то, что находится справа от него, в переменную, находящуюся слева.

Например:

```
inputVariable = 4;
inputVariable2 = analogRead(2);
```

Первая строка присваивает переменной с именем `inputVariable` значение, равное 4. Вторая строка присваивает переменной с именем `inputVoltage2` значение входного напряжения на контакте **A2**.

Когда переменные объявлены, их не нужно инициализировать (присваивать значение), но если вы сделаете это, часто это может быть полезным.

Область видимости переменной

Термин «область видимости» при использовании в отношении переменных в Arduino означает, до какой степени данная переменная доступна для конкретного скетча. Это зависит от того, какой из двух типов переменной используется.

Глобальные переменные

К первому типу относится глобальная переменная. Такие переменные могут использоваться *всеми* функцией в скетче, отсюда и название. Глобальные переменные объявляются в самом начале скетча, перед функцией `setup()`.

Например:

```
int pin = 13;          // Это глобальная переменная
void setup()
{
  pinMode(pin, OUTPUT);
}
void loop()
{
  digitalWrite(pin, HIGH);
}
```

Переменные, которые должны быть доступны для более чем одной функции, должны быть объявлены как глобальные переменные. Это осуществляется путем объявления их в самом начале скетча.

Как можно видеть, переменная `pin` используется в функциях `setup()` и `loop()`. Поскольку обе функции ссылаются на одну и ту же переменную, изменив ее значение в одной функции, мы изменим ее и в другой, поскольку переменная является глобальной.

Переменную можно объявлять в любом количестве позиций в коде скетча. Местоположение определяет, какие фрагменты кода скетча смогут ее использовать.

Локальные переменные

Ко второму типу относятся локальные переменные. Их объявляют внутри функции. Видеть и использовать их может только эта функция.

Например:

```
void setup()
{
  int pin = 13;
  pinMode (pin, OUTPUT);
  digitalWrite (pin, HIGH);
}
```

Переменные, объявленные внутри функции, известны как локальные переменные.

Переменная `pin` видна только функции `setup()`. Следовательно, она не может быть использована никакой другой частью кода скетча.

Типы данных

Типы данных

Объявляя переменную, мы сообщаем Arduino, что хотим сохранить в переменной некоторую информацию. В то же время мы также должны сообщить системе о типе данных, чтобы она могла выделить достаточное количество места в памяти.

`int` — наиболее часто используемый тип данных в Arduino.

Компьютеры работают в двоичной системе, и все данные представлены в последовательности 0 и 1. Они называются битами. Переменные, которые, по сути, просто являются битами информации, состоят из этих бит. Чем больше бит используется переменной, тем больше значений, или данных, она может содержать.

Диапазон значений, которые могут храниться в памяти Arduino — это 2 в степени количества используемых бит. Например, с помощью 8 бит мы можем сохранить до 256 различных значений, а с помощью 16 бит — до 65 536 значений.

Переменные в Arduino могут использовать 8, 16, или 32 бита. Поскольку объем памяти в Arduino ограничен, при кодировании переменных необходимо использовать наименьшее количество бит.

Чтобы помочь вам сделать это, список, приведенный ниже, перечисляет некоторые общие типы данных Arduino, их битовое значение, а также их функциональное назначение.

Внимание! Целочисленные значения (`int`) будут меняться, если их будут подталкивать к минимальным или максимальным значениям. Например, если X = 32 767 и другая инструкция добавляет к X единицу, X станет равен –32,768.

- **Boolean (8 бит)** — простые логические значения `true/false` (истина/ложь)
- **Byte (8 бит)** — беззнаковое число в диапазоне от 0 до 255

- **Char (8 бит)** — относительное число в диапазоне от –128 до 127
- **Word (16 бит)** — беззнаковое число в диапазоне от 0 до 65 535
- **Int (16 бит)** — относительное число в диапазоне от –32 768 до 32 767
- **Unsigned long (32 бита)** — беззнаковое число в диапазоне от 0 до 4 294 967 295
- **Long (32 бита)** — относительное число в диапазоне от –2 147 483 648 до 2 147 483 647
- **Float (32 бита)** — относительное число в диапазоне от –3 402 8235E38 до 3 402 8235E38

На заметку. `long` (32 бита) представляет собой расширенный тип данных для целых чисел в диапазоне от 2 147 483 647 до –2 147 483 648.

Обратите внимание, что относительные переменные используют как положительные, так и отрицательные значения, в то время как беззнаковые переменные допускают использование только положительных значений.

Инструкции

Простые инструкции

Инструкции бывают двух типов — простые и условные. К простым относятся единичные инструкции. Простая инструкция занимает одну строку и всегда заканчивается точкой с запятой.

Например:

```
digitalWrite(ledPin, HIGH); // Присваивает ledPin +5 В
```

Обратите внимание, что за кодом инструкции следует комментарий, который описывает, что обязана выполнять инструкция.

После инструкции должна быть указана точка с запятой. Если точка с запятой отсутствует, компилятор вернет сообщение об ошибке.

Условные инструкции

К условным относятся инструкции, содержащие условие, за которым следует ряд инструкций, которые выполняются тогда, когда условие соблюдено. Они позволяют логически управлять потоком скетча.

Например:

```
while (switchValue = = LOW) {   // Эта строка является
                                   условием
   digitalWrite(ledPin, HIGH);  // Эта строка включает
                                   светодиодный
                                   индикатор
   delay(1000);                 // Никаких действий
                                   в течение 1000 мс
   digitalWrite(ledPin, LOW);   // Эта строка отключает
                                   светодиодный
                                   индикатор
}
```

Первая строка, `while (switchValue = = LOW)`, представляет собой условие, и за ней на трех последующих строках указаны три инструкции. Понять принцип действия условных инструкций можно, разобравшись в их логике.

Поэтому наш пример можно прочитать следующим образом: «когда значение переключателя минимально, включите светодиодный индикатор, не выполняйте никаких действий на протяжении 1000 мс, а затем выключите светодиодный индикатор».

К другим условным инструкциям Arduino относятся:

- `if`
- `if...else`
- `for`
- `do...`
- `while`
- `break`
- `continue`
- `return`

Условные инструкции очень важны при программировании, поэтому мы детально рассмотрим, как они работают.

if проверяет, было ли выполнено условие, т. е. достигло ли значение определенного уровня. Если значение достигло данного уровня, скетч запустит код соответствующего действия. Если же нет, скетч пропустит инструкцию и перейдет к следующей строке кода.

Синтаксис инструкции `if` следующий:

```
if (значениеX > значениеY) {
  выполнитьДанноеДействие;
}
```

В данном примере код читается так: если *значениеX* больше, чем *значениеY*, выполняется действие, указанное в скобках. Обратите внимание, что символ > обозначает оператор «больше чем».

if/else — эта инструкция более функциональна, чем базовая инструкция `if`, поскольку она позволяет сгруппировывать ряд функций сравнения.

Инструкция `if` — самая распространенная из всех инструкций программирования.

Например, когда нужно, чтобы выполнялось одно действие, если *значениеX* равно `HIGH`, и выполнялось другое действие в противном случае, код должен выглядеть следующим образом:

```
if (значениеX = = HIGH) {
  выполнитьДействиеA;}
else {
  выполнитьДействиеБ
}
```

Обратите внимание, что символы == обозначают оператор «равно».

for полезно использовать в ситуациях, в которых блок инструкций необходимо повторить определенное количество раз. Эта инструкция позволяет выполнять блок снова и снова, т. е. зацикливать его, пока не будет выполнено определенное условие.

Оператор `for` часто используется в сочетании с массивами.

Иногда эта инструкция используется в сочетании со счетчиком приращения (инкрементирования), чтобы выполнить приращение и завершить цикл.

Заголовок цикла `for` содержит три элемента — инициализацию, условие и инкрементирование или декрементирование.

Например:

```
for (int x = 0; x < 50; x++) {
   выполнитьДанноеДействие;
}
```

Сначала происходит инициализация, и происходит она только один раз. Выполняется она оператором =. Далее условие проверяется с помощью оператора <. Если условие истинно, инкрементирование выполняется унарным оператором ++. Наконец, выполняется код *выполнитьДанноеДействие*.

Элементы заголовка должны быть разделены точками с запятой.

Вся последовательность повторяется до тех пор, пока условие не станет ложным, после чего цикл завершается.

`switch/case` — инструкция `switch/case` — более функциональная версия инструкции `if`.

Если необходимо учесть три варианта развития событий, вам понадобится три отдельных инструкции `if` — по одной для каждого варианта. Однако вам понадобится всего одна инструкция `switch/case`, поскольку она может выполнять переключение между различными вариантами.

Например:

```
switch (значениеДатчика) {
   case 1:
      // выполняет что-то, когда значениеДатчика равно 1
      break;
   case 2:
      // выполняет что-то, когда значениеДатчика равно 2
      break;
   default:
      // если ничего больше не совпадает, выполняйте
      действие по умолчанию
```

```
    // действие по умолчанию необязательно
}
```

Инструкции `switch/case` помогают сделать ваш код корректным и аккуратным, если использовать их вместо длинных строк кода инструкций `if`.

Как видно из кода, инструкция `switch` сравнивает значение переменной со значениями, указанными в инструкциях `case`. Когда инструкция `case` определит, какое значение совпадает со значением переменной, будет выполнен код инструкции `case`.

`while` выполняет инструкцию, пока определенное условие истинно. Код инструкции будет зациклен, или продолжать выполняться, пока условие будет истинно.

Как только что-либо повлияет на условие и оно станет ложным (к примеру, изменятся входные данные с датчика), выполнение цикла завершится, и инструкция больше не будет выполняться.

Например:

```
while (данноеЗначение > 100) {
   выполнитьДанноеДействие;
}
```

Пока *данноеЗначение* будет больше 100, код *выполнитьДанноеДействие* будет выполняться непрерывно. Когда же *данноеЗначение* упадет ниже 100, произойдет остановка.

`do-while` работает так же, как и инструкция `while`, с той лишь разницей, что условие проверяется не в начале, а в конце.

Например:

```
do {
   сделатьЧтоНибудьПолезное;
} while (x > 50);
```

В данном случае код в фигурных скобках запускается до оценки условия `while (x > 50);`. В результате, даже если условие не соблюдается, код будет выполнен однократно.

break — цель этой инструкции заключается в немедленном прекращении выполнения инструкции, после чего продолжает выполняться код, указанный после инструкции.

Например:

```
while (значениеА > 100) {
  if (значениеБ = = HIGH)
{
  break;
}
```

Инструкцию `break` также можно использовать для разделения фрагментов инструкций `switch/case`.

continue используется внутри циклов. Она пропускает любые инструкции, которые указаны после нее в цикле, и затем переходит к следующей итерации цикла.

Данную инструкцию можно использовать только с итеративными инструкциями, такими как циклы `while`, `do` и `for`.

Например:

```
for (x = 0; x < 255; x ++) {
  if (x > 40 && x < 120) {
    continue;
  }
  digitalWrite(PWMpin, x);
  delay(50);
}
```

Инструкция `continue` может быть использована для пропуска четных или нечетных чисел в цикле `for`.

return используется для завершения работы функции и возврата к вызывающей функции. Инструкция `return` делает это независимо от того, где находится. Также ее можно использовать для извлечения значения из функции.

Например:

```
int checkSensor() {
  if (analogRead(0) > 350) {
```

```
    return 1;
  else {
    return 0;
}
```

Полезно применять инструкцию `return` при тестировании какого-либо фрагмента кода без необходимости «комментировать» большие его участки.

По исполнении инструкции `return` функция немедленно завершает свою работу.

Арифметика и логика

Арифметические операторы

Ваша плата Arduino способна выполнять базовые арифметические действия — сложение, вычитание, умножение и деление. Эти операции возвращают сумму, разность, произведение или частное двух операндов.

Во время вычисления операции выполняются с разными типами данных операндов. Это может повлиять на выполнение операции и, таким образом, на результат.

В математике операндом называется объект математического действия; величина, с которой выполняется действие.

Например, когда операндами являются целые числа (без дробей), результаты отбрасываются, но не округляются, до ближайшего целого числа. Таким образом, результатом 13/3 будет 4, а не 4,333.

Также вычисления могут привести к переполнению, если результат будет больше, чем тип данных сможет сохранить. Например, при добавлении 1 к `int` со значением 32,767 приведет к результату –32,768.

Обратите внимание, что, если используются разные типы операндов, вычисление будет использовать самые большие из типов. Кроме того, если ваши

вычисления включают дробные числа и, следовательно, не могут использовать целые числа, вы можете использовать плавающие переменные. Недостатком, однако, является то, что вычисления с переменными могут быть очень медленными. Также они могут быть очень большого размера, и поэтому задействовать большой объем памяти.

В таблице, приведенной ниже, перечислены арифметические операторы:

Оператор	Действие
+	сложение
–	вычитание
*	умножение
/	деление
%	деление по модулю

На заметку

Число с плавающей запятой — это термин, который используется для описания переменной с дробным значением. Числа, которые созданы путем объявления переменной с плавающей запятой, будут иметь цифры по обе стороны от десятичной запятой в отличие от целочисленной переменной, которая в качестве значения может иметь только целые числа.

Название каждого из этих операторов характеризует выполняемую им операцию, за исключение оператора деления по модулю, %. Он используется для вычисления остатка деления одного целого числа на другое.

Существует несколько способов использования этого оператора. Один из примеров — защита переменной от превышения определенного диапазона значений. Еще один пример — это определение того, что одно значение кратно другому.

Унарные операторы

Унарные операторы используются для объединения арифметических операций с операцией присваивания. Таким образом, с помощью одной инструкции можно фактически сделать то же самое, что делают две.

Соответственно, унарные операторы позволяют писать более компактный код скетча. Как правило, такие операторы используются для несложных операций, таких как увеличение значения переменной.

Унарные операции используются для того, чтобы код был компактным и лаконичным.

Например, чтобы увеличить значение переменной на 1, используя обычные инструкции, вы напишете следующий код:

```
некоеЗначение = некоеЗначение + 1;
```

Но с помощью унарного оператора вы могли бы написать его более компактно:

```
некоеЗначение++;
```

Обратите внимание, что унарные операторы ++ и -- увеличивают или понижают значение только на 1. Если вы заходите использовать большее значение, необходимо применять операторы +=, -=, *= или /=.

Например:

```
некоеЗначение += 5;
```

Полный список унарных операторов приведен в таблице ниже.

Оператор	Действие
++	увеличивает на 1
--	уменьшает на 1
+=	унарное сложение
-=	унарное вычитание
*=	унарное умножение
/=	унарное деление

Операторы сравнения

Операторы сравнения, как понятно из названия, позволяют сравнивать два значения. Часто это выполняется для проверки определенных условий, чтобы увидеть, является значение истинным или ложным.

При использовании операторов сравнения убедитесь, что все операнды имеют одинаковый тип данных, т. е. целые числа необходимо сравнивать с целыми числами, строки со строками и т. д.

В примере, приведенном ниже, условная инструкция используется в сочетании с оператором сравнения > (больше чем):

```
if (некоеЗначение > 20) {
  digitalWrite(LEDpin1, HIGH);
}
```

Не путайте оператор присваивания = (один знак равенства) с оператором сравнения == (двойной знак равенства).

Когда выполняется код, приведенный выше, скетч проверяет, превышает ли *некоеЗначение* число 20. Если это так, скетч будет выполнять инструкцию, которая находится в фигурных скобках, и переведет светодиодный индикатор, подключенный к контакту 1, в режим `HIGH` (т. е. включит его). Если же значение менее 20, инструкция игнорируется.

Список операторов сравнения приведен в следующей таблице.

Оператор	Действие
==	равно
!=	не равно
<	меньше
>	больше
<=	меньше или равно
>=	больше или равно

Логические операторы

Логические, или булевы операторы представляют собой логические инструкции, используемые для проверки различных условий в инструкциях `if`. Каждый оператор будет возвращать либо значение `TRUE` (истина) либо `FALSE` (ложь).

Существует три таких оператора.

Оператор	Действие
Логическое И	Имеет значение `true` (истина), только в том случае, если оба операнда истинны.
Логическое ИЛИ	Имеет значение `true` (истина), если хоть один операнд истинный.
Логическое отрицание	Имеет значение `true` (истина), если операнд ложный.

Обратите внимание, что слова «Логическое И», «Логическое ИЛИ» и «Логическое отрицание» не пишутся в коде скетча. Вместо этого они представлены следующими символами:

- **логическое И** — представлен символами `&&`;
- **логическое ИЛИ** — символами `||`;
- **логическое отрицание** — символами `!`.

Например:

```
if (x==10 && y==20)
```

Код, представленный выше, будет выполняться только в том случае, если *x* равен 10 и *y* равен 20. Если оператора сменить на `||`, он будет выполняться при условии, что *x* равен 10 или *y* равен 20. Пример, приведенный ниже, содержит оператор `!` и будет выполняться, только если *x* имеет значение `false` (ложь).

```
if (!x!=0)
```

Константы

Константы могут помочь упростить чтение кода.

Константа представляет собой значение данных, так же как и переменная. Разница между ними заключается в том, что значение константы используется «только для чтения» и его нельзя изменить. Константы объявляются с помощью слова `const`.

Например:

```
const float pi = 3.14;
```

Обратите внимание, что константы `true` и `false` пишутся в нижнем регистре.

Вы можете написать свои собственные константы или же, как вариант, использовать одну из заранее определенных констант, предоставляемых языком программирования Arduino.

Их можно классифицировать по следующим трем группам.

Логические константы — `true` и `false` используются для представления истины и лжи. Ложь определяется как 0 (ноль), в то время как истина обычно определяется как 1.

Например:

```
if (x = = true); {
   выполнитьДанноеДействие;
}
```

Обратите внимание, что `true` также означает любое целое число, которое не равно 0. Поэтому, к примеру, 1, 56, –300 и т.д. являются истинными значениями.

`HIGH` и `LOW` — при чтении или записи с/на цифровой контакт существует только два возможных значения для контакта — `HIGH` и `LOW`.

