Знакомство с Arduino

От переводчика (figley.migley, arduino.shopium.ua):

- ***Если вы доселе не держали в руках Arduino, вам стоит это прочесть!***
- *Текст взят из файла Getting Started with Arduino.chm, гуляющего по интернету.*
- *В книге описана работа с платой Arduino Duemilanove, но вы вожете применять примеры с любой из плат Arduino, просто внимательно читайте описание выводов вашей платы.*
- *Перевод не литературный, не было стилистической правки, возможно шероховатости. Если вы хотите улучшить текст, присылайте исправления и замечания, буду рад!*

Скачать перевод в форматах FB2, EPUB, MOBI: http://www.rulit.net/tag/computers/znakomstvo-s-arduino-perevod-knigi-getting-started-with-arduino-download-free-67091.html

---

Содержание:

Глава 1. Введение

Arduino - физическая вычислительная платформа и открытым исходным кодом, основанная на просто плате ввода-вывода и среда разработки, которая использует язык Processing (www.processing.org). Arduino может применяться для разработки самостоятельных интерактивных обьектов или может быть связана с программой на вашем компьютере (такой как Flash, Processing, VVVV, или Max/MSP). Платы могут быть собраны самостоятельно или куплены уже собранными; среда разработки (далее IDE) может быть загружена бесплатно с сайта www.arduino.cc

Arduino отличается от других платформ на рынке тем, следующими возможностями:

- Это многоплатформенная среда, она может работать на Windows, Macintosh, и Linux.
- Она основана на IDE языка Processing, лёгкой в использовании среде разработки, для использования художниками и дизайнерами.
- Она программируется через кабель USB, а не через последовательный порт. Это полезно, так как многие современные компьютеры не имеют последовательных портов.
- Это открытые аппаратное и программное обеспечение - если хотите, вы можете скачать схему, купить все компоненты и сделать всё сами, без оплаты разработчикам Arduino.
- Компоненты недорогие. Плата USB стоит около €20 (в настоящее время около US$35) и замена сгоревшей микросхемы на плате легка и стоит не более €5 или US$4. Так что вы можете ошибаться.
- Существует активное общество пользователей, так что вам может помочь большое число людей.
- Проект Arduino развивался как образовательный и поэтому он отлично подходит начинающим для быстрого обучения.

Эта книга была создана для того чтобы начинающие поняли премиущества, которые они могут получить от изучения возможностей применения, а также для того, чтобы они поняли её философию.

1.1 Целевая аудитория

Эта книга была написана для "настоящих" пользователей Arduino: дизайнеров и художников. Поэтому она пытается объяснять вещи таким образом, который может свести некоторых инженеров с ума. Вообще-то, один из них назвал вступительные главы моего первого проекта отстоем. В этом-то и дело. Посмотрим правде в глаза: большинство инженеров не в состоянии объяснить то, что они делают другим инженерам, не говоря уже об обычных людях.

Примечание: Arduino опирается на тезисную работу Эрнандо Баррагана, которую он сделал на платформе Wiring во время учебы под началом Casey Reas и меня в IDII Ivrea.

После того, как Arduino начала становиться популярной, я понял что экспериментаторы, люди с хобби и хакеры всех видов начали использовать её для создания прекрасных и безумных вещей. Я понял, что вы все художники и создатели в собственном праве, так что эта книга для вас.

Arduino был создан для изучения дизайна взаимодействия - дисциплины конструирования, которая ставит прототипирование в центр методологии. Существует множество определения дизайна взаимодействия, но я предпочитаю следующее:

1.1.1 Конструирование взаимодействия - это дизайн любого опыта взаимодействия

В сегодняшнем мире дизайн взаимодействия касается значимого опыта между нами (людьми) и обьектами. Это хороший способ изучить возможность создания красивых и, может быть, даже спорных опытов между нами и технологией. Дизайн взаимодействия предлагает разработку путем интерактивного процесса, основанного на прототипах все возрастающей верности. Такой подход - также часть некоторых типов "обычной" разработки - может быть расширен до включения прототипирования с помощью технологии; в частности, прототипирование в электронике.

Конкретной областью дизайна взаимодействия, которой занимается Arduino, являются физические вычисления (или физический дизайн взаимодействия).

1.2 Что такое физические вычисления?

Физические вычисления используют электронику для прототипирования новых материалов для дизайнеров и художников.

Он включает разработку интерактивных объектов, которые могут общаться с людьми с применением датчиков и исполнительных механизмов, управляемых поведеним, которое реализовано в виде программного обеспечения, запущенного в микроконтроллере (небольшом компьютере на одной микросхеме).

В прошлом использовать электронику означало всё время иметь дело с инженерами и создавать

один маленький компонент в то-же время. Эти вопросы отделяли творческих людей от игр с окружающей средой напрямую. Большинство инструментов были предназначены для инженеров и требовали обширных знаний. В последние годы микроконтроллеры стали более дешевы и легки в применении, позволяя создавать более лучшие инструменты.

Прогресс, который мы сделали с Arudino, должен приблизить эти инструменты на шаг ближе к новичкам, позволяя людям начать создание вещей после двух-трёх дней учёбы.

С Arduino дизайнер иди художник может узнать основы электроники и сенсоров очень быстро и может начать создавать прототипы с очень маленькими вложениями.

Глава 2. Путь Arduino

Философия Arduino основана на создании проектов вместо разговоров о них. Это постоянный поиск более быстрых и ярких способов строить лучшие прототипы. Мы изучили множество способов прототипирования и создали способы мышления с применением рук.

Классический инжиниринг полагается на строгий процесс получения А из Б, а прелесть пути Arduino - возможность уйти с этого пути и вместо него получить В.

Это процесс рукоделия, который мы так любим - играть с окружающей средой в бесконечном поиске и находить неожиданное. В этом поиске путей постройки лучших прототипов мы также выбрали ряд программных пакетов, которые обеспечили этот процесс постоянного манипулирования средой программного и аппаратного обеспечения.

Следующие несколько разделов представляют несколько философий, событий и пионеров, которые вдохновили на Путь Arduino.

2.1 Прототипирование

Прототипирование - сердце пути Arduino: мы делаем вещи и создаём обьекты, которые взаимодействуют с другими обьектами, людьми и сетями. Мы стремимся найти более простой и быстрый путь создания прототипа наиболее дешёвым способом.

Многим новичкам, который знакомятся с электроникой в первый раз, кажется что они должны научиться строить всё с нуля. Это пустая трата энергии: что вам надо, так это очень быстро просто удостовериться в том, что что-то работает, так-что вы сможете мотивировать себя предпринять следующий шаг или даже мотивировать кого-то ещё дать вам побольше денег для реализации задуманного.

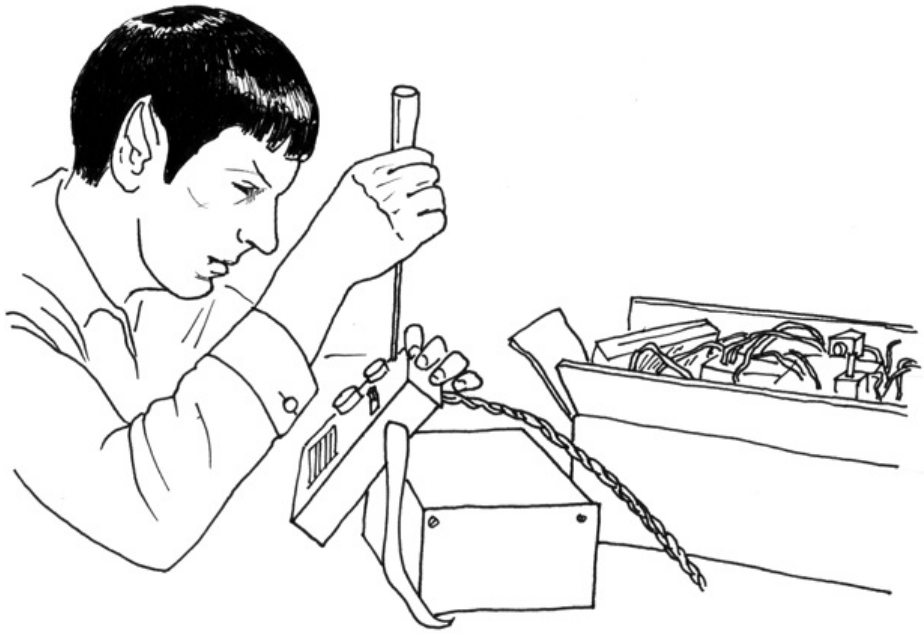
Именно поэтому мы разработали "оппортунистическое прототипирование": зачем тратить время и энергию, строить с нуля (процесс, который требует времени и глубоких технических знаний), когда можно взять готовые устройства и взломать их чтобы использовать большую работу, проделанную крупными компаниями и хорошими инженерами?

Наш герой - Джеймс Дайсон, который создал 5127 прототипов своего пылесоса прежде чем удовлетворился в том, что сделал всё как надо (www.international.dyson.com/jd/1947.asp).

2.2 Самоделкины

Мы полагаем, что для работы с технологиями важно изучать различные возможности прямо на аппаратном и программной обеспечении иногда без точно определённой цели.

Повторное применение существующей технологии - наилучший путь для самоделкина. Получить дешёвые игрушки или списанное оборудование и взломать их для того, чтобы заставить их сделать что-то новое - один из способов достижения великих результатов.



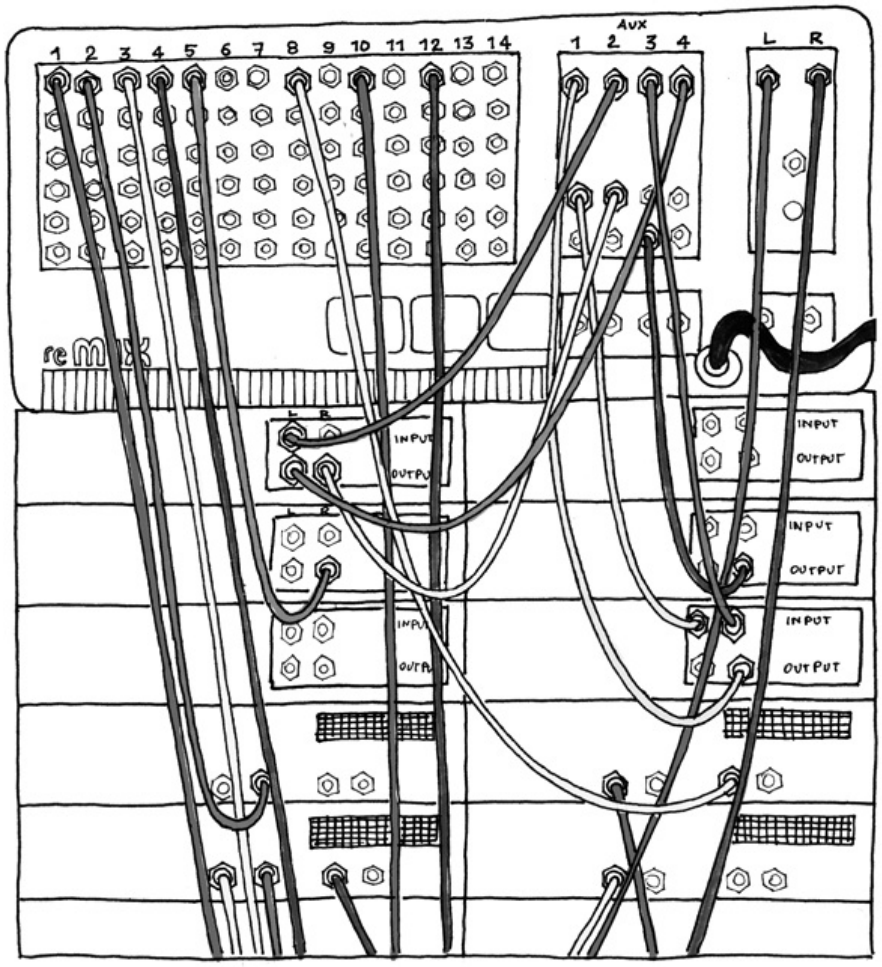
(даже мистер Спок любит Ардуино!)

2.3 Сшивание

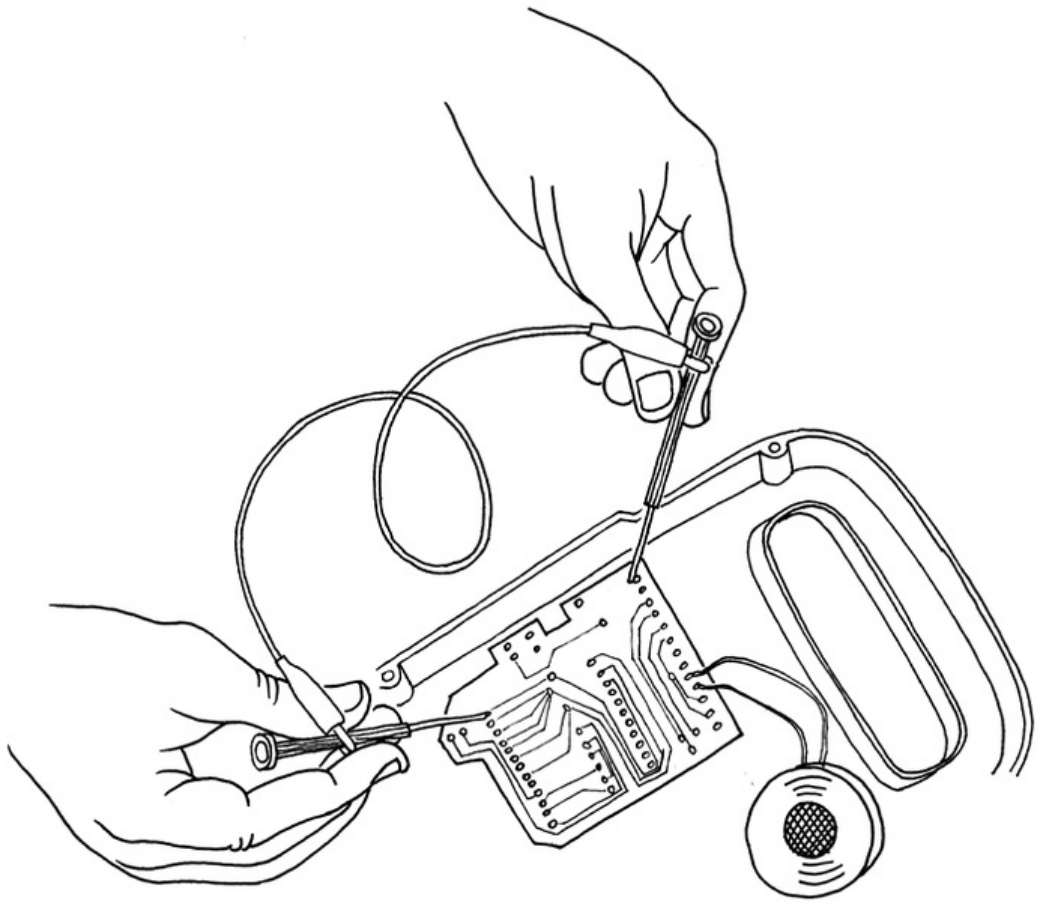
Я всегда был очарован модульностью и возможность построения сложных систем соединением простых устройств. Этот процесс очень хорошо показан Робертом Мугом и его аналоговыми синтезаторами. Музыканты создавали звуки, пробую бесконечные комбинации "сшивая" различные модули при помощи кабелей. При таком подходе синтезатор выглядит как старинный телефонный коммутатор, но в соединении со множеством кнопок, это прекрасная платформа для экспериментов со звуком и инновационной музыкой. Муг определил это как процесс между "свидетельством и открытием" . Я уверен, что большинство музыкантов не знали что делают все эти сотни кнопок, но они пробовали и пробовали, перерабатывая собственный стиль безостановочным потоком.

Снижение количества остановок потока очень важно для творчества - чем более беспрерывный процесс, тем больше получится рукоделия.

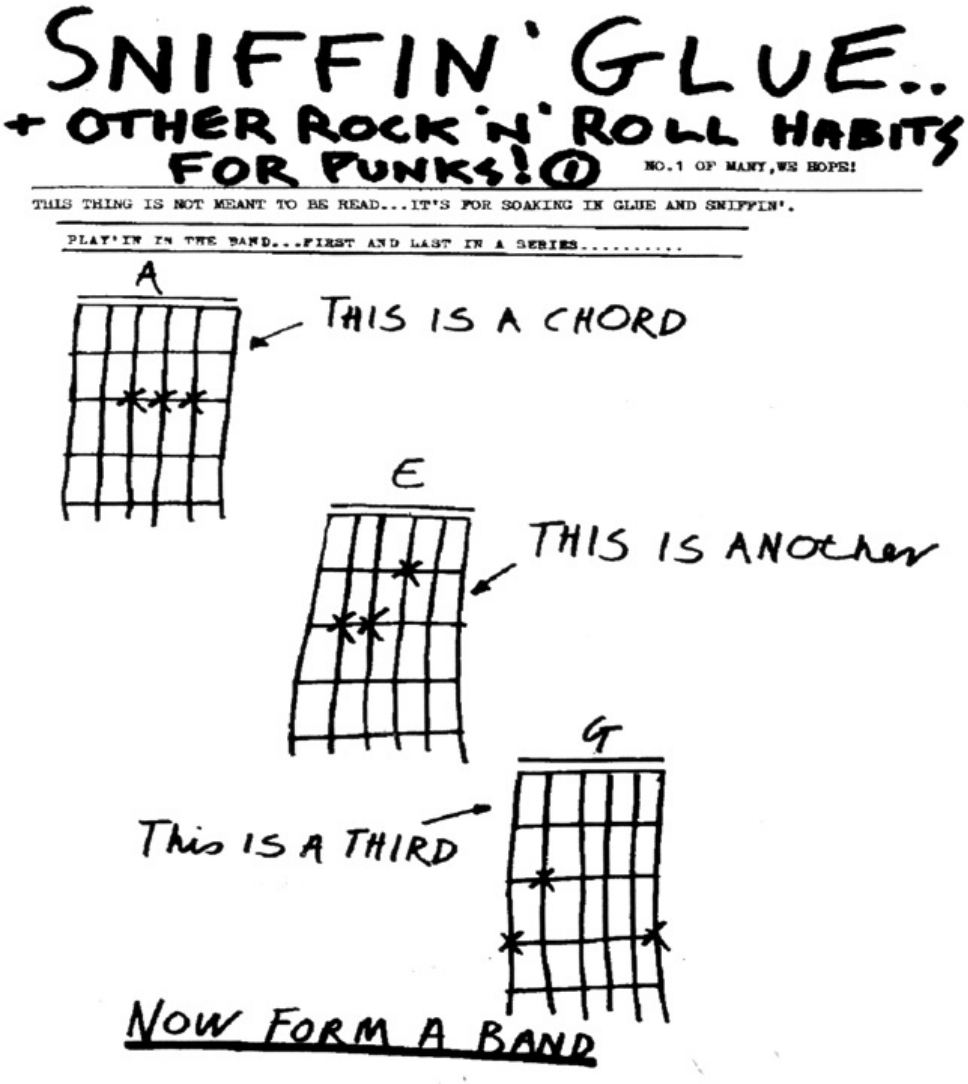
Эта техника была переведена в мир программ при помощи средств "визуального программирования" , таких как Max, Pure Data, или VVVV. Эти инструменты визуализированы как "ящики" с разной фукнциональностью, позволяя пользователю строить связи, соединяя эти ящики вместе. Эти среды позволяют пользователю экспериментировать с программированием без постоянных перерывов, типичных для обычного цикла: "ввести программу, скомпилировать, чёрт побери - тут ошибка, исправить ошибку, скомпилировать, запустить". Если вы нацеливаетесь на визуализацию, рекомендую попробовать их.



