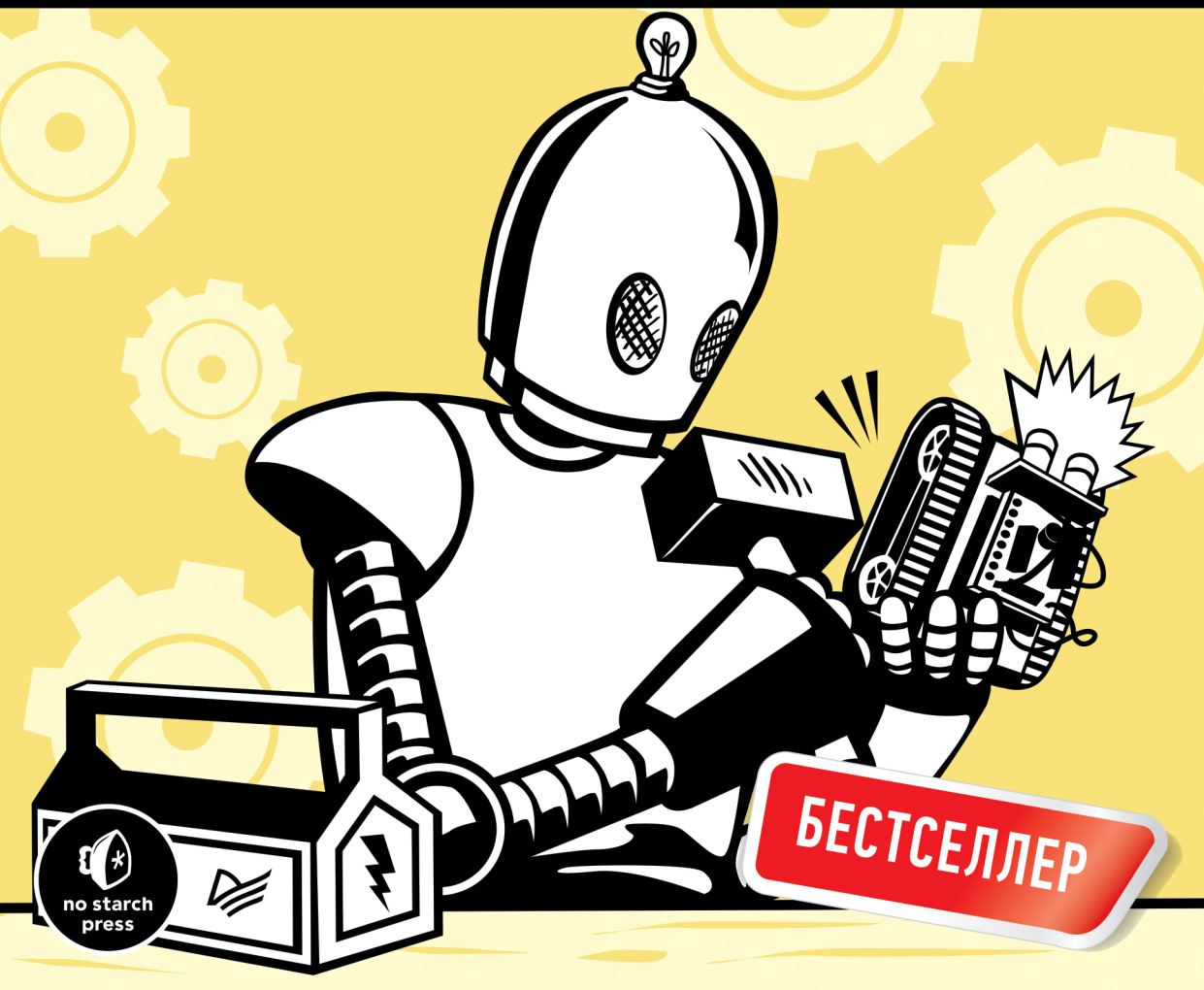
ИЗУЧАЕМ ARDUINO

65 ПРОЕКТОВ СВОИМИ РУКАМИ

ДЖОН БОКСЕЛЛ



ARDUINO WORKSHOP

A Hands-On Introduction with 65 Projects

by John Boxall

no starch press

San Francisco

ДЖОН БОКСЕЛЛ

ИЗУЧАЕМ ARDUINO

65 ПРОЕКТОВ СВОИМИ РУКАМИ

25 лет ПИТЕР®

Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону
Самара • Минск

2017

ББК 32.973.23-018.2
УДК 004.3
Б78

Бокселл Дж.

Б78 Изучаем Arduino. 65 проектов своими руками. — СПб.: Питер, 2017. — 400 с.: ил. — (Серия «Вы и ваш ребенок»).

ISBN 978-5-496-02421-1

Что такое Arduino? За этим словом прячется легкое и простое устройство, которое способно превратить кучу проводов и плат в робота, управлять умным домом и многое другое. Прочитайте эту книгу и овладейте бесчисленными возможностями Arduino, позволяющими электронике взаимодействовать с окружающим миром.

Познакомившись с основами Arduino, вы быстро перейдете к работе с разнообразными электронными компонентами. А конкретные проекты позволят вам сразу закрепить знания на практике. Страница за страницей проекты будут становиться все более изощренными, сложными и интересными.

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.23-018.2
УДК 004.3

ISBN 978-1593274481 англ.
ISBN 978-5-496-02421-1

Краткое содержание

Оглавление

Посвящается маме и моей бесценной Катлин.
Они всегда верили в меня.

Благодарности

Прежде всего я хотел бы выразить огромную благодарность коллективу разработчиков Arduino: Массимо Банци (Massimo Banzi), Дэвиду Куартилье (David Cuartielles), Тому Иго (Tom Igoe), Джанлуке Мартино (Gianluca Martino) и Дэвиду Меллису (David Mellis). Без вашей прозорливости, идей и упорного труда не было бы этого проекта.

Огромное спасибо моему техническому редактору Марку Александру (Marc Alexander) за его содействие, опыт, предложения, поддержку, идеи, терпение и стремление во что бы то ни стало довести этот труд до логического конца.

Я также хочу выразить глубокую признательность за понимание и поддержку следующим компаниям: Adafruit Industries, Keysight Technologies, Gravitech, Freetronics, Oomlout, Seeed Studio, Sharp Corporation и SparkFun. Кроме того, мне хочется отдельно поблагодарить Freetronics за то, что они предоставили мне великолепные электронные компоненты. Спасибо всем тем, кто потратил свое время на создание библиотек для Arduino, которые упрощают работу многим разработчикам.

Выражаю свое уважение и благодарность команде Fritzing за их превосходный открытый инструмент проектирования электронных схем, которым я пользовался на протяжении всего периода работы над книгой.

Моя глубокая благодарность всем, кто вдохновлял и поддерживал меня: Ирафне Чайлдс (Iraphne Childs), Лимору Фриду (Limor Fried), Джонатану Оксеру (Jonathan Oxer), Филипу Линдси (Philip Lindsay), Николь Килах (Nicole Kilah), Кену Ширрифу (Ken Shirriff), Натану Кеннеди (Nathan Kennedy), Дэвиду Джонсу (David Jones) и Натану Зайделю (Nathan Seidle).

И наконец, я хочу сказать спасибо сотрудникам издательства No Starch Press: Сондре Сильверхоук (Sondra Silverhawk) за предложение написать книгу, Серене Янг (Serena Yang) за литературную правку, бесконечное терпение и ценные предложения и Биллу Поллоку (Bill Pollock) за поддержку, советы и способность убедить, что иногда авторское объяснение того или иного аспекта не является лучшим и единственным.

1 Введение

Приходилось ли вам, разглядывая какое-нибудь устройство, задумываться над тем, как оно работает *в действительности*? Возможно, это был катер с дистанционным управлением, лифт, автомат по продаже напитков или электронная игрушка? А может быть, вам хотелось самому создать робота, придумать электронное управление для модели железной дороги? Или у вас вдруг возникало желание организовать получение и анализ долгосрочного прогноза погоды? Как и с чего вы могли бы начать собственный проект?

Плата Arduino (рис. 1.1) поможет на практике раскрыть некоторые секреты электроники. Созданная Массимо Банци и Дэвидом Куартиллье, система Arduino предлагает бюджетный способ создания интерактивных проектов и объектов, таких как дистанционно управляемые роботы, системы записи пройденного маршрута на основе GPS и электронные игры.

Родившись в 2005 году, проект Arduino рос и умножался в геометрической прогрессии. Теперь это процветающая индустрия, поддерживаемая сообществом людей,

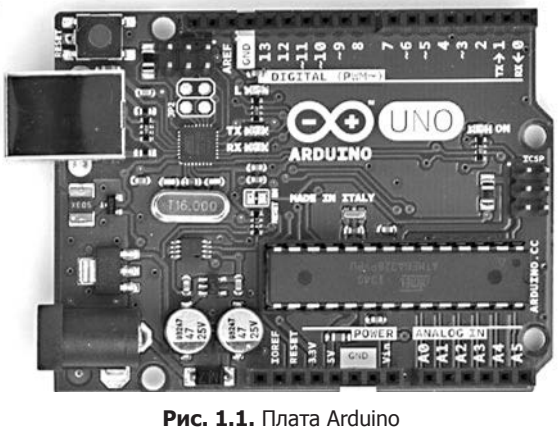


Рис. 1.1. Плата Arduino

объединенных общим стремлением к поиску нового. Здесь вы найдете и отдельных энтузиастов, и целые группы, от простых кружков «по интересам» и клубов до объединений специалистов и образовательных учреждений, заинтересованных в развитии Arduino.

Чтобы получить представление о разнообразии проектов на основе Arduino, достаточно воспользоваться поисковой системой. Вы очень быстро обнаружите множество групп, предлагающих вводные программы и курсы, проводимые творчески мыслящими специалистами.

Бесконечность не предел!

Быстро пролистав эту книгу, вы увидите, что плату Arduino можно использовать для создания различных устройств, от самых простых, незатейливо мигающих светодиодами, до очень сложных, взаимодействующих с сотовым телефоном и между делом решающих множество разнообразных задач.

Например, взгляните на устройство, созданное Филипом Линдси и изображенное на рис. 1.2. Оно принимает текстовые сообщения с сотового телефона и выводит их на большое табло. Это устройство реализовано на плате Arduino и плате расширения поддержки сотовой связи, способной принимать текстовые сообщения с других телефонов (подобное устройство описывается в проекте 65). Текстовое сообщение выводится на пару больших недорогих светодиодных матриц, это делает его доступным большой аудитории.

Рис. 1.2. Табло для вывода содержимого коротких сообщений SMS

Большие табло, легко подключаемые к Arduino, имеются в продаже, поэтому вам не придется конструировать их самим. (За дополнительной информацией обращайтесь по адресу *http://www.labradoc.com/i/follower/p/project-sms-text-scroller*.)

А не хотите ли вы сделать оригинальное предложение руки и сердца? Тайлер Купер сделал такое предложение своей девушке: он сконструировал устройство, которое назвал «шкатулкой с обратным геокэшингом» (reverse geocache box). Оно представляет собой небольшую коробочку, в которой хранится обручальное кольцо

(рис. 1.3). Когда шкатулка оказывается в конкретном месте (определяется с помощью внутреннего модуля GPS), она открывается, и перед изумленной избранницей появляется обручальное кольцо и романтический текст. Вы легко создадите подобное устройство с помощью платы Arduino, приемника GPS и модуля жидкокристаллического дисплея (см. главу 13), кроме того, вам понадобится небольшой сервопривод, который, действуя как защелка, не позволит открыть шкатулку, пока она не окажется в нужном месте. Устройство имеет чрезвычайно простую конструкцию — вы изготовите такой сюрприз за несколько часов. Причем большую часть времени придется потратить на выбор подходящей шкатулки, в которой поместятся все компоненты. (За дополнительной информацией обращайтесь по адресу *http://learn.adafruit.com/reverse-geocache-engagement-box/*.)

Рис. 1.3. Волшебная шкатулка на основе Arduino с предложением руки и сердца

А вот еще один пример. Как-то раз Курт Шульц озаботился определением уровня заряда аккумулятора на своем мопеде. Поразмыслив и придя к выводу, что подобное на плате Arduino реализовать чрезвычайно просто, он трансформировал идею простого определителя заряда в проект устройства, которое назвал «Скутерпьютер», — полноценную систему контроля для мопеда. Его скутерпьютер измеряет напряжение на аккумуляторе плюс отображает скорость, пройденное расстояние, угол наклона, температуру, время, дату, координаты GPS и многое другое. Он также содержит плату расширения сотовой связи для дистанционного управления, что позволяет определить местоположение мопеда и заглушить двигатель, если мопед был похищен. Скутерпьютер, изображенный на рис. 1.4, управляется с помощью небольшого сенсорного экрана. Каждую его функцию можно рассматривать как простой строительный блок, и вы легко создадите подобную систему за пару выходных. (За дополнительной информацией обращайтесь по адресу *http://www.janspace.com/b2evolution/arduino.php/2010/06/26/scooterputer/*.)

Рис. 1.4. Дисплей скутерпьютера (фотография любезно предоставлена Куртом Шульцем)

Джон Сарик, увлекающийся игрой в судоку и конструированием с применением цифровых индикаторов компании Nixie, нашел практическое применение своим увлечениям и создал компьютер для игры в судоку с большим табло, содержащим 81 цифровой индикатор! Пользователь может играть на полноценном игровом поле 9 × 9, управляемом платой Arduino, которая одновременно проверяет допустимость ходов. Несмотря на то что этот проект можно по праву назвать масштабным, его конструкция не особенно сложна, и любой желающий без труда может воспроизвести его. Большие размеры устройства делают его прекрасным дополнением в интерьере, если повесить его на стену, как показано на рис. 1.5. (За дополнительной информацией обращайтесь по адресу *http://trashbearlabs.wordpress.com/2010/07/09/nixie-sudoku/*.)

Группа разработчиков компании Oomlout создала устройство TwypeWriter на основе Arduino. Они приспособили плату Arduino с платой расширения Ethernet, подключенной к Интернету, для поиска сообщений в Twitter по определенным ключевым словам. Обнаруженные сообщения передаются на электронную печатающую машинку. Плата Arduino подключается к электронной схеме управления клавиатурой печатающей машинки, что позволяет ей печатать текст подобно человеку, как показано на рис. 1.6. (За дополнительной информацией обращайтесь по адресу *http://oomlout.co.uk/blog/twitter-monitoring-typewritter-twypwriter/*.)

Это лишь несколько случайных примеров, демонстрирующих возможности применения Arduino. Вашу фантазию ничто не ограничивает, и после прочтения этой книги вы можете приступить к реализации собственного оригинального проекта.

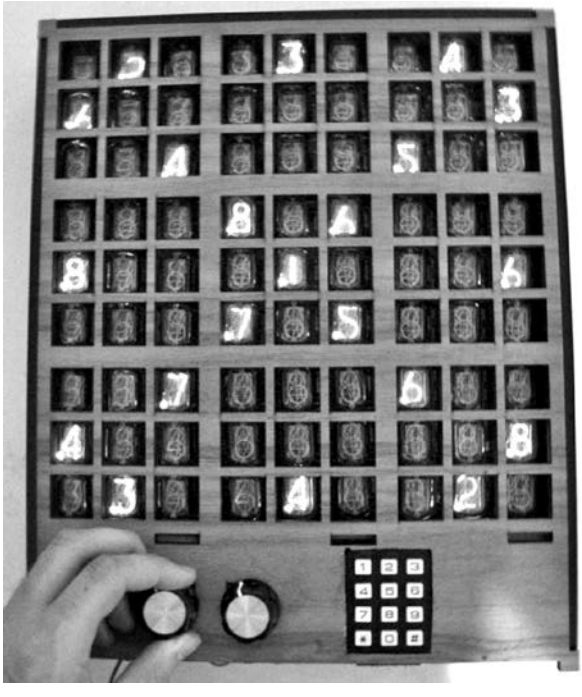


Рис. 1.5. Игра в судоку на цифровых индикаторах Nixie

Рис. 1.6. TwypeWriter

Сила в массовости

Популярность платформы Arduino растет с каждым днем. Если вам необходима поддержка со стороны и хотелось бы найти единомышленников, задайте поиск в Интернете по фразе «группы Arduino», и вы не только встретите группы единоверцев — энтузиастов Arduino, но и увидите, что они делают с применением этой платы. Члены групп Arduino помогут взглянуть на мир Arduino с творческой стороны. Во многих группах создаются маленькие Arduino-совместимые платы. В таких группах вы найдете массу интересного, познакомитесь с новыми людьми и поделитесь своими знаниями об Arduino с другими участниками.