Они устанавливаются константами `HIGH` или `LOW`. `HIGH` определяется как `1`, т. е. вкл. (5 В), тогда как `LOW` определяется как 0, т. е. выкл. (0 В).

Например:

```
digitalWrite(13, HIGH);
```

`INPUT`, `OUTPUT` и `INPUT_PULLUP` — константы `INPUT` и `OUTPUT` используются с функцией `pinMode()` для определения цифрового контакта либо как вход (`INPUT`), либо как выход (`OUTPUT`).

Например:

```
pinMode(13, OUTPUT);
```

Константа `INPUT_PULLUP` представляет собой более сложный вариант. Нагрузочные резисторы используются для обеспечения нахождения входов в Arduino на ожидаемом уровне логики. Они делают это, обеспечивая нахождение контактов либо в определенном состоянии HIGH или LOW, независимо от внешних факторов, например отсоединения устройства, либо путем введения значений полного сопротивления.

Нагрузочные резисторы обеспечивают то, что контакт будет находиться на определенном уровне логики, независимо от того, подключено ли к нему активное устройство.

Такие факторы могут привести к тому, что входное значение на контакте будет «плавать» между HIGH и LOW. Светодиодные индикаторы, подключенные к контакту в этом состоянии, часто слабо мерцают, включаясь и выключаясь.

Чип микропроцессора на плате Arduino Uno имеет встроенные нагрузочные резисторы. Константа `INPUT_PULLUP` позволяет вам активировать эту опцию. Однако при желании вы можете использовать внешние нагрузочные резисторы.

Вопрос, часто задаваемый пользователями, которые начали свое знакомство с Arduino, состоит в следующем: для чего использовать константу, если вы можете использовать более гибкую переменную. Разница заключается в том, что константа содержит неизменное значение.

Нагрузочные резисторы также можно использовать в качестве интерфейса между различными типами логических устройств.

Это позволяет скомпилировать значение непосредственно в скетч, что может не только устранить ошибки, но и оптимизировать скетч, чтобы он более эффективно работал.

Кроме того, в качестве дополнительного преимущества скетч будет использовать меньше памяти.

Массивы

Что такое массив?

Массив представляет собой набор значений, которые сгруппированы вместе, и их можно обозначать как одно целое. Массивы используются, когда необходимо управлять большим количеством связанных данных, поскольку массивы помогают сохранить данные в порядке. Каждое звено в массиве известно как элемент, а количество элементов определяет размер массива.

Массивы могут содержать что угодно, хотя, как правило, хранят переменные. Можно использовать любой тип данных, например `int`, `char`, `word`, но в одном массиве все данные должны быть одного типа. Когда вы объявляете массив, вы указываете, что он будет содержать. Например:

```
int имяМассива[] = {1, 2, 3, 4, 5};
```

Представьте, что массив — это наполненная коробка, содержащая большое количество каталожных карточек, каждая из которых имеет связанные данные, написанные на них.

Указанный массив будет содержать целые числа, он носит название *имяМассива* и содержит пять элементов, каждый из которых был инициализирован (с помощью символа =. Первый элемент равен `1`, второй — `2` и т. д. Когда скетч скомпилирован, количество элементов учитывается компилятором, и, соответственно, автоматически определяется размер массива.

Однако бывают случаи, когда значения неизвестны до тех пор, пока скетч не будет запущен (например, входные данные с датчика). Поэтому, если вы не знаете этих значений, вы не можете их инициализировать. В данном случае, вы объявляете массив, как показано ниже:

```
int имяМассива[5];
```

С помощью этого кода были объявлены пять элементов (по числу в квадратных скобках) со значениями, начиная с `0`. Эти значения будут обновлены автоматически при запуске скетча.

Изменение значений

Чтобы изменить значение элемента вручную, необходимо в квадратных скобках ввести число, связанное с элементом. При этом следует помнить, что элементы нумеруются слева направо, начиная с 0.

Например, чтобы изменить второй элемент (со значением `15.9`) в массиве, представленном ниже, на `17.5`:

```
int имяМассива[] = {10.5, 15.9, 20.3, 34.7, 42.5};
```

вы напишете:

```
int имяМассива[1] = 17.5;
```

Элементы массива нумеруются начиная с нуля. Поэтому первым будет 0, вторым — 1, третьим — 2 и т.д.

Побитовые операторы

В двоичной системе, используемой компьютерами, цифровые значения записываются с помощью цифр 1 и 0. Например, в двоичном виде число 667 записывается как 1010011011. Каждый символ 1 и 0 известен как бит.

Побитовые операторы позволяют выполнять вычисления с байтовыми и целочисленными переменными, используя эти двоичные представления. Основные преимущества этого заключаются в том, что увеличивается скорость работы скетча, а также в том, что требуется меньше памяти Arduino.

Наш пример вычисления с оператором «Побитовое И», представленный слева, выполняется на двух байтовых значениях a и b. Как мы видели на с. 156, типы байтных данных являются 8-битными, что означает, что в числе есть 8 бит. Следовательно, вычисление включает восемь одновременных операций, по одной для каждого бита.

Чтобы продемонстрировать, как это работает, мы взглянем на оператор «Побитовое И». Он указывается в коде в виде символа & и используется для сравнения двух двоичных чисел по битам.

Синтаксис следующий:

```
byte a & b
```

Пример вычисления:

```
byte a = 0 1 0 0 1 1 0 1;
         | | | | | | | |
byte b = 0 1 1 1 0 1 0 0;
         | | | | | | | |
         0 1 0 0 0 1 0 0
```

Если биты в отдельно взятом столбце равны 1, выход будет равен 1. Если же нет, выход будет равен 0.

Побитовыми операторами являются:

- **Побитовое И** (&) — выдает 1, если оба значения в столбце равны 1
- **Побитовое ИЛИ** (|) — выдает 1, если хотя бы одно значение в столбце равно 1
- **Побитовое исключающее ИЛИ** (^) — выдает 1, если биты в столбце отличаются по значению
- **Побитовое НЕ** (~) — изменяет 0 на 1, а 1 на 0
- **Побитовый сдвиг влево** (<<) — сдвигает биты влево на указанное количество
- **Побитовый сдвиг вправо** (>>) — сдвигает биты вправо на указанное количество

Практическое применение оператора «Побитовое И» заключается в доступе к определенным битам в байте данных. Это называется «маскированием».

Интерфейсы ввода/вывода

Существует много ограничений относительно того, что может выполнять плата Arduino сама по себе. Однако подключите ее к внешней цепи, и возможностей станет немного больше. Для этого требуется средство связи, известное как интерфейс. Arduino обеспечивает два типа подключения:

- цифровые контакты (используемые для цифрового сигнала);
- аналоговые контакты (используемые для аналогового сигнала).

Эти контакты можно настроить либо как входы, либо как выходы. Также некоторые из аналоговых контактов можно преобразовать в цифровые.

На заметку

Цифровые контакты по умолчанию настроены на вход. Это означает, что их не нужно определять как входные.

Чтобы позволить вам сделать это, Arduino обеспечивает ряд функций, с помощью которых вы можете настроить контакты входа-выхода для достижения своих целей.

Сначала давайте рассмотрим настройку цифровых контактов.

Цифровые входы и выходы

На с. 26–28 мы видели, что Arduino Uno оборудована 14 цифровыми контактами (интерфейсами). Их можно настроить с помощью следующих функций.

`pinMode()` используется для определения цифрового контакта либо в качестве входа, либо в качестве выхода. Синтаксическая структура очень проста, как можно видеть в примере, приведенном ниже:

```
pinMode(контакт, режим);
```

Данная функция имеет два параметра — *`контакт`* и *`режим`*. *`Контакт`* — это номер настраиваемого контакта в диапазоне от 0 до 13. *`Режим`* определяет функцию контакта как `INPUT`, `INPUT_PULLUP` или `OUTPUT`.

Константа `INPUT` устанавливает контакт на состояние высокого полного сопротивления, вследствие чего контакт предъявляет мало требований

к нагрузочной цепи. Это преимущество в слаботочных цепях, например содержащих датчики.

Константа OUTPUT, с другой стороны, устанавливает контакт на состояние низкого полного сопротивления. Это позволяет контакту обеспечивать значительное количество тока другим цепям.

Константа INPUT_PULLUP устанавливает контакт в качестве входа, а также активирует внутренний нагрузочный резистор контакта.

digitalWrite() — когда цифровой контакт был переведен в режим OUTPUT, его необходимо установить либо во включенное состояние, либо выключенное. Это выполняется с помощью функции digitalWrite(), синтаксис которой следующий:

```
digitalWrite(контакт, режим);
```

Как и в случае с pinMode(), функция имеет два параметра: *контакт* относится к используемому контакту, а *режим* позволяет указать одну из двух констант — HIGH или LOW.

Если вы хотите перевести контакт, скажем, под номером 10, во включенное состояние, код будет выглядеть следующим образом:

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

Прежде чем вы сможете использовать функцию digitalWrite() для включения/выключения контакта, вам необходимо использовать функцию pinMode() для указания режима контакта — INPUT или OUTPUT.

Константа HIGH дает указание Arduino отправить ток напряжением 5 В и силой около 40 мА на указанный контакт. Константа LOW эмулирует на контакте напряжение 0 В, образуя, по сути, заземляющее соединение.

digitalRead() — когда цифровой контакт настроен как вход, состояние контакта можно определить с помощью функции digitalRead(), как показано ниже:

```
digitalRead(контакт);
```

Выводом функции будет либо HIGH, либо LOW, и его можно использовать двумя способами.

Первый способ — это просто присвоить значение контакта переменной. Второй — использовать функцию в качестве переменной. Рассмотрим следующий пример кода:

```
someVariable = digitalRead(pin2);
if (someVariable = = LOW) digitalWrite(ledPin, LOW);
```

В первой строке функция `digitalRead()` считывает значение на входном контакте (`pin2`), а затем присваивает его переменной `someVariable`. Во второй строке мы применяем к переменной проверку условия (`if`), и, если результат равен `LOW`, она настраивает выходной контакт (`ledPin`) как `LOW`.

Аналоговые входы и выходы

Разумеется, не все сигналы являются цифровыми. Вывод многих устройств, например электродвигателей и датчиков, осуществляется в аналоговой форме. Поскольку Arduino представляет собой цифровое устройство, она должна преобразовывать данные аналоговые сигналы в цифровые, прежде чем сможет их использовать.

И наоборот, иногда необходимо преобразовывать цифровые сигналы от Arduino в аналоговые, которые могут использоваться внешними аналоговыми устройствами.

Чтобы преобразовать аналоговый сигнал в цифровой, в Arduino есть аналогово-цифровой преобразователь (АЦП). Для преобразования цифрового сигнала в аналоговый Arduino использует метод широтно-импульсной модуляции (ШИМ). Как мы уже увидели, эта имитация достигается посредством управления рабочим циклом аналогового сигнала.

Arduino Leonardo — это единственная модель из числа плат Arduino, которая поставляется с цифроаналоговым преобразователем. Все остальные модели используют метод ШИМ.

Двумя функциями, которые используются для настройки аналоговых контактов, служат следующие:

`analogRead()` выдает целочисленное значение, которое представляет собой аналоговый сигнал на контакте. Это значение варьируется в диапазоне от 0 до 1023.

Например:

```
value = analogRead(A2)
```

На Arduino Uno метод ШИМ поддерживается только на контактах **3**, **5**, **6**, **9**, **10** и **11**.

Аналоговые контакты Arduino обозначены метками от **A0** до **A0**. Таким образом, «контакт» будет в пределах этого диапазона — `A2`, как показано выше. Переменной `value` присваивается такое же значение, что и на контакте **A2**.

`analogWrite()` — когда вам необходим аналоговый выходной сигнал на определенном контакте, следует использовать функцию `analogWrite()`. Как мы уже упоминали ранее, она использует метод ШИМ для имитации нужного сигнала.

Синтаксис следующий:

```
analogWrite(контакт, некотороеЗначение)
```

Функция записывает *некотороеЗначение* на указанный контакт.

Значение, равное 0, генерирует ток 0 В на указанном контакте, а значение 255 генерирует 5 В.

Время

При запуске приложений, которые работают в режиме реального времени, вашим скетчам точно должно быть известно, что представляет собой время и связанные с ним понятия.

Плата Arduino справляется с этим с помощью четырех связанных функций, как показано ниже:

`delay()` — функция `delay()` приостанавливает скетч, или же ждет, чтобы что-то произошло, в течение количества миллисекунд, указанных в коде.

Например:

```
delay(1000)
```

Число, введенное в круглых скобках, определяет задержку — 1000 мс, или 1 секунду.

`delayMicroseconds()` представляет собой то же самое, что функция `delay()`, но период задержки указывается в микросекундах.

Обратите внимание, что самое большое значение, которое будет вызывать точную задержку, будет равно 16383. Для задержек длительностью свыше нескольких тысяч микросекунд рекомендуется использовать функцию `delay()`, поскольку она будет более точной.

Обе эти функции полезны в ситуациях, когда необходимо, чтобы скетч подождал в течение определенного периода, или же если вам нужно немного замедлить процесс.

`millis()` сообщает о периоде времени в миллисекундах, в течение которого работает текущий скетч.

`micros()` — данная функция аналогична `millis()`, но возвращает необходимое значение не в миллисекундах, а в микросекундах.

На заметку

В функциях `millis()` и `micros()` после определенного периода число выйдет за пределы, то есть будет сброшено до нуля.

Помимо того, что две эти функции сообщают, в течение какого времени работает определенный скетч, они позволяют оперировать периодами времени в скетче.

Другие полезные функции

Ваша плата Arduino предлагает большое количество функций, которые отличаются от упомянутых ранее в этой главе. К ним относятся следующие.

Улучшенная арифметика

Для большинства скетчей базовые арифметические операторы (+, —, x, /) — это обычно все, что вам нужно. Более сложным скетчам, возможно, может

понадобиться нечто более сложное. Для достижения этой цели Arduino позволяет использовать следующие арифметические и тригонометрические функции:

- **`min(x, y)`** — возвращает *x* или *y*, в зависимости от того, чье значение меньше;
- **`max(x, y)`** — возвращает *x* или *y*, в зависимости от того, чье значение больше;
- **`abs(x)`** — возвращает абсолютное значение *x*;
- **`constrain(x, a, b)`** — ограничивает число в пределах диапазона. Значение *x* возвращается в том случае, если оно находится между *a* и *b*, *a* возвращается, если *x* меньше *a*, и *b* возвращается, если *x* больше *b*;
- **`map(x)`** — пересопоставляет значение *x* из одного диапазона в другой;
- **`pow(x, y)`** — выдает значение *x* в степени *y*;
- **`sqrt(x)`** — возвращает квадратный корень значения *x*;
- **`sin(x)`** — возвращает синус значения *x*;
- **`cos(x)`** — возвращает косинус значения *x*;
- **`tan(x)`** — возвращает тангенс значения *x*.

Не используйте другие функции с `abs()`. Из-за метода ее реализации, это может привести к ошибкам.

На заметку. Функции `map(x)` и `constrain()` обычно используются с датчиками, поскольку позволяют сохранять выходные данные датчика в определенном диапазоне.

Случайные числа

Большое количество приложений для обработки данных требуют использования случайных чисел. К стандартным примерам относятся: генерирование ключа и пароля, цифровые игры, перемешивание порядка проигрывания аудиофайлов и т. д.

Большинство языков программирования обеспечивают функцию генерирования случайных чисел (хотя сгенерированные числа по сути псевдослучайны), и язык Arduino не исключение.

Arduino позволяет применять для достижения этой цели две функции.

`random(мин, макс)` — функция `random()` возвращает случайные числа в пределах диапазона, указанного значениями *мин* и *макс*. Обратите внимание, что значение *мин* необязательно. Если вы его не укажете, по умолчанию используется значение 0.

Например:

```
значениеА = random(50, 100);
```

Этот код присваивает переменной *значениеА* случайное число в пределах диапазона от 50 до 100.

`randomSeed(случайноеЗначение)` — функция `randomSeed()` инициализирует генератор псевдослучайных чисел, принудительно запуская генерацию случайной последовательности.

На заметку

Компьютерные генераторы случайных чисел, как правило, создают повторяющиеся случайные последовательности, а не по-настоящему случайные числа. Функция `randomSeed()` в определенной степени устраняет это ограничение, позволяя вставлять переменную, константу или другую функцию в случайную функцию.

случайноеЗначение — это целое число, которое определяет точную последовательность сгенерированных псевдослучайных чисел. Даже небольшое изменение в случайном значении приведет к абсолютно другой случайной последовательности.

Манипуляции на уровне битов

Побитовая операция — это процесс, который управляет битами данных по определенному алгоритму.

Этот процесс часто необходим в задачах, связанных с программированием, как, например, обнаружение ошибок, сжатие данных, шифрование и программная оптимизация.

Язык программирования Arduino обеспечивает широкий спектр функций манипулирования битами.