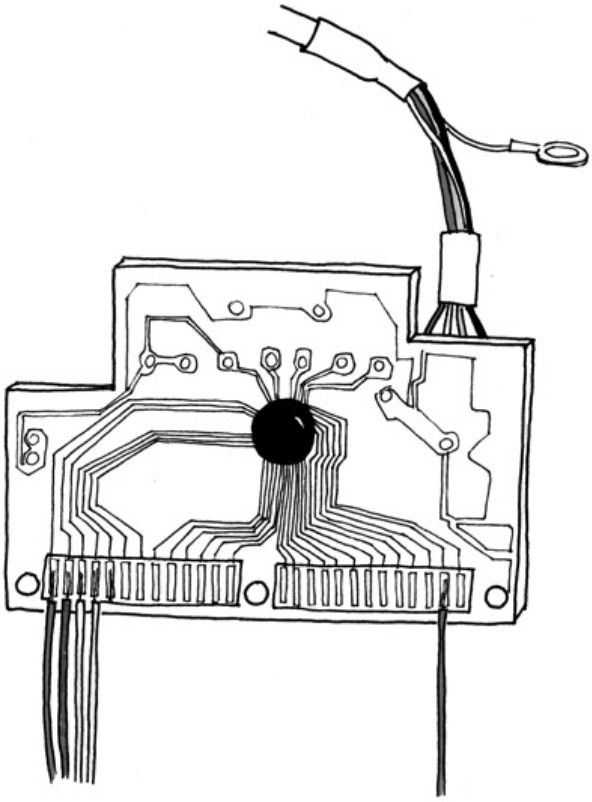
2.4 Искажение схем



Искажение схемы - одна из наиболее интересных форм творчества.. Это творческое короткое замыкание низковольтных аудиоприборов с питанием от батарей, таких как педали гитарных эффектов, детские игрушки и небольшие синтезаторы для получения новых музыкальных инструментов и генераторов звука. Сердце этого процесса - "искусство шанса". Оно было начато Ридом Газала, который случайно закоротил гитарный усилитель железякой в своём ящике стола, что вызвало поток необычных звуков. Что мне нравится в искажателях схем - это то, что они могут создать самые дикие устройства при помощи технологий без понимания что они собственно делают с теоретической стороны.



2.5 Хаки клавиатуры

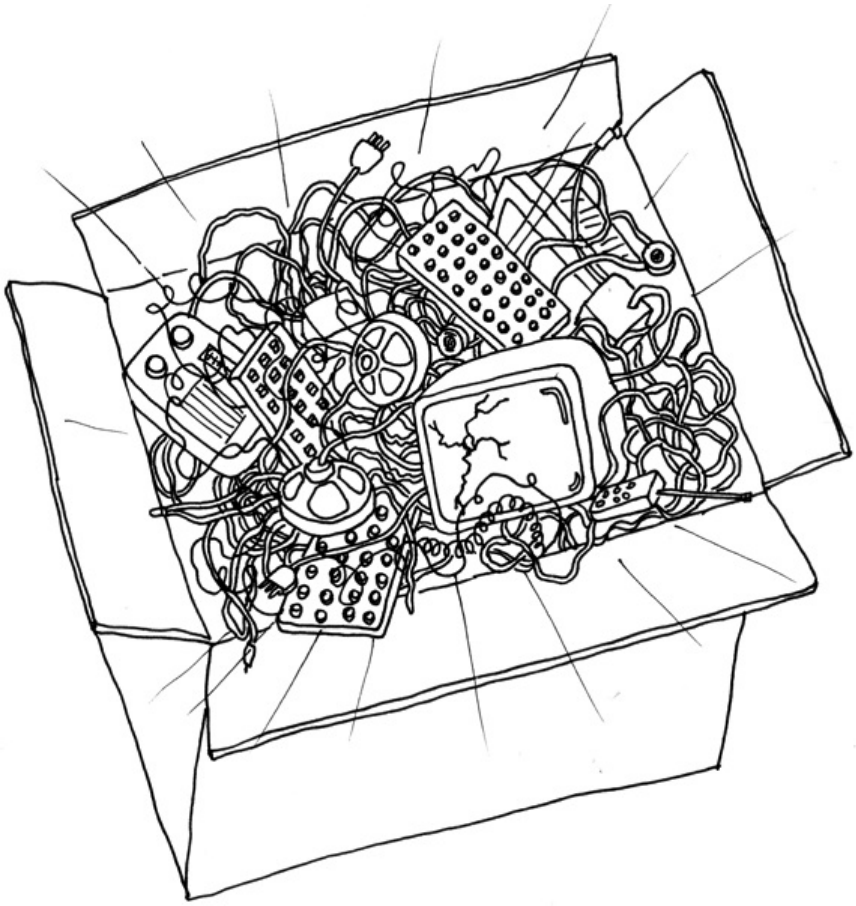


Компьютерные клавиатуры - всё ещё основной способ общения с компьютером на протяжении более 60 лет. Алекс Пентленд, академический глава MIT Media Laboratory, однажды заметил: "Извините за выражение, но мужские писсуары мужчин компьютеров. Компьютеры отделены от того, что вокруг них".

Как экспериментаторы, мы можем реализовывать новые пути общения с программами, заменяя клавиши устройствами, которые могут получать данные от окружающей среды. Клавиатура отдельно от компьютера, является простым (и дешёвым) устройством. Её сердце - маленькая платка. Обычно это дурно пахнущая зелёная или коричневая плата с двумя рядами контактов, которые идут к пластиковым прокладам, осуществляющим соединения между клавишами. Если вы снимете их и используете провод чтобы соединить контакты, на дисплее компьютера появится символ. Если вы приобретёте детектор движения и подключите его к своей клавиатуре, вы увидите, что как только кто-то пройдёт мимо компьютера, будет "нажата" клавиша. Свяжите его с вашей любимой программой, и вы сделаете свой компьютер умнее писсуара. Изучение хаков клавиатуры - ключевой кирпичик прототипирования и физических вычислений.

2.6 Мы любим мусор!

В настоящее время люди выбрасывают множество техники: старые принтеры, компьютеры, странные офисные машины, техническое оборудование и даже множество военных устройств. Это всегда было большим рынком для такой продвинутой технологии, особенно среди молодых и/или бедных хакеров, которые только начинают свой путь. Этот рынок стал очевидным в Иври, где мы разрабатывали Arduino. В городе расположен главный офис компании Оливетти. Они производят компьютеры с 1960-х годов; в середине 1990-х они выбросили всё на свалки района. Они были полны компьютерных частей, электронных компонентов и странных устройств различных видов. Мы потратили здесь бесконечные часы, выбирая все сорта приспособлений за небольшие деньги и применяя их в своих прототипах. Если вы покупаете тысячу динамиков за необльшие деньги, в конце-концов вы родите какую-нибудь идею. Накапливайте мусор и просматривайте его перед началом создания чего-нибудь с нуля.



2.7 Хакайте игрушки

Игрушки - фантастический источник дешёвых технологий для хака и повторного использования, об этом свидетельствует практика искажения схем, упомянутая ранее. При нынешнем притоке очень дешёвых высокотехнологичных игрушек из Китая вы можете быстро реализовать идеи при помощи нескольких мяукающих котов или световых мечей. Я делал это несколько лет для того чтобы заставить понять моих студентов что технология - не страшна, и не сложна в понимании. Один из моих любимых источников - это буклет "Низкотехнологичные сенсоры и актюаторы" авторов Usman Haque и Adam Somlai-Fischer. Я думаю, что они прекрасно описали эту технику в своей книге, и я использую это при любом удобном случае.

2.8 Сотрудничество

Сотрудничество между пользователями является одним из их ключевых принципов мира Arduino - через форум на www.arduino.cc люди из разных уголков мира помогают друг другу в изучении платформы. Команда Arduino побуждает людей к сотрудничеству на местном уровне, а также помогая им создавать группы пользователей в каждом городе, который они посещают. Мы также создали Wiki, названную "Детская площадка" (www.arduino.cc/playground), где пользователи документируют свои результаты. Это так удивительно видеть, как много знаний эти люди выливают в сеть для всех пользователей. Эта культура обмена опытом и помощи друг другу - одна из вещей, которыми я больше всего горжусь в связи с Arduino.

Глава 3. Платформа Arduino

Arduino состоит из двух основных частей - платы Arduino, которая является частью аппаратного обеспечения, над которым вы работаете при создании собственных обьектов; и среды разработки (IDE) Arduino - программного обеспечения, которое вы запускаете на своём компьютере. Вы используете IDE для создания скетчей (маленьких компьютерных программ), которые выгружаются на плату Arduino. Скетч говорит плате что делать.

Не так давно работа на аппаратным обеспечением означала создание схем с нуля, используя сотни различных компонентов со странными названиями, такими как резисторы, конденсаторы, индуктивности, транзисторы и тому подобными.

Каждая схема была спаяна для выполнения одного специфического приложения и изменения

требовали отрезания проводов, перепайки соединений и т.п.

С появлениям цифровых технологий и микропроцессоров эти функции, ранее реализованные проводами, заменены программами.

Программное обеспечение легче менять, чем аппаратное. Несколькими нажатиями вы можете радикально изменить логику устройства и испытать две или три её версии за то-же время, которое потребуется на перепайку нескольких резисторов.

3.1. Аппаратное обеспечение Arduino

Плата Arduino - небольшая плата микроконтроллера, состоящая из небольшой схемы, содержащей целый компьютер в маленьком чипе (микроконтроллер). Этот компьютер по крайней мере в тысячу раз менее мощный чем MacBook, на котором я пишу эту книгу, но он намного дешевле и очень полезен для постройки интересных устройств. Посмотрите на плату Arduino - вы увидите чёрный чип с 28 "ножками" - это ATmega168, сердце вашей платы.

Мы (команда Arduino разместили на этой плате все компоненты, требуемые для нормальной работы и связи с компьютером этого микроконтроллера. Существует много версий этой платы; та, которую мы описываем в книге - ArduinoDuemilanove, сама простая в использовании и наилучшая для изучения. Однако эти-же инструкции подходят к ранним версиям платы, включая последнюю Arduino Diecimila и более старую Arduino NG. На рис. 3-1 показана Arduino Duemilanove, на рис. 2 - Arduino NG.

На иллюстрациях ниже вы видите плату Arduino. Во-первых, все эти разъёмы могут немного напугать. Вот пояснение того, что делает каждый элемент платы:

3.1.1 14 контактов цифрового ввода-вывода (контакты 0-13)

Они могут быть как входами, так и выходами, что определяется вашим скетчем.

3.1.2 6 контактов аналогового входа (контакты 0–5)

Эти отдельные контакты для аналогового входа получают аналоговые значения (например, величину напряжение в датчике) и преобразовывают их в цифры от 0 до 1023.

3.1.3 Контакты аналогового выхода (контакты 3, 5, 6, 9, 10 и 11)

Шесть цифровых контактов, которые могут быть запрограммированы на аналоговый выход при помощи вашего скетча.

Плата может быть запитана от USB-порта компьютера, большинства USB-зарядных устройств, или от AC-адаптера (рекомендуется напряжением 9 вольт, разъём 2,1мм, плюс в центре). Если в разъём питания не подключён источник, плата получает питания от USB-разъёма, но как только вы подключите источник питания, она автоматически переключится на него.

Примечание: Если вы используете старую Arduino-NG или Arduino Diecimila, вам надо установить перемычку выбора питания (помеченную на плате PWR_SEL) на отметку EXT (внешнее) или USB-питание. Эта перемычка расположена между разъмёмом для AC-адаптера и USB-портом.

Рис. 3-1. Arduino Duemilanove

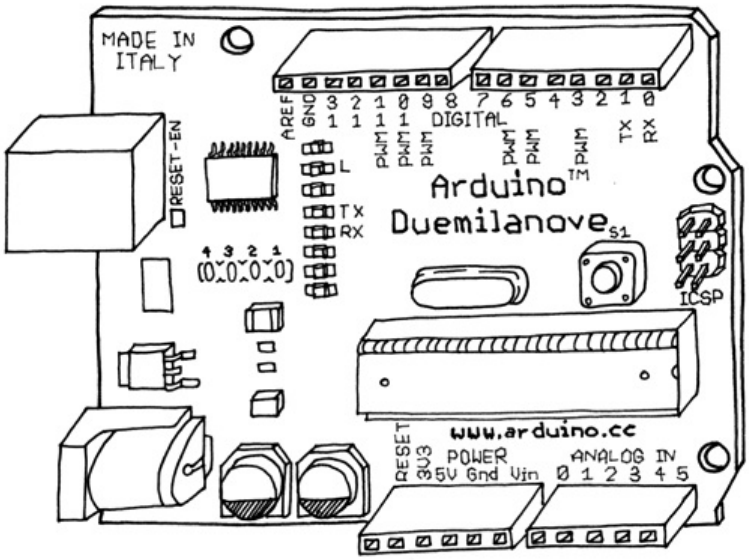
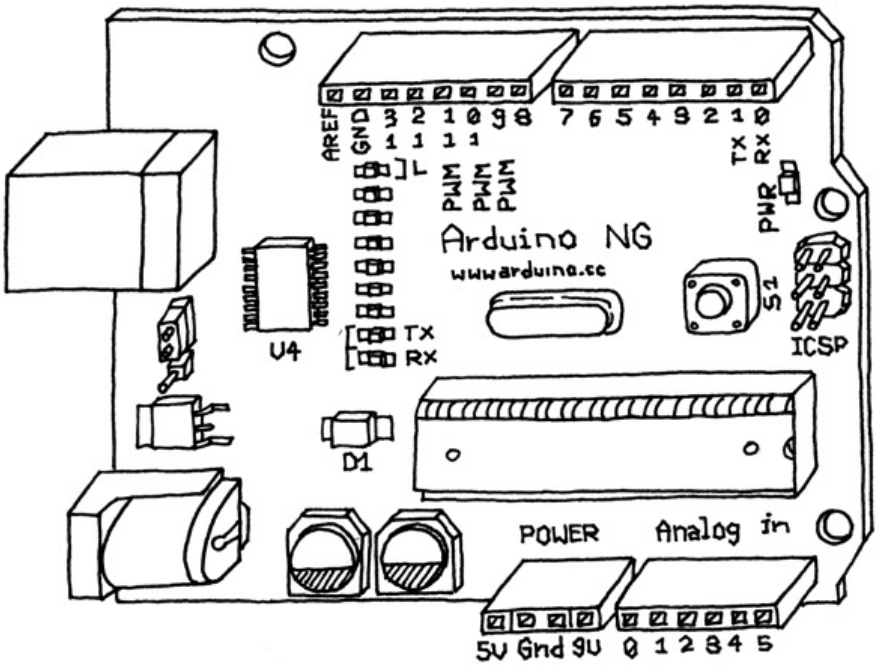


Рис. 3-2. Arduino NG



3.2 Интегрированная среда разработки (IDE)

IDE (интегрированная среда разработки) - это специальная программа, работающая на вашем копьютере, которая позволяет вам писать скетчи для платы Arduino на простом языке по образцу языка Processing. Когда вы нажмёте кнопку выгрузки скетча на плату, случится волшебство - код, который вы написали, будет транслирован в язык C (который немного сложен для начинающих), и будет передан компилятору avr-gcc, важной части открытого программного обеспечения, который и произведёт финальную трансляцию в язык, понятный микроконтроллеру. Последний шаг очень важен, так как Arduino упрощает вам жизнь, скрывая все возможны есложности программирования микроконтроллеров.

Цикл программирования Arduino упрощённо выглядит так:

- Подключите вашу плату в USB-порт своего компьютера.
- Напишите скетч, оживляющий плату
- Выгрузите этот скетч на плату через USB-соединение и подождите несколько секунд для перезапуска платы
- Плата выполнит написанный вами скетч.

Примечание: Установка Arduino на Linux в настоящее время немного усложнена. См. полные

инструкции на www.arduino.cc/playground/Learning/Linux.

3.3. Установка Arduino на ваш компьютер

Чтобы запрограммировать вашу плату Arduino, сначала вы должны скачать среду разработки (IDE) отсюда: www.arduino.cc/en/Main/Software. Выберите подходящую версию для своей операционной системы.

Скачайте файл и дважды щёлкните на нём для распаковки; он создаст папку с именем arduino-[версия], например, arduino-0012. Перетащите эту папку в любое удобное вам место: на рабочий стол, в свою папку или папку приложений (на Mac), или в папку C:\Program Files (на Windows). Теперь, когда вы захотите запустить среду разработки Arduino, откройте эту папку и дважды щёлкните на иконке Arduino. Пока-что не делайте этого, нам требуется выполнить ещё один шаг.

Примечание: Если у вас есть проблемы с запуском Arduino IDE, см. Главу 7, Устранение неполадок.

Теперь вам требуется установить драйверы, которые позволят вашему компьютеру общаться с платой через порт USB.

3.4 Установка драйверов: Macintosh

Загляните в папку Drivers внутри папки arduino-0012 и дважды щёлкните на файле FTDIUSBSerialDriver_x_x_x.dmg (x_x_x означает номер версии драйвера, например, FTDIUSBSerialDriver_v2_2_9_Intel.dmg). Дважды щёлкните на файле .dmg для того, чтобы примонтировать его.

Примечание: Если вы используете Mac на платформе Intel, такой как MacBook, MacBook Pro, MacBook Air, Mac Pro или Mac Mini на базе Intel, или iMac, удостоверьтесь что вы устанавливаете драйвер с текстом "Intel" в его имени, например FTDIUSBSerialDriver_v2_2_9_Intel.dmg. Если ваш Mac не на базе Intel, устанавливайте файл без текста "Intel" в названии.

Далее установите пакет FTDIUSBSerialDriver при помощи двойного щелчка на нём. Следуйте инструкциям, которые показывает программа установки и введите пароль администратора когда он будет запрошен. В конце процесса перезапустите машину чтобы удостовериться в том, что драйверы установлены корректно. Теперь подключите плату к компьютеру. На плате должен загореться светодиод "PWR", а жёлтый светодиод, обозначенный "L", должен начать мигать. Если этого не произошло, смотрите Главу 7, Устранение неполадок.

3.5 Установка драйверов: Windows

Подключите плату Arduino к компьютеру; когда появится окно помощника "Найдено новое оборудование", Windows попробует найти драйвер на сайте Windows Update.

Windows XP спросит вас, проверять-ли сайт Windows Update - если вы не хотите этого делать, выберите "Нет, не в этот раз" и нажмите "Далее".

На следующем экране выберите "Установить из указанного места" и нажмите "Далее".

Отметьте галочкой опцию "Искать в следующих местах", нажмите "Обзор", выберите папку, в которую вы установил Arduino, и выберите папку Drivers\FTDI USB Drivers. Нажмите "OK" и "Далее".

Windows Vista сначала попробует найти драйвер на сервере Windows Update; и если не получится, вы сможете указать папку Drivers\FTDI USB Drivers.

Вам придётся пройти эту процедуру дважды, так как сначала компьютер установит драйвер низкого уровня, а затем установит часть кода, которая заставляет плату выглядет как последовательный порт компьютера.

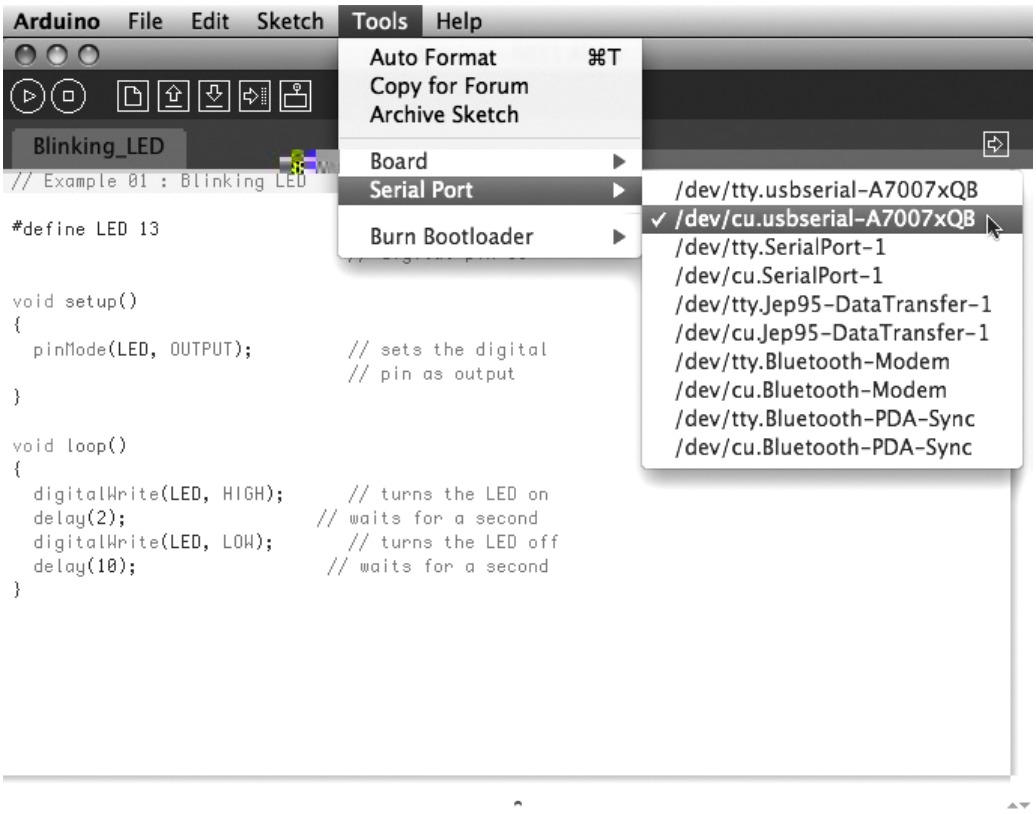
После того, как драйверы установлены, вы можете запускать Arduino IDE и начинать использовать Arduino.