ПРИМЕЧАНИЕ

Если вам потребуется помощь в выборе электронных компонентов либо в установке программного обеспечения или если у вас появились вопросы, касающиеся проектов, представленных в книге, обратитесь к автору этой книги в теме «Arduino Workshop» на форуме: *http://www.tronixforum.com/*. Кроме того, вы можете загрузить файлы скетчей и найти дополнения и обновления к тексту книги на сайте *http://nostarch.com/arduino/*.

Компоненты и аксессуары

Как и любые другие электронные устройства, плата Arduino распространяется множеством розничных продавцов, которые также предложат вам широкий выбор компонентов и аксессуаров. Я рекомендую, делая покупки, выбирать оригинальные версии Arduino, а не копии, иначе вы рискуете приобрести неисправный или некачественный товар; не стоит рисковать и приобретать плату неизвестного производителя, это увеличит как финансовые, так и физические и временные затраты на реализацию проекта. Список поставщиков Arduino можно найти по адресу *http://arduino.cc/en/Main/Buy/*.

Ниже приводится список поставщиков (в алфавитном порядке), к которым я рекомендую обращаться для приобретения компонентов и аксессуаров для создания проектов на основе Arduino:

- Adafruit Industries (*http://www.adafruit.com/*);
- Altronics (*http://www.altronics.com.au/*);
- DigiKey (*http://www.digikey.com/*);
- Jameco Electronics (*http://www.jameco.com/*);
- Jaycar (*http://www.jaycar.com.au/*);
- Newark (*http://www.newark.com/*);
- Oomlout (*http://www.oomlout.co.uk/*);
- SparkFun Electronics (*http://www.sparkfun.com/*);
- Tronixlabs (*http://tronixlabs.com/*).

Как вы увидите далее, я использую несколько Arduino-совместимых продуктов от компании Freetronics (*http://www.freetronics.com/*). Однако все необходимые компоненты широко распространены, и их можно найти у различных поставщиков.

Но давайте пока отложим поход в магазин и познакомимся с несколькими первыми главами, чтобы понять, что же потребуется в первую очередь, и не тратить деньги на то, что вам не нужно.

Необходимое программное обеспечение

Итак, возможность программировать платы Arduino предоставит почти любой компьютер, но при этом необходимо иметь программное обеспечение, которое называют *интегрированной средой разработки* (Integrated Development Environment, IDE). Это программное обеспечение может выполняться в следующих операционных системах:

- Mac OS X или выше.
- Windows XP 32- либо 64-разрядная или выше.
- Linux 32- либо 64-разрядная (Ubuntu или подобная ей).

Если таковой компьютер и программное обеспечение уже имеется, самое время загрузить и установить IDE. Для этого перейдите к разделу с названием операционной системы, установленной у вас, и следуйте инструкциям по установке. Не забудьте вместе с платой Arduino приобрести подходящий кабель USB. Даже если вы еще не приобрели плату Arduino, все равно загрузите IDE и займитесь ее исследованием. Поскольку IDE постоянно развивается, номер версии, приведенный в книге, может не совпадать с номером текущей версии, но сами инструкции по установке, я надеюсь, сохранили свою актуальность.

Mac OS X

В этом разделе вы найдете инструкции по загрузке и настройке Arduino IDE в Mac OS X.

Установка IDE

Чтобы установить IDE в операционной системе Mac OS, следуйте перечисленным ниже инструкциям:

1. В веб-браузере, например Safari, откройте страницу загрузки *http://arduino.cc/en/Main/Software/*, изображенную на рис. 1.7.
2. Щелкните на ссылке Mac OS X. Файл начнет загружаться и появится в окне Downloads (Загрузки), как показано на рис. 1.8.

Рис. 1.7. Страница загрузки IDE в Safari

Рис. 1.8. Загрузка файла завершена

3. По завершении загрузки дважды щелкните на файле, чтобы запустить процедуру установки. Если программа предложит вам открыть файл, щелкните на кнопке Open (Открыть). После этого загруженный файл будет преобразован в приложение.
4. Перетащите ярлык Arduino в папку Applications (Приложения) и отпустите кнопку мыши. На время копирования файла появится окно, отображающее ход копирования.
5. Теперь подключите плату Arduino к компьютеру с помощью кабеля USB. Через мгновение на экране появится диалог, изображенный на рис. 1.9.

Рис. 1.9. Новая плата Arduino определена.
Вместо слова Eleven у вас может появиться слово Uno

6. Щелкните на кнопке Network Preferences... (Настройки сети...) и затем на кнопке Apply (Применить) в диалоге Network (Сеть). Сообщение «not configured» («не настроена») можно игнорировать.

Настройка IDE

После загрузки IDE выполните следующие инструкции, чтобы запустить и настроить IDE:

1. Откройте папку Applications (Приложения) в диспетчере файлов Finder (изображен на рис. 1.10) и дважды щелкните на ярлыке Arduino.

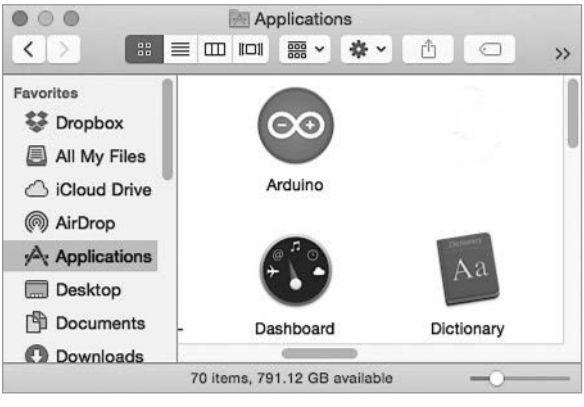


Рис. 1.10. Папка Applications (Приложения)

2. В этот момент может появиться окно, предупреждающее, что вы открываете веб-приложение. В этом случае нажмите Open (Открыть), чтобы продолжить. Перед вами на экране появится среда разработки Arduino IDE, как показано на рис. 1.11.
3. Вот практически и все, осталось всего два шага, после которых ваша Arduino IDE будет готова к работе. Сначала нужно сообщить среде разработки, к какому разъему подключена плата Arduino. Выберите пункт меню Tools ▸ Serial Port (Инструменты ▸ Порт) и затем пункт /dev/tty.usbmodem1d11 в подменю, как показано на рис. 1.12.
4. В заключение сообщите среде разработки тип подключенной платы Arduino. Это очень важный шаг, потому что платы Arduino могут существенно отличаться друг от друга. Например, если вы приобрели наиболее распространенную модель Uno, выберите пункт меню Tools ▸ Board ▸ Arduino Uno (Инструменты ▸ Плата ▸ Arduino Uno), как показано на рис. 1.13. Более подробно о различиях плат Arduino мы поговорим в главе 11.

Теперь аппаратное и программное обеспечение готово к работе. Переходите к разделу «Безопасность» на с. 38.

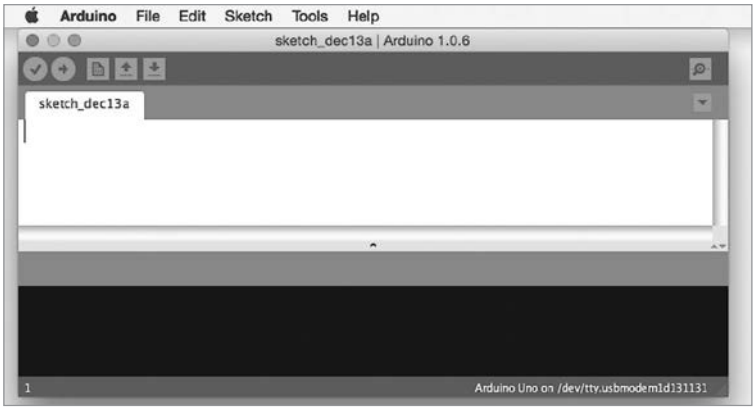


Рис. 1.11. Среда разработки Arduino IDE в Mac OS X

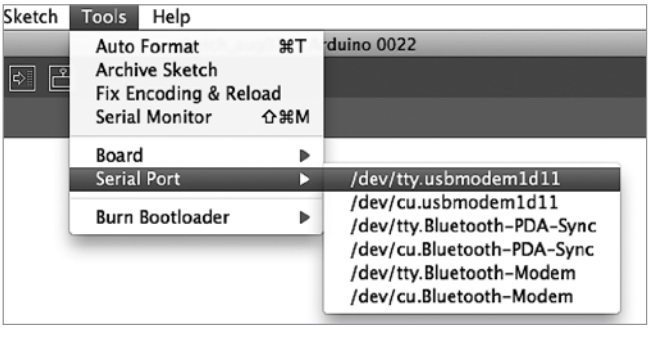


Рис. 1.12. Выбор порта USB

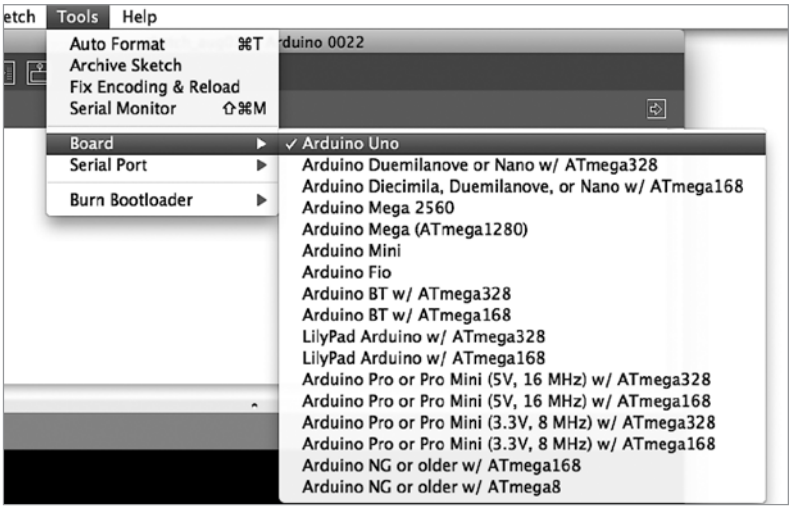


Рис. 1.13. Выбор типа платы Arduino

Windows XP и более поздние версии

В этом разделе вы найдете инструкции по загрузке Arduino IDE, установке драйверов и настройке IDE в Windows.

Установка IDE

Чтобы установить Arduino IDE в операционной системе Windows, следуйте перечисленным ниже инструкциям:

1. В веб-браузере, например Firefox, откройте страницу загрузки *http://arduino.cc/en/Main/Software/*, изображенную на рис. 1.14.

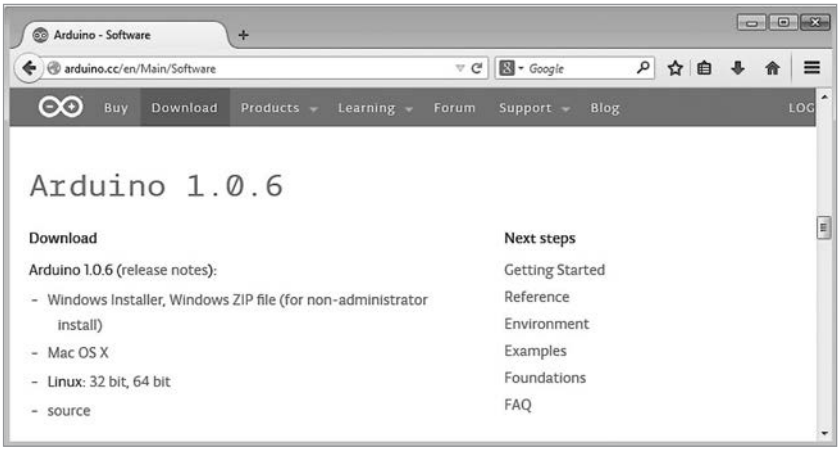


Рис. 1.14. Страница загрузки IDE в Firefox

2. Щелкните по ссылке Windows. На экране появится диалог, изображенный на рис. 1.15. Выберите пункт Open with Windows Explorer (Открыть в Проводнике (по умолчанию)) и затем нажмите OK. Файл начнет загружаться, как показано на рис. 1.16.
3. По завершении загрузки дважды щелкните на файле, и на экране появится окно, изображенное на рис. 1.17.
4. Скопируйте папку с именем *arduino-1.0.6-windows* (или другим, похожим) в место, где вы храните свои приложения. Когда копирование завершится, найдите папку и откройте ее, чтобы увидеть пиктограмму приложения Arduino, как показано на рис. 1.18. Для упрощения запуска приложения в будущем создайте ярлык на рабочем столе.

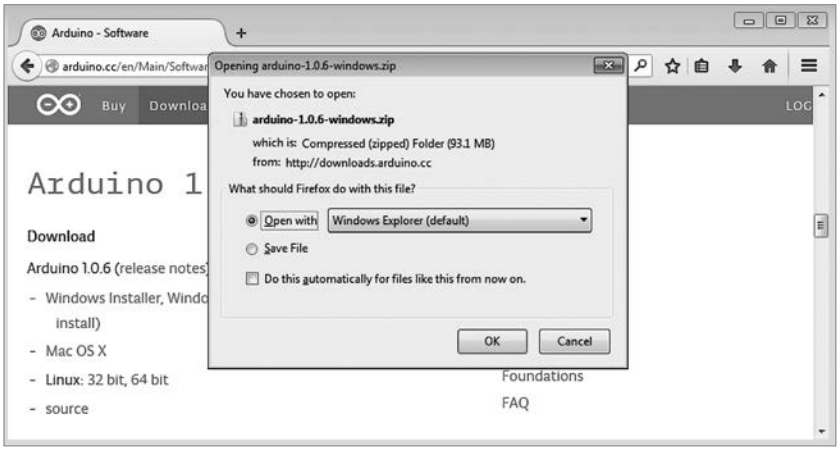


Рис. 1.15. Загрузка файла

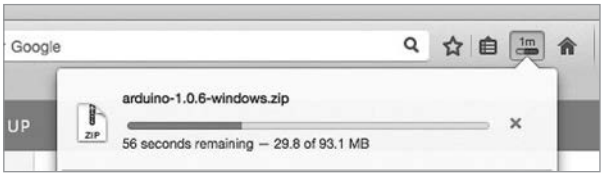


Рис. 1.16. Firefox отображает ход загрузки

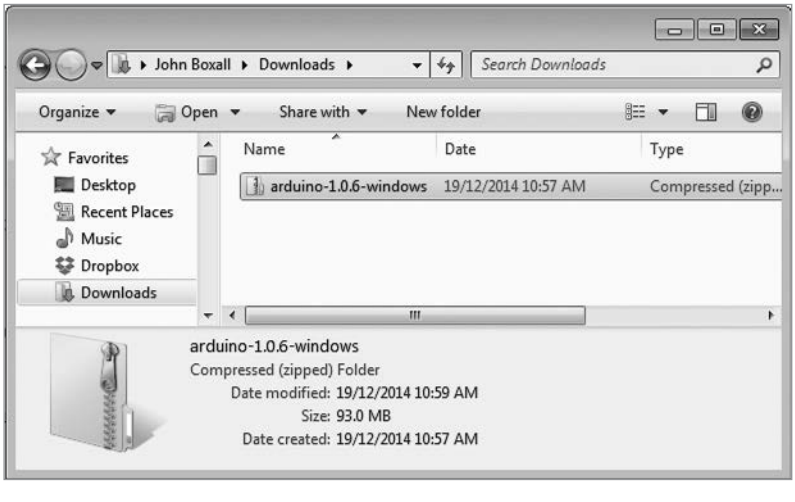


Рис. 1.17. Пакет IDE

Рис. 1.18. Папка с IDE и пиктограммой приложения Arduino

Установка драйверов

Следующая задача — установка драйверов USB для платы Arduino. Для этого:

1. Подключите плату Arduino к компьютеру с помощью кабеля USB. Через несколько мгновений на экране появится сообщение об ошибке с вот таким или подобным текстом: «Device driver software not successfully installed» («Драйвер устройства не был установлен»). Закройте это сообщение.
2. Перейдите в Windows Control Panel (Панель управления Windows). Откройте Device Manager (Диспетчер устройств) и прокрутите содержимое окна, пока не встретите пункт Arduino, как показано на рис. 1.19.