К ним относятся:

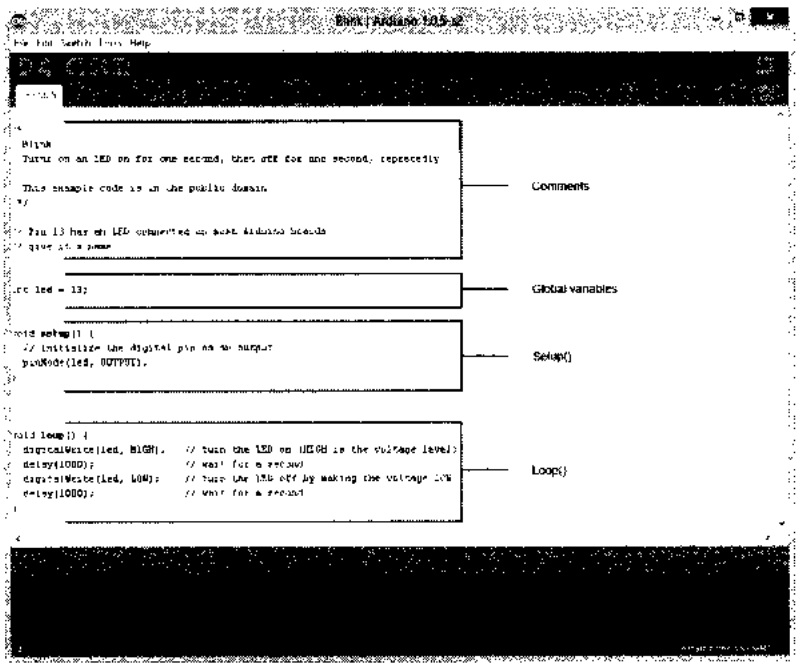
- **`lowByte()`** — возвращает младший байт значения `word`;
- **`highByte()`** — возвращает старший байт значения `word`;
- **`bitRead()`** — считывает бит числа;
- **`bitWrite()`** — записывает бит числового значения;
- **`bitSet()`** — устанавливает (записывает 1 в) бит числового значения;
- **`bitClear()`** — удаляет (записывает 0 в) бит числового значения;
- **`bit()`** — возвращает значение указанного бита.

Структура скетча

В этой главе мы увидели, что основные элементы, которые используются при написании скетчей Arduino — это переменные, циклы, операторы, ввод/вывод и функции. Они отличаются в зависимости от скетча, но неизменной остается основная структура — все скетчи состоят из четырех главных разделов. К ним относятся:

- Комментарии (необязательно);
- Глобальные переменные (если таковые имеются);
- Функция `setup()`;
- Функция `loop()`.

Это продемонстрировано в коде скетча, который показан ниже:



Из всего, приведенного выше, комментарии необязательны. Переменные можно помещать куда угодно, но, если необходимо, чтобы они были доступны для всего скетча, а не в определенной части кода, они должны находиться в верхней части, под общими комментариями.

Функции `setup()` и `loop()` являются необходимыми компонентами скетча. Если хотя бы одна из этих функций будет отсутствовать, скетч не будет работать.

9

Скетчи

Мы начнем эту главу, написав код простого скетча. Затем проанализируем ряд встроенных скетчей Arduino, увидев, какое аппаратное обеспечение необходимо, как они работают и что делают.

- **Создание скетча Arduino**
- **Проверка скетча**
- **Загрузка скетча**
- **Скетч Fade**
- **Скетч DigitalReadSketch**
- **Скетч AnalogReadSerial**
- **Скетч IfStatementConditional**
- **Скетч ForLoopIteration**

Создание скетча Arduino

Вы узнали о различных элементах, используемых в скетчах Arduino — комментариях, функциях, инструкциях, циклах и т. д. Вы уже знаете о синтаксической структуре и насколько важно то, чтобы она была написана правильно. Теперь пришло время объединить все эти знания и написать свой первый скетч.

Основные положения

Первым шагом будет подключение Arduino к компьютеру и запуск среды разработки Arduino. Окно редактора автоматически откроет новый скетч. Название скетча появится на вкладке в верхнем левом углу окна редактора в следующем формате:

```
sketch_мммддх
```

ммм — это первые три буквы месяца, *дд* — это дата, а *х* — это буква, которая отличает скетчи, созданные в один и тот же день.

Когда вы начнете вводить код в окно редактора, обратите внимание, что определенные фрагменты кода будут выделены разным цветом. Например, ключевые слова функций выделены оранжевым цветом, а константы — синим. Благодаря этому проще находить и устранять синтаксические ошибки, которые могут препятствовать компиляции вашего скетча.

Скетч Blinking LED, процесс создания которого описан на с. 183–187, является вариацией скетча Blink, предустановленного в среде разработки Arduino.

Комментарии

Прежде чем вы начнете вводить код, возможно, вы захотите внести несколько комментариев относительно скетча. Например, вы можете присвоить скетчу название, указать, когда и кем он был написан и для каких действий предназначен.

В процессе работы над скетчем вы можете добавлять комментарии в любом месте кода, чтобы дать возможность другим людям понять, что выполняет определенный фрагмент или строка кода, а также различные напоминания и т. д. Они могут оказаться чрезвычайно полезными как для вас, так и для окружающих.

Как мы видим, однострочным комментариям предшествуют две косые черты:

```
// Скетч Blinking LED создан Джоном Смитом 12/11/2014
```

Многострочные комментарии начинаются с символов `/*` и заканчиваются символами `*/`:

```
/*
Скетч Blinking LED создан
Джоном Смитом 12/11/2014
*/
```

1. Напишите свой вариант комментариев, показанных на этой странице, нажмите кнопку **Save** (Сохранить), присвойте скетчу название, а затем щелкните мышью по кнопке **OK**.

Если сейчас вы выберете команду меню **File** → **Sketchbook** (Файл → Папка со скетчами), то увидите папку, содержащую ваш скетч.

Периодически сохраняйте свою работу, особенно когда пишете длинные или сложные скетчи.

Функция `setup()`

Далее необходимо выполнить настройку с помощью функции `setup()`. Это будет первый фрагмент исполняемого кода в скетче, и по этой причине он содержит все инструкции, необходимые для настройки платы. К ним относятся, к примеру, инициализации переменных, настройки режимов контактов и вызовы библиотек.

Эти инструкции вводятся в фигурных скобках и выполняются только один раз. Синтаксис функции `setup()` следующий:

```
void setup() {
}
```

Цель нашего скетча заключается в том, чтобы светодиодный индикатор, установленный на Arduino, мигал, включаясь и выключаясь. Когда светодиодный индикатор подключен по умолчанию к контакту **13**, функция `setup()` сообщит Arduino о необходимости установить контакт **13** как выход. Затем она отправит 5 В на светодиодный индикатор, таким образом включив его.

❶ Введите следующий код в редактор после комментариев:

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

На заметку

Размещение инструкций в функции `setup()` необязательно. Некоторым простым скетчам такие инструкции не требуются. Но наличие самой функции обязательно, даже если в фигурных скобках ничего не указано.

Функция `pinMode()` устанавливает контакт **13** как выход (если бы вместо слова `OUTPUT` мы написали `INPUT`, это был бы вход). Теперь скетч будет выглядеть следующим образом:

```
/*
Скетч Blinking LED создан
Джоном Смитом 12/11/2014
*/

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

Функция `loop()`

Следующей нужно указать функцию `loop()`. Как правило, здесь размещается основная часть кода.

Тогда как функция `setup()` инициализирует входные и выходные контакты Arduino и готовит их к использованию, функция `loop()` приводит их в действие.

Код функции `loop()` выполняется циклически, пока не отключается электропитание или не будет нажата кнопка перезагрузки на плате. Синтаксис функции `loop()` следующий:

```
void loop() {
{
```

Не заметку

Фактически код, находящийся в функции `setup()`, готовит плату Arduino к работе, а код в функции `loop()` приводит ее в действие.

❶ Как и с функцией `setup()`, код цикла указывается в фигурных скобках. Итак, продолжая работу с нашим скетчем Blinking LED, введите следующий код после кода функции `setup()`:

```
void loop() {
  digitalWrite(13, HIGH);
  delay(2000);
  digitalWrite(13, LOW);
  delay(2000);
}
```

В данном коде функция `digitalWrite()` управляет напряжением, которое передается с цифрового контакта (в нашем случае контакт **13**).

Она выполняет это либо путем установки на контакте значения `HIGH`, и в этом случае на контакт передается 5 В, зажигая таким образом светодиодный индикатор; либо `LOW` — в этом случае на контакте не будет напряжения, поэтому светодиодный индикатор будет выключен.

Цикл также содержит две функции `delay()`. Они сообщают скетчу о необходимости приостановки работы в течение указанного периода — 2000 мс, или 2 с.

Последовательность выполнения задач в цикле будет следующей:

- первая функция `digitalWrite()` передает 5 В на контакт **13**, включая таким образом светодиодный индикатор;
- первая функция `delay()` останавливает выполнение кода на 2 с, поэтому светодиодный индикатор продолжает гореть в течение 2 с;
- первая функция `delay()` заканчивает работу, позволяя начать работу второй функции `digitalWrite()`; она убирает электропитание с контакта **13**, таким образом выключая светодиодный индикатор;
- вторая функция `delay()` снова останавливает выполнение кода на 2 с, таким образом, удерживая светодиодный индикатор в отключенном состоянии 2 с.

Затем цикл снова повторяется и будет продолжать делать это, пока его не остановят.

Полный код скетча будет выглядеть следующим образом:

```
/*
Скетч Blinking LED создан
Джоном Смитом 12/11/2014
*/

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

void loop() {
  digitalWrite(13, HIGH);
  delay(2000);
  digitalWrite(13, LOW);
  delay(2000);
}
```

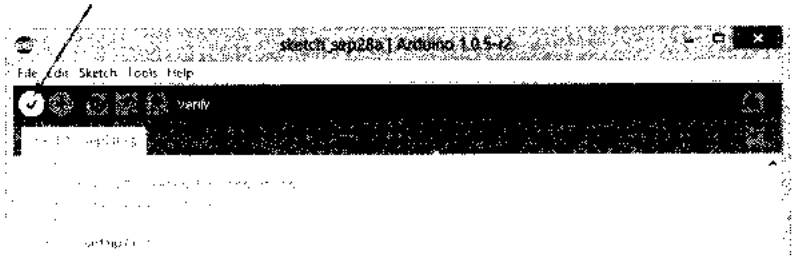
Глобальные переменные (в отличие от локальных) должны указываться перед функцией `setup()`. Вызовы библиотеки указываются в верхней части скетча.

Обратите внимание, что в нашем скетче Blinking LED глобальных переменных нет. Если бы они были (как во многих скетчах), их вводили бы после комментариев, но до функции `setup()`.

Проверка скетча

Операция проверки скетча, известная как компилирование — это процесс проверки платой Arduino корректности синтаксиса, т. е. что код написан в формате, поддерживаемом Arduino. Выполняется это следующим образом.

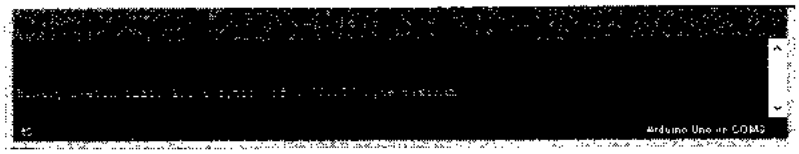
1. Когда вы закончили писать код скетча, нажмите кнопку **Verify** (Проверить) в верхнем левом углу интегрированной среды разработки.



На заметку

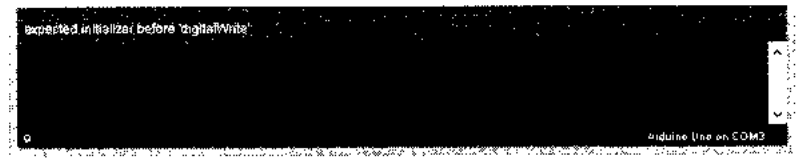
Компилирование скетча не будет выполнено, если он содержит какие-либо синтаксические ошибки.

2. Через несколько секунд (в зависимости от размера скетча) вы должны увидеть следующее сообщение.



3. В сообщении говорится: «Компиляция завершена», а также сообщается размер скетча, который будет загружен в Arduino, в байтах. Это сообщение говорит о том, что скетч отформатирован корректно и должен загрузиться без проблем.

4. Однако если вы получите сообщение об ошибке, как показано ниже, в таком случае вам необходимо будет исправить ошибку до завершения процедуры — см. с. 223–230.



Внимание

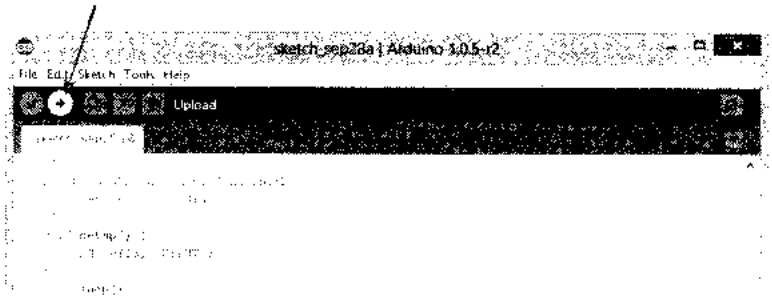
Размер скетча может быть очень важен из-за ограниченного объема памяти. Процедура проверки сообщает вам размер скетча.

❺ Сообщение содержит номер ошибочной строки и тип ошибки.

Выгрузка скетча

Проверка выполнена, теперь вы готовы загрузить скетч.

❶ При условии, что между платой и компьютером осуществлено надежное соединение, нажмите кнопку **Upload** (Загрузка) в окне редактора Arduino.



Совет

Если ваш скетч не загружается, откройте меню **Tools → Board** (Инструменты → Плата) и убедитесь, что выбрана плата Arduino Uno (или та плата, которую вы используете).

❷ Компилирование скетча будет выполнено посредством интегрированной среды разработки, а затем скетч будет загружен. Во время этой процедуры вы увидите, как мигают светодиодные индикаторы **TX/RX** на плате Arduino — это указывает на то, что данные передаются на плату и поступают с нее.

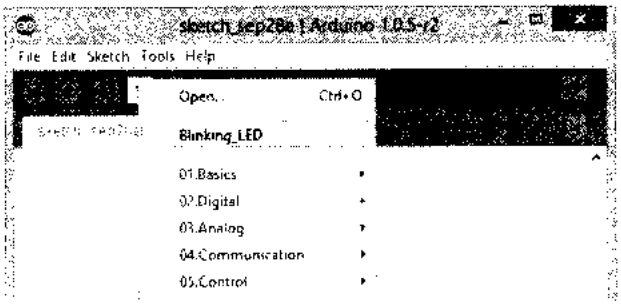
❸ Если все идет по плану, скетч запустится. Вам не нужно нажимать кнопку **Run** (Запуск) — как только загрузка закончится, скетч запустится автоматически. Посмотрите на плату, и вы увидите, как светодиодный индикатор мигает, включаясь и выключаясь.

На заметку

Сочетание клавиш для Windows — **Ctrl+U**. В операционной системе macOS используйте сочетание **Cmd+U**.

Изменение скетча

Возможно, в какой-то момент вы захотите отредактировать код своего скетча, чтобы изменить способ его работы. Интегрированная среда разработки Arduino упрощает этот процесс. Нажмите кнопку **Open** (Открыть), и, если ранее вы сохраняли скетч, вы увидите его в верхней части списка. Выберите соответствующий пункт, чтобы открыть скетч. Выполните необходимые изменения, нажмите кнопку **Save** (Сохранить), а затем щелкните мышью по кнопке **Upload** (Загрузка).



Скетч Fade

Наш скетч Blinking LED представляет собой очень простой пример, предназначенный для знакомства с основными компонентами скетча, процедурами его проверки и загрузки. Теперь давайте взглянем на более сложный скетч под названием Fade.

Он демонстрирует, как можно использовать функцию `analogWrite()` для постепенного угасания и зажигания светодиодных индикаторов с помощью возможностей широтно-импульсной модуляции (ШИМ) платы Arduino. Как мы уже увидели, ШИМ обеспечивает метод преобразования цифрового сигнала в аналоговый. Затем аналоговый сигнал будет использоваться, чтобы постепенно изменять яркость светодиодного индикатора с постепенным его потускнением и загоранием.

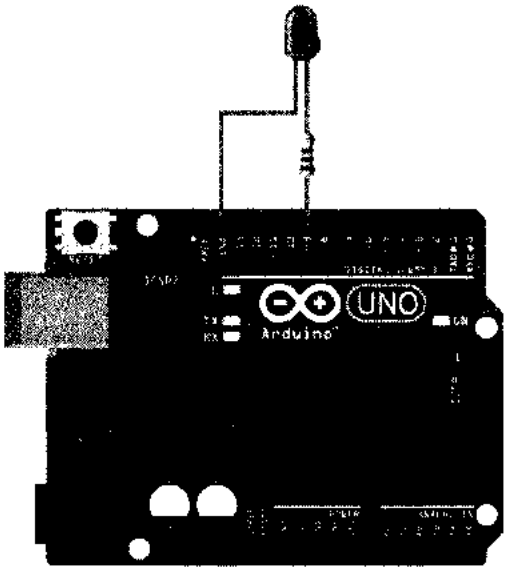
Настоящий цифровой сигнал представляет собой прямоугольную волну и, таким образом, может быть либо включен, либо выключен. Поэтому,

если он используется для электропитания светодиодных индикаторов, светодиодный индикатор будет либо включен, либо выключен, без изменения яркости между двумя этими положениями.

Функция `analogWrite()` обеспечивает способ имитации аналогового сигнала на цифровом контакте.

Цепь

Для создания цепи вам потребуется резистор 220 Ом и светоизлучающий диод (светодиод) любого цвета.



❶ Подключите короткую ножку светодиодного индикатора к контакту **GND**, а другую ножку — к контакту **9** через резистор.

Резистор 220 Ом имеет полосы красного и красно-коричневого цветов.

❷ Теперь подключите плату к своему компьютеру с помощью USB-кабеля и откройте среду разработки Arduino.

❸ Выберите команду меню **File → Examples → 01.Basics → Fade** (Файл → Примеры → 01.Basics → Fade).

❹ **Нажмите кнопку Upload (Загрузка), а затем взгляните на плату Arduino. Вы увидите, что яркость светодиодного индикатора постепенно увеличивается, а затем уменьшается.**

В редакторе вы увидите следующий код:

```
/*
  Fade
  Этот пример демонстрирует, как сделать так, чтобы
  светодиод на контакте 9 начал медленно гаснуть
  с помощью функции analogWrite().
  Образец этого кода находится в открытом доступе.
  */

int led = 9;              // контакт, к которому
                             подключен светодиод
int brightness = 0;       // яркость светодиода
int fadeAmount = 5;       // на сколько единиц выполнить
                             затухание светодиода

// настройка выполняется однократно после нажатия
кнопки перезагрузки:
void setup() {
  // настройка контакта 9 в качестве выхода:
  pinMode(led, OUTPUT);
}

// цикл запускается вновь и вновь до бесконечности:
void loop() {
  // установка яркости на контакте 9:
  analogWrite(led, brightness);

  // изменение яркости для последующей итерации
  цикла:
  brightness = brightness + fadeAmount;

  // обращение направления затухания при достижении
  предела:
  if (brightness == 0 || brightness == 255) {
  fadeAmount = -fadeAmount;
  }
```

```
  // ожидание в течение 30 мс для видимости эффекта
  затемнения
  delay(30);
}
```

Анализ скетча

В начале скетча можно увидеть многострочный, или блочный, комментарий, описывающий назначение скетча.