Далее, вы должны узнать, какой порт назначен плате Arduino - эта информация понадобится вам в дальнейшем. Инструкции по получению этой информации следуют ниже.

3.6 Идентификация порта: Macintosh

Из меню "Tools" в среде разработки Arduino, выберите "Serial Port" и выберите порт, который начинается с /dev/cu.usbserial-; это имя, которое компьютер использует для обращения к плате Arduino. Рис 3-3 показывает список портов.

Рис. 3.3 Список портов в Arduino IDE

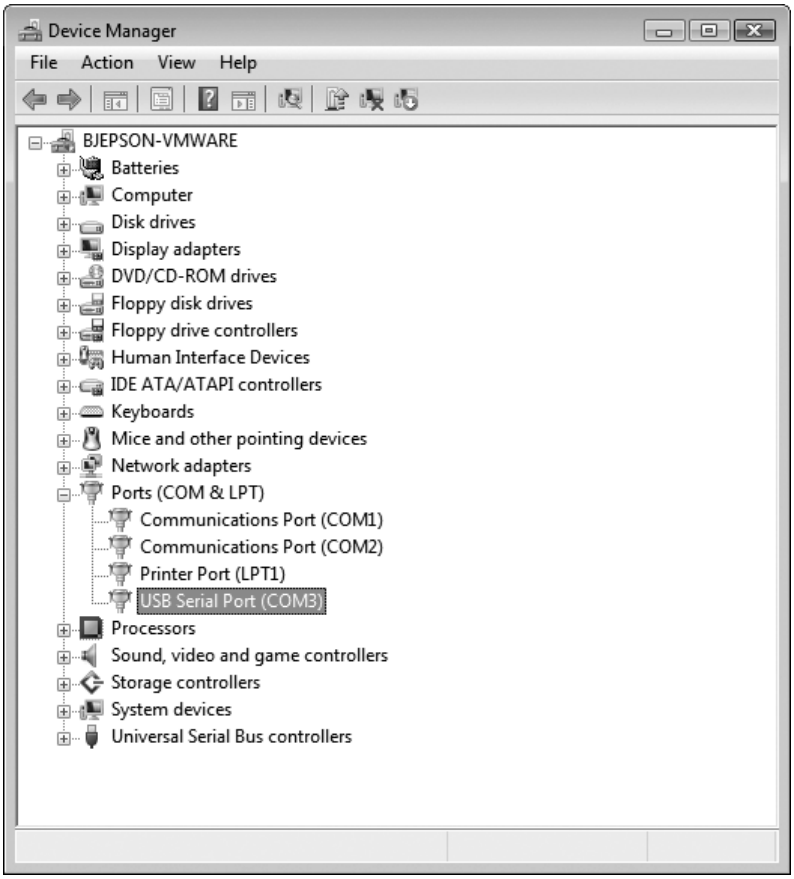


3.7 Идентификация порта: Windows

В Windows процесс чуть сложнее - по крайней мере, в начале. Откройте Диспетчер устройств: щёлкнув кнопку "Пуск", правой кнопкой щёлкните "Компьютер" (Vista) или "Мой компьютер" (XP), и выбрав "Свойства". Для Windows XP, щёлкните "Оборудование" и выберите "Диспетчер устройств". В Vista, щёлкните на "Диспетчер устройств" (в списке приложений в левой части окна).

Найдите устройство Arduino в списке "Порты (COM и LPT)". Arduino будет видна как последовательный порт USB и будет иметь имя вида COM3, как показано на рис. 3.4.

Рис. 3.4 Диспетчер устройств Windows показывает все доступные последовательные порты



Примечание: На некоторых машинах с Windows порт COM может получить номер больше 9; такая нумерация создаёт некоторые проблемы при попытках общения с Arduino. В Главе 7. Устранение неполадок описано решение этой проблемы.

После того, как вы определили COM-порт, вы можете выбрать его из меню "Tools" > "Serial Port" в IDE Arduino.

Теперь ваша среда разработки Arduino может общаться с платой Arduino и программировать её.

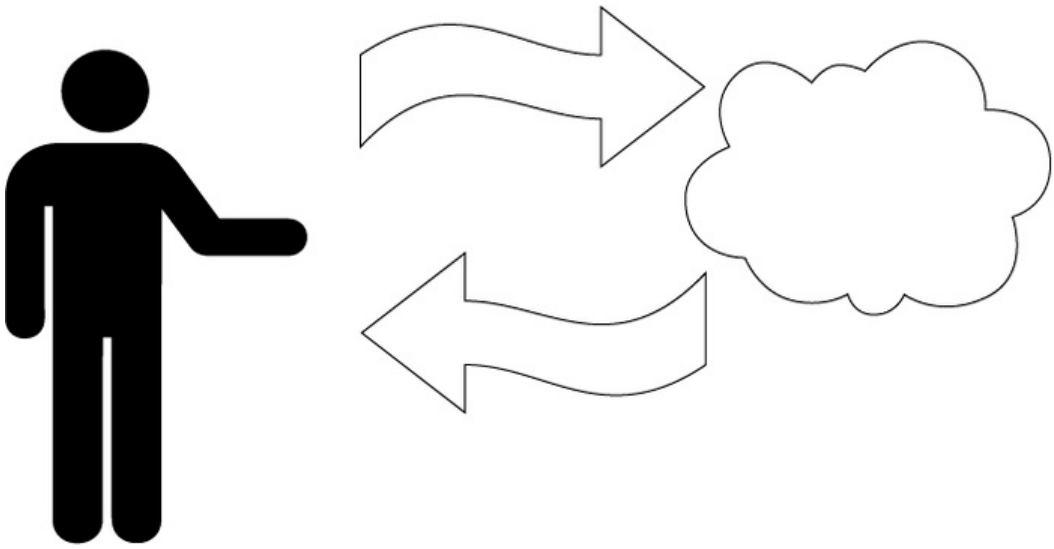
Глава 4. Знакомство с Arduino - теперь точно!

Теперь вы научитесь как создавать программу и программировать интерактивное устройство.

4.1 Анатомия интерактивного устройства

Все обьекты, которые мы построим с применением Arduino, следуют простому шаблону, который назовём "интерактивное устройство". Интерактивное устройство - это электронная схема, которая может "ощущать" окружающую среду, используя сенсоры (электронные компоненты, которые преобразуют измерения реального мира в электрические сигналы) . Устройство обрабатывает информацию, полученную от этих датчиков образом, определённым в программном обеспечении. Устройство может взаимодействовать с окружением посредством актюаторов - электронных компонентов, которые преобразуют электрические сигналы в физические действия.

Рис. 4-1. Интерактивное устройство



4.2 Сенсоры и актюаторы

Сенсоры и актюаторы - это электронные компоненты, которые позволяют электронике взаимодействовать с окружающим миром.

Так как микроконтроллер - это очень простой компьютер, он может обрабатывать только электрические сигналы (похоже на электрические импульсы, которые посылаются между нейронами в нашем мозге). Для того, чтобы определить освещённость, температуру, или другие физические величины, ему требуется что-то для преобразования их в электричество. В нашем теле, для примера, глаз преобразовывает свет в сигналы, которые отправляются в мозг при помощи нервов. В электронике мы можем использовать простой прибор, называемый светочувствительный резистор (или фоторезистор), который может измерить количество света, попадающего на него и передать это как понятный микроконтроллеру сигнал.

После того, как датчик будет прочитан, устройство имеет информацию, требуемуя для определения своего поведения. Процесс выбора решения выполняется микроконтроллером, а действие выполняется актюатором. В нашем теле, например, мускулы получают электрические сигналы от мозга и преобразуют их в движение. В мире электроники эти функции могут быть выполнены лампой или электродвигателем.

В следующих разделах вы научитесь считывать датчики различных типов и управлять разными актюаторами.

4.3 Мигание светодиодом

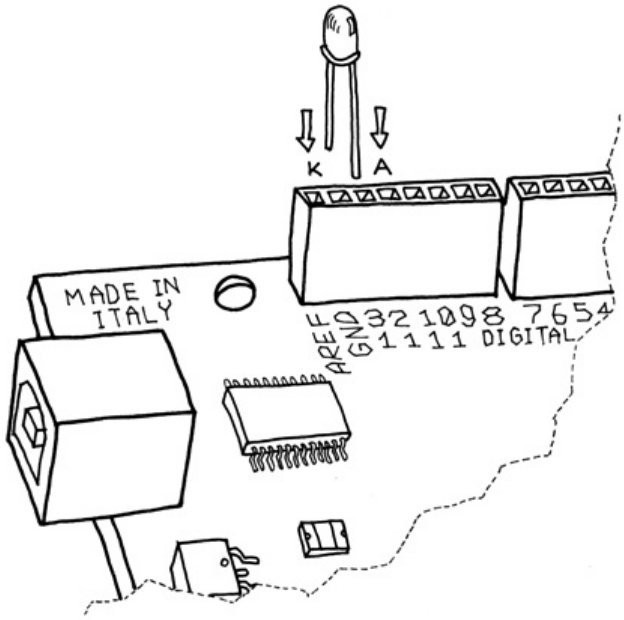
Скетч, мигающий светодиодом - первая программа, которую вам следует запустить для проверки того, что плата Arduino работает и настроена правильно. Это также самое первое программное упражнение, которое выполняет каждый изучающий программирование микроконтроллеров. Светоизлучающий диод (светодиод) - это маленький электрический компонент, который похож на лампочку, но более эффективен и требует меньшего напряжения для работы.

Ваша плата Arduino уже имеет установленный светодиод. Он обозначен "L". Вы также можете добавить свой собственный светодиод - подключите его как показано на рис. 4-2.

"K" обозначает катод (отрицательный вывод), или более короткий вывод; "A" обозначает анод (положительный вывод), более длинный вывод.

После того, как светоидод подключён, вам надо сказать плате Arduino что делать. Это делается кодом - списком команд, которые мы даём микроконтроллеру чтобы он делал то что мы хотим.

Рис. 4.2 Подключение светодиода к Arduino



Откройте папку, в которую вы поместили IDE Arduino на своём компьютере. Дважды щёлкните на значке Arduino для запуска, выберите "File" > "New". У вас запросят имя папки со скетчами: это место, где будет храниться скетч Arduino. Введите имя Blinking_LED и нажмите "OK". Затем введите следующий текст (пример 4-1) в редакторе скетчей (в главном окне IDE Arduino). Вы также можете загрузить его со страницы www.makezine.com/getstartedarduino. Он должен выглядеть как на рис. 4-3.

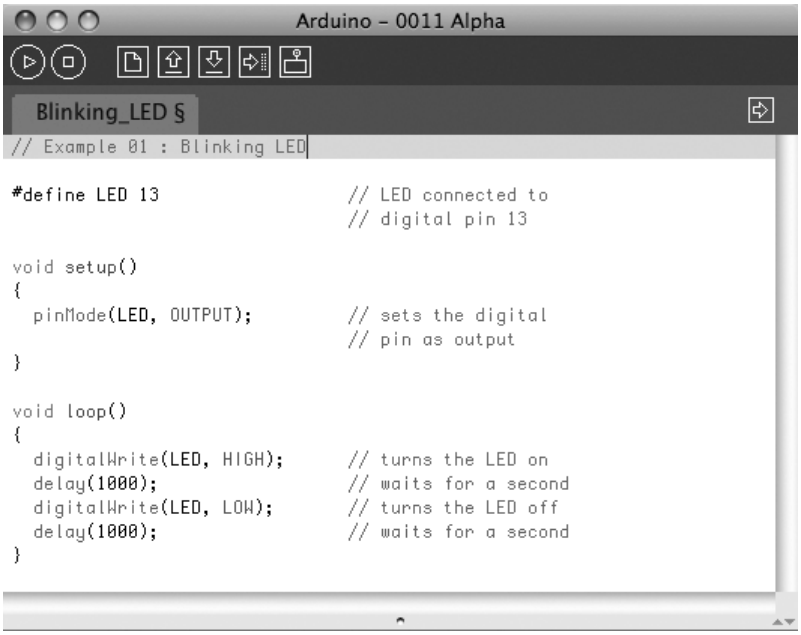
Пример 4-1. Мигающий светодиод

```
#define LED 13  // LED connected to
                // digital pin 13

void setup()
{
  pinMode(LED, OUTPUT);     // sets the digital
                            // pin as output
}

void loop()
{
  digitalWrite(LED, HIGH);    // turns the LED on
  delay(1000);                // waits for a second
  digitalWrite(LED, LOW);     // turns the LED off
  delay(1000);                // waits for a second
}
```

Рис. 4-3. Среда разработки Arduino со скетчем



Теперь, когда ваш код находится в IDE, вам надо проверить его. Нажмите кнопку "Verify" (рис. 4-3 показывает где она находится); если всё верно, вы увидите внизу окна сообщение "Done compiling". Это сообщение говорит о том, что IDE Arduino оттранслировала ваш скетч в выполняемую программу, которая может быть запущена на плате, почти как .exe-файлы в Windows или файлы .app на Mac.

Теперь вы можете выгрузить скетч на плату: нажмите кнопку "Upload to I/O Board" (см. рис. 4-3). Произойдт перезапуск платы, который заставляет плату остановить выполнение кода и слушать инструкции по порту USB. IDE Arduino отправляет текущий скетч на плату, которая сохраняет его в своей памяти и в конце концов выполняет его.

Вы увидите несколько сообщений в чёрной области внизу экрана IDE, и прямо над этой областью вы увидите сообщение "Done uploading". Это означает, что процесс выгрузки успешно завершён. На плате установлено два светодиода, обозначенные " RX" и "TX"; они мигают всякий раз при отправке или получении байта платой. Во время выгрузки они мерцают.

Если вы не видите что светодиоды мерцают, или получаете сообщение об ошибке вместо "Done uploading", значит существует проблема связи между вашим компьютером и Arduino. Удостоверьтесь что выбран верный COM-порт в меню "Tools" > "Serial Port" (см. главу 3). Также проверьте пункт меню "Tools" > "Board" - в нём должна быть выбрана верная модель Arduino.

Если вы всё ещё испытываете проблемы, см. Главу 7. Устранение неполадок.

После того, как ваш код был выгружен в Arduino, он будет оставаться в ней до тех пор, пока не будет выгруден следующий скетч. Скетч останется на месте если плата будет перезапущена или выключена, почти как на жёстком диске вашего компьютера.

Допустим что скетч был выгружен успешно, вы видите включающийся на одну секунду светодиод "L", а затем на то-же время выключающийся. Если вы установили отдельный светодиод как показано на рис. 4-2, этот светодиод также будет мигать. То, что вы только-что написали и запустили, и есть "компьютерная программа", или скетч, как называются программы Arduino. Arduino, как мы определили раньше, это маленький компьютер и она может быть запрограммирована делать то что вы хотите. Это делается при помощи написания последовательности инструкций на языке программирования в среде разработки Arduino, которая преобразует эту последовательность в выполняемый код для платы Arduino.

Далее я покажу вам как понять скетч. Во-первых, Arduino выполняет код сверху вниз, так-что первая строка сверху будет прочтена первой; затем движется вниз. ПАналогия - ползунок текущей позиции в программе-видеоплейере (QuickTime Player или Windows Media Player), он движется слева направо, показывая ваше положение в фильме.

4.4 Передайте мне пармезан

Уделите внимание фигурным скобкам, которые использованы для группировки строк кода. Они, в

частности, полезны когда вы хотите дать имя группе инструкций. Если в обед вы просите кого-нибудь "Передай мне пармезан, пожалуйста", ваша фраза описывает серию действий. Поскольку мы люди, всё это происходит естественно, но все отдельные крошечные требуемые действия должны быть сообщены плате Arduino по причине того что плата не така мощная как наш разум. Так-что для группировки команд установите **{** перед началом вашего кода и добавьте **}** после него.

Вы можете увидеть два блока кода, выделенного как мы только-что описывали. Перед каждым из них стоит странная команда

```
void setup()
```

Эта строка даёт имя блоку кода. Если вы пишете иструкцию плате для подачи пармезана, вам стоит написать

```
void passTheParmesan()
```

в начале блока и этот блок станет инструкцией, которую вы сможете вызывать из любого места в коде Arduino. Такие блоки называются функциями. После этого, если вы напишете

```
passTheParmesan()
```

в любом месте кода, Arduino выполнит эти инструкции и продолжит работу с того места, где прервалась.

4.5 Arduino не остановить

Arduino ожидает наличия двух функций - одна называется setup(), а вторая - loop().

В функции setup() вам следует располагать код, который вы хотите выполнить один раз при запуске своей программы, а loop() содержит ядро программы, которое выполняется снова и снова. Это сделано из-за того, что Arduino - не обычный компьютер - она не может выполнять много программ одновременно и программы не могут быть завершены. Когда вы подаёте питание на плату - программа запускается, когда вы хотите её остановить - просто выключите плату.

4.6 Настоящие самоделкины пишут комментарии

Любой текст, начинающийся с "//", игнорируется Arduino. Эти строки являются комментариями, т.е. заметками, которые вы оставляете в программе сами для пояснения что вы делаете, написав этот код для себя или кого-то ещё.

Очень часто (я знаю потому что и сам так делаю всё время) мы пишем код, загружаем его в плату и думаем: "ОК, я больше не буду иметь дело с этой фигнёй", и только через полгода понимаем, что в программе надо исправить баг. Теперь откройте программу, и если вы не оставили никаких комментариев, ваша мысль будет: "Блин, и с чего мне начать?". По мере нашего продвижения вперёд вы увидите некоторые трюки для того чтобы сделать свои программы более читабельными и лёгкими в обслуживании.

4.7 Код, шаг за шагом

Во-первых, вы можете посчитать подобные пояснения ненужными, почти как когда я был в школе и должен был изучать Божественную комедию Данте (каждый итальянский студент должен пройти через это, как и через другую книгу - "I promessi sposi", или "Невеста" - о, кошмар). Для каждой строфы в поэме было написано тысячи строк комменнтариев! Однако, пояснения будут более полезными когда вы начнёте писать свои собственные программы.

```
// Example 01 :Blinking LED
```

Комментарий полезен для записи небольших заметок. Предыдущий комментарий просто напоминает что это за программа - Пример 4-1, мигающая светодиодом.

```
#define LED 13 // LED connected to
               // digital pin 13
```

#define - эта директива подобна автоматическому поиску и замене в вашем коде; в данном случае она говорит Arduino вставить заменить на число 13 все слова LED в коде. Такая замена - первое, что происходит когда вы нажимаете кнопку "Verify" или "Upload to I/O Board" (вы не увидите результатов этой замены, т.к. она происходит "за сценой"). Мы используем эту команду для того, чтобы показать, что светодиод, которым мы будем мигать, подключён к 13 выводу платы Arduino.

```
void setup()
```

Эта строка говорит Arduino, что следующий блок кода будет называться setup().