Рис. 1.19. Диспетчер устройств

3. Щелкните правой кнопкой на пункте Arduino Uno в разделе Other Devices (Прочие устройства) и выберите пункт Update Driver Software (Обновить драйве-

ры...). Затем выберите пункт Browse my computer for driver software (Выполнить поиск драйверов на этом компьютере) в следующем появившемся диалоге. Появится другой диалог Browse For Folder (Искать драйверы в следующем месте); нажмите кнопку Browse (Обзор) и перейдите в папку *drivers*, находящуюся в папке вновь установленной Arduino IDE (как показано на рис. 1.20). Щелкните на кнопке OK.

Рис. 1.20. Выбор папки drivers

4. Щелкните на кнопке Next (Далее) в диалоге, чтобы перейти к следующему шагу. Если на экране возникнет надпись «cannot verify the publisher of the driver software» («Не удалось проверить издателя этих драйверов»), выберите Install this software anyway (Установить программное обеспечение драйвера в любом случае). После непродолжительной паузы Windows сообщит, что драйвер установлен, и укажет номер COM-порта, к которому подключена плата Arduino, как показано на рис. 1.21.

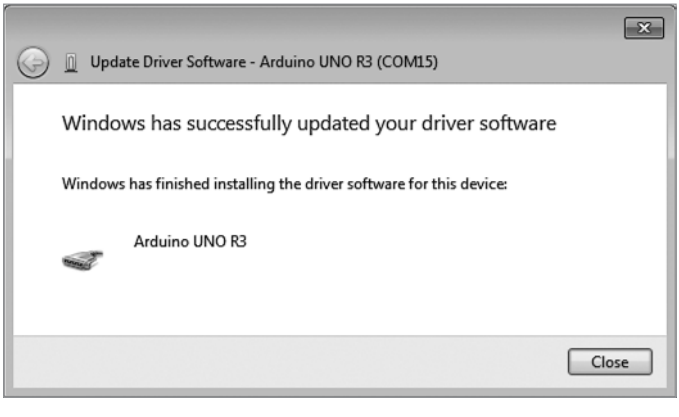


Рис. 1.21. Драйверы успешно установлены

Настройка IDE

Вот практически и все — осталось всего два шага до завершения настройки IDE.

1. Откройте Arduino IDE. Сначала сообщите среде разработки, к какому разъему подключена плата Arduino, для этого выберите пункт меню Tools ▸ Serial

Port (Инструменты ▸ Порт) и затем пункт COM с номером порта, который был указан в окне Update Driver Software (Обновить драйверы...).

2. В заключение сообщите среде разработки тип подключенной платы Arduino. Это очень важный шаг, потому что платы Arduino могут существенно отличаться друг от друга. Например, если вы приобрели наиболее распространенную модель Uno, выберите пункт меню Tools ▸ Board ▸ Arduino Uno (Инструменты ▸ Плата ▸ Arduino Uno), как показано на рис. 1.13. Более подробно о различиях плат Arduino рассказывается в главе 11.

Теперь Arduino IDE настроена. Переходите к разделу «Безопасность» на с. 38.

Ubuntu Linux 9.04 и выше

Если вы пользуетесь операционной системой Ubuntu Linux, ниже вы найдете инструкции по загрузке и настройке Arduino IDE в этой среде.

Установка IDE

Воспользуйтесь следующими инструкциями, чтобы установить IDE:

1. В веб-браузере, например Firefox, откройте страницу загрузки *http://arduino.cc/en/Main/Software/*, изображенную на рис. 1.22.

2. На странице загрузки найдите последнюю стабильную версию IDE, которая на момент написания этих строк имела номер 1.0.6. Затем щелкните на ссылке

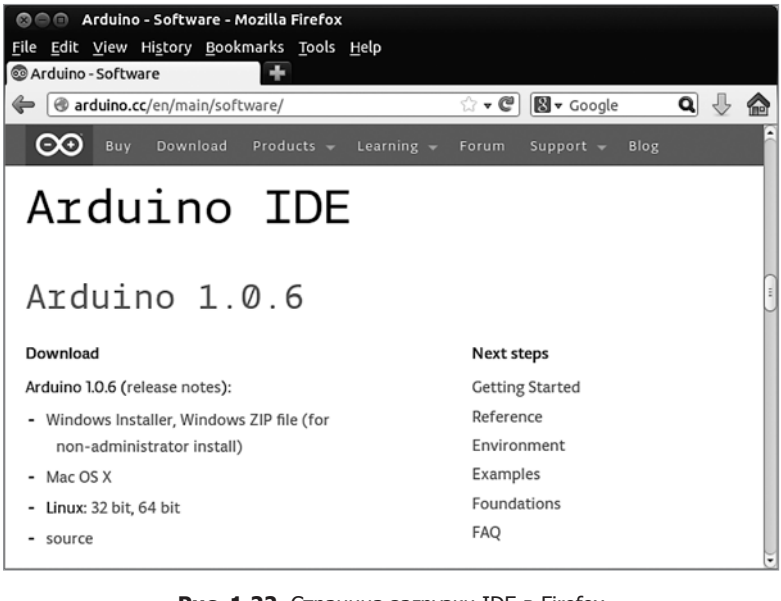


Рис. 1.22. Страница загрузки IDE в Firefox

Linux 32-bit или Linux 64-bit, в зависимости от разрядности вашей системы. Когда появится диалог, изображенный на рис. 1.23, выберите пункт Open with Archive Manager (Открыть в диспетчере архивов) и нажмите OK.

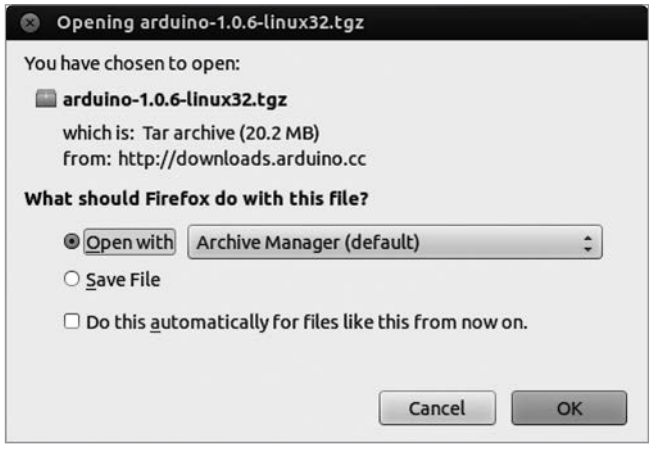


Рис. 1.23. Загрузка файла

3. По завершении загрузки файла откроется диспетчер архивов, как показано на рис. 1.24. Скопируйте папку с именем *arduino-1.0.6-windows* (или другим, похожим) в место, где вы храните свои приложения, или в свою домашнюю папку.

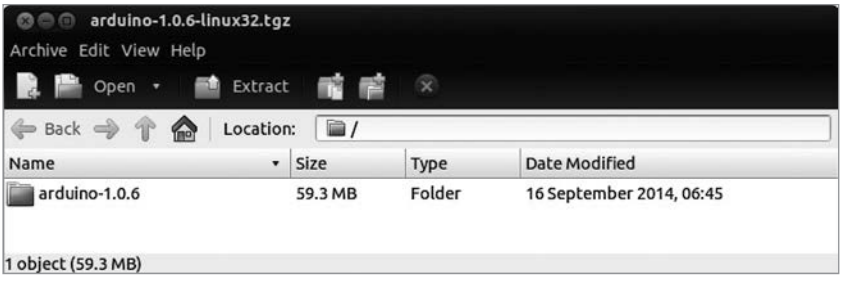


Рис. 1.24. Пакет IDE

Настройка IDE

Следующий этап — настройка IDE.

1. Подключите свою плату Arduino к компьютеру с помощью кабеля USB. Теперь можно запустить Arduino IDE, для этого найдите папку *arduino-1.0.6*, скопированную прежде, и дважды щелкните на файле *arduino*, который выделен на рис. 1.25.

Рис. 1.25. Папка с Arduino IDE и выделенным файлом arduino

2. Если появится диалог, изображенный на рис. 1.26, щелкните на кнопке Run (Выполнить), и перед вами на экране появится среда разработки Arduino IDE, как показано на рис 1.27.

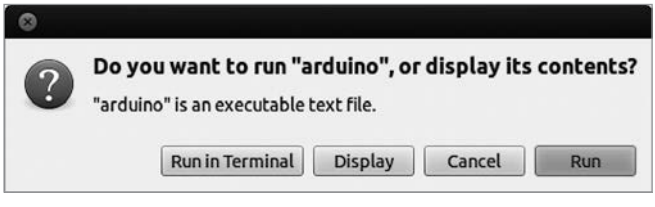


Рис. 1.26. Запрос разрешения на запуск IDE

Рис. 1.27. Окно IDE в Ubuntu

3. После запуска IDE ей нужно сообщить, к какому разъему подключена плата Arduino. Выберите пункт меню Tools ▸ Serial Port (Инструменты ▸ Порт) и затем порт /dev/ttyACMx в подменю, где x — цифра (для выбора должен быть доступен только один порт с подобным именем).
4. В заключение сообщите среде разработки тип подключенной платы Arduino. Это очень важный шаг, потому что платы Arduino могут существенно отличаться друг от друга. Например, если вы приобрели наиболее распространенную модель Uno, выберите пункт меню Tools ▸ Board ▸ Arduino Uno (Инструменты ▸ Плата ▸ Arduino Uno), как показано на рис. 1.13. Более подробно о различиях плат Arduino рассказывается в главе 11.

Теперь аппаратное и программное обеспечение готово к работе.

Безопасность

Каким бы хобби и ремеслом вы ни занимались, позаботьтесь о своей собственной безопасности и безопасности окружающих. Как вы вскоре увидите, я буду использовать в работе простые ручные инструменты, электрические устройства с питанием от аккумуляторов, разного рода ножи и резаки и иногда — паяльник. Ни в одном из представленных далее проектов вам не придется работать с напряжением обычной электросети. Оставим это специалистам-электрикам, это их вотчина. Но не забывайте, что прикосновение к оголенным проводам под высоким напряжением может убить вас.

Забегая вперед

Вы собираетесь пуститься в увлекательнейшее путешествие, в процессе которого будете создавать устройства, прежде казавшиеся фантастическими. В этой книге вас ждут 65 проектов на основе платы Arduino, от очень простых до относительно сложных. Все они были спроектированы специально, чтобы помочь вам научиться создавать что-нибудь полезное. Итак, в путь!

2 Знакомство с платой Arduino и IDE

В этой главе мы займемся исследованием платы Arduino и среды разработки Arduino IDE, которая будет использоваться для создания *скетчей* (так в мире Arduino называются программы) и выгрузки их в плату Arduino. Вы познакомитесь с базовой структурой скетча и некоторыми основными функциями, которые можно использовать в скетчах, а также создадите и выгрузите свой первый скетч.

Плата Arduino

Что такое Arduino? Согласно определению на веб-сайте Arduino (*http://www.arduino.cc/*), это открытая аппаратная платформа для макетирования электронных устройств, основанная на гибком и простом в использовании аппаратном и программном обеспечении. Она предназначена для художников, проектировщиков, радиолюбителей и всех, кто интересуется созданием интерактивных устройств.

Проще говоря, Arduino — это маленький компьютер, который можно запрограммировать для взаимодействия с различными физическими объектами с помощью входных и выходных сигналов разного вида. Основная модель Arduino, **Uno**, имеет небольшие размеры и легко умещается на ладони, как можно видеть на рис. 2.1.

Хотя на первый взгляд плата выглядит не очень внушительно, она позволяет создавать модели, взаимодействующие с окружающим миром. Используя практически неограниченный спектр устройств ввода и вывода, датчиков, индикаторов, дисплеев, электродвигателей и многих других, вы сможете запрограммировать любые взаимодействия, необходимые для создания функционального устройства. Например, художник может создать инсталляцию с множеством светодиодов, мигающих в такт движениям проходящих мимо посетителей, студенты — сконструировать автономного робота, который будет обнаруживать открытый огонь и гасить его,

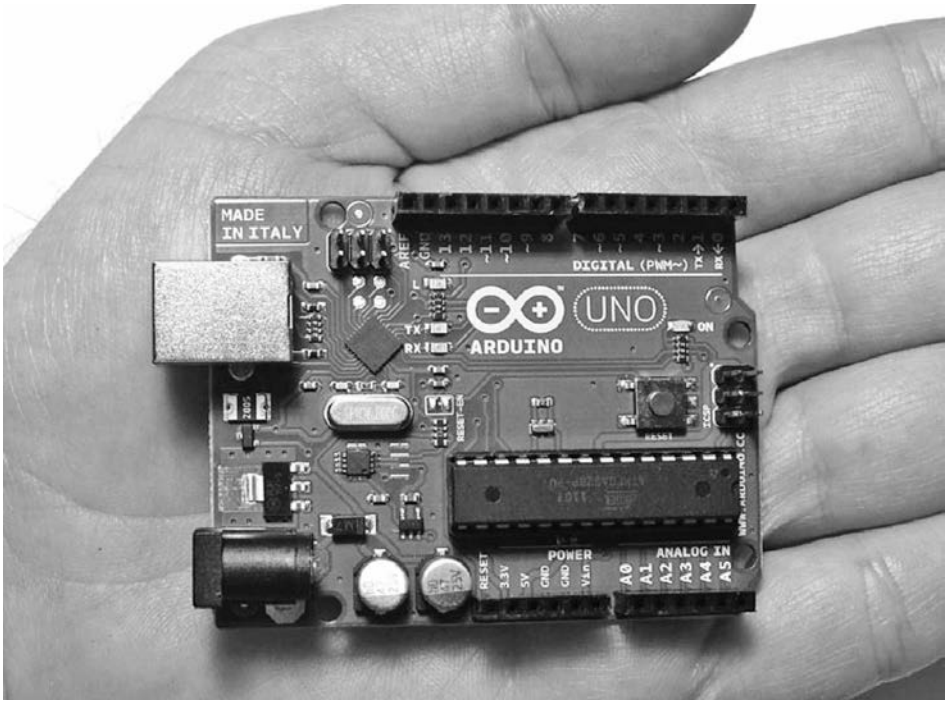


Рис. 2.1. Плата Arduino Uno имеет небольшие размеры

а синоптики — спроектировать систему измерения температуры и влажности и передавать эти данные в свои системы в виде текстовых сообщений. С помощью простого поиска в Интернете вы найдете бесчисленное множество примеров применения платы Arduino.

Теперь двинемся дальше, подробнее исследуем *аппаратную* часть Arduino Uno (то есть ее физическое устройство) и посмотрим, что мы имеем. Если что-то из материалов этой главы вам покажется непонятным, не расстраивайтесь, потому что все, о чем будет рассказываться, мы детально разберем в следующих главах.

Давайте осмотрим плату Uno со всех сторон. Поверните ее к себе стороной с разъемами, как показано на рис. 2.2.

Слева находится разъем подключения универсальной последовательной шины (Universal Serial Bus, USB). С его помощью плата подключается к компьютеру, это позволяет подать на нее напряжение питания, выгрузить скетч с инструкциями и отправить или принять данные с компьютера. Справа находится разъем подключения блока питания. С его помощью можно подключить плату к стандартному блоку питания.

В центре, чуть ниже середины, находится микроконтроллер, изображенный на рис. 2.3.