```
/*
  Fade
  Этот пример демонстрирует, как сделать так, чтобы
  светодиод на контакте 9 начал медленно гаснуть
  с помощью функции analogWrite().
  Образец этого кода находится в открытом доступе.
  */
```

Далее расположен фрагмент кода, который объявляет и инициализирует три глобальные переменные `int`:

```
int led = 9;           // контакт, к которому подключен
                          светодиод
int brightness = 0;    // яркость светодиода
int fadeAmount = 5;    // на сколько единиц выполнить
                          затухание светодиода
```

Переменная `led` определяет, к какому контакту подключен светодиодный индикатор. Переменная `brightness` хранит текущую яркость светодиодного индикатора, в то время как переменная `fadeAmount` определяет уровень изменения яркости светодиодного индикатора.

Следующий фрагмент кода — это функция `setup()`:

```
// настройка выполняется однократно после нажатия
кнопки перезагрузки:
void setup() {
  // настройка контакта 9 в качестве выхода:
  pinMode(led, OUTPUT);
}
```

Эта функция делает только одно: использует локальную функцию `pinMode()` для определения контакта 9 в качестве выхода.

Функция `pinMode()` использует два значения: номер контакта (1, 2, 3 и т.д.) и режим (`INPUT` или `OUTPUT`).

Далее расположен код функции `loop()`:

```
// цикл запускается вновь и вновь до бесконечности:
void loop() {
  // установка яркости на контакте 9:
  analogWrite(led, brightness);

  // изменение яркости для последующей итерации
  цикла:
  brightness = brightness + fadeAmount;

// обращение направления затухания при достижении
предела:
if (brightness == 0 || brightness == 255) {
  fadeAmount = -fadeAmount;
}
// ожидание в течение 30 мс для видимости эффекта
затемнения
delay(30);
}
```

Функция `loop()` обычно содержит основную часть кода проекта.

Первый фрагмент кода — `analogWrite(led, brightness)` — активирует функцию широтно-импульсной модуляции Arduino (ШИМ), которая позволяет варьировать мощность на выходе (контакт 9), преобразовывая цифровой сигнал в аналоговый. Это, в свою очередь, позволяет изменять яркость на контакте 9, а не просто находиться в состоянии «включено» или «выключено», как было бы с прямоугольным цифровым сигналом.

Количество энергии, обеспечиваемое ШИМ, варьируется в диапазоне от 0 до 255.

Объясним это подробнее:

Функция `analogWrite(led, brightness)` содержит два аргумента — `led` и `brightness`. Возвращаясь к трем переменным в начале скетча, мы видим, что в первой, `int led = 9`, мы объявили, что светодиодный индикатор подключен к контакту **9**. Во второй, `int brightness = 0`, мы объявили, что яркость светодиодного индикатора равна 0, или же отсутствует. Функция `analogWrite(led, brightness)` вызывает значения, установленные в двух этих переменных.

Далее расположен код `brightness = brightness + fadeAmount;`. Возвращаясь к переменным, эта функция берет текущую яркость светодиодного индикатора, которая равна 0, а затем добавляет значение `fadeAmount`, равное 5. Таким образом, мы имеем значение, равное 5, которое сохраняется в переменную `int brightness`.

На заметку

К некоторым важным моментам функции `analogWrite()` относится следующее:

- Она не имеет ничего общего с аналоговыми контактами.
- Работает с контактами **3**, **5**, **6**, **10** и **11**.
- Использует функцию ШИМ, чтобы обеспечить регулируемую мощность на выходном контакте.

Поэтому, тогда как изначально значение равно 0 и светодиодный индикатор не горит при первом запуске цикла, в следующий раз значение будет равно 5, и светодиодный индикатор начнет светиться. Поскольку код повторяется в цикле, значение, сохраненное в переменную `int brightness`, стремительно увеличивается. Каждый раз, когда это происходит, возрастает мощность, подаваемая на светодиодный индикатор, заставляя его светиться ярче.

В итоге достигается предел функции `analogWrite()`, равный 255, и светодиодный индикатор уже не будет светить ярче. Таким образом, необходимо проверить значение переменной `brightness`, а затем изменить его, если оно достигло предела.

Здесь вступает в игру следующий участок кода — инструкция `if`. Она проверяет условие, и, если оно соблюдается, выполняет действие. Если условие не соблюдается, инструкция не выполняет ничего.

```
if (brightness == 0 || brightness == 255) {
  fadeAmount = -fadeAmount;
```

Условие, которое необходимо проверить (`brightness`), указано в круглых скобках `()`, а действие, которое необходимо выполнить (`fadeAmount`), если условие соблюдено, — в фигурных `{}`. Инструкции `if` либо истинны, в этом случае выполняется код в скобках; либо ложны — в этом случае код не выполняется.

Если мы посмотрим на первую строку, то увидим символы `||`. Это обозначение оператора «Логическое ИЛИ». Следовательно, условие утверждает, что «если переменная `brightness` равна нулю или переменная `brightness` равна 255».

Также обратите внимание на символы ==. Это обозначение оператора сравнения «равно», который спрашивает, одинаковы ли эти два значения.

Внимание! Не путайте символы = и ==. Первый используется для присвоения значения, тогда как второй представляет собой оператор сравнения «равно».

В нашем скетче значение яркости достигло максимума, равного 255, поэтому условие `if (brightness == 0 || brightness == 255)` было соблюдено. В этом случае код выполняет действие, которым является переменная `fadeAmount = -fadeAmount;`.

Все, что происходит в действии, это то, что отрицательный знак (`-`) ставится перед значением `fadeAmount`, превращая его таким образом из положительного значения в отрицательное. Каждый раз, когда этот код повторяется в цикле, мы берем значение равное `5` из переменной `brightness`, постепенно уменьшая мощность, подаваемую на светодиодный индикатор, и, таким образом, уменьшаем его яркость.

Когда переменная `brightness` достигает значения `0`, действие переменной `fadeAmount` переключается на положительное, и весь процесс начинается снова.

Наконец, скорость, с которой происходит процесс затухания, необходимо сократить до очень медленной. Если мы не сделаем этого, светодиодный индикатор будет гореть непрерывно. Это делается с помощью последнего фрагмент кода:

```
    // ожидание в течение 30 мс для видимости эффекта
    затемнения
    delay(30);
}
```

Функция `delay` просто применяет задержку в 30 мс, достаточную, чтобы сделать изменение яркости светодиодного индикатора различимым для человеческого глаза.

Скетч DigitalReadSerial

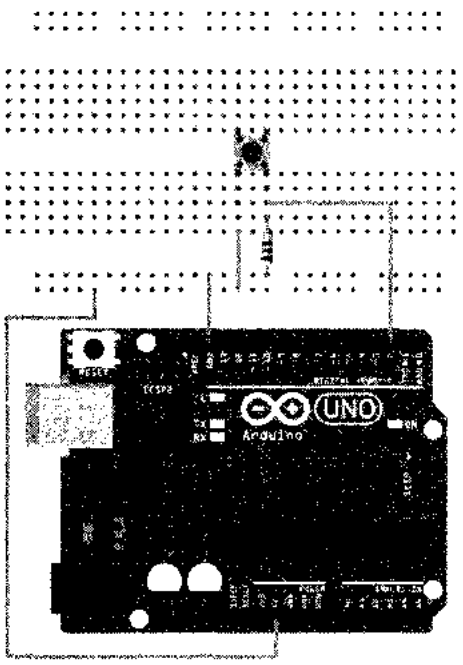
Этот скетч позволяет управлять состоянием выключателя или датчика, т. е. определять, включен он или выключен. Это позволяет инициировать действия соответственно: если кнопка не нажата, выполнять действие А, если же нажата — действие Б.

Чтобы выполнить эту задачу, необходимо знать состояние контактов Arduino и отслеживать любые изменения, если они происходят. Окно **Serial Monitor** (Монитор порта) позволяет справиться с этой задачей.

Цепь

Понадобятся следующие компоненты:

- Кнопка запуска
- Резистор 10,000 Ом (10 кОм)
- Перемычки
- Макетная плата



Установите переключатель на макетную плату, как показано на рисунке. Затем подключите один его конец к контакту **5V** на плате Arduino, а другой конец — к контакту **GND** через резистор 10 кОм. Также подключите этот конец к контакту **2** на плате Arduino.

Подключите плату к компьютеру с помощью USB-кабеля и запустите интегрированную среду разработки. Выберите команду меню **File** → **Examples** →

01.Basics → digitalReadSerial (Файл → Примеры → 01.Basics → digitalReadSerial). Затем нажмите кнопку **Upload** (Загрузка).

После того как скетч загрузился, выберите команду **Tools → Serial Monitor** (Инструменты → Монитор порта) в строке меню. Откроется окно **Serial Monitor** (Монитор порта), и вы увидите последовательность нескольких 0. Нажмите кнопку на плате и удерживайте ее нажатой — будет отображаться последовательность единиц.

На заметку. Плата Arduino непрерывно считывает состояние контакта **2**, чтобы определить, открыта или закрыта кнопка.

Когда она открыта (не нажата), контакт **2** подключается к **GND** через резистор и таким образом считывает `LOW`, или 0. Когда она закрыта (нажата), контакт **2** подключается к **5V**, и таким образом контакт считывает `HIGH`, или 1.

В редакторе вы увидите следующий код:

```
/*
  DigitalReadSerial
  Считывает цифровой вход на контакте 2, выводит
  результат в монитор порта.
  Этот образец кода находится в открытом доступе.
  */

  // к цифровому контакту 2 подключена кнопка.
  Присвоение имени:
  int pushButton = 2;

  // настройка выполняется однократно после нажатия
  кнопки перезагрузки:
void setup() {
  //инициализация последовательной связи на скорости
  9600 бит/с:
  Serial.begin(9600);
  // настройка контакта кнопки в качестве входа:
  pinMode(pushButton, INPUT);
}

  // цикл запускается вновь и вновь до бесконечности:
void loop() {
```

```
  // считывание контакта входа:
  int buttonState = digitalRead(pushButton);
  // вывод состояния кнопки:
  Serial.println(buttonState);
  delay(1);    // задержка между считываниями для
  стабильности
}
```

Анализ скетча

Скетч начинается с блочного комментария, который описывает назначение скетча, т. е. считывание значения на контакте 2 и последующий его вывод в окне **Serial Monitor** (Монитор порта).

За ним следует фрагмент кода, который объявляет и инициализирует переменные скетча. В этом скетче присутствует только одна:

```
// к цифровому контакту 2 подключена кнопка.
Присвоение имени:
int pushButton = 2;
```

Комментарий сообщает вам все, что необходимо знать, — что кнопка запуска в цепи подключена к контакту 2.

Далее следует код функции `setup()`:

```
// настройка выполняется однократно после нажатия
кнопки перезагрузки:
void setup() {
  //инициализация последовательной связи на скорости
  9600 бит/с:
  Serial.begin(9600);
  // настройка контакта кнопки в качестве входа:
  pinMode(pushButton, INPUT);
}
```

На заметку

Библиотека представляет собой группу функций, выполняющих определенные действия. Ваша плата Arduino поставляется с рядом встроенных библиотек (включая библиотеку `Serial`). Также вы можете загрузить и использовать библиотеки, разработанные другими программистами.

Код содержит две функции: `Serial.begin()` и `pinMode()`. Функция `Serial.begin()` принадлежит к связанной группе библиотечных функций — в данном случае библиотеки `Serial`. Слово `Serial` — это название библиотеки, а слово после точки — имя функции.

Поэтому код `Serial.begin(96000);` — это вызов функции `begin` из библиотеки `Serial`. Она активирует монитор порта, который позволяет вам увидеть в окне **Serial Monitor** (Монитор порта) на компьютере то, что происходит на плате Arduino.

За кодом функции `Serial.begin(9600)` следует функция `pinMode()`. Она позволяет установить контакт в качестве либо входа, либо выхода. В нашем случае слово `pushbutton` вызывает значение, определенное в переменной `int pushbutton = 2` в начале скетча.

Таким образом, контакт **2** выбирается и устанавливается как ввод (`INPUT`).

Далее следует функция `loop()`:

```
// цикл запускается вновь и вновь до бесконечности:
void loop() {
  // считывание контакта входа:
  int buttonState = digitalRead(pushButton);
  // вывод состояния кнопки:
  Serial.println(buttonState);
  delay(1);    // задержка между считываниями для
  стабильности
}
```

Первая строка кода представляет собой следующую переменную:

```
int buttonState = digitalRead(pushButton)
```

Она объявляет целочисленное значение и присваивает ему имя `buttonState`. Цель переменной — хранение состояния кнопки, т. е. включена она или выключена.

Переменная инициализируется путем установки оператора присваивания между ней и выводом функции `digitalRead()`. Эта функция возвращает значение равное либо `1`, либо `0`, где `1` указывает, что напряжение на контакте будет высоким, а `0` — низким. Контактом, о котором идет речь, конечно же является контакт **2**. Это происходит потому, что `pushButton` представляет собой значение, указанное в скобках переменной, и `pushButton` уже инициализировано в качестве контакта **2** в глобальной переменной в начале скетча.

В результате, когда нажата кнопка запуска, она замыкает цепь между контактом **2** и выходом **5V** платы Arduino. Это считывается функцией `digitalRead(pushbutton)` и интерпретируется как значение `HIGH`. Когда кнопка не нажата, контакт **2** подключен к контакту **GND**, поэтому на нем нет напряжения. Это считывается и интерпретируется как значение `LOW`.

Далее следует функция `Serial.println()`:

```
Serial.println(buttonState);
```

Мы берем функцию `Serial.println()` из той же библиотеки Serial, что и функцию `Serial.begin()`. Она считывает состояние кнопки и выводит его в монитор порта в виде единиц и нулей.

На заметку

Пусть вас не вводит в заблуждение имя функции `Serial.println()` — она не имеет ничего общего с вашим принтером. Данная функция отправляет выходные данные платы Arduino на монитор вашего компьютера.

Наконец, функция `delay()` замедляет процесс работы, в данном случае на 1 мс, чтобы было проще прочитать данные.

```
delay(1); // задержка между считываниями для
стабильности
}
```

Скетч AnalogReadSerial

Скетч digitalReadSerial, который мы только что рассмотрели, позволяет нам определять цифровые значения, т. е. включено что-либо или выключено. В принципе, данная информация представляет собой интерес, но иногда нам необходимо знать больше.

К примеру, возьмем звуковой датчик, который предоставляет информацию о высоте звука и его амплитуде. Датчики такого типа возвращают аналоговые значения, которые могут охватить широкий диапазон, а не просто состояние включено или выключено.

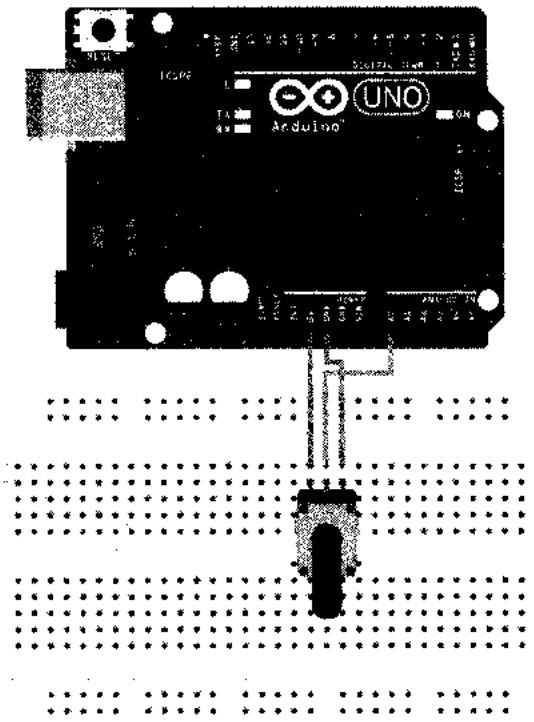
Для этого используем скетч AnalogReadSerial. Его код считывает сигналы на аналоговых входных контактах Arduino — от **A0** до **A5**.

Аналоговые входы Arduino могут выдавать значения в диапазоне от 0 до 1023. Значение 0 соответствует 0 В, 512–2,5 В, а 1023–5 В.

Цепь

Для конструирования цепи вам потребуются следующие компоненты:

- Потенциометр
- Макетная плата
- Перемычки



Установите потенциометр на макетную плату, как показано на рисунке. Затем подключите один из двух наружных выводов потенциометра к контакту **5V** на плате Arduino, а другой — к контакту **GND**. Наконец, подключите средний вывод потенциометра к контакту **A0** на плате Arduino.

Подключите Arduino к компьютеру с помощью USB-кабеля, запустите интегрированную среду разработки Arduino и выберите команду меню **File** → **Examples** → **01.Basics** → **AnalogReadSerial** (Файл → Примеры → 01.Basics → AnalogReadSerial).

Щелкните мышью по кнопке **Upload** (Загрузка).

На заметку. Контакты аналоговых входов подключены к аналого-цифровому преобразователю. Он берет сигналы аналоговых входов и превращает их в цифровую форму, которую поддерживает Arduino.

Теперь выберите команду **Tools → Serial Monitor** (Инструменты → Монитор порта) в строке меню. Так вы откроете окно **Serial Monitor** (Монитор порта), которое отображает информацию, считываемую с платы Arduino, — в данном случае диапазон чисел между 0 и 1023. Поворот ручки потенциометра изменит эти цифры.

В этом скетче мы используем потенциометр, поскольку он выдает результат, который может изменяться. Вместо него вы можете использовать датчик любого типа.

В окне редактора вы увидите следующий код:

```
/*
Считывается аналоговый вход на контакте 0, результат
выводится в монитор порта.
Подключаем центральный контакт потенциометра
к контакту A0, а внешние контакты к +5В и к GND.
Этот образец кода находится в открытом доступе.
*/

// настройка выполняется однократно после нажатия
кнопки перезагрузки:
void setup() {
  //инициализация последовательной связи на скорости
  9600 бит/с:
  Serial.begin(9600);
}

// цикл запускается вновь и вновь до бесконечности:
void loop() {
  // считывание ввода на аналоговом контакте 0:
  int sensorValue = analogRead(A0);
  // вывод значения, которое вы считываете:
  Serial.println(sensorValue);
  delay(1); // задержка между считываниями для
  стабильности
}
```

Анализ скетча

Как обычно, скетч начинается с блочного комментария, который описывает, что скетч должен делать.

Обратите внимание, что после блочного комментария ни одна переменная не объявляется, в отличие от большинства скетчей. Причина этого заключается в том, что в данном скетче переменная объявляется в функции `loop()`.

За блочным комментарием следует функция `setup()`.