{ - с такой открывающей скобкой начинается блок кода.

```
pinMode(LED, OUTPUT); // sets the digital
                      // pin as output
```

И, наконец, действительно интересная команда. pinMode сообщает Arduino как настроить отдельный вывод. Цифровые выводы могут использоваться как ВХОД (INPUT) и как ВЫХОД (OUTPUT). В данном случае нам требуется вывод для управления светодиодом, так-что мы указываем в скобках номер вывода 13 и его режим OUTPUT. pinMode - это функция, а слова (числа) в её скобках - аргументы. INPUT и OUTPUT - это константы языка Arduino (подобно переменным, константам назначены величины, только величина константы предопределена и никогда не изменяется).

} - закрывающая скобка обзначает конец функции setup().

```
void loop()
{
```

loop() - это функция, где вы определяете основное поведение вашего интерактивного устройства. Она будет повторяться снова и снова до выключения платы.

```
digitalWrite(LED, HIGH);   // turns the LED on
```

Как видно из комментария, digitalWrite() может включить (или выключить) любой вывод, настроенный как ВЫХОД (OUTPUT). Первый аргумент (в данном случае, LED) указывает какой вывод должен быт ьвключён или выключен (помните, LED - это константа со значением, которое указывает на вывод 13, так-что переключаться будет именно он). Второй аргумент может включить вывод (HIGH) или выключить его (LOW).

Представьте себе, что каждый вывод - это крошечная электрическая розетка, такая как те что есть на стенах вашей квартиры. У европейцев там 230 вольт, у американцев - 110 вольт, а Arduino работает с 5 В. В этот момент и происходит волшебство - когда программное обеспечение превращается в аппаратное. Когда вы пишете digitalWrite(LED, HIGH), эта функция подаст на вывод 5 вольт, и если вы подключите к нему светодиод, он загорится. Итак, в этом месте вашего кода инструкция программы влияет на физический мир посредством управления потоком электричества на выводе. Включение и выключение вывода по желанию даёт нам возможность перевести это в что-то видимое для человека; светодиод - наш актюатор.

```
delay(1000);       // waits for a second
```

Arduino имеет очень простую структуру. Поэтому, если вы хотите чтобы всё происходило с определённой регулярностью, вы говорите: "сиди тихо и ничего не делай до тех пор, пока не придёт время следующего шага". delay() указывает процессору сидеть и ничего не делать столько миллисекунд, сколько было указано в аргументе. Миллисекунды - это тысячные доли секунды; поэтому 1000 миллисекунд равно одной секунде. Итак, светодиод будет включён на одну секунду.

```
digitalWrite(LED, LOW);    // turns the LED off
```

Эта инструкция выключает светодиод, подобно тому как мы его включили раньше. зачем использовать HIGH и LOW? Это старое соглашение в электронике. HIGH означает что вывод включён, и в случае с Arduino на него будет подано 5 В. LOW означает 0 В. Мысленно вы можете заменить эти аргументы на ВКЛ и ВЫКЛ.

```
delay(1000); // waits for a second
```

Здесь мы производим ещё одну задержку. Светодиод будет выключен одну секунду.

} - эта закрывающая скобка обозначает конец функции loop.

Подводя итоги, наша программа делает вот что:

- Включает вывод 13 на вывод (только один раз в начале программы)
- Входит в цикл loop
- Переключает светодиод, подключённый к выводу 13
- Ожидает одну секунду
- Выключает светодиод на выводе 13
- Ожидает одну секунду
- Возвращается к началу цикла

Надеюсь, это было несложно. Вы узначете больше о программировании в следующих примерах.

Перед тем как мы перейдём к следующему разделу, я хочу чтобы вы поиграли с кодом. Например, уменьшите величину задержки используя различные цифры для команд включения и выключения, и вы увидите различные виды мигания. В частности, вы должны увидеть что происходит если вы сделаете задержку очень маленькой, но используете разные величины для задержек при включенном светодиоде и при выключенном... будет момент, когда произойдёт странная вещь; это "нечто" будет нам очень полезно когда мы будет изучать широтно-импульсную модуляцию.

4.8 Что мы будем создавать

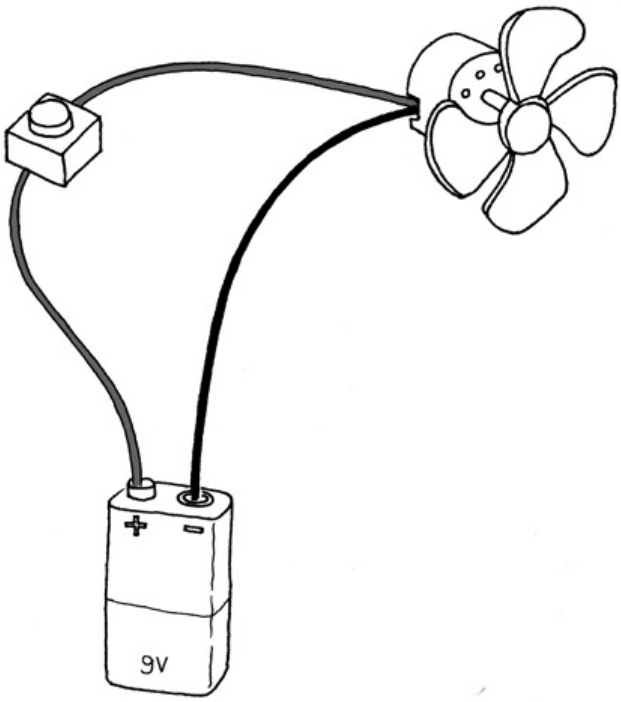
Меня всегда очаровывали свет и возможность управлять разными источниками света при помощи технологии. Мне посчастливилось работать над интересными проектами, которые включают управление светом и его взаимодействие с людьми. Arduino действительно хороша в этом. В этой книге мы будем работать над вопросом разработки "интерактивного света ", используя Arduino как способ понять основы построения интерактивных устройств.

В следующем разделе я постараюсь пояснить основы электричества способом, скучным инженеру, но не отпугивающим начинающих программистов Arduino.

4.9 Что такое электричество?

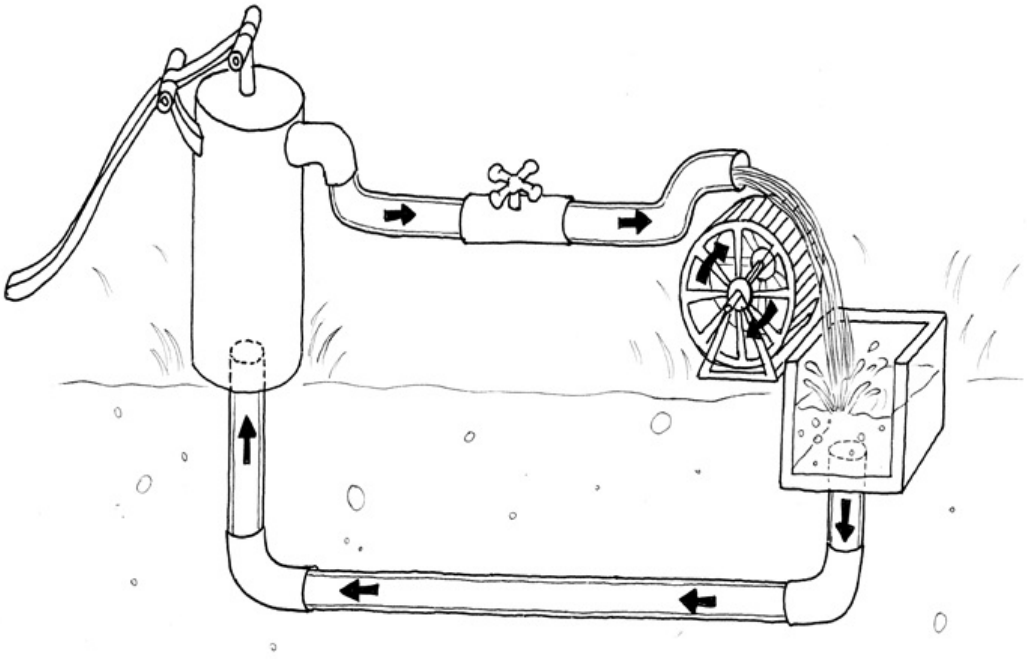
Если вы делали что-нибудь по дому, электроника вам не покажется сложной для понимания. Чтобы понять как работают электричество и электрические схемы, лучше всего представлять вещи как "водяная аналогия". Давайте создадим простое устройство, такое как портативный вентилятор с питанием от батарей (рис. 4-4).

Рис. 4-4. Портативный вентилятор



Если разобрать вентилятор на части, мы увидим что он состоит из маленькой батарейки, нескольких проводов и электромоторчика. Один из проводов, идущих от батареи к моторчику, разорван выключателем. Если у вас новая батарейка и вы включите выключатель, моторчик начнёт крутиться, охлаждая вас. Как это работает? Представьте себе что батарейка - это водяной резерувар с насосом, выключатель - кран, а электромотор - колесо, подобное тем что вы видели у водяных мельниц. Когда вы откроете кран, вода потечёт из насоса и будет приводить водяное колесо в движение.

Рис. 4-5. Гидравлическая система



Вы быстро поймёте что если вам надо вращать колесо быстрее, требуется увеличить размер труб (но это работает только до определённого предела) и увеличить давление насоса. Увеличение диаметра труб позволит пройти через них большему потоку воды; увеличивая трубу мы уменьшаем её сопротивление потоку. Это работает до определённого предела, при котором колесо не будет крутиться ещё быстрее, так как давленые воды недостаточно велико. Когда мы достигнет этой точки, нам надо насос помощнее. Такой метод ускорения водяной мельницы работает также до некоторой точки, в которой водяное колесо сломается из-за слишком сильного напора воды. Другая вещь, которую вы можете заметить, это что ось колеса немного нагревается, поскольку незавимо от того насколько точно мы установили колесо, трение между осью и колесом будет

создавать тепло. Важно понять что в подобной системе не вся энергия насоса будет превращена в движение колеса, некоторая часть будет потеряна из-за неэффективности системы и превратится в основном в тепло в некоторых её частях.

Итак, какие часты системы важны? Давление, поизводимое насосом; сопротивление труб и колеса потоку воды, и, собственно, сам поток воды (определяемый литрами воды, которая вытекает за секунду) и другие. Электричество работает подобно воде. У вас есть что-то подобное насосу (любой источник электричества, такой как батарейка или розетка в стене), который толкает электрические заряды (представим их как "капельки" электричества) по трубам, которые мы можем представить как провода, и устройства, способные производить тепло (пример - термоодеяло), свет (лампа в вашей комнате), звук (ваша стереосистема), движение (вентилятор) и многое другое.

Теперь, если вы прочтёте на батерейке "9 В", думайте об этом как о давлении воды, которое может выдать наш "насос". Напряжение измеряется в вольтах - единицах названных в честь Александра Вольта, создателя первой батареи.

В точности как давление воды имеет эквивалент в электричестве, скорость потока воды также его имеет. Он называется током, который измеряется в амперах (по имени Андре Мари Ампера, первооткрывателя электромагнетизма). Связь между напряжением и током может быть показана если мы вернёмся к водяному колесу: если большее напряжение (давление) позволяет вам крутить колесо быстрее, то больший поток воды (ток) позволяет крутить большее колесо.

И, наконец, сопротивление, противостоящее течению электричества на его пути, через который ток проходит, называется - вы знали это! - соротивлением, и измеряется в омах (по имени немецкого физика Георга Ома). Герр Ом также виновен в формулировке самого важного закона в электричестве, и вам надо запомнить только одну его формулу. Он смог показать, что напряжение, ток и сопротивление в цепи связаны друг с другом, и, в частности, что сопротивление цепи определяет количество тока, который будет течь через неё при определенном напряжении питания.

Это легко понять если вы задумаетесь. Возьмите батарейку на 9 вольт и включите её в простую схему. Измеряя ток, вы увидите что чем с большим сопротивлением резистор вы добавите в схему, тем меньший ток будет проходить через него.Возвращаясь к аналогии с водой, при данном насосе, если я установлю клапан (который соотносится с сопротивлением в электронике), то чем больше я буду закручивать этот клапан - увеличивая сопротивление потоку воды - тем меньше воды протечёт по трубе. Ом подвёл итог своего закона в формулу:

R (сопротивление) = V (напряжение) / I (ток)
V = R * I
I = V / R

Это единственное правило, который вам надо запомнить и выучить, поскольку в большинстве ваших работ оно единственное вам и понадобится.

4.10 Использование кнопки для управления светодиодом

Мигать светодиодом несложно, но я не думаю что вам понравится если настольная лампа будет бесконечно мигать тогда как вы пытаетесь читать книгу. Поэтому вам надо понять как управлять ею. В нашем предыдущем примере светодиод был актюатором и Arduino управляла им. Чего не хватает для полноты картины, так это сенсора.

В данном случае мы будем использовать простейший из доступных сенсоров - кнопку.

Если вы разберёте кнопку на части, вы увидите что это очень простое устройство: два кусочка металла, разделённые пружинкой, и пластиковый наконечник, который при нажатии соединяет эти контакты. Когда металлические части разделены, ток через кнопку не протекает (она подобна закрытому крану для воды); когда мы нажимаем её, мы осуществляем соединение.

Чтобы узнать состояние выключателя, существует новая для нас команда Arduino, которую нам следует изучить: функция digitalRead().

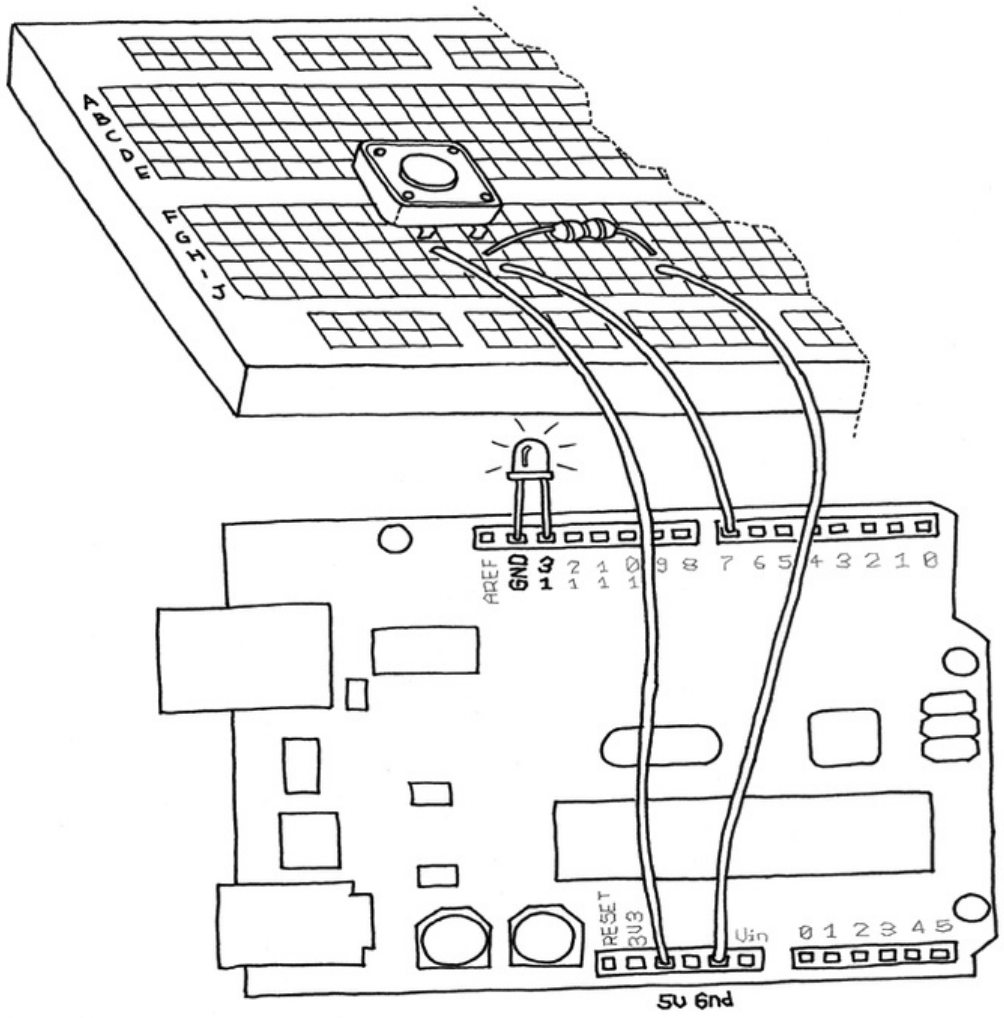
digitalRead() проверяет, подключено-ли напряжение к контакту, который вы указали в скобках, и возвращает значение "HIGH" или "LOW". Другие инструкции, которые мы использовали до этого, не возвращали никакой информации - они просто выполняли то что мы просили. Но такой тип функций немного ограничен, так как они заставляют нас придерживаться предсказуемой, строго

определённой последовательности команд, без ввода данных из окружения. С функцией digitalRead() мы можем "задать вопрос" Arduino и получить ответ, который можно сохранить где-нибудь в памяти и принять решение немедленно или позже.

Составьте схему по рис. 4-6. Для этого у вас должны быть некоторые детали (и они потребуются в следующих проектах):

- Беспаечная макетная плата. ПриложениеА - инструкция по применению такой платы.
- Набор нарезанных проводов
- Резистор на 10 кОм
- Кнопка

Рис. 4-6. Подключение кнопки



Примечание: чтобы не покупать набор проводов, вы можете купить небольшую катушку провода 22 AWG (диаметр 0.65 мм), нарезать куски требуемой длины и зачистить концы самостоятельно.

Давайте рассмотрим код, который используется для управления светодиода кнопкой:

Пример 4-2. Включение светодиода при нажатии кнопки

```
#define LED 13   // the pin for the LED
#define BUTTON 7 // the input pin where the
                 // pushbutton is connected
int val = 0;     // val will be used to store the state
                 // of the input pin

void setup() {
```

```
  pinMode(LED, OUTPUT);    // tell Arduino LED is an output
  pinMode(BUTTON, INPUT);  // and BUTTON is an input
}

void loop(){
  val = digitalRead(BUTTON);  // read input value and store it

  // check whether the input is HIGH (button pressed)
  if (val == HIGH) {
    digitalWrite(LED, HIGH); // turn LED ON
  } else {
    digitalWrite(LED, LOW);
  }
}
```

В среде Arduino выберите "File" > "New" (если у вас уже был открыт какой-то скетч, сохраните его). Когда Arduino спросит у вас имя папки для нового скетча, введите PushButtonControl. Напечатайте код Примера 4-2 в Arduino (или скачайте его с www.makezine.com/getstartedarduino). Если всё сделано правильно, светодиод будет загораться когда вы нажмёте кнопку.

4.11 Как это работает?

В этом примере программы я показал две новых концепции: функция, которая возвращает результат своей работы, и выражение "if".

Выражение "if" - возможно, самая важная инструкция в языке программирования, так как она позволяет компьютеру (а мы помним что Arduino - это маленький компьютер) делать выбор. После ключевого слова "if" вы должны написать "вопрос" в круглых скобках, и если "ответ", или результат, верен, будет выполнен первый блок кода; и напротив, если ответ неверен, будет выполнен блок кода после "else". Обратите внимание, что я использовал символ "==" вместо "=". Первый используется при сравнении двух значений и возвращает "TRUE" (ИСТИНА) или "FALSE" (ЛОЖЬ); второй присваивает значение переменной. Удостоверьтесь что вы пользуетесь ими правильно, так как очень легко совершить подобную ошибку и использовать простое равно. В этом случае программа никогда не будет работать. Я знаю это по опыту 25 лет программирования и всё ещё могу ошибиться.

Держать палец на кнопке пока вам надо свет - не очень практично. Хотя это заставило-бы вас задуматься о том, сколько энергии тратися впустую когда вы оставляете лампу включённой, нам надо подумать о том, как-бы сделать чтобы кнопка "залипала".

4.12 Одна схема, тысяча применений

Огромное преимущество цифровой программируемой электроники над классической стало теперь очевидным: я покажу вас как реализовать множество различных "поведений" с использованием той-же электрической схемы из предыдущего раздела, просто изменяя программу.