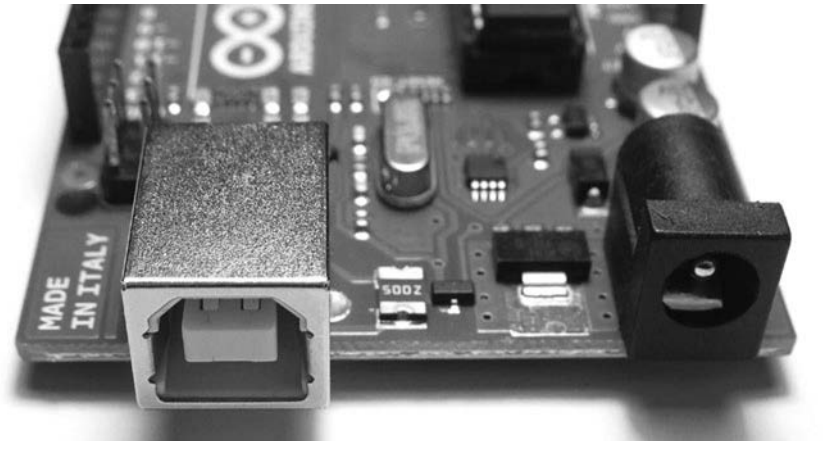


Рис. 2.2. Разъем USB и контакты подключения питания

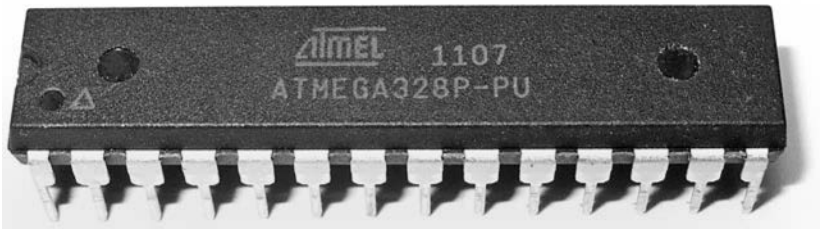


Рис. 2.3. Микроконтроллер

Микроконтроллер — это «мозг» Arduino, маленький компьютер, содержащий микропроцессор, выполняющий инструкции, включающий несколько видов памяти для хранения данных и инструкций и имеющий различные входы и выходы для вывода или ввода данных. Чуть ниже микроконтроллера располагаются два ряда разъемов, или контактов, как показано на рис. 2.4.

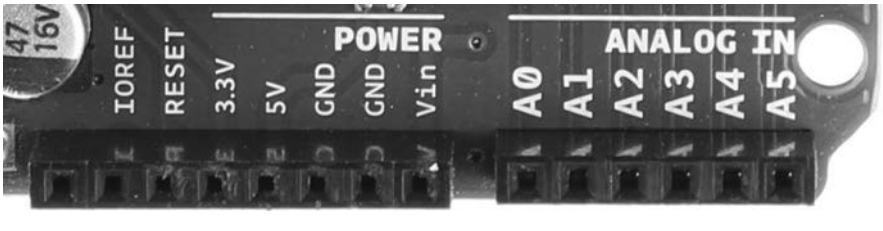


Рис. 2.4. Контакты электропитания и аналоговых входов

В первом ряду находятся контакты электропитания и контакты внешней кнопки сброса. Во втором ряду — шесть контактов аналоговых входов, они используются

для измерения уровней напряжения электрических сигналов. Кроме того, контакты A4 и A5 служат для обмена данными с другими устройствами. Вдоль верхнего края платы располагаются еще два ряда контактов, как показано на рис. 2.5.

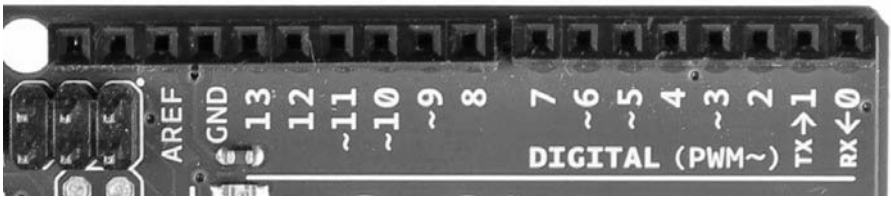


Рис. 2.5. Контакты цифровых входов/выходов

Контакты (или разъемы) с номерами от 0 до 13 — это цифровые входы/выходы. С их помощью можно определять наличие или отсутствие входных электрических сигналов или генерировать выходные сигналы. Контакты с номерами 0 и 1, также известные как *последовательный порт*, могут использоваться для обмена данными с другими устройствами, например с компьютером, через схему подключения к разъему USB. Контакты, отмеченные знаком тильды (~), могут также генерировать электрический сигнал переменного напряжения, — это нужно, например, для создания световых эффектов или управления электродвигателями.

Чуть ниже находятся несколько хорошо знакомых нам устройств, которые называют *светодиодами*. Как известно, они испускают свет, когда через них течет ток. На плате Arduino имеется четыре светодиода: один, с подписью ON, находится возле правого края и служит индикатором подключенного к плате электропитания, а другие три располагаются ближе к левому краю, рядом друг с другом, как показано на рис. 2.6.

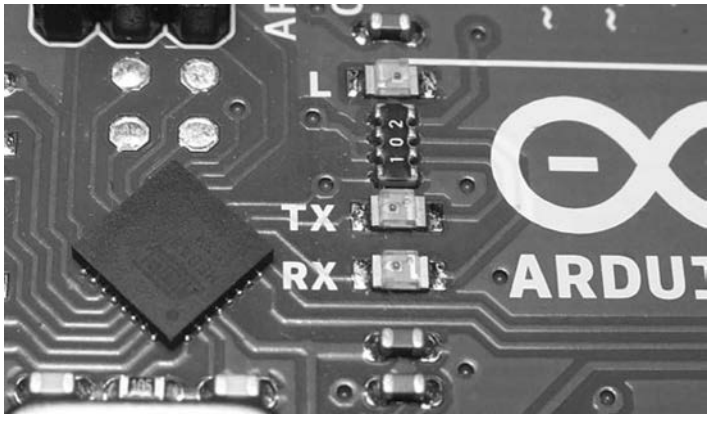


Рис. 2.6. Светодиоды на плате

Светодиоды с подписями RX и TX зажигаются в том случае, когда происходит обмен данными между платой Arduino и подключенными устройствами через последовательный порт и USB. Светодиод L предназначен для нужд пользователя (он подключен к контакту цифрового входа/выхода с номером 13). Небольшой черный квадрат слева от светодиодов — это микроконтроллер, управляющий интерфейсом USB. Именно он позволяет плате отправлять данные на компьютер или принимать их с компьютера, но вам едва ли придется иметь с ним дело.

И наконец, на рис. 2.7 показана кнопка RESET (сброс).

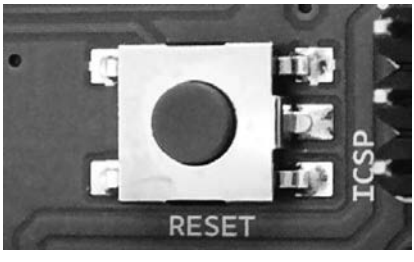


Рис. 2.7. Кнопка RESET (сброс)

Как это иногда случается с компьютерами, в плате Arduino тоже может что-то пойти не так. Когда все ваши попытки «привести ее в чувство» не дадут положительного результата, останется только нажать кнопку сброса и перезапустить Arduino. Простая кнопка RESET на плате (см. рис. 2.7) предназначена для перезапуска Arduino и решения проблем, не устранимых иными способами.

Одна из замечательных особенностей системы Arduino — простота ее расширения, то есть в нее легко добавляются новые аппаратные функции. Два ряда контактов вдоль каждой стороны Arduino позволяют подключить *плату расширения* — другую плату с контактами, которые можно включить в разъемы на плате Arduino. Например, на рис. 2.8 показана плата расширения с интерфейсом Ethernet, с помощью которой Arduino может обмениваться данными через сети и Интернет.

Примечательно, что плата расширения Ethernet также имеет ряды контактов. Это позволяет подключать сверху дополнительные платы расширений. Например, на рис. 2.9 показана еще одна плата с большими цифровыми индикаторами, датчиком температуры, дополнительным устройством хранения данных и большим светодиодом, вставленная сверху.

Обращаю ваше внимание, что вы должны знать и помнить назначение контактов платы расширения, — это позволит избежать «конфликтов». В продаже также имеются совершенно пустые платы, на которых можно собирать свои схемы. Об этой возможности мы поговорим в главе 8.

Аппаратную часть Arduino дополняет *программная* — коллекция инструкций, сообщающих аппаратной части, что делать и как делать. Существует два типа программного обеспечения, с которым вам придется работать: интегрированная среда разработки (Integrated Development Environment, IDE), которая обсуждается в этой главе, и ваши собственные скетчи.

Среда разработки устанавливается на персональный компьютер и используется для составления скетчей и выгрузки их в плату Arduino.

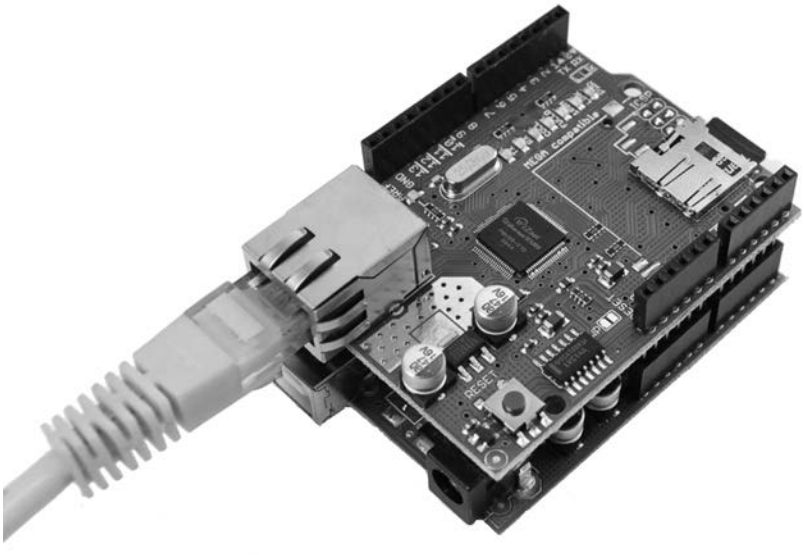


Рис. 2.8. Плата расширения Ethernet для Arduino

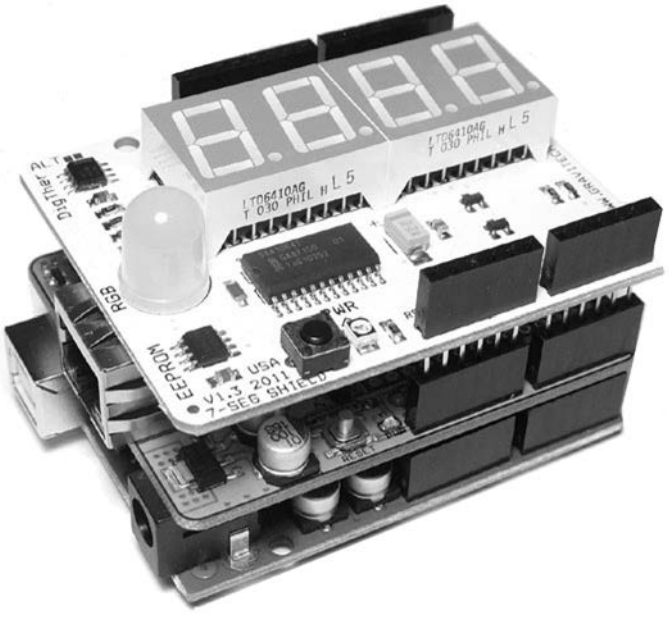


Рис. 2.9. Плата с цифровыми индикаторами и датчиком температуры

Обзор среды разработки

Как вы видите на рис. 2.10, среда разработки Arduino IDE напоминает простой текстовый процессор. Окно IDE делится на три основные области: область управления, область ввода текста и область вывода сообщений.

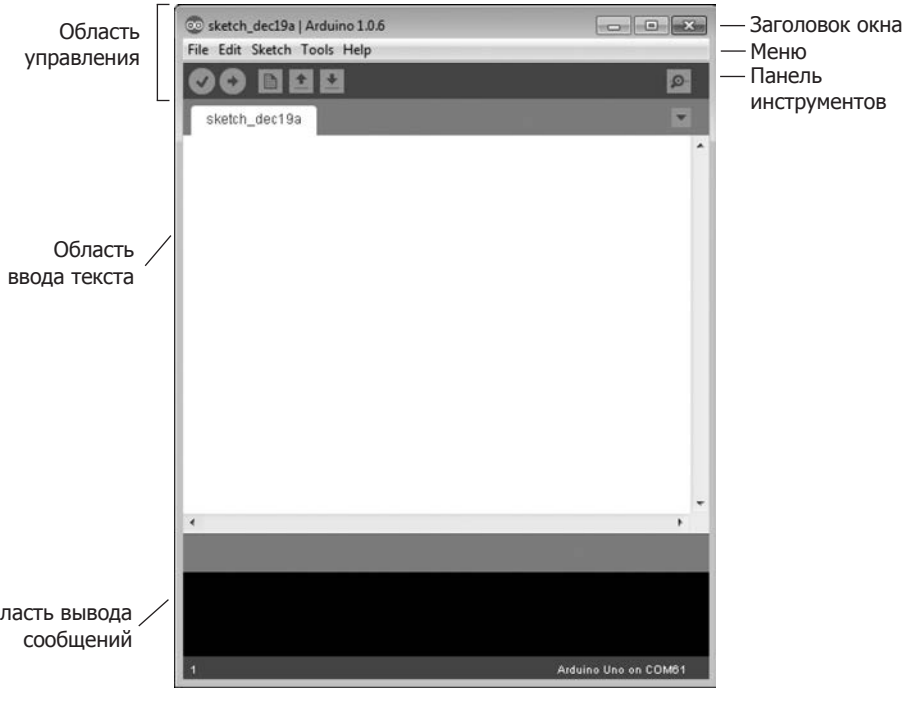


Рис. 2.10. Среда разработки Arduino IDE

Область управления

Область управления (вверху на рис. 2.10) содержит заголовок окна, меню и панель инструментов. В заголовке окна отображается имя файла скетча (*sketch_jun28a*) и версия IDE (*Arduino 1.0.5*). Ниже находится полоса меню (File (Файл), Edit (Правка), Sketch (Скетч), Tools (Инструменты) и Help (Помощь)) и панель инструментов с пиктограммами.

Меню

Так же как в любом текстовом процессоре или редакторе, щелчком на любом элементе меню выводится список соответствующих команд:

- File (Файл) содержит команды для сохранения, загрузки и печати скетчей; богатый набор примеров скетчей, которые можно открыть, а также подменю Preferences (Настройки);
- Edit (Правка) содержит команды копирования, вставки и поиска, обычные для любого текстового процессора;
- Sketch (Скетч) содержит команду для проверки скетча перед выгрузкой в плату, команду доступа к папке со скетчем и команды импортирования;
- Tools (Инструменты) содержит множество разных команд, а также команды для выбора типа платы Arduino и порта USB;
- Help (Помощь) содержит ссылки на разные темы и версию IDE.

Панель инструментов

Под полосой меню находится панель инструментов с шестью пиктограммами. Если навести указатель мыши на любую пиктограмму, появится название соответствующей команды. Ниже кратко описываются пиктограммы в порядке слева направо:

- Verify (Проверить) запускает проверку скетча на корректность и отсутствие программных ошибок;
- Upload (Загрузить) запускает проверку скетча и последующую его выгрузку в плату Arduino;
- New (Новый) открывает новый пустой скетч в новом окне;
- Open (Открыть) открывает ранее сохраненный скетч;
- Save (Сохранить) сохраняет открытый скетч. Если скетч еще не имеет имени, вам будет предложено ввести его;
- Serial Monitor (Монитор порта) открывает новое окно для обмена данными между платой Arduino и IDE.

Область ввода текста

Область ввода текста (посередине на рис. 2.10) служит для ввода исходного кода скетчей. Имя текущего скетча отображается во вкладке вверху слева над областью ввода текста. (По умолчанию скетчу присваивается имя, содержащее текущую дату.) В этой области вводится исходный код скетча, как в обычном текстовом редакторе.

Область вывода сообщений

Область вывода сообщений (внизу на рис. 2.10) имеет вид прямоугольника черного цвета в нижней части окна IDE. В ней будут выводиться самые разные со-

общения, включая результаты проверки скетчей, успешность выгрузки в плату и др.

Справа внизу под областью вывода сообщений отображается тип вашей платы Arduino и порт USB, к которому она подключена, — в данном случае *Arduino Uno on COM4*.