```
// настройка выполняется однократно после нажатия
кнопки перезагрузки:
void setup() {
  //инициализация последовательной связи на скорости
  9600 бит/с:
  Serial.begin(9600);
}
```

Все, что у нас есть в коде `setup()` — это функция `Serial.begin()`, которая инициализирует монитор порта на скорости 9600 бод. Как мы уже увидели, монитор порта — это канал связи, который соединяет компьютер с платой Arduino и позволяет вам в точности видеть то, что происходит на плате.

Обратите внимание, что в данном скетче мы не использовали функцию `pinMode()`, чтобы настроить контакт **A0** как вход. Причина этого заключается в том, что в этом нет необходимости, поскольку по умолчанию плата Arduino устанавливает все контакты как входы.

Однако для большей ясности, как правило, рекомендуется делать именно так. В нашем скетче код `pinMode(A10, INPUT)` должен указываться внутри фигурных скобок в функции `setup()`.

Необязательно специально настраивать контакты платы Arduino в качестве входов. Это уже настроено по умолчанию.

Далее следует код функции `loop()`:

```
// цикл запускается вновь и вновь до бесконечности:
void loop() {
  // считывание ввода на аналоговом контакте 0:
  int sensorValue = analogRead(A0);
```

```
    // вывод значения, которое вы считываете:
    Serial.println(sensorValue);
    delay(1);  // задержка между считываниями для
    стабильности
}
```

Как и в случае со скетчем digitalReadSerial, функция `loop()` начинается с объявления и инициализации переменной:

```
int sensorValue = analogRead(A0);
```

Ей присваивается имя `sensorValue`, и она инициализируется путем установки символа = между ней и выводом функции `analogRead()`.

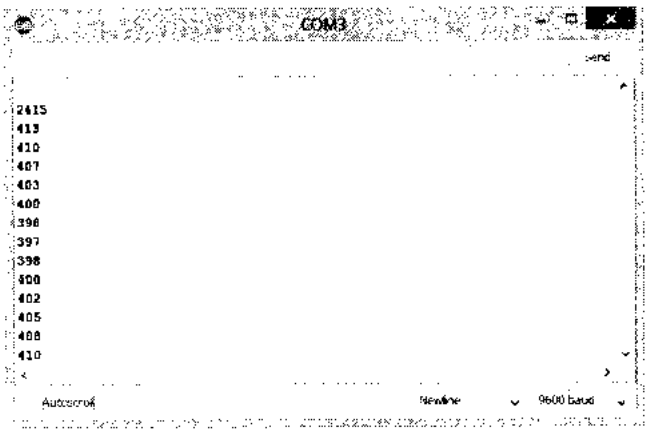
Функция `analogRead()` считывает значение на определенном контакте — **A0** в нашем скетче. Этот контакт подключен к среднему выводу потенциометра, напряжение на котором соответствует некоему числу в диапазоне от 0 до 1023. Это осуществляется с помощью цифроаналогового преобразователя. Затем переменной `sensorValue` присваивается число (значение).

На заметку

Число, генерируемое аналоговым входом, пропорционально напряжению на входе.

Обратите внимание, что фактическое напряжение на контакте **A0** находится в диапазоне от 0 до 5 В, в зависимости от положения ручки настройки потенциометра.

Также цикл содержит функцию `Serial.println()`. Она активирует монитор порта, с помощью которого мы можем просматривать значения, генерируемые скетчем на контакте **A0**, как показано на рисунке ниже:



Если вы увидите, что значения прокручиваются вниз слишком быстро, увеличьте значение задержки в вызове функции `delay()`.

Если вы покрутите ручку потенциометра, то увидите изменение значений в окне **Serial Monitor** (Монитор порта).

Функция `delay(1)` добавляет задержку в 1 мс между считываниями, чтобы замедлить процесс и упростить чтение информации.

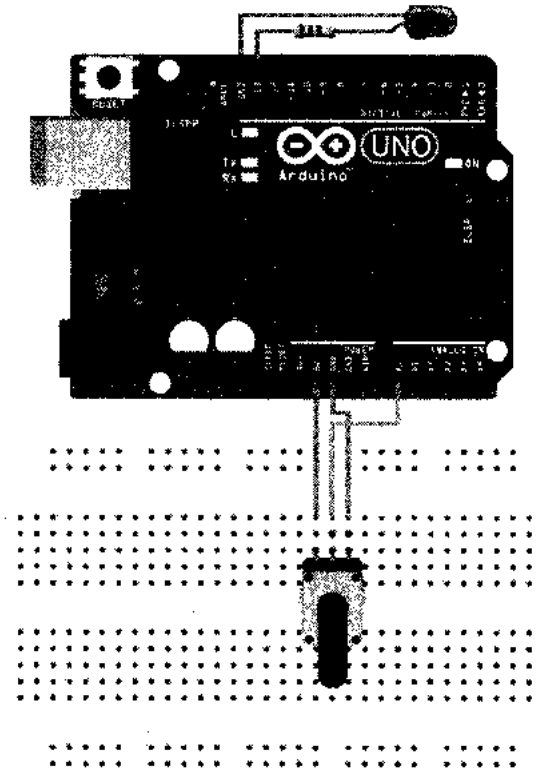
Скетч IfStatementConditional

Скетч Fade продемонстрировал, как инструкция `if` используется для выполнения действия, если было соблюдено определенное условие. В скетче ifStatementConditional мы разберемся, как располагать различные условия для обеспечения гибкости в ваших скетчах.

Цепь

Для создания цепи вам понадобятся следующие компоненты:

- Потенциометр
- Резистор 220 Ом
- Светодиодный индикатор
- Макетная плата
- Перемычки



Установите потенциометр на макетную плату, как показано на рисунке. Подключите один из его внешних выводов к контакту **5V** на плате, а другой вывод — к контакту **GND**. Затем подключите

его средний вывод к контакту **A0**. Подключите светодиодный индикатор и резистор так, как вы делали в скетче Fade, но в этот раз используйте контакт **13** вместо **9**.

Теперь подключите Arduino к своему компьютеру с помощью USB-кабеля и запустите интегрированную среду разработки Arduino. Выберите команду меню **File → Examples → 05.Control → ifStatementConditional** (Файл → Примеры → 05.Control → ifStatementConditional). Щелкните мышью по кнопке **Upload** (Загрузка).

На заметку

Обратите внимание, что в листинг мы не включили блочный комментарий, размещенный в начале скетча.

В редакторе вы увидите следующий код:

```
// Эти константы не меняются:
const int analogPin = A0;  // контакт, к которому
подключен датчик
const int ledPin = 13;     // контакт, к которому
подключен светодиод
const int threshold = 400; // уровень произвольного
порога в диапазоне аналогового входа

void setup() {
  // инициализация контакта светодиода в качестве
  выхода:
  pinMode(ledPin, OUTPUT);
  // инициализация последовательной связи:
  Serial.begin(9600);
}

void loop() {
  // считывание значения потенциометра:
  int analogValue = analogRead(analogPin);

  // если аналоговое значение достаточно высокое,
  включается светодиод:
  if (analogValue > threshold) {
  digitalWrite(ledPin, HIGH);
  } else {
```

```
    digitalWrite(ledPin, LOW);
    }

    // вывод аналогового значения:
    Serial.println(analogValue);
    delay(1);    // задержка между считываниями для
    стабильности
}
```

Анализ скетча

Первое, что необходимо понять в данном случае — это принцип работы потенциометра. На одном терминале напряжение составляет 5 В, а на другом — 0 В. Когда мы вращаем ручку настройки, напряжение на центральном контакте (выходе) варьируется от 0 до 5 В.

На заметку

Потенциометр, по сути, представляет собой трехтерминальный резистор с подвижным или вращающимся контактом, который выступает в качестве делителя напряжения.

Второе — это назначение скетча, которое, по сути, очень просто. Напряжение на аналоговом контакте **A0** (выход потенциометра) измеряется, и когда оно достигает порогового значения (400 в примере), загорается светодиодный индикатор, подключенный к контакту **13**. Когда оно ниже порогового значения, светодиодный индикатор отключается.

Первый блок кода содержит переменные:

```
// Эти константы не меняются:
const int analogPin = A0;  // контакт, к которому
                              подключен датчик
const int ledPin = 13;     // контакт, к которому
                              подключен светодиод
const int threshold = 400; // уровень произвольного
                              порога в диапазоне
                              аналогового входа
```

Здесь у нас есть три переменные: первые две, `analogPin` и `ledPin`, представляют собой контакты, к которым соответственно подключены потенциометр и светодиодный индикатор. Третья, `threshold`, представляет собой

произвольное число, выбранное в качестве условия, при котором светодиодный индикатор включается или выключается.

Константы обеспечивают подобие системы поддержки, которая защищает ваш скетч от последствий непредвиденных изменений значений.

Вы видите слово `const`, которое предшествует каждой из этих переменных. Это ключевое слово, обозначающее константу. Назначение этих констант заключается в том, чтобы быть уверенным в неизменности значений, т. е. светодиодный индикатор всегда подключен к контакту **13**, порог всегда равен 400 и т. д.

Далее следует функция `setup()`.

```
void setup() {
  // инициализация контакта светодиода в качестве
  выхода:
  pinMode(ledPin, OUTPUT);
  // инициализация последовательной связи:
  Serial.begin(9600);
}
```

Первая функция, `pinMode()`, настраивает контакт, к которому подключен светодиодный индикатор, как выход. Вторая, `Serial.begin()`, инициализирует монитор порта на скорости 9600 бод.

После кода функции `setup()` указана функция `loop()`. Как и в большинстве скетчей, в ней приведена основная часть кода скетча.

```
void loop() {
  // считывание значения потенциометра:
  int analogValue = analogRead(analogPin);

  // если аналоговое значение достаточно высокое,
  включается светодиод:
  if (analogValue > threshold) {
  digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
```

Сжато код цикла можно интерпретировать так: если условие А истинно, выполняйте действие Б. В противном случае выполняйте действие В.

В верхней части кода цикла находится функция с именем `analogRead()`. Она считывает значение на контакте **A0** и присваивает его целочисленной переменной с именем `analogValue`. Поскольку напряжение на контакте **A0** — это вывод потенциометра, данная строка кода проверяет уровень напряжения.

Далее, значение на контакте **A0** сравнивается с пороговым значением, равным `400`. Это осуществляется с помощью инструкции `if`, определяющей, что если значение на контакте **A0** больше порогового значения, т. е. равно `HIGH`, то светодиодный индикатор будет включен.

Заключительную часть кода цикла занимает инструкция `else`. В ней говорится, что если условие в инструкции не соблюдается, т. е. порог на контакте **A0** не превышает 400, или `LOW`, то светодиодный индикатор выключается.

Из этого можно понять, что инструкция `else` определяет поведение при несоблюдении условия в инструкции `if`.

Заключительный фрагмент кода выглядит так:

```
    // вывод аналогового значения:
    Serial.println(analogValue);
    delay(1);    // задержка между считываниями для
    стабильности
}
```

Этот скетч демонстрирует оператор сравнения в действии; в данном случае речь идет об операторе «больше чем», >.

В нем используется функция `println()` для передачи данных скетча в монитор порта, где их можно просмотреть. Так вы можете с точностью контролировать работу скетча.

Функция `delay()` замедляет процесс, чтобы упростить считывание данных, в данном случае на 1 мс.

Скетч ForLoopIteration

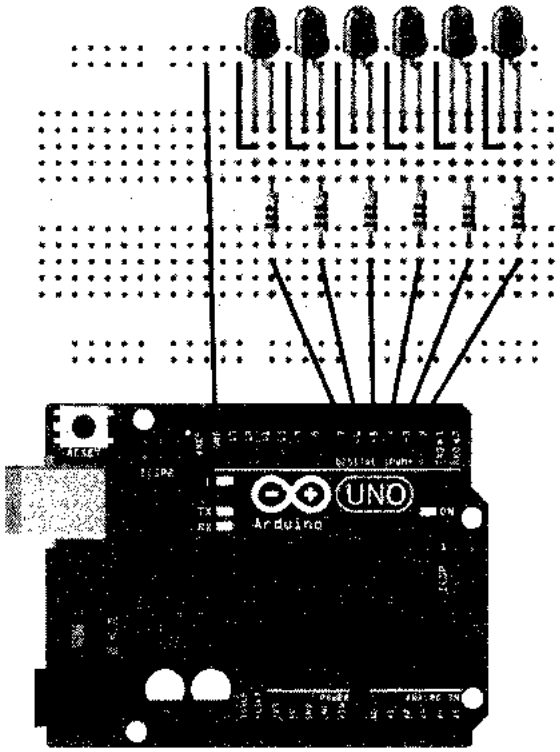
Существует большое количество функций, которые можно использовать при написании скетчей Arduino. Некоторые из них полезнее, чем другие, и в эту категорию определенно попадает цикл `for`.

По сути, он позволяет точно указать, сколько именно раз необходимо выполнить действие. Это не только экономит программисту время и избавляет от необходимости писать код для каждого экземпляра, но также делает код короче и, соответственно, упрощает его чтение.

Цепь

Цепь для этого скетча потребует наличия следующих компонентов:

- Шесть резисторов 220 Ом
- Шесть светодиодных индикаторов
- Макетная плата
- Перемычки



На заметку

Нет необходимости располагать компоненты в точности так, как показано здесь. Воспользуйтесь собственным вариантом размещения, но убедитесь в том, что длинная ножка каждого светодиодного индикатора подключена к цифровому контакту через резистор, и том, что другие ножки подключены к общей шине электропитания.

Возьмите шесть светодиодных индикаторов и подключите их так, как показано на рисунке. Длинная ножка каждого светодиодного индикатора подключается к цифровому контакту (используйте контакты от **2** до 7) через резистор с сопротивлением 220 Ом, а короткая ножка — к общей шине электропитания макетной платы. Наконец, подключите общую шину электропитания к контакту **GND** на плате Arduino.

Подключите Arduino к своему компьютеру с помощью USB-кабеля, запустите интегрированную среду разработки и выберите команду меню **File → Examples → 05.Control → ForLoopIteration** (Файл → Примеры → 05.Control → ForLoopIteration). Нажмите кнопку **Upload** (Загрузка) и дождитесь загрузки скетча.

Вы увидите, как светодиодный индикатор последовательно включается и выключается.

На заметку

Обратите внимание, что в листинге не указан блочный комментарий в начале скетча.

В редакторе отобразится следующий код:

```
int timer = 100;    // Чем больше число, тем ниже
темп.

void setup() {
  // используется цикл for для инициализации каждого
  контакта в качестве выхода:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
  pinMode(thisPin, OUTPUT);
  }
}
```

```
void loop() {
  // циклическая обработка от младшего контакта
  до старшего:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
  // включение контакта:
  digitalWrite(thisPin, HIGH);
  delay(timer);
  // отключение контакта:
  digitalWrite(thisPin, LOW);
  }

  // циклическая обработка от старшего контакта
  до младшего:
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
  // включение контакта:
  digitalWrite(thisPin, HIGH);
  delay(timer);
  // отключение контакта:
  digitalWrite(thisPin, LOW);
  }
}
```

Анализ скетча

Первая строка кода — это переменная:

```
int timer = 100;    // Чем больше число, тем ниже темп.
```

Это целое число, которое определяет время, выделяемое на включение и выключение светодиодного индикатора. Судя по комментарию, чем больше число, тем ниже частота повторения импульсов.

Далее следует функция `setup()`:

```
void setup() {
  // используется цикл for для инициализации каждого
  контакта в качестве выхода:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
  pinMode(thisPin, OUTPUT);
  }
}
```

Ее код содержит цикл `for`. В скобках указаны три отдельных инструкции: первая, `int thisPin = 2`, инициализирует контакт, к которому подключен первый светодиодный индикатор, в данном случае контакт **2**.

Внимание! При использовании операторов ++ и -- для увеличения или уменьшения значения по умолчанию применяется 1 — данное значение не нужно указывать. Но все прочие значения указать нужно. Например, `thisPin++` или `thisPin--`.

Вторая инструкция, `thisPin < 8`, определяет, запустится ли цикл. Если условие истинно, т. е. 2 меньше 8, что так и есть, код в скобках цикла запустится. Если же нет, инструкция не будет выполнена и скетч перейдет к следующей строке кода.

Последняя инструкция, `thisPin++`, обеспечивает возможность приращения. Символы ++ означают добавление 1 к значению `thisPin`.

Выделяя основные моменты из вышеизложенного, скажем, что, если условие в цикле соблюдается, код в скобках будет выполняться, а переменная `thisPin` будет увеличена на 1. Затем произойдет итерация цикла, и, предполагая, что условие все еще соблюдается, код снова будет выполняться, а переменная вновь увеличиваться. Эта последовательность будет повторяться до тех пор, пока переменная `thisPin` не получит значение, равное 8. В этот момент условие станет ложным, и, следовательно, цикл прекратит работу.

Этот скетч демонстрирует преимущества цикла `for`. Без него нам пришлось бы писать отдельный экземпляр переменной `pinMode` для каждого светодиодного индикатора. При использовании цикла `for` нам необходим всего один экземпляр.

Каждый раз, когда переменная `thisPin` увеличивается на 1, значение переменной `pinMode` () увеличивается соответственно. Следовательно, контакты со **2** по **7** последовательно устанавливаются как выходы.

После функции `setup()` расположена функция `loop()`:

```
void loop() {
  // циклическая обработка от младшего контакта
  до старшего:
```

```
    for (int thisPin = 2; thisPin < 8; thisPin++) {
      // включение контакта:
      digitalWrite(thisPin, HIGH);
      delay(timer);
      // отключение контакта:
      digitalWrite(thisPin, LOW);
  }
```

Функция `digitalWrite(thisPin, HIGH)` имеет два аргумента — номер контакта и выход (`HIGH` или `LOW`). Константа `HIGH` подает 5 В на `thisPin`, включая таким образом светодиодный индикатор, подключенный к контакту. Контакт, о котором идет речь, имеет номер 2, как указано в переменной `thisPin`.

Происходит задержка, отчего светодиодный индикатор продолжает гореть в течение короткого периода времени. Это достигается с помощью функции `delay(timer)`, которая на 100 мс приостанавливает выполнение кода. Затем функция `digitalWrite(thisPin, LOW)` передает значение `LOW` на контакт 2, снимая напряжение и отключая светодиодный индикатор.

Рекомендуется использовать функцию `delay()` только с короткими задержками, так как она может полностью прекратить выполнение программы. К альтернативным методам относятся следующие функции:

- `millis()`
- `micros()`
- `delayMicroseconds()`

Цикл возобновляет свою работу, увеличивая переменную `counter` на 3, включая таким образом светодиодный индикатор, подключенный к контакту 3. Цикл продолжает работать до тех пор, пока не будет включен последний светодиодный индикатор (на контакте 7).

Теперь мы должны обратить процесс вспять, чтобы вернуться к контакту 2. Это достигается с помощью финального фрагмента кода, который представлен еще одним циклом.