Как мы поняли раньше, непрактично держать палец на кнопке чтобы свет оставался включённым. Поэтому мы должны осуществить что-то похожее на "память" в виде механизма программы, который будет запоминать что мы нажали кнопку и продолжать светить даже елси мы отпустим её.

Чтобы сделать это, нам придётся использовать нечто, называемое переменной (мы уже использовали её, но я не пояснял ничего о ней). Переменная - это место в памяти Arduino, в котором мы можем хранить данные. Думайте о ней как о липкой бумаге для заметок, которую вы иногда используете для записи чего-нибудь: например, телефонного номера - вы берёте листик, пишете на ней "Аня, 02 555 1212" и приклеиваете на компьютер. В языке Arduino это так-же легко: вы просто определяете тип данных, которые будут храниться (например, число или какой-то текст), даёте ей имя, и теперь при надобности вы можете сохранить в переменной данные, или получить их. Например:

```
int val = 0;
```

"int" означает, что в переменной будет храниться целое число, "val" - это имя переменной, а "= 0" -

назначение переменной нулевого начального значения.

Переменная, как следует из названия, может быть изменена в любом месте вашего кода, так-что позднее в своей программе вы можете написать:

```
val = 112;
```

что изменит значение переменной с нуля на 112.

Примечание: Вы заметили, что Arduino каждая инструкция, кроме #define, заканчивается точкой с запятой? Это делается для того чтобы компилятор (часть Arduino, которая превращает ваш скетч в программу, которую может выполнить микроконтроллер) знал где заканчивается одно ваше выражение и начинается другое. Не забудьте использовать точку с запятой (кроме тех строк, которые начинаются с #define).

#define заменяются компилятором перед трансляцией кода в исполняемую программу.

В следующей программе val используется для хранения результата функции digitalRead(); что-бы ни получала Arduino со входа попадает в переменную и остаётся там до тех пор, пока другая строка кода не изменит её. Отметьте, что эти переменные хранятся в оперативной памяти, называемой (RAM). Она очень быстрая, но когда вы выключите свою плату, все данные в оперативной памяти будут потеряны (что означает что все переменные будут сброшены в начальные значения при включании платы). Ваша программа хранится во флеш-памяти (такой-же тип памяти используется в сотовых телефонах для хранения записной книжки) которая не изменяется при отключении платы от питания.

Давайте используем другую переменную для запоминания должен-ли светодиод оставаться включённым когда мы отпускаем кнопку. Пример 4-3 - это наша первая попытка:

Пример 4-3. Включить светодиод при нажатии кнопки и оставить его включённым при отпускании кнопки

```
#define LED  13  // the pin for the LED
#define BUTTON 7 // the input pin where the
                 // pushbutton is connected
int val = 0;     // val will be used to store the state
                 // of the input pin
int state = 0;   // 0 = LED off while 1 = LED on

void setup() {
  pinMode(LED, OUTPUT);   // tell Arduino LED is an output
  pinMode(BUTTON, INPUT); // and BUTTON is an input
}

void loop() {
  val = digitalRead(BUTTON); // read input value and store it

  // check if the input is HIGH (button pressed)
  // and change the state
  if (val == HIGH) {
    state = 1 - state;
  }

  if (state == 1) {
    digitalWrite(LED, HIGH); // turn LED ON
  } else {
    digitalWrite(LED, LOW);
  }
}
```

Попробуйте запустить этот код. Вы увидите что оно работает ... как-то. Вы увидите что светодиод изменяет своё состояние так быстро, что правильно установить его нажатием кнопки тяжело.

Посмотри на интересную часть кода: state - это переменная, которая хранит значение 0 или 1 для запоминания включён светодиод или нет. После отпускания кнопки мы устанавливаем её в 0 (светодиод выключен).

Далее мы считываем текущее состояние кнопки, и если она нажата (val == HIGH), мы изменяем state с 0 на 1, или наоборот. Поскольку state может быть равна только 1 или 0, используем небольшой трюк. Он заключается в маленьком математическом выражении, идея которого состоит в том, что 1 - 0 = 1, а 1 - 1 = 0:

```
state = 1 - state;
```

Такая строка не имеет смысла в математике, но он есть при программировании. Знак "=" означает "присвоить результат выражения после меня переменной передо мной" - в данном случае, новое значение state будет вычислено как единица минус старое значение state.

Далее в программе вы видите, что мы используем state для выяснения должен-ли светодиод быть включён или выключен. Как я говорил, это приводит к странному результату.

Результат странный из-за способа считывания кнопки. Arduino очень быстрая; она выполняет свои команды со скоростью 16 миллионов в секунду - вполне может быть, что и несколько миллионов строк кода за секунду. Это означает что пока ваш палец нажимает кнопку, Arduino может снять данные с кнопки несколько сотен раз и изменить столько-же раз состояние светодиода. Результат непредсказуем; светодиод может остаться выключённым когда вы хотите его включить и наоборот. Поскольку даже сломанные часы показывают верное время дважды в день, программа может выдавать верный результат каждый раз какое-то время, но и долгое время - неправильный.

Как ним исправить эту ситуацию? Требуется определить момент нажатися конпки - именно в этот момент следует изменять state. Способ, который мне нравится, таков - хранить старое значение val перед считыванием нового; это позволяет мне сравнить текущее положение кнопки с предыдущим и изменить state только когда кнопка стала "HIGH" после того, как была "LOW".

Пример 4-4 содержит следующий код:

Пример 4-4. Включить светодиод при нажатии кнопки и оставить его включённым после отпускания кнопки с новой, улучшеной формулой!

```
#define LED  13   // the pin for the LED
#define BUTTON 7  // the input pin where the
                  // pushbutton is connected
int val = 0;      // val will be used to store the state
                  // of the input pin
int old_val = 0;  // this variable stores the previous
                  // value of "val"
int state = 0;    // 0 = LED off and 1 = LED on

void setup() {
  pinMode(LED, OUTPUT);     // tell Arduino LED is an output
  pinMode(BUTTON, INPUT);   // and BUTTON is an input
}
void loop(){
  val = digitalRead(BUTTON); // read input value and store it
                             // yum, fresh


  // check if there was a transition
  if ((val == HIGH) && (old_val == LOW)){
    state = 1 - state;
  }

  old_val = val;  // val is now old, let's store it

  if (state == 1) {
```

```
    digitalWrite(LED, HIGH); // turn LED ON
  } else {
    digitalWrite(LED, LOW);
  }
}
```

Попробуйте код, мы почти закончили!

Возможно вы заметили, что результат не отличный из-за другой проблемы механических переключателей. Кнопки очень простые устройства - два кусочка металла разделены пружинкой. При нажатии кнопки эти контакты соединяются и через них может протекать электричество. Это звучит красиво и просто, но в реальной жизни соединения не так прекрасны, особенно если кнопка нажата не полностью, и генерируют ложные сигналы, называемые дребезг.

Когда кнопка "дребезжит", Arduino видит быструю последовательность сигналов включения и выключения. Существует множество видов антидребезга, но для нашего, простого коде, я заметил что достаточно добавить 10...50-миллисекундную задержку чтобы код определил изменение.

Окончательный код показан в примере 4-5.

Пример 4-5. Включить светодиод при нажатии кнопки и оставить его включённым после отпускания кнопки, включая простой антидребезг. Теперь с новой, улучшенной формулой!

```
#define LED  13   // the pin for the LED
#define BUTTON 7  // the input pin where the
                  // pushbutton is connected
int val = 0;      // val will be used to store the state
                  // of the input pin
int old_val = 0;  // this variable stores the previous
                  // value of "val"
int state = 0;    // 0 = LED off and 1 = LED on

void setup() {
  pinMode(LED, OUTPUT);    // tell Arduino LED is an output
  pinMode(BUTTON, INPUT);  // and BUTTON is an input
}

void loop(){
  val = digitalRead(BUTTON);  // read input value and store it
                              // yum, fresh

  // check if there was a transition
  if ((val == HIGH) && (old_val == LOW)){
    state = 1 - state;
    delay(10);
  }

  old_val = val; // val is now old, let's store it

  if (state == 1) {
    digitalWrite(LED, HIGH); // turn LED ON
  } else {
    digitalWrite(LED, LOW);
  }
```

5. Продвинутый ввод-вывод

Всё, что мы изучили в четвёртой главе - в большинстве простейшие операции, которые мы можем делать с Arduino: управлять цифровым выводом и читать цифровод ввод. Если-бы Arduino был разговорным языком, это было-бы всего-лишь двумя буквами алфавита. Учитывая то, что в этом алфавите всего пять букв, вы можете увидеть сколько еще работы надо сделать чтобы писать поэмы на Arduino.

5.1 Пробуем другие датчики включения-выключения

5.1.1 Выключатели

То-же самое что и кнопка, но не изменяет автоматически своё состоянии при отпускании.

5.1.2 Термостаты

Выключатель, который срабатывает при достижении установленной температуры.

5.1.3 Магнитные переключатели, также известные как "герконы"

Имеют два контакта, которые соединяются если рядом расположить магнит, используются в сигнализация против взлома для определения открытого окна.

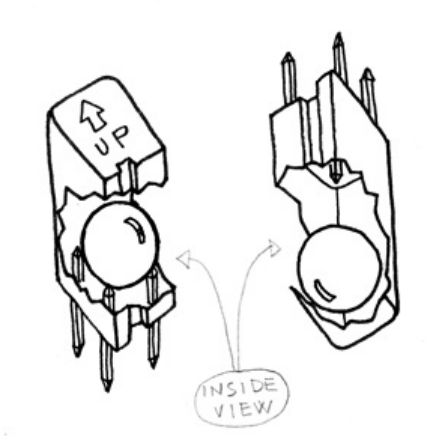
5.1.4 Ковровые переключатели

Маленькие пластинки, которые можно положить под ковёр чтобы определить присутствие человека (или тяжёлого кота).

5.1.5 Датчики наклона

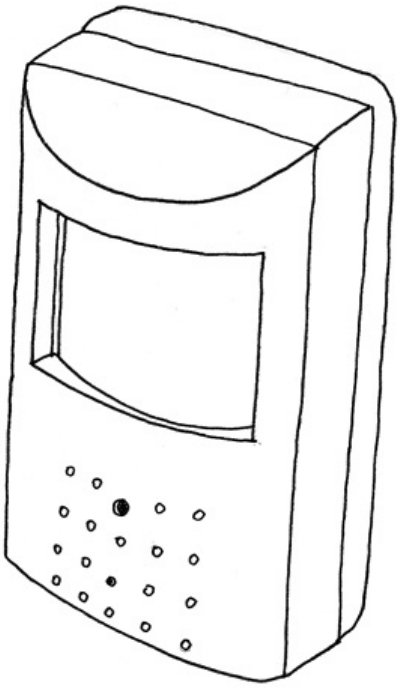
Простой электронный компонент, сделанный из двух контактов и маленького металлического шарика (или капельки ртути, но я не рекомендую использовать такие). Рис. 5-1 показывает внутренности типичной модели. Когда датчик расположен вертикально, шарик замыкает оба контакта, и устройство работает как если-бы вы нажати кнопку. Когда вы наклоняете датчик, шарик движется и контакты размыкаются, как если-бы вы отпустили кнопку. Используя такой простой компонент, вы можете создавать, например, интерфейсы управления жестами, которые реагируют на движения или встряхивание обьекта.

Рис. 5-1. Конструкция датчика наклона



Другой датчик, который вы можете захотеть испытать, это инфракрасный датчик от охранной сигнализации (также известный как пассивный инфракрасный датчик, или ИК-датчик, см. рис. 5-2). Эти маленькие устройства срабатывают когда человек (или животное) движется в пределах его видимости. Это простой способ определить движение.

Рис. 5-2. Типичный инфракрасный датчик



Теперь вы можете поэкспериментировать, найдя всевозможные устройства, имеющие два замыкающихся контакта, такие как термостат, управляющий температурой в комнате (используйте старый, который не подключён к системе), или просто расположив рядом два контакта и капнув на них водой.

Например, используя пример из главы 4 и ИК-датчик, вы можете создать лампу, которая реагирует на присутствие человека, или вы можете применить датчик наклона чтобы она включалась при наклоне в определённую сторону.

5.2 Управление светом при помощи ШИМ

С пониманием того что вы уже изучили, можно создать интерактивную лампу, которая управляется, а не только скучно включается и выключается, и более элегантно. Одно из ограничений примера с мигающий светодиодом - это то, что вы можете только включить или выключить его. Хорошая интерактивная лампа должна иметь плавную регулировку. Чтобы решить эту проблему, мы можем использовать маленький трюк, который делает возможным множество таких вещей, как телевидение или кино: инерцию зрительного восприятия.

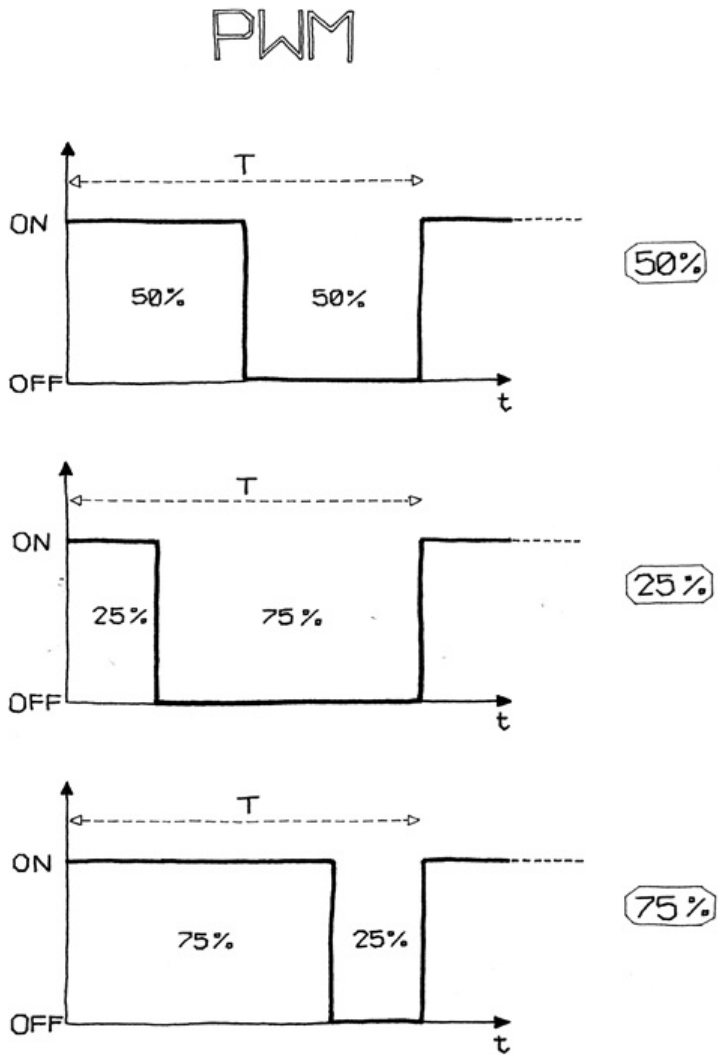
Как я подсказывал в первом примере главы 4, если вы измените цифры в функции задержки чтобы светодиод перестал мигать, вы заметите что светодиод, похоже, светится в половину своей нормальной яркости. Теперь измените цифры так, чтобы светодиод был включен на четверть того времени, пока он выключен. Запустите скетч и вы увидите, что яркость понизилась до 25%. Такая техника называется широтно-импульсная модуляция (ШИМ) - фантастический способ, определяемый так: если мигать светодиодом достаточно быстро, вы не увидите самих миганий, но вы можете изменять яркость светодиода меняя отношение между временем его включенного и выключеного состояния.

Рис. 5-3 показывает как это работает.

Такая техника работает не только со светодиодами. Например, таким-же способом вы можете изменять скорость электордвигателя.

Экспериментируя, вы увидите, что мигать светодиодом, расставляя задержка в коде, немного неудобно, так как когда вам понадобится считать датчик или отправить данные по последовательному порту, светодиод будет мерцать из-за задержек чтения датчика. К с частью, процессор на плате Arduino, имеет аппаратную часть, которая может очень эффективно мигать тремя сетодиодами в то время, когда ваш скетч делает что-то другое. Эта часть реализована на выводах 9, 10 и 11, которые могут управляться командой analogWrite().

Рис. 5-3. ШИМ в действии

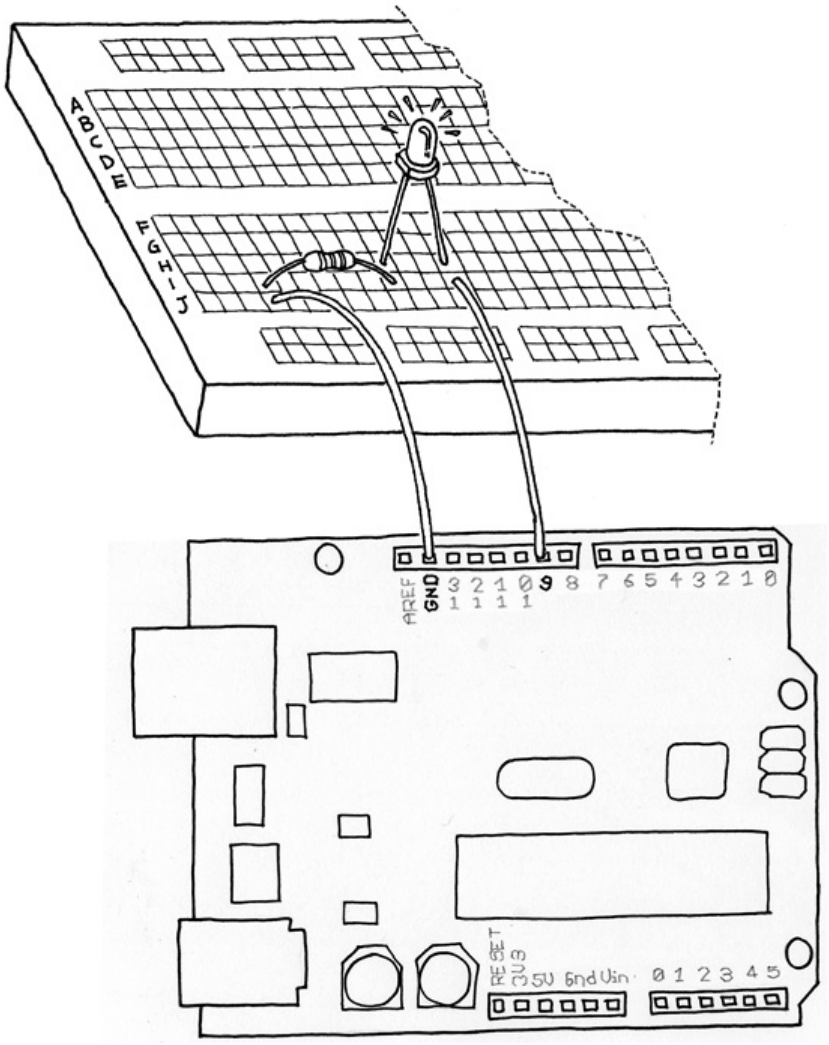


Например, запись analogWrite(9,128) установить яркость светодиода, подключённого к выводу 9 в 50%. Почему 128? analogWrite() требует число от 0 до 255 в качестве аргумента, где 255 означает полную яркость, а 0 - выключено.

Примечание: Наличие трёх каналов - просто отлично, так как вы можете купить красный, зелёный и синий светодиоды и смешивать их цвета для получения любого другого цвета по своему желанию!

Давайте попробуем. Соберите схему по рис. 5-4. Заметьте, что у светодиода есть полярность: длинная ножка (анод) должна быть справа, а короткая (катод) - слева. Также большинство светодиодов имеют плоскую фаску на стороне катода, как показано на рисунке.

Рис. 5-4. Светодиод, подключённый к выводу ШИМ



Далее, создайте новый скетч в Arduino и используйте пример 5-1 (его вы также можете загрузить с www.makezine.com/getstartedarduino):

Пример 5-1. Плавное включение и выключение светодиода, подобное режиму сна в компьютере Apple.

```
#define LED 9 // the pin for the LED
int i = 0;    // We'll use this to count up and down

void setup() {
  pinMode(LED, OUTPUT); // tell Arduino LED is an output
}

void loop(){

  for (i = 0; i < 255; i++) { // loop from 0 to 254 (fade in)
    analogWrite(LED, i);      // set the LED brightness
    delay(10); // Wait 10ms because analogWrite
               // is instantaneous and we would
               // not see any change
  }

  for (i = 255; i > 0; i--) { // loop from 255 to 1 (fade out)

    analogWrite(LED, i); // set the LED brightness
    delay(10);           // Wait 10ms
  }
}
```

Теперь вы повторили чудесную возможность ноутбука (может быть, это не очень экономно

использовать Arduino для такой простой задачи). Давайте используем эти знания чтобы улучшить нашу лампу.