Создание первого скетча в IDE

Скетч для Arduino — это набор инструкций, определяющий пути решения стоящей перед вами задачи; иными словами, скетч — это *программа*. В этом разделе вы создадите и выгрузите в плату простой скетч, который заставит мигать светодиод на плате (рис. 2.11), включая и выключая его через интервалы времени в одну секунду.

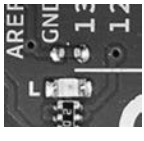


Рис. 2.11. Светодиод с меткой L на плате Arduino

ПРИМЕЧАНИЕ

Не стремитесь досконально разобраться в командах, которые вы увидите в скетче, представленном здесь. Его цель — показать, насколько просто заставить Arduino работать по заданной вами программе, поэтому, встретив что-то непонятное, не останавливайтесь и продолжайте читать.

Для начала подключите плату Arduino к компьютеру с помощью кабеля USB. Затем запустите IDE, выберите пункт меню Tools ▸ Serial Port (Инструменты ▸ Порт) и проверьте выбор соответствующего порта USB. Это позволит убедиться, что плата правильно подключена.

Комментарии

Сначала введем комментарий, напоминающий о назначении скетча. *Комментарий* — это примечание произвольной длины, находящееся непосредственно в исходном коде скетча и написанное для пользователя. Комментарии в скетчах позволяют оставлять примечания для себя или других, инструкции или описание важных деталей. Создавая скетчи для Arduino, всегда полезно добавлять комментарий, описывающий ваши намерения; подобные комментарии пригодятся вам и в том случае, если вы вновь вернетесь к скетчу.

Чтобы добавить однострочный комментарий, введите два слеша и текст комментария, например:

```
// Скетч мигает светодиодом. Автор: Мэри Смит, создан 01.07.13
```

Два идущих подряд слеша сообщают среде разработки, что она должна игнорировать текст, следующий за ними, во время проверки скетча. (Как упоминалось выше, выполняя команду проверки скетча, вы требуете от IDE проверить отсутствие ошибок в исходном коде скетча.)

Чтобы ввести комментарий, занимающий несколько строк, введите символы `/*` в строке, предшествующей комментарию, и символы `*/` в строке, следующей за комментарием, например:

```
/*
Скетч мигает светодиодом
Автор: Мэри Смит, создан 01.07.13
*/
```

Так же как два слеша в однострочных комментариях, комбинации символов `/*` и `*/` сообщают среде разработки, что она должна игнорировать текст, заключенный между ними.

Введите комментарий с описанием одним из этих способов и затем сохраните скетч, выбрав пункт меню File ▸ Save As (Файл ▸ Сохранить как...). Введите короткое имя скетча (например, *blinky*) и затем нажмите OK.

По умолчанию среда разработки сохраняет скетч в файлах с расширением *.ino* и добавляет его автоматически. То есть в данном случае скетч должен сохраниться в файле с именем *blinky.ino* и появиться в папке с вашими скетчами.

Функция setup

Следующий этап в создании любого скетча — добавление функции `void setup()`. Эта функция содержит инструкции, которые плата Arduino выполняет только один раз после каждого сброса или включения питания. Чтобы создать функцию `setup`, добавьте после комментария следующие строки:

```
void setup()
{
}
```

Управление аппаратными компонентами

Наша программа должна включать и выключать светодиод L на плате Arduino. Этот светодиод подключен к контакту цифрового входа/выхода с номером 13. Цифровые входы/выходы либо определяют наличие электрического сигнала, либо генерируют сигнал по команде. В этом проекте мы будем генерировать сигнал, включающий светодиод. На первый взгляд задача выглядит сложной, но не волнуйтесь, в сле-

дующих главах вы узнаете больше о цифровых входах/выходах. А пока просто продолжайте создание скетча.

Введите следующую строку между фигурными скобками ({ и }):

```
pinMode(13, OUTPUT); // настроить контакт 13 как цифровой выход
```

Число 13 в листинге представляет нужный нам контакт. Этот контакт настраивается на работу в режиме OUTPUT, то есть он будет генерировать (выводить — output) электрический сигнал. Если потребуется определять наличие входящего электрического сигнала, то следует указать режим INPUT. Обратите внимание, что вызов функции pinMode() завершается точкой с запятой (;). Вызовы любых функций в скетчах Arduino должны завершаться точкой с запятой.

Сохраните скетч еще раз, чтобы случайно не потерять результаты своих трудов.

Функция loop

Как вы помните, наша цель состоит в том, чтобы заставить светодиод включаться и выключаться снова и снова. Для этого создадим функцию loop, которую плата Arduino будет выполнять снова и снова, пока кто-то не выключит питание или не нажмет кнопку RESET (сброс).

Введите код, выделенный полужирным шрифтом в следующем листинге, чтобы создать пустую функцию loop. Не забудьте завершить новый раздел закрывающей фигурной скобкой (}) и затем снова сохраните скетч.

```
/*
Скетч мигает светодиодом
Автор: Мэри Смит, создан 01.07.13
*/
void setup()
{
 pinMode(13, OUTPUT); // настроить контакт 13 как цифровой выход
}
void loop()
{
 // поместите сюда код, который должен выполняться в цикле:
}
```

ВНИМАНИЕ

Среда разработки Arduino IDE не поддерживает автоматическое сохранение скетчей, поэтому старайтесь почаще сохранять свою работу.

Теперь поместите внутрь тела цикла void loop() вызовы функций.

Введите следующий код между фигурными скобками, ограничивающими тело функции loop, и затем щелкните на пиктограмме Verify (Проверить), чтобы убедиться, что при вводе не было допущено ошибок:

```
digitalWrite(13, HIGH); // включить напряжение на выходе 13
delay(1000); // пауза в одну секунду
digitalWrite(13, LOW); // выключить напряжение на выходе 13
delay(1000); // пауза в одну секунду
```

Давайте поближе познакомимся с этими командами. Функция `digitalWrite()` управляет напряжением, которое подается на цифровой выход: в данном случае на контакт `13`, к которому подключен светодиод `L`. Второй параметр в вызове этой функции (`HIGH`) указывает, что она должна установить «высокий» (high) цифровой уровень; в результате через контакт и светодиод `L` потечет электрический ток, и светодиод включится. (Если во втором параметре передать `LOW`, ток перестанет течь через светодиод, и он выключится.)

После включения светодиода вызовом `delay(1000)` выполняется пауза длительностью в 1 секунду. Функция `delay()` приостанавливает работу скетча на указанный период времени — в данном случае на 1000 миллисекунд, или 1 секунду.

Затем мы снимаем напряжение со светодиода вызовом `digitalWrite(13, LOW);`. В заключение вызовом `delay(1000);` выполняется еще одна пауза в одну секунду, в течение которой светодиод остается выключенным.

Ниже приводится полный исходный код скетча:

```
/*
Скетч мигает светодиодом
Автор: Мэри Смит, создан 01.07.13
*/
void setup()
{
 pinMode(13, OUTPUT); // настроить контакт 13 как цифровой выход
}
void loop()
{
 digitalWrite(13, HIGH); // включить напряжение на выходе 13
 delay(1000); // пауза в одну секунду
 digitalWrite(13, LOW); // выключить напряжение на выходе 13
 delay(1000); // пауза в одну секунду
}
```

Прежде чем продолжить, сохраните скетч!

Проверка скетча

Проверка скетча позволяет убедиться, что он не содержит ошибок и понятен плате Arduino. Для проверки законченного скетча щелкните на пиктограмме Verify (Проверить) в IDE и немного подождите. Когда проверка скетча будет закончена, в области вывода сообщений должен появиться текст, как показано на рис. 2.12.

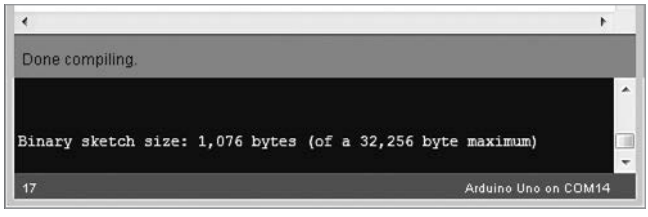


Рис. 2.12. Скетч успешно проверен

Сообщение «Done compiling» (Компиляция завершена) говорит о том, что скетч проверен и готов к выгрузке в плату Arduino. Здесь также показано, какой объем памяти будет использован (в данном случае 1076 байт) из имеющихся 32 256 байт.

А что, если проверка не увенчалась успехом? Допустим, к примеру, что вы забыли добавить точку с запятой после второго вызова функции `delay(1000)`. Если во время проверки в скетче обнаружатся ошибки, в области вывода сообщений должно появиться сообщение об ошибке, как показано на рис. 2.13.

Рис. 2.13. Информация об ошибке в области вывода сообщений

Данное сообщение говорит, что ошибка обнаружена в функции `void loop`, перечисляет номера строк, где, по мнению IDE, находится ошибка (`blinky:16` означает: «строка 16 в скетче blinky»), и выводит текст описания ошибки (`expected ';' before '}' token` — «отсутствует точка с запятой перед }»). Кроме того, среда разработки также выделяет в исходном коде строку с ошибкой или строку, следующую за ней. Это помогает быстро найти ошибку и устранить ее.

Загрузка и запуск скетча

Убедившись, что скетч был введен без ошибок, сохраните его, проверьте подключение платы Arduino и щелкните на пиктограмме Upload (Загрузить) на панели инструментов IDE. В ответ на это среда разработки проверит скетч еще раз и за-

тем загрузит его в плату. В процессе загрузки мигание светодиодов TX/RX на плате (рис. 2.6) покажет, что идет обмен информацией между Arduino и компьютером.

После этого наступит момент истины: плата Arduino начнет выполнять ваш скетч. Если все было сделано правильно, светодиод L начнет мигать с секундными интервалами.

Поздравляю! Теперь вы знаете, как ввести, проверить и загрузить скетч Arduino.

Изменение скетча

После запуска скетча у вас может возникнуть желание поменять его поведение, например изменить длительность интервала времени, когда светодиод находится во включенном или в выключенном состоянии. Поскольку IDE во многом напоминает текстовый процессор, вы можете открыть сохраненный скетч, изменить необходимые значения, снова сохранить скетч и загрузить его в плату. Например, чтобы увеличить частоту мигания, уменьшите параметры в обоих вызовах функции `delay` до одной четверти секунды, то есть до 250 миллисекунд, как показано ниже:

```
delay(250); // пауза в одну четвертую секунды
```

Затем снова загрузите скетч. После этого светодиод начнет мигать чаще, с интервалами в одну четверть секунды.

Забегая вперед

Вооруженные новоприобретенными знаниями и умениями ввода, правки, сохранения и выгрузки скетчей Arduino, вы готовы перейти к следующей главе, где познакомитесь с множеством других функций и принципами проектирования, сконструируете простую электронную схему и узнаете еще много интересного.

3 Первые шаги

В этой главе вы познакомитесь:

- ✓ с принципами проектирования;
- ✓ с основными свойствами электричества;
- ✓ с компонентами электронных схем: резисторами, светодиодами, транзисторами, выпрямительными диодами и реле;
- ✓ с макетными платами для навесного монтажа электронных схем;
- ✓ с целочисленными переменными, циклами `for` и цифровыми выходами

и узнаете, как с их помощью создавать разные эффекты с использованием светодиодов.

Далее вам предстоит вдохнуть жизнь в свою плату Arduino. Вы убедитесь, что круг решаемых задач не ограничивается одной только платой Arduino, и узнаете, как планировать проекты, чтобы воплотить идеи в жизнь. Затем перейдете к краткому учебнику по электричеству. Электричество — это движущая сила всего, о чем рассказывается в этой книге, поэтому для создания собственных проектов очень важно иметь полное понимание основ. Вы также познакомитесь с компонентами электронных схем, которые помогают воплощать проекты в действительность. А кроме того, вас ждут новые функции — строительные блоки скетчей Arduino.

Планирование проектов

На первых порах у вас может возникнуть соблазн немедленно приступить к написанию скетча, реализующего новую идею. Но прежде чем начать писать код, следует выполнить несколько подготовительных шагов. В конце концов, плата Arduino не способна читать ваши мысли — ей нужны точные инструкции, и даже если Arduino сможет выполнить эти инструкции, полученные результаты могут сильно вас удивить, если вы упустите из виду какую-нибудь мелочь.

Что бы вы ни задумали — проект, просто мигающий светодиодом или автоматизирующий модель железной дороги, в основе успеха всегда лежит подробный план. Планирование проекта для Arduino должно включать следующие простые шаги:

1. **Определение цели.** Определите, что вы хотите, что будет являться конечным результатом вашего проекта.
2. **Разработка алгоритма.** *Алгоритм* — это набор инструкций, описывающих работу проекта. Алгоритм должен перечислять шаги, выполнение которых приведет к желаемому результату.
3. **Выбор аппаратуры.** Определите, какое дополнительное аппаратное обеспечение будет подключаться к плате Arduino.
4. **Разработка скетча.** Создайте начальную версию программы, которая сообщит плате Arduino, что она должна делать.
5. **Соединение компонентов проекта.** Подключите дополнительные аппаратные компоненты, схемы и другие элементы к плате Arduino.
6. **Тестирование и отладка.** Что-то не работает? На этом этапе вы должны выявить ошибки и определить причины их возникновения, они могут быть заключены в скетче, в аппаратуре или в алгоритме.

Чем больше времени вы потратите на предварительное проектирование, тем меньше — на тестирование и отладку.

ПРИМЕЧАНИЕ

Даже детально проработанные проекты иногда становятся жертвой попытки расширения возможностей. Это происходит, если разработчик решает дополнить проект новыми функциональными возможностями и пытается внедрить новые элементы в имеющуюся конструкцию. Если вам понадобится изменить проект, не пытайтесь «втиснуть» в него новые элементы или что-то поменять в последний момент. Вместо этого возьмитесь за новый проект и начните с определения новых целей.

Об электричестве

Давайте теперь немного поговорим об электричестве, так как совсем скоро вам предстоит конструировать электрические схемы для своих проектов на Arduino. Выражаясь простым языком, *электричество* — это форма энергии, которую можно использовать и преобразовывать в тепло, свет или механическую работу. Электричество имеет три основных аспекта, важные для нас: сила тока, напряжение и мощность.

Сила тока

Электрическим током называется упорядоченное движение заряженных частиц. Несмотря на то что в большинстве материалов за протекание электрического тока отвечают отрицательно заряженные электроны, в электротехнике принято условно

считать, что электрический ток течет от положительного полюса источника питания, например батарейки, к отрицательному. Такой ток называют *постоянным током*. В примерах из этой книги мы не будем иметь дело с *переменным током*. На некоторых схемах отрицательный полюс обозначается как *земля* (ground, GND). Сила тока измеряется в *амперах* (А). Небольшие величины силы тока измеряются в *миллиамперах* (мА): 1000 миллиампер равна 1 амперу.

Напряжение

Напряжение представляет собой разность потенциалов между положительным и отрицательным полюсами схемы. Измеряется в *вольтах* (В). Чем выше напряжение, тем больше сила тока.

Мощность

Мощность является выражением скорости, с которой электрическое устройство преобразует энергию из одной формы в другую. Мощность измеряется в *ваттах* (Вт). Например, лампочка мощностью 100 Вт светит ярче, чем лампочка мощностью 60 Вт, потому что более мощная лампочка преобразует больше электрической энергии в свет и тепло за единицу времени.

Между напряжением, силой тока и мощностью существует простое математическое соотношение:

$$\text{мощность (Вт)} = \text{напряжение (В)} \times \text{ток (А)}.$$

Электронные компоненты

Теперь, когда вы вспомнили все, что знали об электричестве, давайте посмотрим, как действуют некоторые электронные компоненты и устройства. Электронные *компоненты* — это различные элементы, управляющие электрическим током в схемах и помогающие воплотить наши замыслы в реальность. Так же как различные детали в автомобиле, управляющие подачей топлива, мощностью двигателя и движением колес, помогают нам вести его, электронные компоненты, управляющие электрическим током и использующие его, помогают создавать полезные устройства.

На протяжении всей книги я буду описывать специализированные компоненты в той последовательности, в какой они будут появляться в наших проектах. Но о некоторых основных компонентах я расскажу прямо сейчас, в нескольких следующих разделах.