```
// циклическая обработка от старшего контакта
до младшего:
for (int thisPin = 7; thisPin >= 2; thisPin--) {
  // включение контакта:
```

```
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // отключение контакта:
    digitalWrite(thisPin, LOW);
    }
}
```

По сравнению с первым циклом он работает иначе. Код начинается с переменной `thisPin`, установленной на контакте 7. Теперь проверочное условие можно прочитать так: «данный контакт больше или равен 2». Инструкция `thisPin--` вычитает `1` из значения `thisPin` каждый раз, когда цикл выполняется. Когда значение возвращается к `2`, тестовое условие не соблюдается, и цикл завершается. Далее происходит возвращение к верхней части первого цикла, и вся процедура повторится снова.

10

Устранение неисправностей и отладка

Существует множество причин, из-за которых проекты Arduino могут начать работать со сбоями. Мы рассмотрим некоторые распространенные проблемы и методы их решения.

- Перед началом работы
- Аппаратное обеспечение
- Проблемы с настройкой
- Ошибки синтаксиса
- Монитор порта
- Отладка

Перед началом работы

Опытные пользователи Arduino знают, что потенциальные проблемы не заставят себя ждать. Чем сложнее проект, тем выше вероятность возникновения сложностей. Разумеется, у новичков будут возникать трудности при работе даже с простыми проектами.

Знания

Не имеет значения, какой опыт работы с платой Arduino у вас есть, проблемы также могут возникнуть из-за ряда различных навыков, необходимых для многих проектов, — электроники, программирования, вычислительной техники, механики, столярного дела, металлообработки и т. д. Все это в сочетании добавляет еще больше проблем и сложностей.

Очевидно, что, чем больше вы понимаете, что именно делаете и как взаимодействуют друг с другом разные компоненты проекта, тем больше вероятности в том, что вы сможете устранять проблемы по мере их поступления. Таким образом, знания — это самое главное оружие в вашем арсенале. Узнавайте как можно больше, в частности из области электроники, вычислительной техники и программирования. Не обладая хорошими знаниями в этих областях, при работе с Arduino вы столкнетесь с большими трудностями.

Довольно часто правильное направление укажет вам анализ именно работающих компонентов, а нет тех, что неисправны.

Технические приемы

Важно также то, как именно вы справляетесь с неполадками. Многие люди пытаются идти напролом, довольно неуклюже подходя к решению проблемы, возникающей при работе с тем или иным компонентом. Как правило, в конечном итоге ситуация становится только хуже.

Гораздо более разумный подход, который почти гарантировано даст лучший результат, заключается в первоначальном логическом рассмотрении того, что происходит (или не происходит, вероятность чего тоже существует). Исключите элементы проекта, не имеющие никакого отношения к проблеме. Если

вы сможете исключить 50% таких элементов, то вероятность того, что вы отыщете проблему, будет на 50% выше. Исключите как можно больше элементов.

Интернет-форумы представляют собой отличное место, где можно узнать, возникала ли подобная проблема у других людей. Также туда можно обратиться за советом. В интернете есть большие сообщества Arduino, и вы обязательно получите помощь.

Проанализируйте любые изменения, которые вы только что внесли в ранее работающее оборудование. Если возможно, отмените эти изменения и затем проверьте, решилась ли проблема.

Составьте список всех причин, которые, по вашему мнению, могли вызвать проблему, в порядке их вероятности, а затем проработайте данный список.

Если компонент все еще не работает, попробуйте задать вопрос на одном из онлайн-форумов Arduino, таких как **forum.arduino.cc**.

Эти источники информации могут быть очень полезными.

Аппаратное обеспечение

Неисправности в ваших проектах могут быть связаны либо с программным, либо с аппаратным обеспечением. Проверку аппаратных проблем можно выполнить, как описано ниже. Проблемы, связанные с программным обеспечением, мы рассмотрим далее в этой главе.

Плата Arduino

В зависимости от характера возникшей проблемы, для начала вы можете убедиться в том, что ваша плата Arduino работает и правильно подключена к компьютеру.

Неисправные или некорректные кабельные соединения часто становятся причиной возникновения аппаратных проблем. Всегда проверяйте эти подключения в первую очередь.

Проработайте следующий список.

- Убедитесь, что компьютер включен (вы удивитесь, сколько людей пренебрегает этим самым элементарным шагом).
- Подключите Arduino к компьютеру с помощью USB-кабеля.
- Проверьте индикатор **PWR** на плате. Он должен гореть мягким зеленым светом. Если это так, то он указывает, что плата Arduino подключена к компьютеру и на нее подается электропитание.

  Если индикатор **PWR** выключен или светит тускло, в первую очередь проверьте, надежно ли подключен USB-кабель к плате Arduino и к компьютеру. Затем попробуйте подключить кабель к другому порту USB, этот вариант часто помогает. Наконец, попробуйте заменить кабель.
- Если для электропитания Arduino вы используете внешний адаптер электропитания, а не компьютер, убедитесь, что адаптер работает правильно. Это можно сделать, подключив его к другому устройству. Также убедитесь, что используемый соединительный кабель хорошего качества и что он подключен к корректным контактам на плате Arduino.

Внешнее аппаратное обеспечение

К внешнему аппаратному обеспечению относятся цепи, подключенные к плате Arduino, датчики, электродвигатели и т. д. Всем им для работы необходимо электропитание, и это первое, что необходимо проверить. Если электропитание отсутствует, очень часто проблема заключается всего лишь в некачественном подключении или его отсутствии.

Замена неработоспособных компонентов — отличный способ диагностики неисправностей.

Как только вы исключили источники электропитания и соединения, определите проблему, заменяя компоненты там, где это возможно. Таким образом можно проверить датчики и электродвигатели.

Диагностика неисправностей на печатной плате требует специального тестового оборудования, такого как мультиметр (и умения им пользоваться). См. с. 94–96.

Проблемы установки

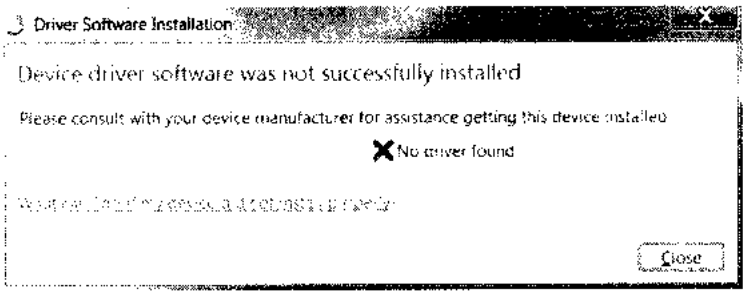
На с. 44–48 объяснено, как настраивать плату Arduino с помощью компьютера. У вас могут возникнуть проблемы, когда вы будете делать это, что во многом зависит от операционной системы, установленной на вашем компьютере.

Windows

Как правило, для пользователей операционной системы Windows не составляет труда выполнить настройку Arduino. В частности это возможно, если вы используете Windows 8 или более поздней версии — эта ОС делает это для вас автоматически.

Совет. Если у вас возникает проблема при попытке наладить связь между платой Arduino и компьютером, очень часто причиной этого является неработающий драйвер.

Если же ваш компьютер работает под управлением операционной системы Windows 7, Windows Vista или Windows XP, то, скорее всего, у вас возникнут проблемы. Эти операционные системы могут испытывать сложности при поиске и установке драйвера Arduino, что проявляется в виде следующего сообщения об ошибке (или подобного ему).



Диспетчер устройств Windows позволяет вам управлять всем подключенным к вашему компьютеру аппаратным обеспечением, включая плату Arduino.

Решением в данной ситуации служит установка драйвера вручную. Вы можете сделать это следующим образом.

1. Откройте меню **Start** (Пуск) и щелкните мышью по пункту **Control Panel** (Панель управления). Затем откройте компонент **Device Manager** (Диспетчер устройств). В его окне перечислено все оборудование, установленное на вашей системе, включая плату Arduino.

2. Покрутите список вниз, пока не увидите надпись **Arduino**; если вы не видите этой надписи, она может находиться под надписью **Ports** (Порты). Рядом с ней может быть указан восклицательный знак, оповещающий, что плата была установлена неправильно.

3. Щелкните правой кнопкой мыши по строке **Arduino** и выберите пункт **Update Driver Software** (Обновить драйвер) в контекстном меню. Затем выберите пункт **Browse My Computer for Driver Software** (Выполнить поиск драйверов на этом компьютере).

4. Нажмите кнопку **Browse** (Обзор) и перейдите к каталогу *C:\Program Files (x86)\Arduino\Drivers*. Здесь вы найдете файл *Arduino.int*. Выберите его и нажмите кнопку **Next** (Далее). Операционная система Windows выполнит установку драйвера.

Теперь между вашей платой Arduino и компьютером должно быть установлено соединение.

Также вы можете ввести слово Arduino в служебной программе поиска Windows. Вы сможете найти ее в меню **Start** (Пуск).

macOS

Процедура настройки Arduino в операционной системе macOS Snow Leopard, Leopard, Mountain Lion, Lion и более поздних версий представляет собой аналогичный процесс и не должна вызвать затруднений.

Однако, как и с более ранними версиями операционных систем Windows и Linux, настройка Arduino в более ранних версиях операционной системы macOS может привести к возникновению проблем.

Программное обеспечение Arduino для macOS поставляется в zip-архиве, и при распаковке или разархивации файлов может произойти повреждение.

Иногда вы получите сообщение об ошибке и таким образом будете осведомлены, какого рода проблема возникла. Но часто сообщение может отсутствовать.

В любом случае, попробуйте распаковать архив с помощью другой программы для работы с архивами. Таких программ множество. Хорошими примерами являются программы 7-Zip и WinRAR.

Другой возможной проблемой может быть устаревшая версия Java на вашем компьютере. В этом случае вы должны получить сообщение об ошибке; если это произойдет, просто загрузите последнюю версию программы с веб-сайта Java.

Если вы получите сообщение об ошибке Link (dyld), решением в данной ситуации будет обновление до самой актуальной версии macOS. Ранние версии содержат несовместимые системные библиотеки.

Если вы рассматриваете вариант использования платы Arduino с операционной системой Linux, лучшее, что можно посоветовать, — это использовать самую актуальную версию. Это поможет вам избежать множества проблем, которые возникли бы с ранними версиями.

Linux

Как мы упоминали на с. 47, настройка Arduino в ранних версиях операционных систем Linux может стать очень сложной задачей, а для описания решений проблем в этой книге не хватит места.

В то же время существует целый ряд онлайн-ресурсов, где можно найти помощь, например **www.linux.org**.

Ошибки синтаксиса

Плата Arduino представляет собой компьютер, хотя и достаточно простой. Как и все компьютеры, он не обладает способностью думать сам, а всего лишь реагирует на инструкции, которые задает ему пользователь. Эти инструкции (код скетчей) должны быть написаны так, чтобы компьютер их распознал. В противном случае он просто проигнорирует их и не предпримет никаких действий.

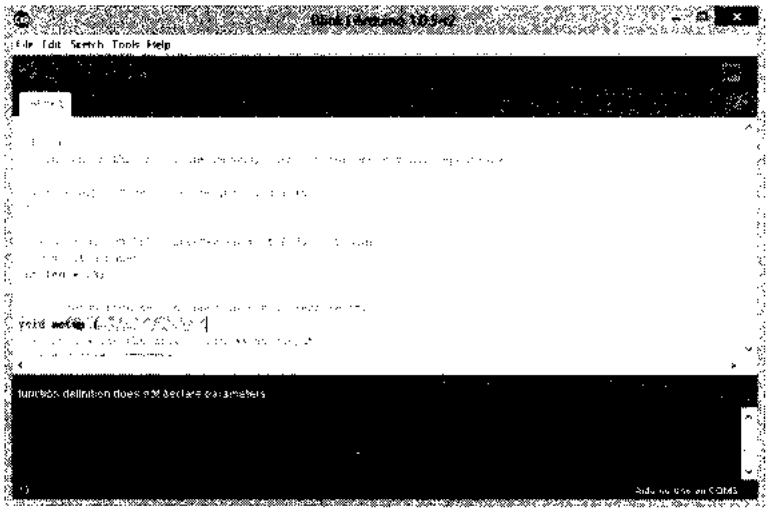
Правила, которые определяют способы написания программного кода, называются синтаксисом, и вам необходимо изучить их, чтобы программировать

вашу плату Arduino. Если вы не сделаете этого, вы не сможете написать скетч, который будет принят процессом проверки Arduino или компилирующей программой.

Синтаксис — это термин, используемый для описания набора правил оформления программного кода.

Неправильный синтаксис — вероятно, самая распространенная ошибка, которую допускают новички, и она всегда приводит к одному результату — процесс проверки просто прекращается, когда обнаруживается ошибка.

К счастью, компилирующая программа сообщает вам, где именно находится ошибка, а также тип возникшей ошибки. Рассмотрим пример, приведенный ниже.



К типичным синтаксическим ошибкам относятся: пропуск точки с запятой в конце инструкции, отсутствие открывающих/закрывающих скобок и ошибки/опечатки в именах команд.

Компилирующая программа обнаружила, что отсутствуют скобки в конце функции `void setup`. Данная функция должна иметь вид `void setup()`. Программа выделила строку и на панели в нижней части окна указала, что параметр для функции из строки 13 не был объявлен.

На заметку

Описание ошибки часто непонятно и может ничего не значить для новичка.

Это вся информация, которая вам необходима, чтобы исправить ошибку.

Монитор порта

Монитор порта — это инструмент среды разработки Arduino, позволяющий поддерживать связь с платой Arduino через последовательной порт. С его помощью вы можете отправлять и принимать данные. Полученные данные можно просмотреть на компьютере, как вы узнали в главе 9.

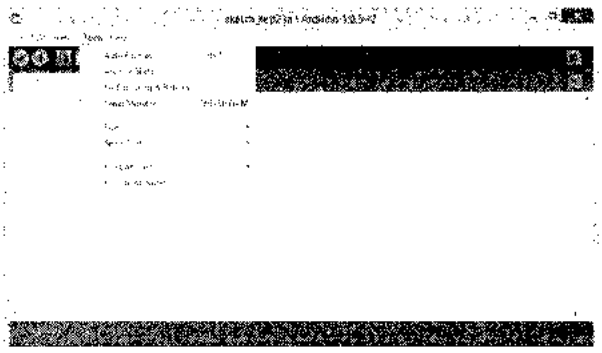
Способность отправлять/принимать данные и, в частности, просматривать полученные данные превращает монитор порта в очень полезный инструмент отладки. С его помощью вы можете анализировать код на наличие ошибок и быстро их исправлять.

Прежде чем вы сможете его использовать, вам необходимо подключить свою плату Arduino к компьютеру с помощью USB-кабеля. Затем вам необходимо открыть окно **Serial Monitor** (Монитор порта). Это можно сделать тремя различными способами.

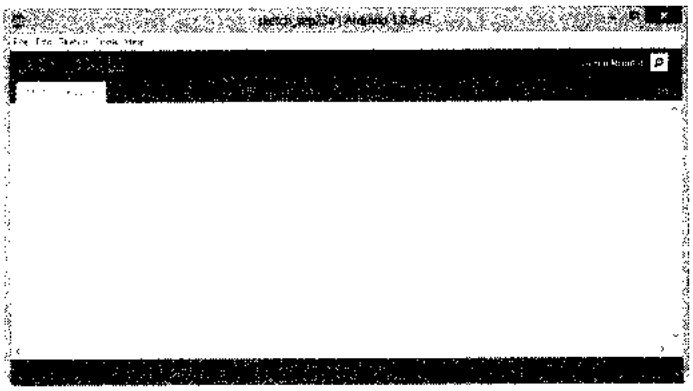
На заметку

Окно **Serial Monitor** (Монитор порта) используется для отладки скетчей Arduino и для просмотра данных, переданных работающими скетчами.

1. Откройте среду разработки Arduino и из меню в верхней части выберите команду **Tools → Serial Monitor** (Инструменты → Монитор порта).



❷ Щелкните мышью по вкладке **Serial Monitor** (Монитор порта) на панели инструментов.



Фактически окно **Serial Monitor** (Монитор порта) представляет собой отдельный терминал.

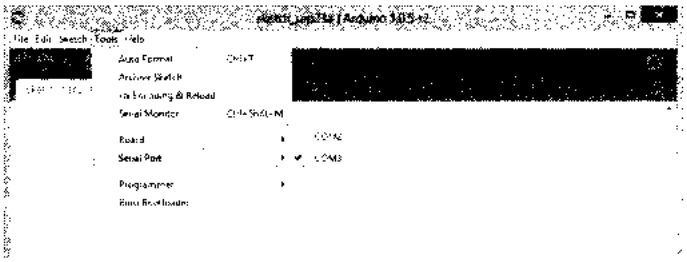
❸ На клавиатуре нажмите сочетание клавиш **Ctrl+Shift+M**.

Какой бы из способов вы ни выбрали, появится окно **Serial Monitor** (Монитор порта), показанное ниже.

Сочетание клавиш **Ctrl+Shift+M** предоставляет быстрый способ открытия окна **Serial Monitor** (Монитор порта).

Если же окно **Serial Monitor** (Монитор порта) не появилось, проблема может заключаться в том, что Arduino использует неправильный порт. Запустите инструмент **Device Manager** (Диспетчер устройств) в операционной системе Windows (см. с. 221) и щелкните мышью по треугольнику слева от пункта **Ports (COM & LPT)** — Порты (COM & LPT). Так вы узнаете, какой порт операционная система присвоила вашей плате Arduino.

Вернитесь к среде разработки Arduino и выберите команду меню **Tools → Serial Port** (Инструменты → Последовательный порт).



Когда вы запускаете окно **Serial Monitor** (Монитор порта), каждый загруженный в данный момент скетч автоматически перезагрузится. Обратите на это внимание.

Убедитесь, что выбран именно тот порт, который присвоен вашей плате Arduino в окне **Device Manager** (Диспетчер устройств). Теперь окно **Serial Monitor** (Монитор порта) должно открыться.

Передача текста на плату Arduino через окно **Serial Monitor** (Монитор порта) осуществляется путем ввода текста в поле в верхней части окна и нажатием кнопки **Send** (Отправить).

Обратите внимание, что инструмент **Serial Monitor** (Монитор порта) не сможет отправлять/принимать данные, если вы не включили в скетч соответствующий код, т. е. не сообщили скетчу, что это необходимо выполнить — см. с. 229.

На заметку

Вы должны сообщить инструменту **Serial Monitor** (Монитор порта), что вы хотите отправлять/принимать данные, включив в скетч соответствующий код.

В нижней части окна **Serial Monitor** (Монитор порта) вы увидите три параметра.

- **Скорость, бит/с** — это скорость, с которой инструмент **Serial Monitor** (Монитор порта) получает и передает данные на плату Arduino. Значение по умолчанию равно 9600 бод.
- **Определение конца строки** — этот раскрывающийся список предоставляет вам четыре варианта, с помощью которых вы можете установить метку конца строки, которую отправляет инструмент **Serial Monitor** (Монитор порта). Помимо пункта **New line** (Новая строка), вы можете выбрать варианты: **No line ending** (Не найден конец строки), **Carriage return** (Возврат каретки) и **Both NL & CR** (Оба варианта).

  «Возврат каретки» и «Новая строка» — это ASCII-символы, которые отправляются нажатием клавиши **Enter** на клавиатуре. «Возврат каретки» указывает, что курсор вернется в начало строки, тогда как «Новая строка» определяет, что курсор будет перемещен к началу новой строки.
- **Autoscroll** (Автопрокрутка) — установите этот флажок, если вы хотите, чтобы в скетче всегда были видны последние строки. В противном случае, вам придется выполнять прокрутку кода вручную.

Хотя инструмент **Serial Monitor** (Монитор порта), предоставляемый платой Arduino, и полезен, существуют не менее удобные альтернативы. К ним относятся:

На заметку

При установлении связи с Arduino вы не ограничены инструментом **Serial Monitor** (Монитор порта). Существует ряд программ сторонних разработчиков, которые можно использовать.

Processing — бесплатное приложение для операционной системы Windows, macOS и Linux;

CoolTerm — бесплатное приложение для операционной системы Windows, macOS и Linux;

puTTY — приложение с открытым исходным кодом для операционной системы Windows и Linux;

CuteCom — приложение с открытым исходным кодом, которое работает в операционной системе Linux.

Отладка

Прежде чем использовать инструмент **Serial Monitor** (Монитор порта) для устранения неисправностей в скетче, или для отладки (как еще называют этот процесс), вы должны задействовать его и сообщить, каких действий вы от него ожидаете. Это означает, что необходимо добавить два фрагмента кода в скетч, который следует проанализировать.

Первый фрагмент кода задействует монитор порта и выглядит следующим образом:

```
Serial.begin();
```

Функция `Serial.begin()` представляет собой одну из множества функций, доступных во встроенной библиотеке `Serial` в Arduino. В скобках необходимо ввести желаемую скорость передачи данных (информации). 9600 — это стандартное значение, составляющее примерно 1000 символов в секунду. Код будет выглядеть так:

```
Serial.begin(9600);
```

На заметку

Ваш скетч должен вызвать функцию `Serial.begin()`, прежде чем он сможет использовать монитор порта. Обычно этот код помещают внутрь функции `setup()`.

Какую бы скорость передачи информации вы ни ввели в скобках, убедитесь, что такая же скорость выбрана в окне **Serial Monitor** (Монитор порта) — если скорости будут отличаться, в окне **Serial Monitor** (Монитор порта) вы увидите лишь непонятный набор символов.

Второй фрагмент кода важен с точки зрения отладки. Он имеет следующий вид:

```
Serial.println()
```

На заметку

Если выбранная скорость передачи данных не соответствует значению, указанному в коде вашего скетча, символы в окне **Serial Monitor** (Монитор порта) будут нечитабельны.

Функция `Serial.println()` сообщает монитору порта о необходимости отобразить текст в окне **Serial Monitor** (Монитор порта) (без него окно **Serial Monitor** (Монитор порта) откроется, но останется пустым).

Если в скобках вы укажете инструкцию, например вывод датчика, функция `Serial.println()` отобразит это значение в окне **Serial Monitor** (Монитор порта). Это позволит вам сразу же увидеть, корректно ли значение. Если значение не отображается, вы будете знать, что существует проблема, а также ее приблизительное расположение.

Для отладки вы можете использовать функцию `Serial.println()` следующим образом: если вы понятия не имеете, в каком именно фрагменте кода возникла проблема, вы можете просто поместить множество экземпляров функции `Serial.println()` на протяжении всего кода скетча, а затем, когда скетч запущен, просто следить за окном **Serial Monitor** (Монитор порта).

Если же вы подозреваете ошибку в определенном разделе или строке кода, вы можете устранять неисправности избирательнее.

На заметку

Библиотека `Serial` в Arduino содержит ряд функций, которые могут помочь при отладке кода.

11

Проекты Arduino

Чтобы увидеть, что можно сделать с платой Arduino достаточно посмотреть на проекты, созданные другими людьми. В этой главе мы покажем вам небольшую подборку таких проектов. Конечно же, вы можете найти гораздо больше подобного материала в интернете.

- **Введение**
- **GSM-сигнализация**
- **Светодиодный куб**
- **Беспроводное интернет-радио**
- **Газонокосилка с дистанционным управлением**
- **Twitter-пекарня**
- **Робот-древолаз**
- **Беспроводной музыкальный ночник**

Введение

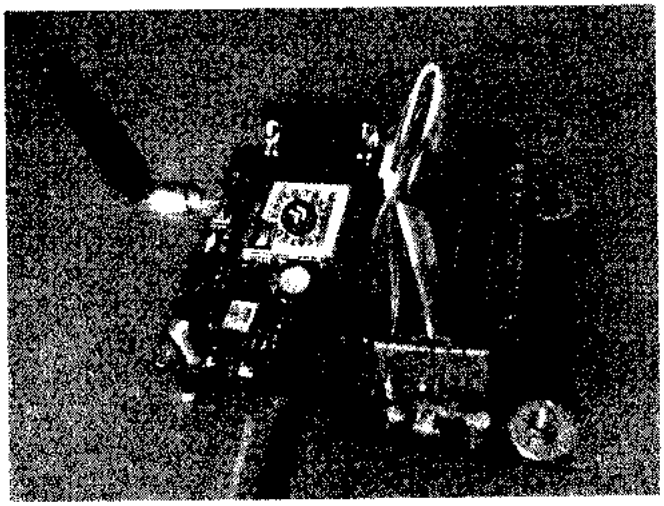
В предыдущих главах мы попытались поделиться с вами всеми знаниями и умениями, которые необходимы для создания электронных цепей, а также использования вашей платы Arduino в полном объеме. Сейчас мы продемонстрируем несколько реальных проектов, созданных на базе Arduino. Они не только показывают, что можно сделать с помощью этой платы, но также демонстрируют весь спектр навыков, необходимых для создания сложных проектов Arduino.

На заметку

Большая часть проектов Arduino требуют ряда навыков. К ним относится программирование, создание электронных цепей, знание сенсорных интерфейсов, создание корпусов и т. д.

GSM-сигнализация

Цель данного проекта — создание охранной сигнализации с большим радиусом действия, которую можно использовать дома (или где-либо еще с такой же целью).



Она состоит из платы Arduino Uno, стандартного GSM/GPRS-модема на базе чипсета SIM900A и детектора проникновения (к примеру, инфракрасного

датчика приближения или датчика освещенности). Система питается от блока питания постоянного тока напряжением 12 В или от батареи.

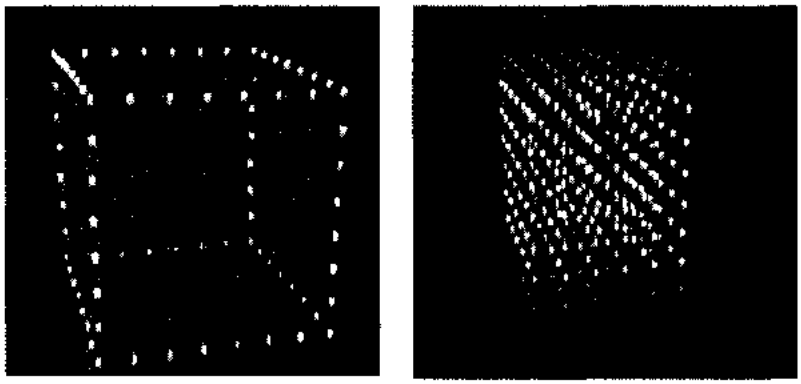
Когда система срабатывает при попытке проникновения, на указанный в коде номер мобильного телефона отправляется SMS-сообщение. Также система оборудована функцией «тревожный звонок», которая инициирует телефонный звонок при активации. С помощью нее генерируется предупреждение «пропущенный вызов».

*Благодарим **www.electroschematics.com** за разрешение разместить проект «GSM-сигнализация» в этой книге.*

Светодиодный куб

Сооруженный из 512 светодиодных индикаторов, а также большого количества интегральных схем с регистром сдвига, резисторов, транзисторов, конденсаторов и многих метров проводов, светодиодный куб, тем не менее, не служит никакой полезной цели.

Однако это интересный объект, предлагающий почти неограниченное количество световых комбинаций, которые окажутся довольно занимательными — детям, к примеру, понравится световое шоу, которое можно устроить с его помощью.



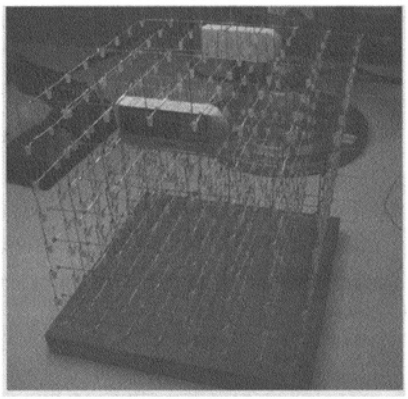
Светодиодный куб состоит из восьми слоев светодиодных индикаторов, расположенных на равном расстоянии и соединенных с помощью провода диаметром 0,22 мм. Благодаря пространству между светодиодными индикаторами, мы можем видеть их все на восьми слоях одновременно, что придает кубу эффект трехмерности.

На заметку

Если вы решили собрать это устройство, помните, что вам понадобятся сотни светодиодных индикаторов. Их можно купить в интернете по очень низким ценам. Однако качество таких диодов может быть довольно посредственным. Если вы не хотите постоянно заменять неисправные светодиодные индикаторы, мы советуем вам приобретать только качественные компоненты.

Это классический проект с применением платы Arduino, требующий использования ряда различных навыков. Вам не только необходимо знать, как программировать плату Arduino, вы также должны понимать, как проектировать и создавать электрические цепи.

Кроме того, каждый слой должен быть таким же, как предыдущий, а это требует очень точной сборки.



Совет

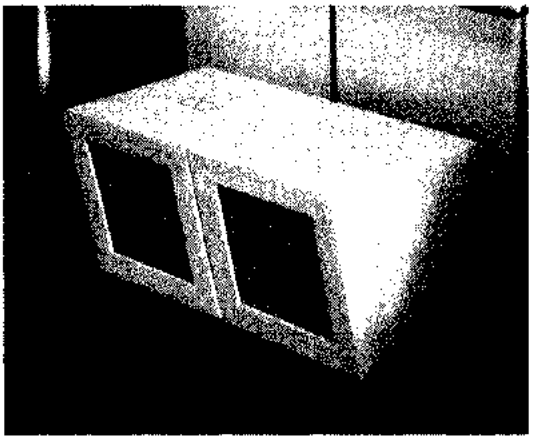
Практическая сторона этого проекта, т. е. создание и укладка слоев в жесткую конструкцию, довольно сложна. При создании куба гораздо меньших размеров, 2 × 2 × 2 сначала рекомендуется сделать все необходимые расчеты.

Благодарим Калеба Пику за разрешение разместить проект «Светодиодный куб» в этой книге.

Беспроводное интернет-радио

Проект Skube, разработанный в Копенгагенском институте интерактивного дизайна, входит в состав более крупного модуля по созданию осязательного пользовательского интерфейса. Он демонстрирует, насколько полезной может быть Arduino для прототипирования и разработки.

На заметку. Skube — это портативное беспроводное интернет-радио. Его можно подключить к учетной записи Last.fm пользователя и загрузить музыку, а также использовать платформу Spotify для поиска и воспроизведения аудиозаписей.



Два устройства Skube подключены друг к другу

Мотивацией для разработки устройства Skube было осознание того, что с развитием тенденции прослушивания цифровой музыки в интернете, современные портативные музыкальные плееры не приспособлены для этой среды и, следовательно, непригодны.

Добавьте к этому тот факт, что процедура обмена музыкой в общественных местах не является ни удобной, ни простой. Особенно когда у людей такие разные музыкальные пристрастия.

Результатом явился музыкальный плеер, который позволяет взаимодействовать с цифровыми музыкальными сервисами, такими как Spotify, без необходимости использования компьютерных устройств. Всеми возможностями можно управлять непосредственно с устройства Skube.

Когда один Skube подключается к другому, их плейлисты перемешиваются. Если вам нравится композиция, записанная в памяти Skube вашего друга, просто нажмите кнопку в виде сердца, чтобы добавить музыку в свой плейлист.

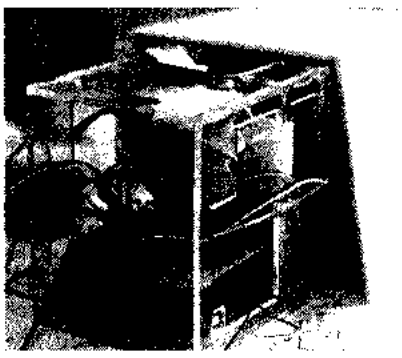
Каждое устройство Skube предоставляет два режима — «Плейлист» и «Поиск». Мы можем выбрать один из режимов, просто коснувшись верхней части Skube. Режим «Плейлист» воспроизводит композиции с помощью Skube, а режим «Поиск» выполняет поиск композиций, похожих на те, которые уже записаны в памяти Skube. Устройство с легкостью помогает отыскать новую музыку, которая придется вам по вкусу.

Когда два устройства Skube подключены друг к другу, они работают как единый проигрыватель, в котором все дорожки из плейлистов перемешаны. Это позволяет управлять несколькими Skube с одного устройства.

Интерфейс разработан таким образом, чтобы быть интуитивно понятным и простым в использовании. Перевернув Skube, вы измените режимы, прикосновение вызовет воспроизведение, либо пропуск композиции, а если перевернуть Skube на лицевую сторону произойдет отключение устройства.

Сердце системы — плата Arduino, установленная во всех устройствах Skube. Также внутри установлен модуль Xbee, который обеспечивает устройства беспроводной связью. Он позволяет устройствам Skube обмениваться данными друг с другом.

Skube содержат ряд датчиков, реагирующих на различные действия, с помощью которых пользователь управляет устройством (например, легкое постукивание по поверхности устройства, переворот и т. д.). Входные данные с этих датчиков поступают на Arduino, она интерпретирует их и отправляет необходимые команды остальным частям системы.



Эти устройства также содержат шилд FM-радио. Назначение данного модуля заключается в проигрывании музыки.

Модули беспроводной связи Xbee, которые упоминались ранее, также используются для подключения и управления Skube с помощью компьютера.

На заметку

Модули беспроводной связи Xbee задействуют сетевой протокол IEEE802.15.4 для быстрого сетевого соединения типа «точка-группа точек» или «точка-точка».

Это можно сделать с помощью визуального языка программирования под названием Max, который был специально разработан для музыки и мультимедиа.

Два известных музыкальных сервиса, Spotify и Last.fm, предоставляют специальные интерфейсы программирования приложений (API). Max извлекает данные из этих API и использует их для обеспечения работы функций «Плейлист» и «Поиск» в Skube.

На заметку

Сетевой API-интерфейс Spotify позволяет приложениям извлекать данные об исполнителях, альбомах и композициях непосредственно из каталога Spotify. API также обеспечивает доступ к данным пользователя, например плейлистам и музыке, сохраненной в его библиотеке.

Проект Skube требует таких навыков, как программирование, постройка цепей, беспроводная связь, обмен данными с внешними устройствами и создание корпуса.

Благодарим Эндрю Спица за разрешение разместить проект Skube в этой книге.

Газонокосилка с дистанционным управлением

Кому-то нравится косить траву. Другие люди ненавидят это занятие. Если вы из последних, почему бы не решить проблему? Если вам интересно, как именно, изучите этот проект.

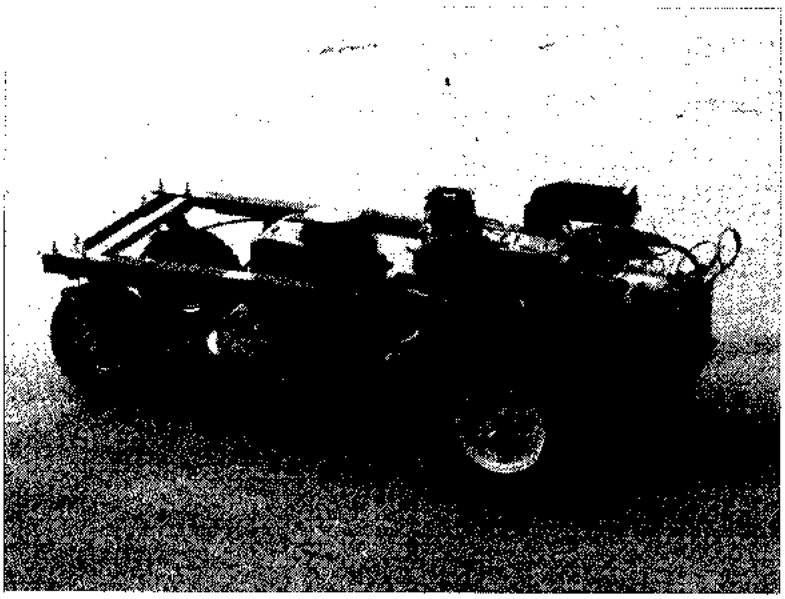
На заметку

Газонокосилка Lawnbot400 станет хорошим испытанием ваших навыков работы с механическими и электронными устройствами.

Lawnbot400 представляет собой дистанционно управляемую газонокосилку, которая состоит из следующих компонентов:

- газонокосилка с колесами и без ручки управления;
- два 12 В аккумулятора для обеспечения электропитания напряжением 24 В;

- прочный металлический каркас и поддон для крепления механизмов газонокосилки и аккумуляторов;
- два электродвигателя для перемещения косилки;
- передатчик и приемник дистанционного управления, с помощью которых можно управлять газонокосилкой;
- электронные компоненты, включая плату Arduino.



Металлический каркас и колеса — основная часть проекта. Конструкция требует хороших навыков работы с механикой, не говоря уже о необходимых инструментах. Здесь нет строгих и жестких правил — изобретательность будет вашим лучшим другом.