Добавьте в схему кнопку (назад в главу 4) к данной плате. Попробуйте сделать это не читая текст дальше, так как я хочу чтобы вы начали думать о том факте, что любая простая схема, которую я показывал здесь, является "маленьким кирпичиком" для создания всё больших и больших проектов. Если вам потребовалось заглянуть вперёд, не расстраивайтесь, самое важное - это то что вы потратили некоторое время на обдумывание того как может выглядеть такая схема.

Чтобы создать эту данную, вам надо соединить схему, которую вы только-что построили (рис. 5-4) со схемой с кнопкой, показанной на рис. 4-6. Если хотите, можете просто собрать отдельно обе эти схемы на плате; у вас на ней достаточно места. Однако, одно из преимуществ подобной платы (см. приложение А) - это то, что в ней есть пара шин, которые проходят горизонтально поверху и понизу. Одна из них отмечана красным (плюс), а вторая - синим (минус).

Эти шины используются для подвода питания и общей земли куда потребуется. В случае схемы, которую надо собрать для данного примера, у вас есть два компонента (оба - резисторы), которые надо подключить в выводу GND (земля) платы Arduino. Так как у Arduino есть два вывода GND, вы можете просто соединить эти две схемы в точности как показано на обоих рисунках, а затем взять провода, подключённые на иллюстарциях к GND, и соединить их вместе.

Если вы не готовы сделать это, не расстраивайтесь: просто подключите схемы к Arduino как показано на рисунках 4-6 и 5-4. Вы увидите пример, использующий отрицательную и положительную шины в главе 6.

Возвращаясь к нашему следующему примеру, если у нас есть только одна кнопка, как мы будем управлять лампой? Мы изучим ещё одно действие в технике создания: определение того, как долго была нажата кнопка. Чтобы сделать такое, нам надо изменить пример 4-5 из главы 4 чтобы добавить плавную регулировку. Идея состоит в том чтобы создать "интерфейс", в котором нажатие и отпускание кнопки приводит к включению и выключению света, а нажатие и удерживание той-же кнопки изменяет яркость.

Давайте рассмотрим скетч:

Пример 5-2. Включить светодиод при нажатии кнопки, оставить включённым после её отпускания с использованием простого антидребезга. Если кнопка нажата, изменять яркость.

```
#define LED 9     // the pin for the LED
#define BUTTON 7  // input pin of the pushbutton

int val = 0;      // stores the state of the input pin

int old_val = 0;  // stores the previous value of "val"
int state = 0;    // 0 = LED off while 1 = LED on

int brightness = 128;         // Stores the brightness value
unsigned long startTime = 0;  // when did we begin pressing?

void setup() {
  pinMode(LED, OUTPUT);   // tell Arduino LED is an output
  pinMode(BUTTON, INPUT); // and BUTTON is an input
}

void loop() {

  val = digitalRead(BUTTON); // read input value and store it
                             // yum, fresh

  // check if there was a transition
  if ((val == HIGH) && (old_val == LOW)) {

    state = 1 - state; // change the state from off to on
                       // or vice-versa
```

```
    startTime = millis(); // millis() is the Arduino clock
                          // it returns how many milliseconds
                          // have passed since the board has
                          // been reset.

   // (this line remembers when the button
   // was last pressed)
   delay(10);
}
// check whether the button is being held down
  if ((val == HIGH) && (old_val == HIGH)) {

    // If the button is held for more than 500ms.
    if (state == 1 && (millis() - startTime) > 500) {

      brightness++; // increment brightness by 1
      delay(10);    // delay to avoid brightness going
                    // up too fast

      if (brightness > 255) { // 255 is the max brightness

        brightness = 0; // if we go over 255
                        // let's go back to 0
      }
    }
  }

  old_val = val; // val is now old, let's store it

  if (state == 1) {
    analogWrite(LED, brightness); // turn LED ON at the
                                  // current brightness level
  } else {
    analogWrite(LED, 0); // turn LED OFF
  }
}
```

Попробуйте скетч. Как видно, наша модель наша модель взаимодействия обретает форму. Если вы нажмёте и немедленно отпустите кнопку, вы включите или выключите лампу. Если держать кнопку нажатой, яркость светодиода будет изменяться. Просто отпустите кнопку, когда установите желаемую яркость.

А теперь давайте разберёмся как использовать более интересные сенсоры.

5.3 Использование фотодатчика вместо кнопки

Сейчас мы произведём интересный эксперимент. Возьмём фотодатчик, как на рис. 5-5.

В темноте сопротивление светочувствительного датчика весьма высокое. Если на него посветить, сопротивление быстро снижается и становится достаточным для прохождения электричества. Таким образом, у нас есть выключатель, активируемый светом.

Соберите схему для примера 4-2 (см. "Использование кнопки для управления светодиодом" в главе 4), затем загрузите код из примера 4-2 в вашу Arduino.

Теперь подключите фотодатчик в плату вместо кнопки. Вы увидите, что если закрыть датчик рукой, светодиод выключится. Откройте датчик, и светодиод зажжётся. Вы только-что собрали свой первый светодиод, управляемый датчиком. Это важно, так как первый раз в книге мы используем электронный компонет, не являющийся просто механическим устройством: это полнофункциональный датчик.

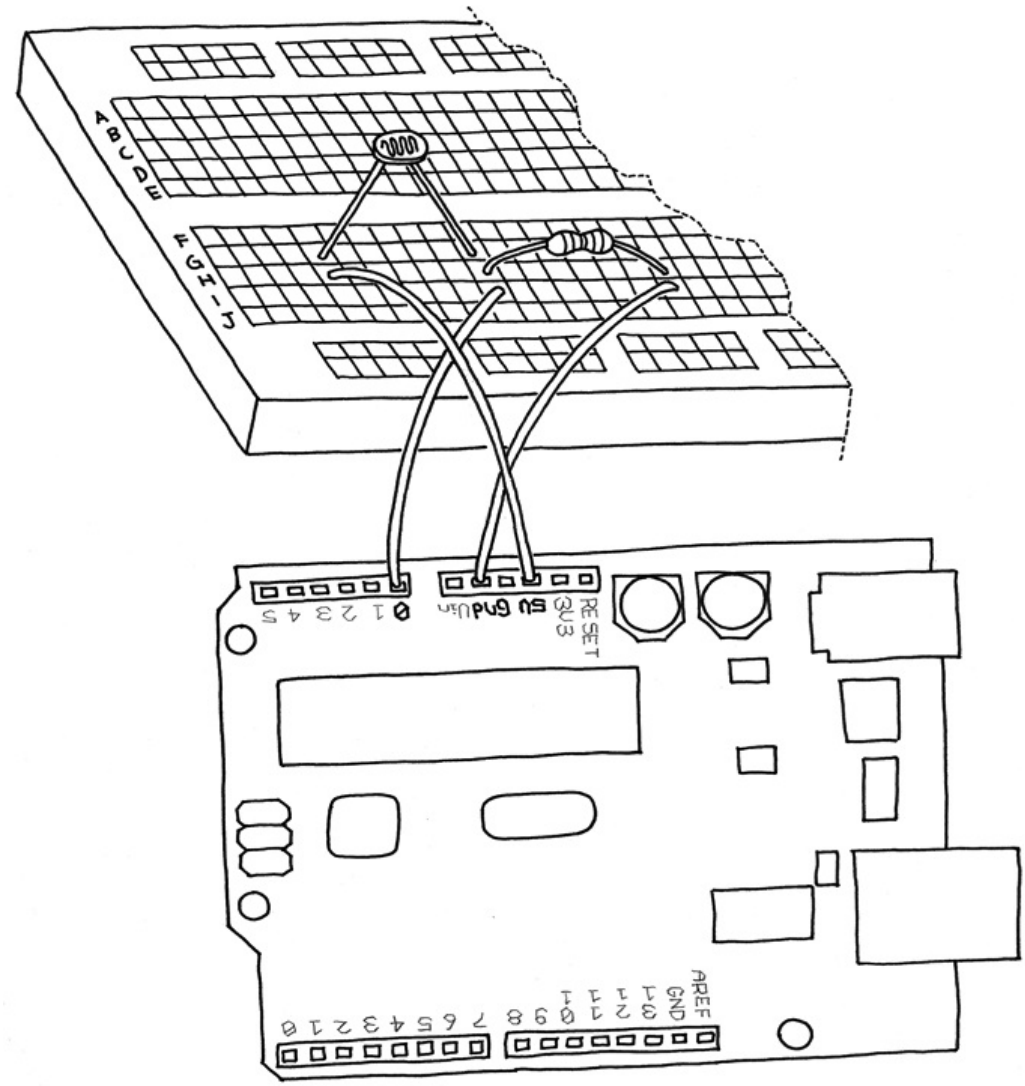
5.4 Аналоговый ввод

Как вы узнали из предыдущего раздела, Arduino может определять, было-ли приложено напряжение к одному из её выводов и сообщать это функции digitalRead(). Тип ответа "да/нет" хорош для многих приложений, но датчик света, который мы недавно использовали, может сообщить нам не только что свет есть, но также и сколько его. В этом и состоит разница между датчиками вк/выкл (который сообщает что что-то мы что-то имеем) и аналоговым датчиком, данные которого постоянно изменяются. Для чтения таких датчиком нам требуется другой тип выводов.

В правой нижней части платы Arduino вы видите шесть выводов, обозначенных "Analog In"; это специальные выводы, которые могут сообщить нам не только что к ним приложено напряжение, но и его величину. Используя функцию analogRead(), мы можем считывать это напряжение. Данная функция возвращает числа от 0 до 1023, которые соответствует напряжениям в пределах от 0 до 5 вольт. Например, для напряжения величиной 2,5 В, приложенного к выводу 0, функция analogRead(0) вернёт 512.

Теперь, если вы соберёте схему по рис. 5-6, используя резистор 10 кОм и запустите код из примера 5-3, вы увидите что светодиод на плате (также вы можете вставить светодиод в выводы 13 и GND как показано в разделе "Мигающий светодиод" из главы 4) мигает с частотой, зависящей от освещённости фотодатчика.

Рис. 5-6. Схема с аналоговым датчиком



Пример 5-6. Мигание светодиодом со скоростью, зависящей от величины на аналоговом входе

```
#define LED 13 // the pin for the LED

int val = 0;   // variable used to store the value
               // coming from the sensor
void setup() {
  pinMode(LED, OUTPUT); // LED is as an OUTPUT
```

```
  // Note: Analogue pins are
  // automatically set as inputs
}

void loop() {

  val = analogRead(0); // read the value from
                       // the sensor

  digitalWrite(13, HIGH); // turn the LED on

  delay(val); // stop the program for
              // some time

  digitalWrite(13, LOW); // turn the LED off

  delay(val); // stop the program for
              // some time
}
```

Теперь попробуйте пример 5-4, но перед этим вам придётся изменить свою схему. Посмотрите на рис. 5-4 ещё раз и соедините светодиод с выводом 9. Так как у вас уже есть собранная схема, вам надо найти точку платы где светодиод, провод и резистор не будет соединён с фотодатчиком.

Пример 5-4. Установка яркости светодиода, определяемая данными на аналоговом входе.

```
#define LED 9 // the pin for the LED

int val = 0;  // variable used to store the value
              // coming from the sensor

void setup() {

  pinMode(LED, OUTPUT); // LED is as an OUTPUT

  // Note: Analogue pins are
  // automatically set as inputs
}

void loop() {

  val = analogRead(0); // read the value from
                       // the sensor
  analogWrite(LED, val/4); // turn the LED on at
                           // the brightness set
                           // by the sensor

  delay(10); // stop the program for
             // some time
}
```

Примечание: мы указываем яркость, деля val на 4 из-за того, что analogRead() возвращает число до 1023, а analogWrite() принимает 255 максимум.

5.5 Попробуйте другие аналоговые датчики

Используя ту-же схему что вы видели в предыдущем разделе, вы можете подсоединить множество других резистивных датчиков, которые работают более-менее подобным образом. Например, вы можете подключить термистор (простое устройство, изменяющее своё сопротивление в зависимости от температуры). В схеме показано как изменение сопротивления изменяет напряжение в ней. Это напряжение может быть измерено Arduino.

Если вы работаете с термистором, будьте внимательны с правильным соотношением данных, которые вы прочитали и действительно измеренной температурой. Если вам надо точные значения, вам потребуется записать числа, считываемые с аналогового входа и сравнить их с настоящим термометром. После этого можно составить табличку и выработать способ калибровки аналоговых результатов и фактической температуры окружающей среды.

До сих пор мы использовали в качестве устройства вывода светодиод, но сейчас как нам прочесть действительные данные значений, полученный Arduino с датчика? Мы не можем сделать плату, мигающую кодом Морзе (ну, вообще-то можем, но для людей есть более простые способы чтения данных). Для этого мы может заставить Arduino общаться с компьютером по последовательному порту, который и будет описан в следующем разделе.

5.6 Последовательная связь

Как вы прочли в начале книги, у Arduino есть USB-подключение, используемое средой разработки для загрузки кода в процессор. Хорошая новость в том, что это подключение также может использоваться скетчами для отправки назад на компьютер данных или получения команд от него. Для этой цели мы изучим обьект serial (обьект - это набор свойств, которые обьединены вместе для удобства пишущих скетчи людей).

Этот обьект содержит весь код, необходимый для отправки и получения данных. Сейчас мы используем предыдущую схему с фоторезистором и будем отправлять значения, которые прочли, на компьютер. Вставьте следующий код в новый скетч (его также можно скачать с www.makezine.com/getstartedarduino):

Пример 5-5. Отправка на компьютер данных, прочитанных с аналогового входа 0. После выгрузки скетча на плату нажмите кнопку "Serial Monitor" в IDE.

```
#define SENSOR 0 // select the input pin for the
                 // sensor resistor

int val = 0; // variable to store the value coming
             // from the sensor

void setup() {

 Serial.begin(9600); // open the serial port to send
                     // data back to the computer at
                     // 9600 bits per second
}

void loop() {

  val = analogRead(SENSOR); // read the value from
                            // the sensor

  Serial.println(val); // print the value to
                       // the serial port

  delay(100); // wait 100ms between
              // each send
}
```

После того, как вы выгрузите скетч в Arduino, нажмите кнопку "Serial Monitor" в Arduino IDE (самая правая в панели кнопок); в открывшемся вы увидите постоянно увеличивающийся список цифр. Теперь любая программа, способная работать с последовательным портом, может вести диалог с Arduino. Существует много языков программирования, которые позволяют писать программы, общающиеся с последовательным портом. Processing (www.processing.org) является отличным дополнением к Arduino из-за того что и язык, и IDE очень похожи.

5.7 Управление большими нагрузками (электродвигатели, лампы и тому подобное)

Каждый из выводов Arduino может быть использован для запитки устройств с током до 20 миллиампер - это очень маленькая величина тока, достаточная только для питания светодиода. Если вы попробуете управлять чем-нибудь другим, например, моторчиком, вывод немедленно перестанет работать, и, вероятно, сожжёт весь процессор. Для управления большими нагрузками, такими как электродвигатели или лампы накаливания нам требуется внешний компонент, который может включать и выключать другие компоненты и который управляется выводом Arduino. Одним из таких приборов является MOSFET-транзистор (МОП-транзистор) - не обращайте внимания на смешное название - это электронный выключатель, который может управляться приложением напряжения на один из его трёх выводов, каждый из которых называется затвором. Он подобен выключателю света в доме, когда движение руки для включения или выключения света заменено выводом Arduino, подабщим напряжение на затвор такого транзистора.

Примечание: MOSFET означает "metal–oxide–semiconductor field-effect transistor" (металл-оксид-полупроводник (МОП) полевой транзистор). Это специальный вид транзистора, который может работать на принципе полевого эффекта. Это означает, что электричество будет протекать через частицу полупроводника (между выводами стока и истока) при прикладывании напряжения к выводу затвора. Так как затвор изолирован от остальных выводов плёнкой оксида металла, из Arduino в MOSFET ток не протекает, что позволяет создать простой интерфейс. Они идеальны для включения и выключения больших нагрузок с высокой частотой.

На рис. 5-7 видно как мы применили MOSFET IRF520 для включения и выключения маленького моторчика, присоединённого к вентилятору. Вы также должны заметить, что моторчик получает питание от разъёма 9 В на плате Arduino. Это ещё одно преимущество MOSFET: он позволяет управляет приборами, у которых питание осуществляется от других источников. Так как MOSFET подключён к выводу 9, мы может использовать команду analogWrite() для управления скоростью моторчика при помощи ШИМ.

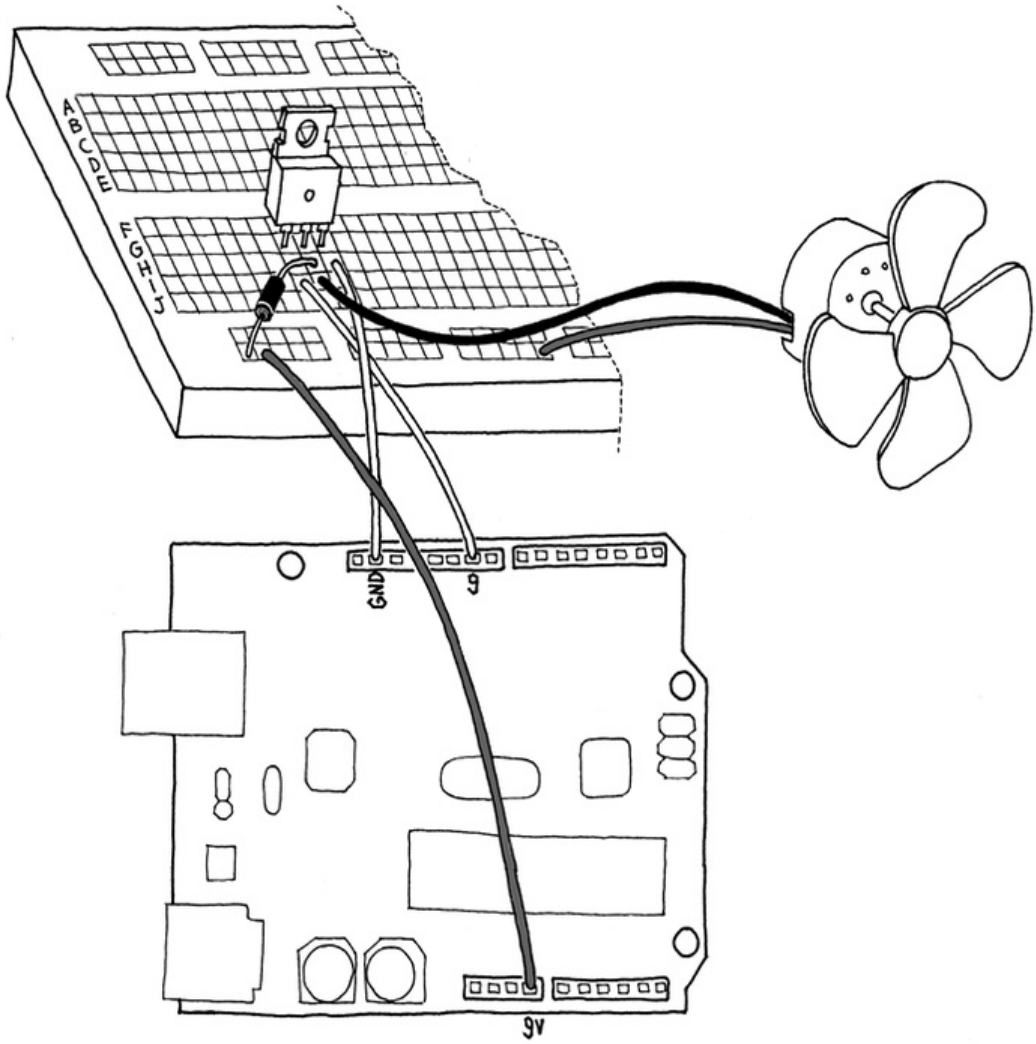
5.8 Сложные сенсоры

Мы определим сложные сенсоры как такие, которые могут выдавать информацию, что требует больше чем просто функции digitalRead() или analogRead(). Обычно они представляют собой маленькую схему с небольшим микроконтроллером внутри, который предварительно обрабатывает информацию.