Резистор

Некоторые компоненты, такие как светодиоды на плате Arduino, требуют для работы небольшой силы тока — обычно около 10 мА. Когда светодиод получает избы-

точный ток, он преобразует избытки электрической энергии в тепловую, большое количество которой может его уничтожить. Чтобы уменьшить величину тока на таких компонентах, как светодиоды, между источником напряжения и компонентом следует добавить *резистор*. Ток свободно течет по медным проводникам, но когда на его пути встречается резистор, величина протекающего тока уменьшается. Часть электроэнергии преобразуется в тепловую энергию и рассеивается резистором. На рис. 3.1 показана пара типичных резисторов.

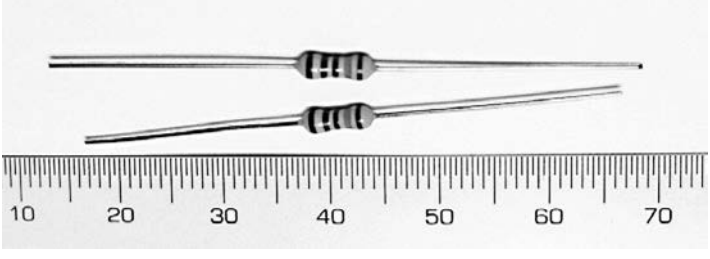


Рис. 3.1. Типичные резисторы

Сопротивление

Сопротивление резистора может быть постоянным или переменным. Сопротивление измеряется в омах (Ом) и может иметь величину от нуля до тысячи ом (*килоом*, или кОм) и даже миллионов ом (*мегаом*, или МОм).

Обозначение номиналов резисторов

Резисторы имеют очень маленький размер, поэтому их номинал (уровень сопротивления) обычно невозможно напечатать на корпусе самого компонента. Конечно же, сопротивление резистора можно измерить с помощью мультиметра, но его также можно определить по обозначениям, нанесенным на корпус. Часто роль таких обозначений играют цветные полоски, которые читаются слева направо, как описывается ниже.

- **Первая полоса.** Представляет первую цифру сопротивления.
- **Вторая полоса.** Представляет вторую цифру сопротивления.
- **Третья полоса.** Представляет множитель (для резисторов с четырьмя полосами) или третью цифру сопротивления (для резисторов с пятью полосами).
- **Четвертая полоса.** Представляет множитель для резисторов с пятью полосами.
- **Пятая полоса.** Определяет диапазон отклонений значений относительно номинала (точность).

В табл. 3.1 перечислены значения цветных полос, используемых для маркировки резисторов.

Таблица 3.1. Значения цветных полос, используемых для маркировки резисторов

Цвет	Цифра
Черный	0
Коричневый	1
Красный	2
Оранжевый	3
Желтый	4
Зеленый	5
Синий	6
Фиолетовый	7
Серый	8
Белый	9

Пятая полоса определяет *диапазон отклонений* значений относительно номинала. Она отражает точность соответствия сопротивления резистора его номиналу. Так как на практике довольно сложно производить резисторы с точным значением сопротивления, при покупке резисторов вы можете выбрать диапазон отклонений значений в процентах. Коричневая полоса соответствует диапазону отклонений в 1 %, золотая — в 5 % и серебряная — в 10 %.

На рис. 3.2 изображена схема нанесения цветных полос на корпус резистора. Желтая, фиолетовая и оранжевая полосы соответствуют цифрам **4**, **7** и **3** соответственно, как указано в табл. 3.1. Вместе эти три полосы соответствуют номиналу **47** × 10^3 = 47 000 Ом, или 47 кОм. Последняя, коричневая, полоса, обозначает погрешность сопротивления, в данном случае 1 %.

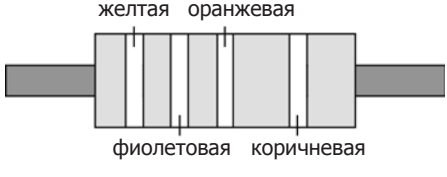


Рис. 3.2. Схема нанесения цветных полос на корпус резистора

Бескорпусные резисторы

На бескорпусных резисторах, используемых для поверхностного монтажа, номинал печатается в виде цифро-буквенного кода, как показано на рис. 3.3. Первые две цифры представляют одно число, а третья — количество нулей, следующих за этим числом. Например, резистор на рис. 3.3 имеет номинал 10 000 Ом, или 10 кОм.

Рис. 3.3. Пример бескорпусного резистора для поверхностного монтажа

ПРИМЕЧАНИЕ

Если на поверхности бескорпусного резистора вы увидите цифро-буквенное обозначение (например, *01C*), поищите в Google по фразе «маркировка EIA-96», чтобы найти таблицы с расшифровкой этой более сложной системы маркировки.

Мультиметры

Мультиметр — чрезвычайно полезный и относительно недорогой прибор, с помощью которого измеряется напряжение, сопротивление, ток и другие характеристики. Например, на рис. 3.4 изображен мультиметр, с помощью которого измеряется сопротивление резистора.

Рис. 3.4. С помощью мультиметра измеряется сопротивление резистора с номиналом 560 Ом и точностью 1%

Если вы страдаете нарушением цветоощущения, то мультиметр станет вашим основным инструментом. Приобретая мультиметр, выбирайте модель от производителя с надежной репутацией, а не самую дешевую, какую только сможете найти в Интернете.

Мощность

Мощность резистора измеряется в ваттах и определяет мощность, которую резистор способен рассеивать, сохраняя работоспособность. Резисторы, изображенные на рис. 3.1, имеют мощность 1/4 Вт и чаще других используются в системах на основе Arduino.

Подбирая резистор, помните об отношении между мощностью, током и напряжением. Чем выше ток и/или напряжение, тем выше должна быть мощность резистора.

Обычно более мощные резисторы имеют большие размеры. Например, резистор на рис. 3.5, мощностью 5 Вт, имеет 26 мм в длину и 7,5 мм в ширину.

Рис. 3.5. Резистор мощностью 5 Вт

Светодиод

Светодиоды — очень распространенные и бесконечно полезные компоненты, преобразующие электрический ток в свет. Светодиоды имеют разную форму, размеры и цвет. На рис. 3.6 показан типичный светодиод.

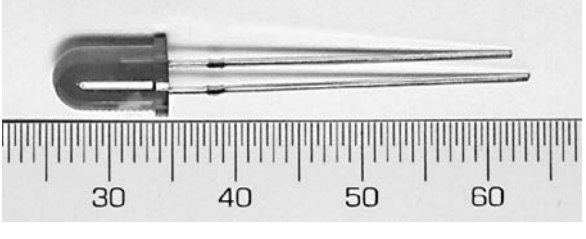


Рис. 3.6. Красный светодиод диаметром 5 мм

Включение светодиода в схему требует определенного внимания, так как при этом требуется соблюдать *полярность*; то есть ток может входить в светодиод и покидать его только в определенном направлении. Ток должен входить через *анод* (плюс) и покидать его через *катод* (минус), как показано на рис. 3.7. Попытка подать слишком большое напряжение в обратном направлении (равно как и пропустить слишком большой ток в прямом) приведет к выходу светодиода из строя.

К счастью, конструкция светодиода позволяет определить, какой контакт является анодом, а какой катодом. Ножка, связанная с анодом, более длинная, а рядом с катодом на бортике корпуса имеется плоская площадка, как показано на рис. 3.8.

Используя светодиоды в проекте, не забывайте о рабочем напряжении и силе тока. Например, типичные светодиоды красного цвета требуют напряжения около 1,7 В и тока от 5 до 20 мА. Это представляет некоторую проблему для нас, потому что на

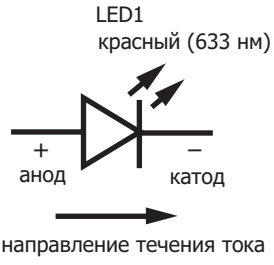


Рис. 3.7. Направление течения тока через светодиод

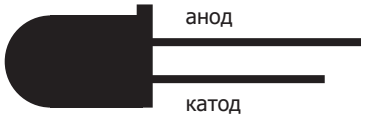


Рис. 3.8. Конструкция светодиода позволяет различить анод (более длинная ножка) и катод (плоская площадка)

выходах Arduino выводится напряжение 5 В и намного больший ток. К счастью, можно добавить *ограничительный резистор*, чтобы уменьшить силу тока, протекающего через светодиод. Но какой номинал резистора выбрать? В этом вам поможет закон Ома.

Вычисление сопротивления ограничительного резистора для светодиода выполняется по следующей формуле:

$$R = (V_s - V_f) / I,$$

где V_s — напряжение питания (Arduino выводит напряжение 5 В); V_f — падение напряжения на светодиоде (например, 1,7 В) и I — сила тока, требуемая для светодиода (10 мА). (Значение I должно выражаться в амперах, то есть 10 мА нужно преобразовать в 0,01 А.)

Давайте теперь воспользуемся этой формулой и рассчитаем ограничительный резистор для исходных значений $V_s = 5$ В, $V_f = 1{,}7$ В и $I = 0{,}01$ А. Подстановка этих значений в формулу дает значение сопротивления 330 Ом. Однако светодиод прекрасно будет работать и при силе тока меньше 10 мА. Считается хорошим тоном использовать меньшую силу тока, когда это возможно, чтобы обеспечить дополнительный уровень защиты схемы, поэтому для ограничения тока, протекающего через светодиод, мы будем использовать резистор с сопротивлением 560 Ом и мощностью 1/4 Вт, который уменьшит силу тока до 6 мА.

ПРИМЕЧАНИЕ

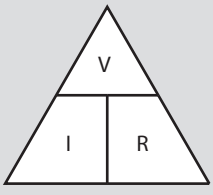
В случае сомнений всегда выбирайте резистор с большим сопротивлением, потому что тусклый светодиод лучше сгоревшего!

ТРЕУГОЛЬНИК ЗАКОНА ОМА

Закон Ома устанавливает отношение между силой тока, сопротивлением и напряжением:

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

Зная две характеристики, можно вычислить третью. Для запоминания закона Ома часто используется треугольная диаграмма, изображенная на рис. 3.9.



V = I × R
I = V ÷ R
R = V ÷ I

Рис. 3.9. Треугольник закона Ома

Треугольная диаграмма закона Ома позволяет быстро вспомнить, как вычислить напряжение, силу тока или сопротивление по двум другим известным значениям. Например, если нужно вычислить сопротивление, закройте пальцем ячейку *R*, и у вас останется формула, в которой напряжение делится на силу тока. А чтобы вычислить напряжение, закройте ячейку *V*, и у вас останется формула, в которой сила тока умножается на сопротивление.

Макетная плата для навесного монтажа

Для конструирования электрических схем необходима основа, на которую будут монтироваться электронные компоненты. Для этой цели отлично подходит *макетная плата для навесного (беспаечного) монтажа*. Плата имеет пластиковую основу с рядами гнезд, имеющих пружинные клеммы. Платы выпускаются разных размеров, форм и цветов, как показано на рис. 3.10.

Ключом к использованию макетной платы является знание того, как соединены гнезда — по вертикали или по горизонтали вдоль краев и в центре. На разных платах соединения могут быть выполнены по-разному. Например, на плате, изображенной вверху на рис. 3.11, пять столбцов гнезд соединены по вертикали, но изолированы по горизонтали. Если подключить два провода к одному вертикальному ряду, они окажутся электрически связанными. Аналогично, два длинных ряда в центре связаны по горизонтали. Поскольку схемы часто требуется подключать к источнику питания, эти длинные горизонтальные ряды гнезд идеально подходят для подачи напряжения питания.

При конструировании сложных схем не всегда получается разместить компоненты на макетной плате именно там, где хотелось бы. Эта проблема легко решается с помощью отрезков провода. Продавцы макетных плат обычно предлагают также комплекты отрезков провода разной длины типа показанных на рис. 3.12.

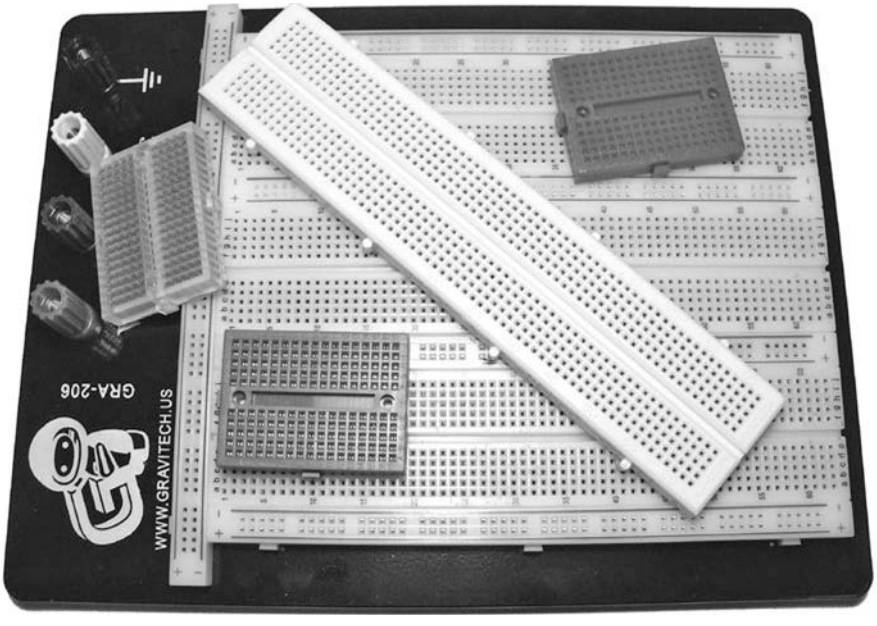


Рис. 3.10. Макетные платы разных форм и размеров

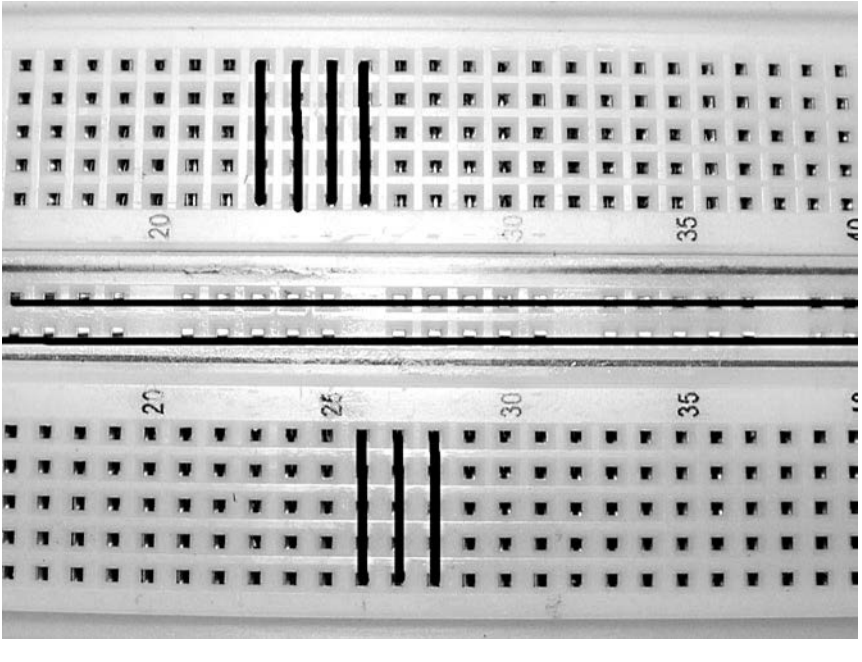


Рис. 3.11. Внутренние соединения на макетной плате

Рис. 3.12. Комплект отрезков провода

Проект № 1: Эффект бегущей волны из огоньков светодиодов

Давайте применим на деле несколько светодиодов и резисторов. В этом проекте используется пять светодиодов для имитации эффекта бегущей световой волны на решетке радиатора автомобиля KITT из американского телесериала «Knight Rider»[1].

Алгоритм

Алгоритм для этого проекта следующий:

1. Включить светодиод 1.
2. Ждать полсекунды.
3. Выключить светодиод 1.
4. Включить светодиод 2.
5. Ждать полсекунды.
6. Выключить светодиод 2.

[1] В России известен под названием «Рыцарь дорог». — *Примеч. пер.*

7. И так далее, пока не будет задействован светодиод 5, затем выполнить те же действия в обратном порядке, от светодиода 5 до светодиода 1.
8. Повторять предыдущие пункты до бесконечности.