Как только каркас будет готов, вам потребуется установить электродвигатели. Можно использовать любые — те, которые установлены на Lawnbot400, были взяты с инвалидного кресла. Какие бы электродвигатели вы ни использовали, их положение должно быть регулируемым, чтобы вы могли настроить натяжение приводной цепи. Для этого потребуется изготовить установочную пластину.

Скорость газонокосилки регулируется специально созданным для этой цели контроллером электродвигателя. Он подает на электродвигатели переменное напряжение. Сам контроллер управляется с помощью сигнала широтно-импульсной модуляции (ШИМ), передаваемого платой Arduino. Контроллер электродвигателя преобразует значения ШИМ 0–5 В в напряжение 0–24 В постоянного тока, которое подается на электродвигатели.

Следующий этап — это установка косилки на каркас. Опять же это будет проверкой ваших навыков работы с механикой и смекалки.

Заключительный этап по сборке данной механической конструкции — установка аккумуляторов. Вес аккумуляторов велик, поэтому их установка за задними колесами значительно улучшит контроль над механизмом, поскольку батареи будут играть роль противовеса.

Управлять газонокосилкой довольно просто. Переместите левый рычаг управления вверх, и левое колесо начнет двигаться вперед. Переместите правый рычаг управления назад, и правое колесо станет двигаться назад. Переместите оба рычага вперед, и газонокосилка поедет вперед. Lawnbot400 может разворачиваться с радиусом, равным нулю.

Безопасность — очень важный фактор в работе над этим проектом.

Газонокосилка Lawnbot400 — весьма небезопасный механизм. По этой причине очень важно, чтобы каркас и крепления косилки создавались по высоким стандартам качества.

На данном этапе есть возможность улучшить Lawnbot400. Например, мы могли бы полностью автоматизировать ее, встроив систему GPS и датчики. Также можно было бы подключить электродвигатель к ведущему валу косилки, чтобы автоматически заряжать аккумуляторы.

Чтобы пользователь не потерял газонокосилку, в нее встроен предохранитель. Он представляет собой еще одну плату Arduino, которая управляет силовым реле на 60 А. Этот предохранитель отключает электропитание, подаваемое на контроллер электродвигателя, если сигнал, подаваемый с Arduino, станет очень слабым.

Наконец, на передатчике есть аварийный блокиратор, который прекращает подачу электропитания на электродвигатели, если возникает такая необходимость.

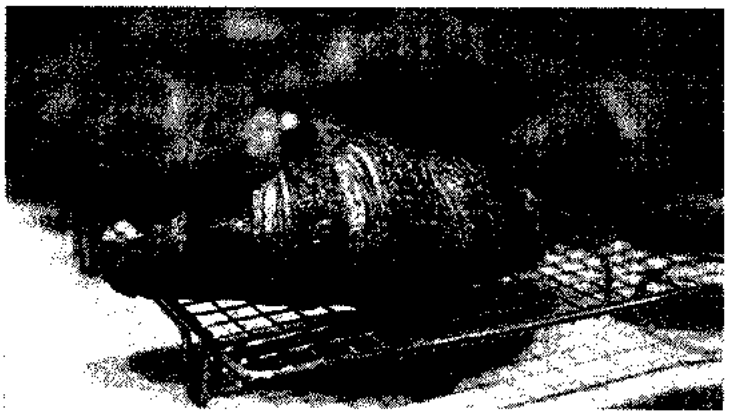
Благодарим Джона-Дэвида Уоррена за разрешение разместить проект Lawnbot400 в этой книге.

Twitter-пекарня

Выпечку и другие хлебобулочные изделия приятнее всего покупать, когда они еще свежие — только из печи. Помня об этом неоспоримом факте, ребята из английского офиса компании POKE решили сделать так, чтобы в их любимой пекарне (которая находится через дорогу) была возможность немедленно сообщать им о готовности своей продукции.

На заметку

Система Baker Tweet помогла поместить кафе Albion Bakery на карту.



Для этого они спроектировали и создали настенное Wi-Fi-устройство Baker Tweet с большой ручкой регулировки, изготовленной из прорезиненного пластика. В нем установлена плата Arduino, а также элементы управления, позволяющие сообщить о прибытии свежеиспеченной продукции в Twitter, как только она будет готова.

Все, что необходимо сделать пекарю — это повернуть ручку регулировки в необходимое положение и нажать кнопку в центре прибора. Все пользователи, которые подписаны на Twitter-канал пекарни, получат уведомление о свежей выпечке.

На заметку

Существует множество потенциальных применений для устройства с возможностью подключения к сетям Wi-Fi, которое может автоматически публиковать обновления в социальных сетях типа Twitter.

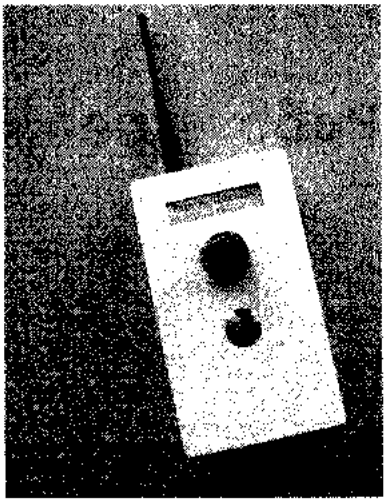
Устройство Baker Tweet было установлено в кафе Albion на Баундари-стрит в лондонском районе Шордитч и стало очень хорошим подспорьем в бизнесе. Другие пекарни, без сомнения, воспользуются этим изобретением для получения дополнительной прибыли.

Однако выгоду здесь получить могут не только пекарни. Baker Tweet позволяет любой компании общаться с покупателями в режиме реального времени через Twitter.

При авторизации через веб-интерфейс (которым комплектуется устройство) вы можете отправлять информацию о предложениях, ценах и наличии товара с настенного устройства, которое достаточно прочно, чтобы выдержать суровые кухонные условия, а в управлении гораздо проще, чем ноутбук или смартфон.

Устройство Baker Tweet состоит из следующих компонентов:

- Плата Arduino Duemilanove
- Шилд Arduino Ethernet
- Шилд Ladyada Proto
- Wi-Fi адаптер беспроводной связи Linksys
- Соответствующие электронные компоненты, проводка, ручка, переключатель, корпус и т. д.



На заметку

Устройство Baker Tweet было собрано на основе платы Arduino Duemilanove. Однако вам также не составит труда создать подобное устройство на базе платы Arduino Uno.

Одна из особенностей Baker Tweet заключается в том, что для простоты использования вся соответствующая информация доступна на веб-сайте, созданном с помощью системы управления контентом.

Это означает, что пользователь может войти в свою учетную запись на сайте **bakertweet.com** и указать информацию о своей продукции, например количество, наименование и т. д.

После этого информация на устройстве обновляется путем простого поворота ручки в положение **Update Items List** (Обновить список продукции) и нажатия кнопки.

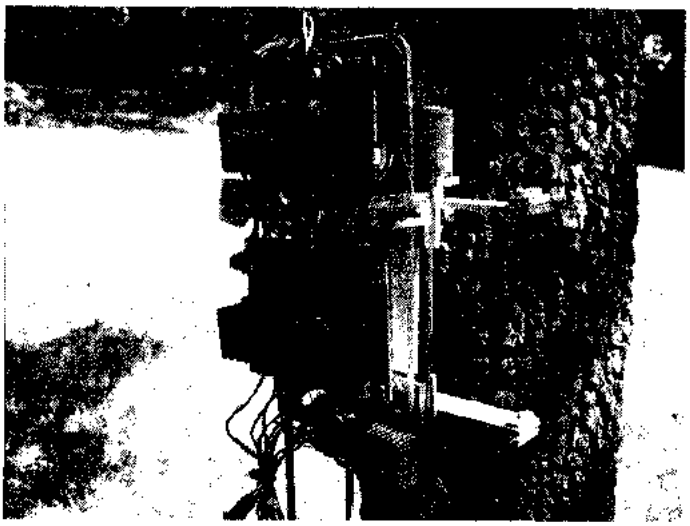
*Благодарим команду POKE (**www.pokelondon.com**) за разрешение разместить проект Baker Tweet в этой книге.*

На заметку

Django — система управления контентом с открытым исходным кодом, которая распространяется бесплатно и используется для публикации контента в интернете.

Робот-древолаз

Многие проекты Arduino представляют собой всего лишь эксперименты, позволяющие увидеть, возможно ли воплотить идею в жизнь, — они могут не служить никакой практической цели, хотя в действительности могут представлять собой трудную задачу. Робот, взбирающийся на дерево — типичный пример такого проекта.



Все, что может выполнять данное устройство — это взбираться вверх по стволу дерева, а затем спускаться вниз, как показано на рисунке ниже. Однако

добиться этого вовсе не так просто, как кажется, и от вас потребуются все ваши навыки в проектировании, электронике и механике.

На заметку

Робот имеет четыре пары ног. Каждая пара управляется непосредственно отдельным электродвигателем.

Этот проект требует выполнения большого количества первоначальных плановых и проектных работ. Для этого существует множество подходящих программ, но в данном случае использовалась программа под названием SketchUp.

Требовалось спроектировать следующее.

Робот, состоящий из двух сегментов, соединенных позвоночным стержнем, который может растягиваться и сокращаться. Оба сегмента имеют четыре ноги с очень острыми концами наподобие когтей.

Для осуществления подъема ноги на верхнем сегменте стягиваются, и концы ног впиваются в кору, закрепляя положение робота. Затем позвоночный стержень сокращается, подтягивая нижний сегмент. Ноги на нижнем сегменте сцепляются с деревом, а верхний сегмент освобождает захват. Наконец, позвоночный стержень растягивается, толкая верхний сегмент вверх. Затем процедура повторяется.

На заметку

Принцип подъема подобен передвижению гусеницы.

Основные компоненты робота:

- Плата Arduino Uno
- Контроллер электродвигателя
- Батареи и регулятор мощности
- Четыре электродвигателя
- Аппаратное обеспечение для конструирования робота

Контроллер электродвигателя изготовлен специально для данного проекта, и разработан на базе чипа L298HN Dual Full Bridge. Он может двунаправленно управлять четырьмя электродвигателями постоянного тока, по одному для каждой из ног.

Робот получает электропитание от двух разных источников. Плата Arduino и логические схемы контроллера электродвигателя питаются от батареи 9 В, тогда как электродвигатели рассчитаны на электропитание от литий-ионного аккумулятора 12 В.

На заметку

По той причине, что для вращения резьбового позвоночного стержня используется стандартный электродвигатель постоянного тока, а не сервопривод, или шаговый электродвигатель, роботу может быть неизвестна степень растяжения позвоночного стержня. Таким образом, для предотвращения чрезмерного растяжения позвоночного стержня используются концевые выключатели.

Робот имеет четыре пары ног, каждая пара управляется одним электродвигателем. Ноги сделаны из отрезков алюминиевого стержня. Основной каркас, который удерживает всю конструкцию, также сделан из алюминия.

Чтобы двигаться вверх и вниз, робот растягивается и сокращается путем вращения резьбового стержня, который закреплен на верхнем сегменте. Когда стержень вращается по часовой стрелке, два сегмента стягиваются и отодвигаются друг от друга при вращении против часовой стрелки. Резьбовой стержень управляется электродвигателем 12 В.

К слову об электронике, датчики вращения — ключевой фактор для функционирования робота. Для каждого электродвигателя в роботе используется отдельный датчик, поэтому робот определяет точное местоположение каждой ноги в любой момент времени, позволяя таким образом осуществлять точное управление.

Проводка используется для подключения электродвигателей, контроллера электродвигателя, датчиков, регулятора и переключателей.

На заметку

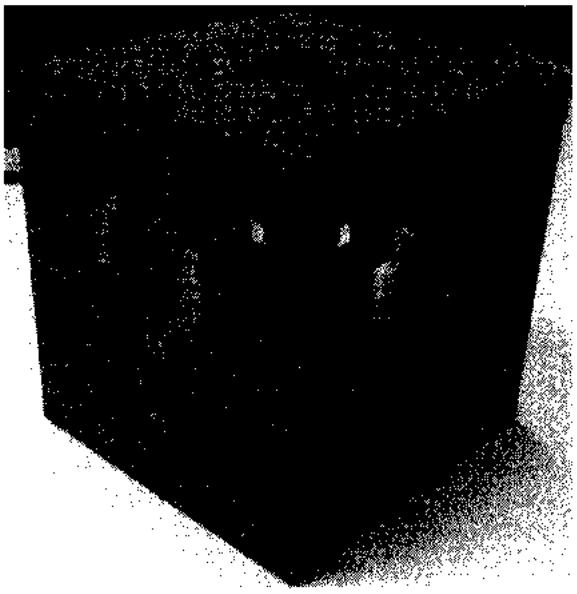
К данным функциям относятся: `closeTop()`, `closeBottom()`, `openTop()`, `openBottom()`, `Lift()` и `Push()`. Комбинируя эти функции в нужном порядке, можно сделать так, чтобы робот поднимался и опускался.

Что касается программирования, в данном проекте для каждого из основных действий робота была написана отдельная функция — см. примечание.

Благодарим Бена Кетца за разрешение разместить проект «Робот, взбирающийся на дерево» в этой книге.

Беспроводной музыкальный ночник

8BitBox — портативное Bluetooth-устройство, управлять которым можно с помощью любого Android-устройства с поддержкой Bluetooth, такого как смартфон или планшет. С помощью 8BitBox вы можете удаленно воспроизводить и переключать музыкальные композиции. Также благодаря встроенному светодиодному индикатору устройство служит превосходным ночником.



На заметку

Этот проект поможет вам детально разобраться, как дистанционно управлять проектами на основе Arduino через Bluetooth-интерфейс.

Корпус сам по себе интересен, поскольку был напечатан с помощью 3D-принтера (хотя вы можете создать его из любых других материалов). 3D-печать представляет собой нечто абсолютно новое для большинства людей.

Чтобы рассеивать свет от светодиодного индикатора и создать приглушенный световой эффект, во внутренней части со всех четырех сторон корпуса прикреплена веленевая бумага.

Если использовать RGB-светодиод, вы сможете настраивать цвет свечения светодиодного индикатора.

К электронной начинке проекта относится плата Arduino в роли контроллера, протошилд Arduino для размещения компонентов, RGB-светодиод для реализации подсветки, пьезозуммер для воспроизведения звука, модуль EZ-Link для поддержки Bluetooth-соединения и зарядная цепь LiPo для контроля заряда аккумулятора.

После сборки все компоненты помещаются в корпус. Для активизации устройства следует нажать кнопку на верхней стороне устройства.

*Благодарим компанию Adafruit (**www.adafruit.com**) за разрешение разместить проект 8BitBox в этой книге.*