Некоторые из таких сложных сенсоров являются ульразвуковыми или инфракрасными датчиками расстояния, акселерометрами. Вы можете увидеть примеры на нашем сайте в разделе "Tutorials" (www.arduino.cc/en/Tutorial/HomePage).

Книга "Making Things Talk" (автор - Tom Igoe, издательство O'Reilly) подробно разъясняет принципы этих сенсоры, а также множество других.

Рис. 5-7. Схема двигателя вентилятора с Arduino



Глава 6. Разговоры с облаками

В предыдущих главах вы познакомились с основами Arduino и основнами строительных блоков. Позвольте мне напомнить вам, из чего состоит "алфавит Arduino":

6.1 Цифровой вывод

6.1.1 Цифровой вывод

Мы использовали его для управления светодиодом, но, с правильной схемой он может использоваться для управления моторами, создания звуков и многого другого.

6.1.2 Аналоговый вывод

Даёт возможность управлять яркостью светодиода, а не просто включать или выключаеть его. Мы даже можем контролировать с его помощью скорость электродвигателя.

6.1.3 Цифровой ввод

Позволяет нам читать состояние простых сенсоров, таких как кнопки или датчики наклона.

6.1.4 Аналоговый ввод

Мы можем читать сигналы с датчиков, которые постоянно посылают данные, а не просто "вкл/выкл", таких как потенциометр или датчик света.

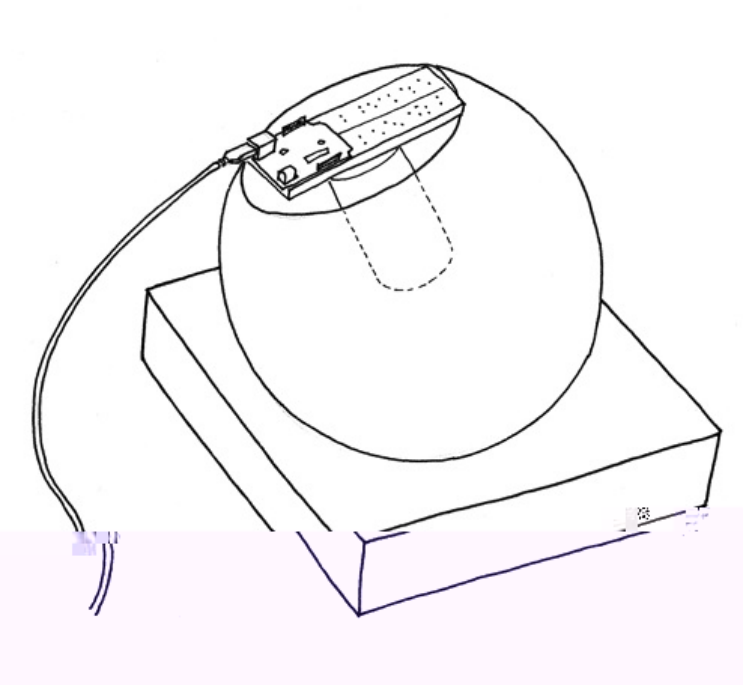
6.1.5 Последовательная связь

Позволяет связываться с компьютером и обмениваться данными, или просто следить за тем, что происходит в скетче, выполняющемся Arduino.

В этой главе мы посмотрим как обьединить работающие приложения используя всё что мы выучили в предыдущих. Эта глава покажет вам как каждый отдельный пример может быть использован как кирпичик при построении сложного проекта.

Вот где во мне проявляется желание быть дизайнером. Мы создадим версию двадцать первого века классической лампы моего любимого итальянского дизайнера, Джо Коломбо. Обьект, который мы будем создавать, называется "Атон" с 1964 года.

Рис. 6-1. Закончення лампа



Лампа, которую вы можете видеть на рис. 6-1 - это простой шар, установленный на подставке с большим отверстием для того чтобы шар не скатился с вашего стола. Такая конструкция позволяет направлять лампу куда угодно.

В терминах функциональности, мы хотим создать устройство, которое должно подключаться к интернету, получать текущий список статей в блоге Make blog (blog.makezine.com) и подсчитывать использованное количество слов "peace", "love" и "Arduino". С этими значениями мы будет создавать цвет и показыавть его при помощи лампы. Лампа имеет кнопку для включения и выключения, а также датчик света для автоматического включения.

6.2 Планирование

Давайте посмотрим чего мы хотим добиться и что нам для этого надо. Во-первых, нам требуется чтобы Arduino могла подключаться к интернету. Поскольку у платы Arduino есть только USB-порт, мы не можем подключить её к интернету напряму, так-что нам надо подумать как соединить их. Обычно люди запускают приложение на компьютерах, которые подключаются к интернету, обрабатывают данные и посылают Arduino пакеты обработанной информации.

Arduino - это простой компьютер с небольшой памятью, она не может легко обрабатывать большие файлы, и мы подключимся к RSS-ленте для получения понятного XML-файла, который требует больше оперативной памяти. Мы создадим промежуточный элемент (прокси) для упрощения XML с применением языка Processing.

> Processing
>
> Processing - это то, откуда появился Arduino. Мы любим этот язык и используем его как для обучения программирования начинающих, так и для создания прекрасного кода. Другое преимущество Processing - это то, что у него открытый код и он работает на всех основных платформах (Mac, Linux и Windows). Он также может создавать отдельные приложения, которые будут выполняться на этих платформах. Более того, сообщество Processing довольно оживлено и полезно, к тому-же вы можете найти тысячи уже написанных примеров программ.

Прокси сделает для нас следующая работу: скачает ленту RSS с сайта makezine.com и извлечёт все слова из полученного XML-файла.Затем, пройдя по файлу, он подсчитает количество слов "peace", "love" и "Arduino" в тексте. Исходя из полученных трёх чисел мы можем вычислить значение цвета, которое пошлём Arduino. Плата отправит обратно величину света, измеренную датчиком и покажет её на экране компьютера.

Со стороны аппаратного обеспечения, мы обьединим пример с кнопкой, управлением светодиодом при помощи ШИМ (умноженное на три) и последовательную связь.

Поскольку Arduino - простое устройство, нам требуется закодировать цвет простым способом. Мы будем использовать стандартный способ, представленный в языке HTML: знак решётки # с шестью шестнадцатеричными цифрами после него.

Шестнадцатеричные цифры удобны потому, что каждое 8-битовое число сохраняется в двух битах; для десятичных чисел количество знаков меняется от одного до трёх. Предсказуемость делает код проще: ждём, пока не увидим решётку "#", затем считываем шесть знаков после неё в буфер (переменную для временного хранения данных). И, наконец, превращаем каждую группу из двух цифр в один байт, который будет соответствовать яркости одного из трёх светодиодов.

6.3 Программирование

Вы будете запускать два скетча: один скетч на языке Processing и один а языке Arduino. Вот код для Processing. Вы можете скачать его с www.makezine.com/getstartedarduino.

Пример 6-1. Честь кода для лампы с подключением к сети, вдохновлённые записью блоге Tod E. Kurt (todbot.com)

```
import processing.serial.*;

String feed = "http://blog.makezine.com/index.xml";

int interval = 10;  // retrieve feed every 60 seconds;
int lastTime;       // the last time we fetched the content

int love    = 0;
int peace   = 0;
int arduino = 0;

int light = 0;  // light level measured by the lamp

Serial port;
color c;
String cs;

String buffer = ""; // Accumulates characters coming from Arduino

PFont font;

void setup() {
  size(640,480);
  frameRate(10);   // we don't need fast updates

  font = loadFont("HelveticaNeue-Bold-32.vlw");
  fill(255);
  textFont(font, 32);
  // IMPORTANT NOTE:
  // The first serial port retrieved by Serial.list()
  // should be your Arduino. If not, uncomment the next
  // line by deleting the // before it, and re-run the
  // sketch to see a list of serial ports. Then, change
  // the 0 in between [ and ] to the number of the port
  // that your Arduino is connected to.
  //println(Serial.list());
```

```
  String arduinoPort = Serial.list()[0];
  port = new Serial(this, arduinoPort, 9600); // connect to Arduino

  lastTime = 0;
  fetchData();
}

void draw() {
  background( c );
  int n = (interval - ((millis()-lastTime)/1000));

  // Build a colour based on the 3 values
  c = color(peace, love, arduino);
  cs = "#" + hex(c,6); // Prepare a string to be sent to Arduino

  text("Arduino Networked Lamp", 10,40);
  text("Reading feed:", 10, 100);
  text(feed, 10, 140);

  text("Next update in "+ n + " seconds",10,450);
  text("peace" ,10,200);
  text(" " + peace, 130, 200);
  rect(200,172, peace, 28);

  text("love ",10,240);
  text(" " + love, 130, 240);
  rect(200,212, love, 28);

  text("arduino ",10,280);
  text(" " + arduino, 130, 280);
  rect(200,252, arduino, 28);

  // write the colour string to the screen
  text("sending", 10, 340);
  text(cs, 200,340);
  text("light level", 10, 380);
  rect(200, 352,light/10.23,28); // this turns 1023 into 100

  if (n <= 0) {
    fetchData();
    lastTime = millis();
  }

  port.write(cs); // send data to Arduino

  if (port.available() > 0) { // check if there is data waiting
    int inByte = port.read(); // read one byte
    if (inByte != 10) { // if byte is not newline
      buffer = buffer + char(inByte); // just add it to the buffer
    }
    else {

      // newline reached, let's process the data
      if (buffer.length() > 1) { // make sure there is enough data

        // chop off the last character, it's a carriage return
        // (a carriage return is the character at the end of a
        // line of text)
        buffer = buffer.substring(0,buffer.length() -1);

       // turn the buffer from string into an integer number
       light = int(buffer);
```

```
      // clean the buffer for the next read cycle
      buffer = "";

      // We're likely falling behind in taking readings
      // from Arduino. So let's clear the backlog of
      // incoming sensor readings so the next reading is
      // up-to-date.
      port.clear();
    }
   }
  }

}
void fetchData() {
  // we use these strings to parse the feed
  String data;
  String chunk;

  // zero the counters
  love = 0;
  peace = 0;
  arduino = 0;
  try {
    URL url = new URL(feed); // An object to represent the URL
    // prepare a connection
    URLConnection conn = url.openConnection();
    conn.connect(); // now connect to the Website

    // this is a bit of virtual plumbing as we connect
    // the data coming from the connection to a buffered
    // reader that reads the data one line at a time.
    BufferedReader in = new
      BufferedReader(new InputStreamReader(conn.getInputStream()));

    // read each line from the feed
    while ((data = in.readLine()) != null) {

      StringTokenizer st =
        new StringTokenizer(data,"\"<>,.()[] ");// break it down
      while (st.hasMoreTokens()) {
        // each chunk of data is made lowercase
        chunk= st.nextToken().toLowerCase() ;

        if (chunk.indexOf("love") >= 0 ) // found "love"?
          love++; // increment love by 1
        if (chunk.indexOf("peace") >= 0) // found "peace"?
          peace++; // increment peace by 1
        if (chunk.indexOf("arduino") >= 0) // found "arduino"?
          arduino++; // increment arduino by 1
     }
    }

    // Set 64 to be the maximum number of references we care about.
    if (peace > 64)   peace = 64;
    if (love > 64)    love = 64;
    if (arduino > 64) arduino = 64;
    peace = peace * 4;     // multiply by 4 so that the max is 255,
    love = love * 4;       // which comes in handy when building a
    arduino = arduino * 4; // colour that is made of 4 bytes (ARGB)
  }
  catch (Exception ex) { // If there was an error, stop the sketch
    ex.printStackTrace();
    System.out.println("ERROR: "+ex.getMessage());
```

```
  }

}
```

Есть две вещи, которые вам потребуются для правильного запуска скетча Processing. Во-первых, вы должны указать Processing создать шрифт, который мы будем использовать для скетча. Чтобы сделать это, создайте и сохраните этот скетч. Затем, при открытом скетче, щёлкните меню "Tools" и выберите "Create Font". Выберите шрифт с именем "HelveticaNeue-Bold", выберите 32 для размера шрифта, а затем щёлкните "OK".

Во-вторых, вам требуется удостовериться в том, что скетч использует правильный последовательный порт для связи с Arduino. Вам надо подождать, пока мы не соберём схему с Arduino и загрузим в неё скетч. На большинстве систем скетч Processing будет работать нормально. Однако если вы видите что на плате Arduino ничего не происходит и на экран не выводится никаких данных от датчика света, найдите комментарий "IMPORTANT NOTE (ВАЖНОЕ ЗАМЕЧАНИЕ)" в скетче Processing и следуйте следующим инструкциям.

```
  // IMPORTANT NOTE:
  // The first serial port retrieved by Serial.list()
  // should be your Arduino. If not, uncomment the next
  // line by deleting the // before it, and re-run the
  // sketch to see a list of serial ports. Then, change
  // the 0 in between [ and ] to the number of the port
  // that your Arduino is connected to.
  // ВАЖНОЕ ЗАМЕЧАНИЕ
  // Первый последовательный порт, полученный функцией Serial.list(),
  // должен быть подключён к Arduino. Если это не так, раскомментируйте следующую
  // строку удалением // перед ней, и перезапустите
  // скетч чтобы увидеть список последовательных портов. Затем измените
  // 0 в между скобками [ и ] на номер порта, к которому
  // подключена ваша Arduino.
  //println(Serial.list());
```

Вот скетч для Arduino (также доступен на www.makezine.com/getstartedarduino):

Пример 6-2. Сетевая лампа с Arduino

```
#define SENSOR 0
#define R_LED 9
#define G_LED 10
#define B_LED 11
#define BUTTON 12

int val = 0; // variable to store the value coming from the sensor

int btn = LOW;
int old_btn = LOW;
int state = 0;
char buffer[7] ;
int pointer = 0;
byte inByte = 0;

byte r = 0;
byte g = 0;
byte b = 0;

void setup() {
  Serial.begin(9600); // open the serial port
  pinMode(BUTTON, INPUT);
}

void loop() {
  val = analogRead(SENSOR); // read the value from the sensor
```

```
  Serial.println(val);      // print the value to
                            // the serial port

if (Serial.available() >0) {

  // read the incoming byte:
  inByte = Serial.read();

  // If the marker's found, next 6 characters are the colour
  if (inByte == '#') {

    while (pointer < 6) { // accumulate 6 chars
      buffer[pointer] = Serial.read(); // store in the buffer
      pointer++; // move the pointer forward by 1
    }
      // now we have the 3 numbers stored as hex numbers
      // we need to decode them into 3 bytes r, g and b
      r = hex2dec(buffer[1]) + hex2dec(buffer[0]) * 16;
      g = hex2dec(buffer[3]) + hex2dec(buffer[2]) * 16;
      b = hex2dec(buffer[5]) + hex2dec(buffer[4]) * 16;

      pointer = 0; // reset the pointer so we can reuse the buffer

    }
  }

  btn = digitalRead(BUTTON); // read input value and store it

  // Check if there was a transition
  if ((btn == HIGH) && (old_btn == LOW)){
    state = 1 - state;
  }

  old_btn = btn; // val is now old, let's store it

  if (state == 1) { // if the lamp is on

    analogWrite(R_LED, r); // turn the leds on
    analogWrite(G_LED, g); // at the colour
    analogWrite(B_LED, b); // sent by the computer
  } else {

    analogWrite(R_LED, 0); // otherwise turn off
    analogWrite(G_LED, 0);
    analogWrite(B_LED, 0);
   }

  delay(100);             // wait 100ms between each send
}

int hex2dec(byte c) { // converts one HEX character into a number
    if (c >= '0' && c <= '9') {
      return c - '0';
    } else if (c >= 'A' && c <= 'F') {
      return c - 'A' + 10;
    }
}
```

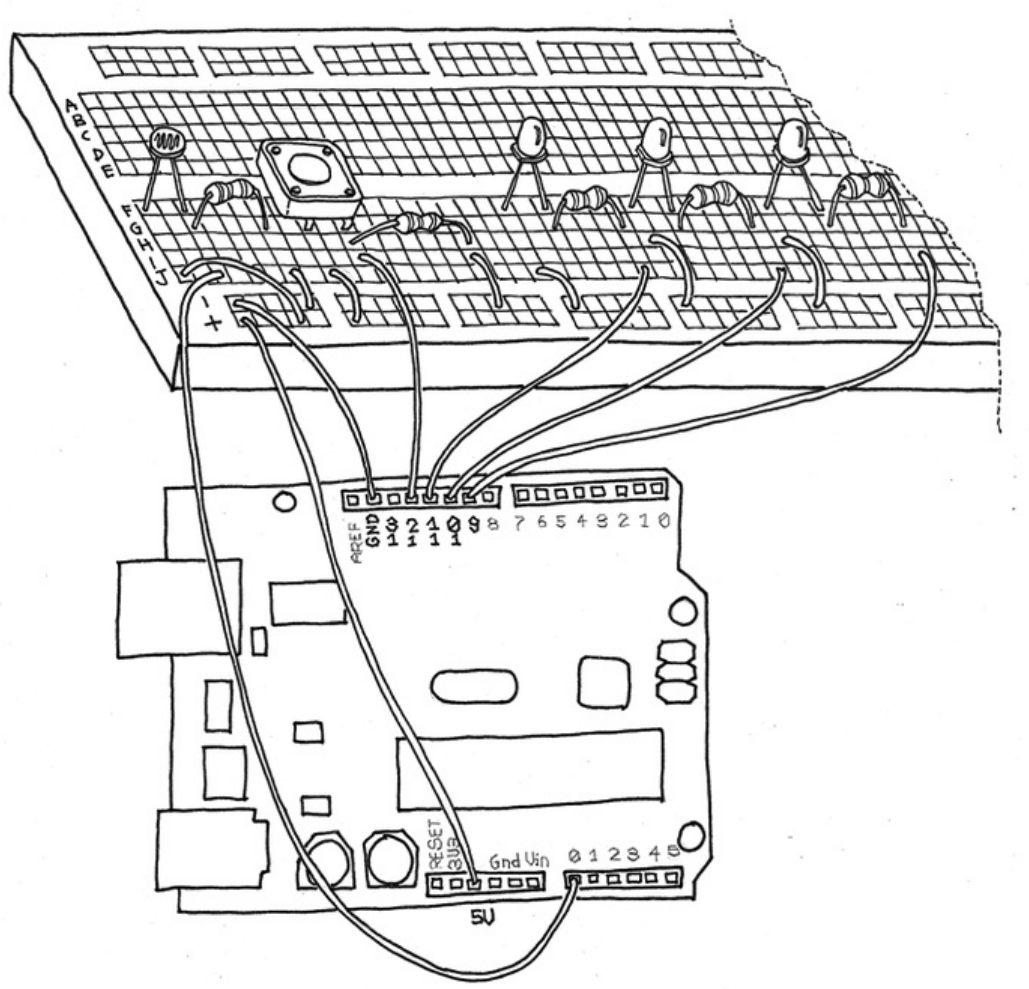
6.4 Сборка схемы

Рис. 6-2 показывает как собрать схему. Вам понадобится резисторы на 10 кОм (все резисторы,

показанные на схеме, хотя для резисторов, подключённых к светодиодам, можно использовать меньшие значения).

Вспомните пример с ШИМ из главы 5: светодиоды имеют полярность: в этой схеме длинный вывод (анод) подключается справа, а короткий (катод) - слева. (большинство светодиодов имеют фаску со стороны катода, как показано на рисунке).

Рис. 6-2. Схема сетевой лампы с Arduino



Соберите схему как показано на рисунке, используя один красный, один зелёный и один синий светодиоды. Затем загрузите скетчи в Arduino и Processing и запустите их. Если случились какие-то проблемы, см. главу 7. Устранение неполадок.