Оборудование

Компоненты, необходимые для реализации проекта:

- Пять светодиодов.
- Пять резисторов с номиналом 560 Ом.
- Одна макетная плата.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Мы соединим светодиоды с цифровыми выходами со 2-го по 6-й через 560-омные ограничительные резисторы.

Скетч

Теперь перейдем к скетчу. Введите следующий код в среде разработки:

```
  // Проект 1 - Эффект бегущей волны из светодиодов
❶ void setup()
  {
    pinMode(2, OUTPUT); // настроить контакты,
    pinMode(3, OUTPUT); // управляющие светодиодами,
    pinMode(4, OUTPUT); // на работу в режиме
    pinMode(5, OUTPUT); // цифровых выходов
    pinMode(6, OUTPUT);
  }

❷ void loop()
  {
    digitalWrite(2, HIGH); // Включить светодиод 1
    delay(500);            // ждать полсекунды
    digitalWrite(2, LOW);  // Выключить светодиод 1
    digitalWrite(3, HIGH); // и повторить то же самое
    delay(500);            // для светодиодов со 2-го по 5-й
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
    delay(500);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    delay(500);
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
```

```
    delay(500);
    digitalWrite(6, LOW);
    digitalWrite(5, HIGH);
    delay(500);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);
    delay(500);
    digitalWrite(4, LOW);
    digitalWrite(3, HIGH);
    delay(500);
    digitalWrite(3, LOW);
    // в этой точке функция loop() завершится
    // и будет вызвана снова
}
```

В функции `void setup()` (❶) выполняется настройка контактов на работу в режиме цифровых выходов, потому что с их помощью мы будем управлять подачей электрического тока на светодиоды. В функции `void loop()` (❷) мы определяем, когда включить каждый светодиод, вызывая функцию `digitalWrite()`.

Схема

Теперь соберем схему. Схему можно описать несколькими способами. В первых нескольких проектах будут использоваться монтажные схемы как на рис. 3.13.

При сопоставлении монтажной схемы с вызовами функций в скетче становится понятно, как она работает. Например, если производится вызов `digitalWrite(2, HIGH)`,

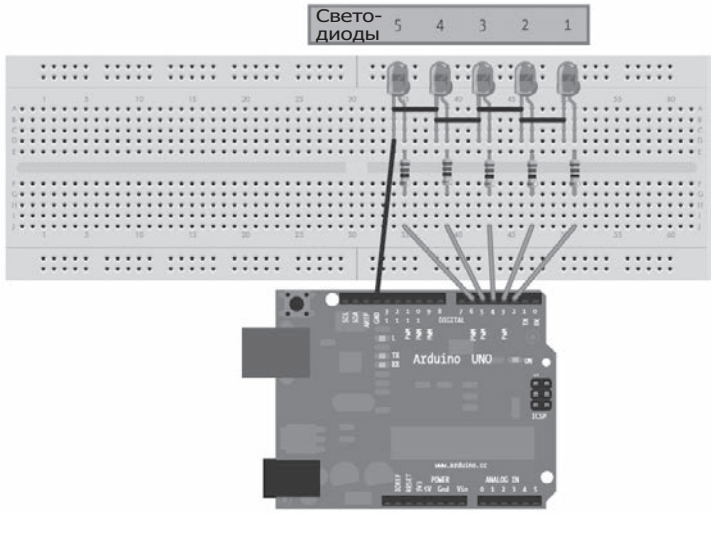


Рис. 3.13. Схема соединений для проекта 1

на цифровой выход 2 подается ток с напряжением 5 В, который течет через ограничительный резистор, через анод светодиода, через катод и завершает свой путь в разъеме GND («земля»), замыкающем цепь. Затем, когда производится вызов `digitalWrite(2, LOW)`, течение тока прекращается и светодиод гаснет.

Запуск скетча

Теперь подключите Arduino к компьютеру и загрузите скетч. Через одну-две секунды светодиоды должны начать зажигаться и гаснуть по одному, слева направо. Успех окрыляет — прочувствуйте его!

Ну а если вдруг ничего не произошло, отключите плату от компьютера и проверьте, не ошиблись ли вы, набирая скетч. Если обнаружится ошибка, исправьте ее и загрузите скетч еще раз. Если в скетче ошибок не обнаружилось, а светодиоды все равно не мигают, проверьте соединения на макетной плате.

Теперь вы знаете, как заставить светодиод мигать с помощью Arduino, но этот скетч недостаточно оптимален. Например, если позднее вам захочется изменить его, чтобы заставить волну бежать быстрее, придется вносить изменения в каждый вызов `delay(500)`. Выберем лучший путь.

Переменные

Переменные используются в компьютерных программах для хранения данных. Например, в скетче для проекта 1 мы вызывали функцию `delay(500)`, чтобы выполнить задержку перед выключением светодиодов.

Слабое место этого скетча — недостаточная гибкость. Если появится желание изменить время задержки, придется вручную изменять каждое его вхождение в скетче. Чтобы исправить эту проблему, добавим в скетч переменную, представляющую значение задержки для функции `delay()`.

Введите следующую строку в скетч проекта 1 перед функцией `void setup()`, сразу за вступительным комментарием:

```
int d = 250;
```

Она присваивает число `250` переменной с именем `d`.

Затем замените все числа `500` в скетче именем переменной `d`. Теперь, выполняя скетч, Arduino будет передавать в вызовы функции `delay()` значение переменой `d`. Когда вы загрузите измененный скетч, волна будет бежать существенно быстрее, потому что значение задержки уменьшилось до `250`.

`int` указывает, что переменная предназначена для хранения целого числа со знаком (integer) в диапазоне от –32 768 до 32 767. Такой переменной можно присвоить любое число, не имеющее дробной части. Теперь, чтобы изменить задержку, до-

статочно лишь изменить объявление переменной в начале скетча. Например, если присвоить переменной число `100`, скорость бегущей волны увеличится еще больше:

```
int d = 100;
```

Поэкспериментируйте со скетчем. Попробуйте менять величину задержки и последовательность `HIGH` и `LOW`. Поиграйте. Но пока не разбирайте схему; она еще пригодится нам в проектах, которые мы рассмотрим ниже в этой главе.

Проект № 2: Повторение команд с помощью цикла for

В скетчах часто приходится повторять одни и те же команды. Можно, конечно, просто скопировать команду в буфер обмена и вставить ее в скетч столько раз, сколько потребуется, но это решение неэффективно и ведет к напрасному расходованию памяти в программах для Arduino. Вместо этого воспользуемся циклом `for`. Главное достоинство цикла `for` — он позволяет определить, сколько раз следует выполнить код внутри него.

Для более близкого знакомства с циклом `for` введите следующий новый скетч:

```
// Проект 2 - Повторение команд с помощью цикла for
int d = 100;

void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop()
{
  for ( int a = 2; a < 7 ; a++ )
  {
    digitalWrite(a, HIGH);
    delay(d);
    digitalWrite(a, LOW);
    delay(d);
  }
}
```

Цикл `for` многократно выполняет код, заключенный в фигурные скобки, следующие сразу за ним, пока не будет выполнено некоторое условие. Здесь мы добавили еще одну целочисленную переменную, `a`, которой в начале цикла присвоили зна-

чение 2. После каждой итерации команда a++ увеличивает значение этой переменной на 1. Цикл продолжает выполняться снова и снова, пока значение a остается меньше 7 (условие). Как только a получит значение, равное или больше 7, Arduino двинется дальше и продолжит выполнение кода, следующего за циклом for.

В цикле for можно также вести отсчет итераций в обратном направлении, от больших значений к меньшим. Для демонстрации такой возможности добавьте в скетч проекта 2 следующий код после первого цикла for:

```
❶ for ( int a = 5 ; a > 1 ; a-- )
  {
    digitalWrite(a, HIGH);
    delay(d);
    digitalWrite(a, LOW);
    delay(d);
  }
```

Здесь цикл for (❶) присваивает переменной a начальное значение 5 и затем вычитает из нее 1 после каждой итерации командой a--. Этот цикл продолжается, пока значение a остается больше 1 (a > 1), и завершается, как только a станет равным 1 или меньше.

Мы воспроизвели проект 1, используя меньше кода. Загрузите скетч и убедитесь в этом сами!

Изменение яркости светодиода с использованием широтно-импульсной модуляции

Светодиоды можно не только включать и выключать вызовом digitalWrite(), но и регулировать яркость их свечения, изменяя промежуток времени, когда светодиод находится во включенном и в выключенном состоянии, используя для этого *широтно-импульсную модуляцию* (ШИМ — Pulse-Width Modulation, PWM). ШИМ можно использовать для создания иллюзии изменения яркости свечения светодиода, быстро включая и выключая его 500 раз в секунду. Видимая нами яркость определяется отношением интервала времени, когда цифровой выход включен, к интервалу времени, когда он выключен, то есть когда светодиод светится и когда он не светится. Наш глаз не способен видеть мерцания с частотой более 50 раз в секунду, поэтому создается ощущение, что светодиод светит непрерывно.

Чем больше *коэффициент заполнения* (отношение времени, когда через контакт течет ток, к времени, когда ток не течет, в каждом цикле), тем выше воспринимаемая яркость светодиода, подключенного к цифровому выходу.

На рис. 3.14 изображены циклы ШИМ с разными коэффициентами заполнения. Области, залитые серым цветом, — это периоды, когда светодиод включен. Как видите, длительность периодов, когда светодиод светится, увеличивается с увеличением коэффициента заполнения.

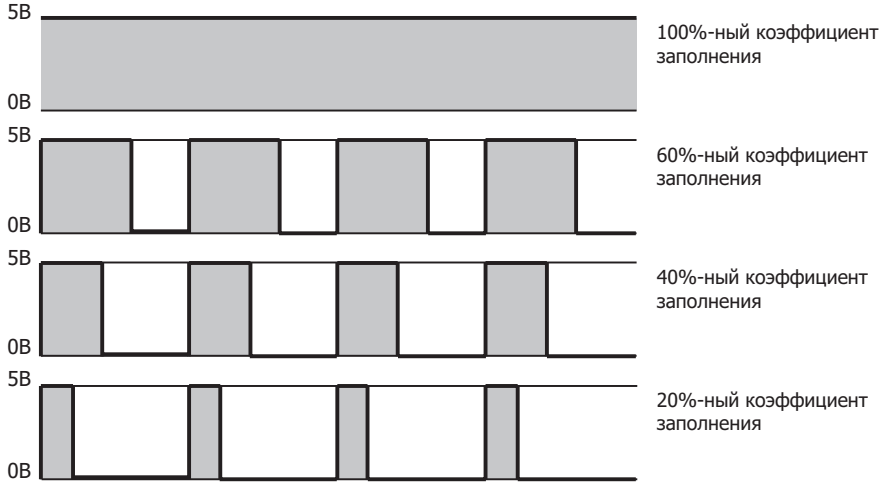


Рис. 3.14. Разные коэффициенты заполнения ШИМ

Только цифровые выходы 3, 5, 6, 9, 10 и 11 на обычной плате Arduino поддерживают ШИМ. Они отмечены на плате символом «тильда» (~), как показано на рис. 3.15.

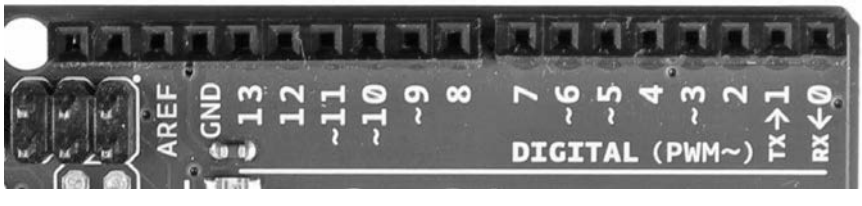


Рис. 3.15. Выходы с поддержкой ШИМ отмечены на плате символом «тильда» (~)

Для создания сигнала ШИМ используется функция `analogWrite(x, y)`, где `x` — номер цифрового выхода, а `y` — значение коэффициента заполнения, в диапазоне от 0 до 255, где 0 соответствует коэффициенту заполнения 0 %, а 255 — 100 %.

Проект № 3: Демонстрация ШИМ

Теперь используем сигналы ШИМ в нашей схеме из проекта 2. Введите в IDE следующий скетч и загрузите его в плату Arduino:

```
// Проект 3 - Демонстрация ШИМ
int d = 5;
void setup()
```

```
{
  // управление светодиодом, подключенным к контакту 3,
  // поддерживающему ШИМ
  pinMode(3, OUTPUT);
}

void loop()
{
  for ( int a = 0 ; a < 256 ; a++ )
  {
    analogWrite(3, a);
    delay(d);
  }
  for ( int a = 255 ; a >= 0 ; a-- )
  {
    analogWrite(3, a);
    delay(d);
  }
  delay(200);
}
```

Яркость свечения светодиода, подключенного к цифровому выходу 3, будет плавно уменьшаться и увеличиваться с увеличением и уменьшением коэффициента заполнения. Иными словами, яркость свечения светодиода будет плавно увеличиваться, пока не достигнет максимума, а потом плавно уменьшаться. Поэкспериментируйте со скетчем и схемой. Например, попробуйте заставить все пять светодиодов изменять яркость одновременно или последовательно.

Дополнительные электронные компоненты

Поверьте мне, проекты проще планировать, если не приходится заботиться о величине тока, потребляемого устройствами, которые управляются цифровыми выходами. Создавая свои проекты, помните, что каждый цифровой выход на плате Arduino Uno может отдавать ток не более 40 мА и не более 200 мА для всех выходов в сумме. Расширить технические возможности Arduino вам помогут три электронных компонента, их-то мы и рассмотрим ниже.

ВНИМАНИЕ

Если попытаться превысить допустимую силу тока — 40 мА на один цифровой выход или 200 мА для всех выходов в сумме, это может привести к выходу из строя микроконтроллера.

Транзистор

Практически каждый из нас слышал слово *транзистор*, но большинство не понимает, как он работает. Я постараюсь объяснить это настолько просто, насколько возможно. Транзистор может включать и выключать намного более мощный ток,

чем плата Arduino Uno. Однако мы можем безопасно управлять транзистором с помощью цифровых выходов Arduino. Типичным примером может служить транзистор BC548, изображенный на рис. 3.16.

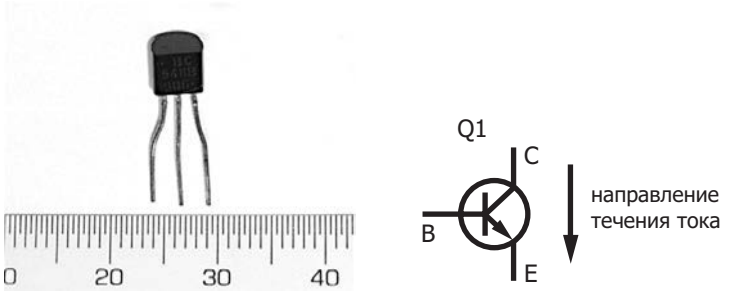


Рис. 3.16. Типичный транзистор: BC548

Подобно светодиоду, выводы транзистора имеют свои уникальные функции и должны подключаться с соблюдением полярности. Если смотреть на транзистор с плоской стороны (как показано слева на рис. 3.16), выводы BC548 называются (слева направо) *коллектором*, *базой* и *эмиттером*. (Обратите внимание, что здесь приводится порядок следования выводов, или цоколевка, для транзистора BC548 — другие транзисторы могут иметь иную схему расположения выводов.) Если на базу подается маленький ток, например, с цифрового выхода Arduino, то нужный нам больший ток будет протекать через цепь коллектор–эмиттер. Когда малый управляющий ток снимается с базы, течение тока через транзистор прекращается.

Транзистор BC548 может переключать ток силой до 100 мА и напряжением до 30 В — это намного больше, чем способна отдать плата Arduino через цифровой выход. В проектах, рассмотреных в следующих главах, мы будем использовать этот и другие транзисторы и тогда познакомимся с ними поближе.

ПРИМЕЧАНИЕ

Всегда обращайте внимание на расположение выводов конкретного транзистора, потому что каждый транзистор может иметь собственную цоколевку.

Выпрямительный диод

Диод — очень простой и очень полезный компонент, позволяющий току течь только в одном направлении. Своей формой он напоминает резистор (рис. 3.17).