Теперь давайте завершим устройство, поместив макетную плату в стеклянный шар. Самый простейший и дешёвый выход - купить настольную лампу "FADO" в магазине IKEA. Сейчас она продаётся за US$14.99/€14.99/£8.99 (ага, прекрасно быть европейцем).

Вместо использования трёх отдельных светодиодов вы можете использовать один RGB-светодиод с четырьмя выводами. Подключите его так-же как и светодиоды на рис. 6-2, с единственным отличием - вместо трёх отдельных выводов, подключённых на вывод GND платы Arduino (называемый общий катод, или земля), у вас будет только один провод на землю.

6.5 Как собрать лампу:

Распакуйте лампу и удалите кабель от лампы к подставке. Вы больше не будете включать её в розетку.

Укрепите Arduino на макетной плате и приклейте плату к лампе.

Припаяйте длинные провода к светодиодам и приклейте их в нужном месте внутри лампы. Подключите провода от светодиода к плате (откуда вы их вытащили). Помните, что для RGB-светодиода требуется только один земляной провод.

Или найдите подходящий кусок дерева с отверстием, который может быть использован как основание для лампы, или просто отрежьте у картонной коробки для лампы верх (примерно 5 см) и сделайте отверстие для неё. Проклейте внутренности картонной коробки клеевым пистолетом по всем граням чтобы придать жёсткость коробке.

Поместите шар на основание и подключите USB-кабель к компьютеру.

Запустите код Processing, нажмите кнопку "вкл/выкл" и следите как оживает лампа.

Как упражнение попробуйте добавить код, который включит лампу в тёмной комнате. Другие возможные улучшения:

- Добавить датчик наклона для включения или выключения лампы поворотом
- Добавить маленький датчик движения для включения лампы если кто-то есть поблизости и для выключения если никого рядом нет
- Создать различные режимы для ручного управления цветом или заставить лампу плавно менять все цвета.

Подумайте о разных вещах, поэкспериментируйте и получите удовольствие!

Глава 7. Устранение неполадок

В ваших экспериментах настанет момент когда ничего не будет работать и вы должны будете разобраться в проблеме. Устранение неполадок и отладка - древние искусства, у которых есть несколько простых правил, но большинство результатов получаются посредством тяжёлой работы.

Чем больше вы работаете с электроникой и Arduino, тем больше вы поймёте и получите опыт, который сделает процесс работы менее болезненным. Не обескураживайтесь проблемами, с которыми вы встретитесь - они не так страшны как выглядели вначале.

Так как любой проект на основе Arduino состоит из программного и аппаратного обеспечений, то для поиска причины неприятностей будет больше одного места. В поисках проблемы вам следует следовать трём правилам:

7.1 Понимание

7.1.1 Понимание

Попробуйте понять как можно больше о том как работают используемые вами части и какой вклад они вносят в готовый проект. Такой подход позволит вам использовать какой-либо способ проверить каждый компонент в отдельности.

7.1.2 Упрощение и разделение

Древние римляне говорили: "divide et imper", что означает "разделяй и властвуй". Попробуйте разбить (мысленно) проект на его компоненты с использованием знаний где начинается и заканчивается ответственность каждого компонента.

7.1.3 Исключение и уверенность

При исследование проверьте каждый компонент отдельно для того чтобы быть абсолютно уверенным что он работает. Вы будете постепенно наращивать понимание того какие части проекта делают свою работу, а какие являются сомнительными.

Отладка (debugging) - это термин, используемый для определения такого процесса применительно к программированию. Легенды говорят, что в первый раз он был использован Грейс Хоппер в 1940-м, когда компьютеры были электромеханическими и один из них перестал работать из-за того что в механизм залезло настоящее насекомое (bug, баг).

Множество из современных багов не являются физическими: они виртуальны и невидимы. Поэтому иногда они требуют длительного и утомительного процесса идентификации.

7.2 Проверка платы

Что делал самый первый пример, "мигание светодиодом"? Разве это был он скучен? Давайте посмотрим что можно сделать.

Перед тем как начать обвинять свой проект, вам стоит проверить несколько вещей в порядке, подобном тому как пилоты самолёта проверят все системы перед вылетом.

Подключите свою Arduino к разъёму USB своего компьютера.

- Удостоверьтесь что компьютер включён (да, звучит глупо, но и такое случается). Если зелёный светодиод, обозначенные "PWR" светится, это означает что компьютер подаёт питание на плату. Если светодиод светится слабо, что-то случилось с питанием: попробуйте другой кабель USB и проверьте USB-порт компьютера, а также разъём USB платы Arduino на предмет повреждений. Если это не помогло, попробуйте другой USB-порт компьютера.
- Если Arduino только-что была куплена, светодиод "L" начнёт мигать в немного нервной манере; это тестовая программа, загруженная продавцом для проверки платы.
- Если вы используете внешний источник питания и старую плату Arduino (Extreme, NG или Diecimila), удостоверьтесь что источник питания включён и перемычка "SV1" замыкает два ближних к разъёму питания контакта.

Примечание: Если вы испытываете проблемы с другими скетчами и хотите проверить, работает-ли плата, откройте первый пример "мигание светодиодом" в IDE Arduino и выгрузите его в плату. Светодиод на плате должен начать мигать.

Если вы прошли все описанные шаги успешно, можете быть уверены что плата Arduino работает корректно.

7.3 Проверка схемы на макетной плате

Теперь подключите плату Arduino на макетной плате проводами от 5 V и GND к негативной и позитивной шинам. Если зелёный светодиод "PWR" выключился, немедленно отключите провода. Это означает что в вашей схеме есть большая ошибка и где-то произошло "короткое замыкание". Когда такое случается, ваша плата потребляет слишком большой ток и для защиты компьютера питание отключается.

Примечание: Если вы опасаетесь повредить свой компьютер, то знайте, что на многих компьютерах токовая защита достаточно хороша и срабатывает быстро. Также плата Arduino оснащена "полисвитчем" - прибором для защиты по току, который автоматически восстанавливается когда проблема будет устранена. Если вы настоящий параноик, вы всегда можете подключать Arduino через USB-хаб с отдельным питанием. В этом случае,если всё пойдёт очень плохо, помрёт только хаб, а не ваш компьютер.

Если у вас случилось короткое замыкание, начните процесс "упрощения и разделения". Всё что вам надо сделать, это пройтись по каждому сенсору в проекте и подключать их поочерёдно.

Первое, с чего следует начать - всегда питание (подключения 5 V и GND). Осмотрите их и удостоверьтесь что все части схемы подключены к ним правильно.

Работать шаг за шагом и производить по одному изменению за раз - правило номер один при устранении неполадок. Это правило было вбито в мою молодую голову моим школьным учителем и первым работодателем, Маурицио Пирола. Каждый раз когда я отлаживал что-то и дела шли плохо (и, поверьте, это случалось часто), в моей голове всплывало его лицо со словами: "одно изменение за раз ... одно изменение за раз" и это обычно помогало. Это очень важно, так как что вы поймёте что устранило проблему (очень легко потерять след всех изменений, которые решили проблему, поэтому так важно делать их один за другим).

Каждый опыт отладки будет выстраивать у вас в голове "базу знаний" неполадок и их возможных решений. И после тем как вы поймёте это, вы станете экспертом. Вы будете выглядет круто, так как новичок скажет "Оно не работает!", а вы бросите беглый взгляд и дадите ответите в несколько секунд.

7.4 Выделение проблемы

Другое важное правило - найти надёжный способ воспроизвести проблему. Если ваша схема ведёт себя странно в разные случайные моменты времени, попробуйте точно определить момент, в

котором неполадка случается и что её могло вызвать. Такой процесс позволит вам подумать о возможной причине. Также очень полезно когда вам требуется пояснить кому-нибудь то, что вы делаете.

Описание проблемы настолько точно, насколько это возможно, это также хороший способ найти решение. Попробуйте пояснить кому-нибудь проблему - во многих случаях решение всплывёт в голове как только вы озвучите проблему. Брайан В. Керниган и Роб Пайк в книге "Практика программирования" (изд-во Addison-Wesley, 1999), рассказали историю, в которой один университет "держал плюшевого мишку возле службы поддержки. Студенты с загадочными проблемами должны были пояснить их мишке прежде чем пообщаться с консультантом".

7.5 Проблемы с IDE

Иногда у вас могут быть проблемы с использованием интегрированной среды разработки Arduino, в частности, в Windows.

Если вы получаете ошибку при двойном щелчке на иконке Arduino, или не происходит вообще ничего, попробуйте запустить двойным щелкчком файл run.bat. Это альтернативный способ запуска Arduino.

Пользователи Windows могут также получить проблему если операционная система назначает COM-порту для Arduino номер COM10 или больше. Если такое произошло, обычно вы можете уговорить Windows назначить более низкий номер порта. Сначал откройте диспетчер устройств, нажав кнопку "Пуск", щёлкнув правой кнопкой на "Компьютер" (Vista) или "Мой компьютер" (XP) и выбрав "Свойства". В Windows XP щёлкните на "Оборудовани" и выберите "Диспетчер устройств". Для Vista щёлкните "Диспетчер устройств" (он расположен в списке приложений в левой части окна).

Найдите последовательные устройства в списке "Порты (COM и LPT)". Найдите последовательное устройство, которое вы не используете и имеющее номер COM9 или ниже. Правой кнопкой мышки щёлкните на нём и выберите "Свойства". Затем выберите закладку "Параметры порта" и нажмите кнопку "Дополнительно...". Установите номер COM в COM10 или выше, нажмите "OK" и ещё раз "OK" чтобы закрыть окно "Свойства".

Теперь сделайте то-же самое с последовательным портом USB, который представляет Arduino, с одним отличием - установить номер порта COM (COM9 или ниже), который вы только-что освободили.

Если эти подсказки вам не помогли, или у ваша проблема не описана выше, просмотрите страничку устранения неполадок Arduino: www.arduino.cc/en/Guide/Troubleshooting.

7.6 Как получить помощь онлайн

Если вы застряли, не тратьте дни на решение проблемы самостоятельно - попросите о помощи. Одна из лучших вещей у Arduino - это сообщество. Вы всегда найдёте помощь если сможете достаточно хорошо описать проблему.

Возьмите за правило копировать и вставлять текст в поисковую систему и убеждаться что кто-то уже говорил об этом.

Например, когда Arduino IDE выплевывает неприятное сообщение об ошибке, скопируйте и вставьте его в поиск Google и посмотрите, что получится. Сделайте то же самое с кода, с которым вы работаете или только определенным именем функции. Осмотритесь: все, что уже было изобретено раньше и оно хранится где-то на веб-страницах.

Для дальнейших изысканий начните с главного сайта www.arduino.cc и посмотрите на часто задаваемые вопросы и ответы на них (FAQ, www.arduino.cc/en/Main/FAQ), затем пройдитесь по "детской площадке" (playground, www.arduino.cc/playground), свободно редактируемой вики, где любой пользователь может поспособствовать улучшению документации. Это одна из лучших частей философии открытого кода. Люди способствуют созданию документации и примеров, которые вы можете применить к Arduino. Перед началом проекта поищите в детской площадке и вы найдёте части кода или схем, которые помогут вам начать.

Если вы всё ещё не можете найти ответ, поищие на форуме (www.arduino.cc/cgi-bin/yabb2/YaBB.pl). Если он не содержит ответа, задайте в форуме вопрос. Выбирайте правильный раздел форума для

своего вопроса: есть несколько разделов для проблем с программным или аппаратным обеспечанием, и даже форумы на разных языках. Пожалуйста, дайте как можно больше информации:

- Какую Arduino вы используете?
- Какую операционную системы вы используете для запуска Arduino IDE?
- Дайте описание того, что вы хотите сделать. Укажите ссылки к базам данных нестандартных компонентов, которые используются.

Количество ответов зависит от того, как вы сформулируете свой вопрос.

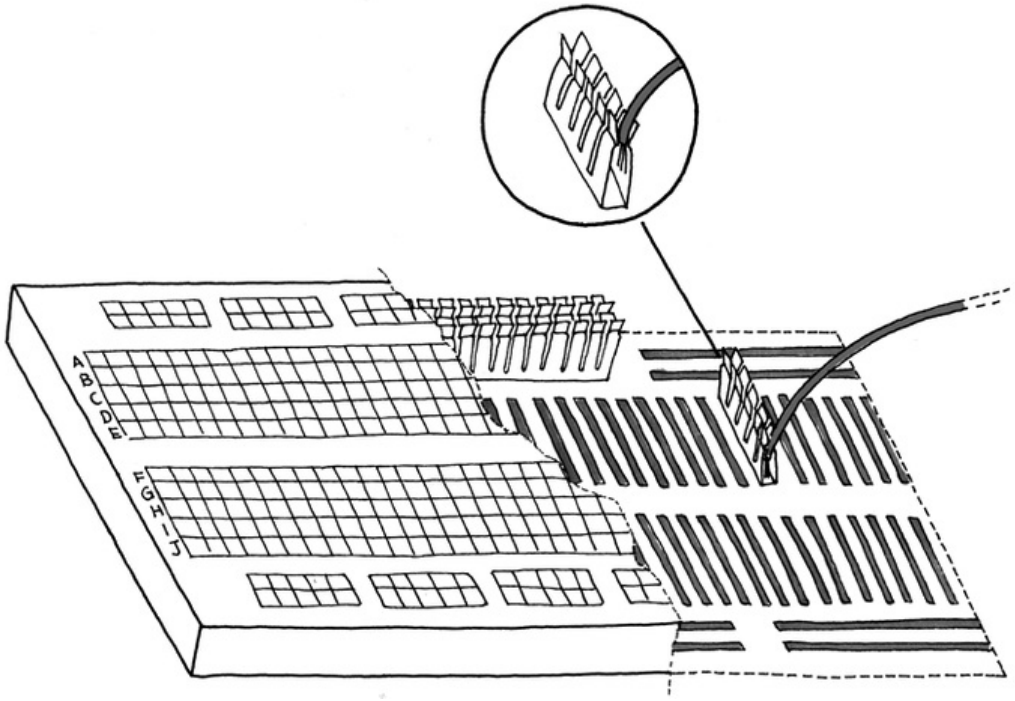
Ваши шансы увеличатся если вы будете избегать следующих вещей (эти правила одинаковы для любых форумой, не только об Arduino):

- Писать своё сообщение ЗАГЛАВНЫМИ БУКВАМИ. Это сильно раздражает людей и подобно хождению по улицам с татуировкой "новичок" на лбу (в онлайн-сообществах писать заглавными буквами означает "кричать")
- Писать одинаковые сообщения в разных разделах форума
- "Подталкивание" своих сообщений вопросами "Что, никто не ответил?", или, что ещё хуже, просто "вверх". Если вы не получили ответа, просмотрите своё сообщение. Понятна-ли тема? Вы хорошо описали проблему? Вы были вежливы? Всегда будьте вежливым.
- Сообщения вида "Хочу построить космический шаттл на Ардуино как мне это сделать". Это означает что вы хотите чтобы на вас работал кто-то ещё, что для настоящего самоделкина невесело. Лучше обьяснить что вы хотите сделать, а затем задать определённый вопрос об одной части проекта.
- Если вы зададите вопрос, то люди будут рады помочь, но если вы попросите их сделать всю работу (и не поделитесь деньгами), ответов не получите.
- Писать сообщения, подозрительно похожие на просьбы сделать вам домашнее задание. Преподаватели, такие как я, бродят по форумам и наказывают провинившихся студентов.

Приложение A. Макетная плата

Процесс создания рабочей схемы включает в себя много изменений до тех пор пока оно не станет нормальным; это очень быстрый, интерактивный процесс, который похож на электронный эквивалент создания зарисовок. Конструкция развивается в ваших руках когда вы пробуете различные комбинацииц. Для достижения лучших результатов используйте систему, позволяющую изменять соединения сежду компонентами самым быстрым, наиболее практичным и наименее разрушающим способом. Такие требования полностью исключают пайку, являющуюся длительной процедурой и прикладывающей напряжение к деталям каждый раз, когда вы нагреваете и охлаждаете их.

Решением данной проблемы является очень практичное устройство, называемое беспаечной макетной платой. Вы можете увидеть её на рис. A-1. Это небольша пластиковая пластина со множеством отверстий, каждое из которых содержит пружинный контакт. Вы можете вставить ножку компонента в одно из отверстий и получить электрический контакт со всеми отверстиями, расположенными в той-же вертикальной колонке отверстий. Каждое отверстие находится на расстоянии 2,54 мм от других.



Поскольку у большинства компонентов ножки (ещё называемые "пинами") расположены со стандартным шагом, чипы со множеством ножек отлично устанавливаются. Не все контакты платы сделаны одинаковыми - есть небольшая разница. Верхний и нижний горизонтальные ряды (помеченные красным и синим цветом и обозначенные "+" и "-") соединены горизонтально и используются для подключения питания по всей плате, так-что когда вам требуется питания или земля, вы можете быстро получить их, проведя перемычку (короткий отрезок провода для соединения двух точек схемы). И последнее что вам следует знать в макетной плате - это то, что посередине платы есть большой разрыв, соизмеримый с самой маленькой микросхемой. Все вертикальные линии прерываются посередине, так-что когда вы установите в плату микросхему, у неё не будут замкнуты ножки по двум стронам микросхемы. Умно, да?

Приложение B. Маркировка резисторов и конденсаторов

Для того чтобы использовать электронные компоненты, вы должны уметь различать их, что может быть трудной задачей для начинающего. Большинство сопротивлений, которые вы можете найти в магазинах, имеют цилиндрическую форму с двумя выводами и странную цветовую маркировку по кругу. Когда производились первые коммерческие резисторы, способа напечатать на них такие маленькие знаки ещё не было, так-что умные инженеры решили что они могут обзоначать величины полосками цветной краски.

Теперь начинающие должны понять как расшифровать такие знаки. "Ключ" очень прост: вообще существует четыре полоски, и каждый цвет обозначает отдельную цифру. Одно из колец, обычно золотистого цвета, обозначает точность сопротивления. Для правильного прочтения кода держите сопротивления так чтобы золотистая полоска была справа. Затем читайте цвета и заменяйте их соответстсвующими цветами. В следующей таблице показано соответствие между цветами и их числовыми значениями:

Цвет	Значение
Чёрный	0
Коричневый	1
Красный	2
Оранжевый	3
Жёлтый	4
Зелёный	5
Синий	6
Красный	7
Серый	8

Белый	9
Серебристый	10%
Золотистый	5%

Например, коричневый, чёрный, оранжевый и золотистый означают 103±5%. Просто, не так-ли? Не очень, так как здесь есть выверт: третья полоска означает количество нулей в значении. Поэтому 1 0 3 означает десятку с тремя нулями, в итоге получается результат 10000 Ом ±5%. Электронные ботаники любят сокращать эти значения в килоомы (тысячи ом) или мегаомы (миллионы ом) , так-что обычно 10000 Ом превращается в 10k тогда как 10000000 становится 10M. Пожалуйста, заметьте что инженеры любят улучшать всё подряд, и на некоторых схемах вы можете увидеть 4k7, что означает 4,7 килоОма, или 4700, что означает то-же самое.

С конденсаторами немного легче: конденсатор в форме бочонка (электролитический) чаще всего имеет надпись со своим номиналом. Емкость конденсатора измеряется в фарадах (F), но большинство конденсаторов, с которыми вы встретитесь, измеряются в микрофарадах (uF). Так-что если вы увидите конденсатор с надписью 100 uF, это конденсатор с ёмкостью 100 микрофарад.

Множество конденсаторов в форме диска (керамические) не имет надписей с указанием единиц и используют трёхзначный код, показывающий количество пикофарад (pF). В одном микрофараде 1000000 пикофарад. Подобно коду резисторов используйте третью цифру для определения числа нулей в значении, с одной разницей: если это цифра от 0 до 5, то она показывает число нулей; цифры 6 и 7 не используются; если-же вы видите 8, то умножайте первые две цифры на 0,01; а если видите девятку, то умножайте на 0,1.

Итак, если конденсатор обозначен маркировкой "104", значит его ёмкость равна 100000 пикофарад, или 0,1 uF. Конденсатор с надписью 229 имеет ёмкость 2,2 пФ.