В проектах этой книги мы используем выпрямительный диод 1N4004. Ток входит в диод со стороны анода и выходит через катод, который обозначен кольцевой меткой на корпусе диода. Подобные диоды могут защищать части схем от тока, текущего в обратном направлении, но за это приходится расплачиваться падением напряжения примерно на 0,7 В. Диод 1N4004 способен пропускать ток силой

до 1 А и выдерживать обратное напряжение до 400 В, что с лихвой покрывает наши потребности. Это надежный, распространенный и недорогой компонент.

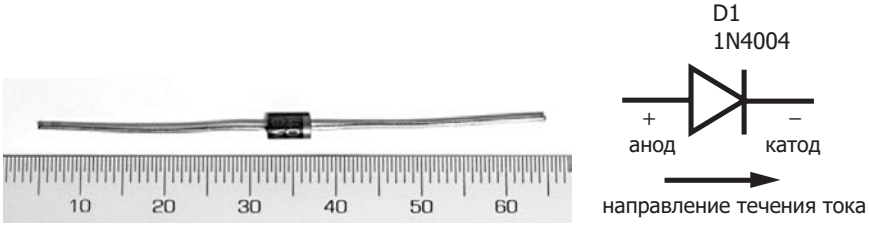


Рис. 3.17. Выпрямительный диод 1N4004

Реле

Реле используются с той же целью, что и транзисторы, — для управления током большей силы и напряжения. Преимущество реле заключается в электрической изолированности от управляющей схемы, что позволяет Arduino переключать токи очень большой силы и напряжения. Изоляция иногда необходима, чтобы защитить схему от воздействия токов большой силы и напряжения, которые могут повредить плату. Внутри реле имеется пара интересных элементов: контакты механического переключателя и низковольтная катушка индуктивности (рис. 3.18).

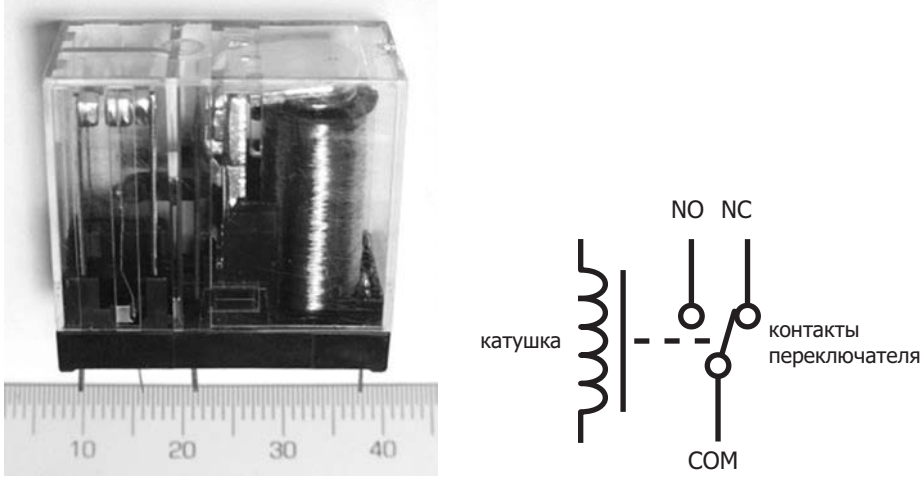


Рис. 3.18. Типичное реле

Когда на катушку подается ток, она начинает действовать как электромагнит и притягивает металлический язычок, который работает как переключатель. Когда электромагнит включен, язычок притягивается к нему и замыкает контакты, когда

выключен — язычок возвращается в исходное положение и размыкает контакты. Это позволяет управлять контактами, подавая напряжение на катушку и снимая его. Включение и выключение реле происходит с характерным щелчком, который можно услышать, например, включив сигнал поворота в старом автомобиле.

Схемы управления более высоким напряжением

Теперь, после знакомства с транзистором, выпрямительным диодом и реле, используем их для управления током большей силы и напряжения. Схема соединения компонентов очень простая, это показано на рис. 3.19.

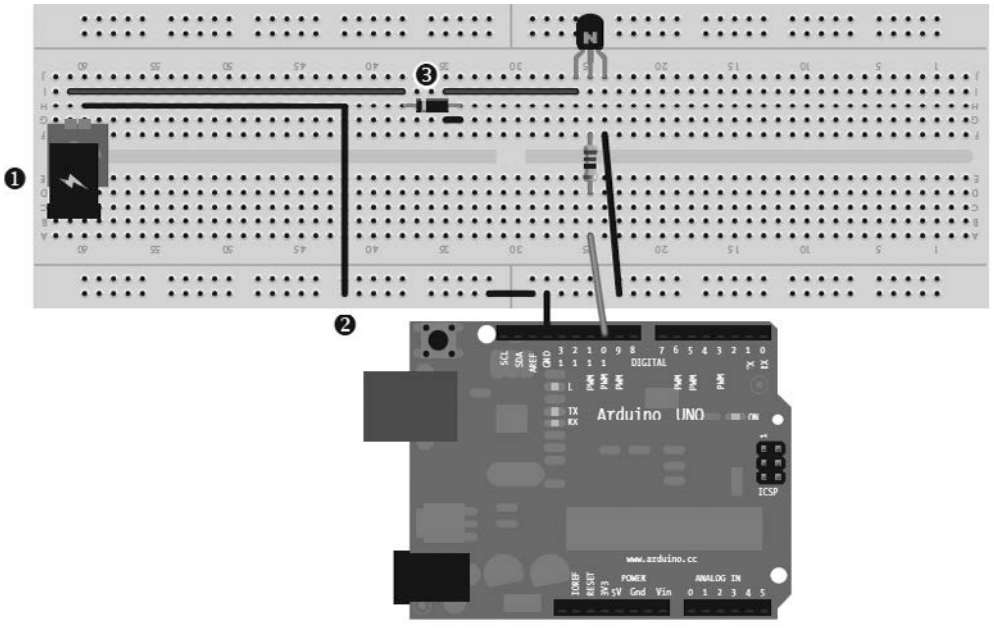


Рис. 3.19. Схема управления реле

Эта схема управляет реле, имеющим 12-вольтовую катушку. Схема позволяет управлять лампой или вентилятором, подключив их к контактам реле. Цифровой выход 10 на плате Arduino подключен к базе транзистора через резистор с номиналом 1 кОм. Транзистор управляет током, протекающим через катушку, включая и выключая реле. Не забывайте, что выводы этого транзистора располагаются в порядке (слева направо): коллектор–база–эмиттер, если смотреть со стороны плоской поверхности транзистора. Компонент ❶ слева на макетной плате — это источник питания с напряжением 12 В для катушки реле. Отрицательный вывод этого источника питания ❷ связан с эмиттером транзистора и контактом GND на плате Arduino. Наконец, катод выпрямительного диода 1N4004 ❸ связан через катушку

реле с положительным выводом источника питания. Загляните в документацию с описанием реле, чтобы определить, какие выводы соединены с управляющими контактами, и подключите к ним управляемое устройство.

Диод здесь служит для защиты схемы. Когда с катушки снимается напряжение, возникает кратковременный всплеск паразитного тока высокого напряжения, который должен быть куда-то направлен. Диод пропускает паразитный ток по кругу через катушку, пока он не будет рассеян в виде небольшого количества тепла. Это предотвращает повреждение транзистора или выхода на плате Arduino от кратковременного всплеска паразитного тока.

ВНИМАНИЕ

Если вы захотите с помощью реле управлять приборами, питающимися от бытовой электросети (110–250 В), посоветуйтесь с опытным электриком, чтобы он помог вам сделать эту работу. Даже небольшая ошибка может привести к трагическим последствиям.

Забегая вперед

Вот и третья глава подошла к концу. Я надеюсь, вы получили удовольствие от экспериментов со светодиодами. В этой главе мы разными способами воспроизвели эффект бегущей световой волны и узнали, как эффективнее использовать функции и циклы для управления компонентами, подключенными к Arduino. Получив новые знания, вы, я надеюсь, готовы преодолеть путь к вершинам в следующих главах.

В главе 4 вас ждет много интересного. Вы создадите несколько практических проектов, в том числе светофор, термометр, тестер для батареи и многое другое. Итак, если вы готовы подняться на следующий уровень, перелистните страницу!

4 Строительные блоки

В этой главе вы:

- ✓ научитесь читать принципиальные схемы и понимать язык описания электронных конструкций;
- ✓ познакомитесь с конденсатором;
- ✓ изучите работу цифровых входов;
- ✓ узнаете, как использовать арифметические и логические выражения;
- ✓ освоите приемы принятия решений с помощью инструкций `if-then-else`;
- ✓ узнаете, чем отличаются аналоговые и цифровые сигналы;
- ✓ научитесь измерять напряжение на аналоговых входах с различной степенью точности;
- ✓ познакомитесь с переменными резисторами, пьезоэлектрическими зуммерами и датчиками температуры;
- ✓ используете полученные знания для создания светофора, тестера для батареек и термометра.

Эта глава познакомит вас с потенциальными возможностями платы Arduino. Мы продолжим изучение электроники, в том числе новых компонентов, приемов чтения принципиальных схем (путеводителя по электронным конструкциям) и типов сигналов, которые можно измерять. Затем мы обсудим дополнительные возможности Arduino, такие как хранение значений, выполнение математических операций и принятие решений. В заключение мы исследуем дополнительные компоненты и затем на их основе создадим несколько интересных проектов.

Принципиальные схемы

В главе 3 рассказывалось, как собирать электронные устройства, используя монтажные схемы, на которых изображена макетная плата и компоненты на ней. Подобные монтажные схемы могут показаться самым простым способом представления электронных схем, но увеличение числа компонентов и непосредственное их изображение делает монтажные схемы весьма запутанными. Так как наши проекты со временем будут усложняться, мы перейдем к использованию *принципиальных схем* (также известных как *электрические схемы*). На рис. 4.1[1] приводится пример такой схемы.

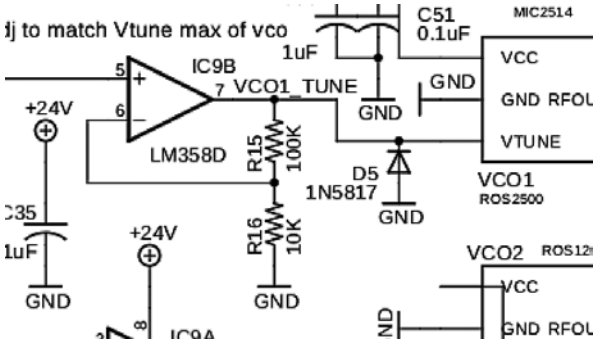


Рис. 4.1. Фрагмент принципиальной схемы

Принципиальные схемы можно считать своеобразными «путеводителями», которые показывают, как электрический ток протекает через различные компоненты. Вместо изображений компонентов и проводов на принципиальных схемах используются символы и линии.

Обозначение компонентов

Зная значение символов, вы легко сможете читать принципиальные схемы. Для начала давайте познакомимся с символами, обозначающими компоненты, которые мы уже использовали.

Arduino

На рис. 4.2 показано, как на схемах обозначается сама плата Arduino. Как видите, здесь отмечены и аккуратно подписаны все контакты на плате.

---

[1] Здесь и далее в книге сохранено оригинальное оформление электрических схем. — *Примеч. ред.*

Рис. 4.2. Обозначение платы Arduino Uno

Резистор

На рис. 4.3 показан символ, обозначающий резистор.

На схемах принято изображать рядом с символом номинал резистора и его уникальный идентификатор в пределах схемы (в данном случае 220Ω и R1). Это упрощает чтение и анализ принципиальной схемы. Часто номинал в омах обозначается не символом Ω, а буквой R, например 220 R.

Рис. 4.3. Обозначение резистора

Выпрямительный диод

На рис. 4.4 показан символ, обозначающий выпрямительный диод.

Как отмечалось в главе 3, выпрямительный диод имеет полярность и пропускает электрический ток в направлении от анода к катоду. На рис. 4.4 анод изображен слева, а катод справа. Проще запомнить это, если представить, что треугольник — это воронка, в которую втекает электрический ток. Ток не может течь в обратном направлении, потому что вертикальная черта, как «заслонка», не пускает его.

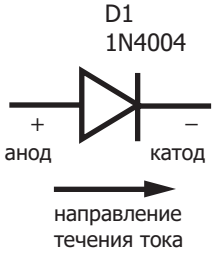


Рис. 4.4. Обозначение выпрямительного диода

Светодиод

На рис. 4.5 показан символ, обозначающий светодиод.

Для обозначения всех компонентов из семейства диодов используется один и тот же символ: треугольник и вертикальная черта. Однако символ светодиода включает также две параллельные стрелки, направленные в сторону от треугольника и изображающие испускаемый свет.

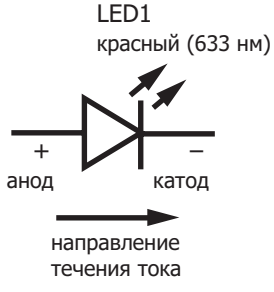


Рис. 4.5. Обозначение светодиода

Транзистор

На рис. 4.6 показан символ, обозначающий транзистор. Этот символ далее будет использоваться для обозначения транзистора BC548.

Вертикальная линия в верхней части символа (с буквой C) обозначает коллектор (collector), горизонтальная линия слева (с буквой B) — базу (base) и вертикальная линия в нижней части символа (с буквой E) — эмиттер (emitter). Стрелка внутри символа, направленная вправо вниз, сообщает, что этот транзистор относится к типу NPN-транзисторов, основной отличительной особенностью которых является направление течения электрического тока от коллектора к эмиттеру. (Другой тип транзисторов — PNP-транзисторы — пропускают электрический ток в направлении от эмиттера к коллектору.)

Для обозначения транзисторов на схемах используется буква Q, а для резисторов — буква R.

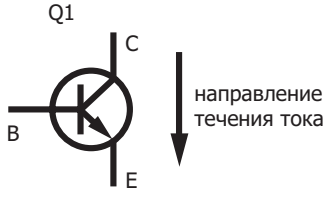


Рис. 4.6. Обозначение транзистора

Реле

На рис. 4.7 показан символ, обозначающий реле.

Реле на схемах обозначаются по-разному и могут иметь несколько групп контактов, но все обозначения содержат несколько общих элементов. Первый — это обозначение *катушки* в виде витой вертикальной линии слева. Второй элемент — это *контакты* реле. Входной контакт часто обозначается как COM (common — общий), а выходные — как NO (normally open — нормально разомкнутый) и NC (normally closed — нормально замкнутый).

Символ реле всегда изображается в выключенном состоянии, когда ток *не течет* через катушку, — то есть с перемычкой между контактами COM и NC. При подаче напряжения на катушку реле замыкает контакты COM и NO.

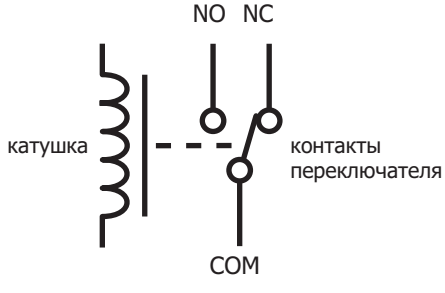


Рис. 4.7. Обозначение реле

Проводники на схемах

Пересекающиеся и соединяющиеся проводники изображаются на схемах по-разному, это показано на следующих примерах.

Пересечение проводников

Когда два проводника пересекаются на схеме, не образуя электрического соединения, их пересечение может быть обозначено двумя способами, показанными на рис. 4.8. В данном случае предписанного способа обозначения не существует; выбор зависит от личных предпочтений.

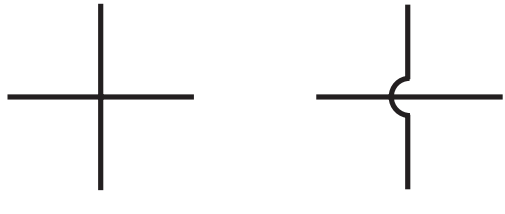


Рис. 4.8. Пересечение проводников

Электрическое соединение проводников

Когда подразумевается физическое соединение проводников, в месте пересечения изображается точка соединения, как показано на рис. 4.9.

Рис. 4.9. Электрическое соединение проводников

Проводник, подключенный к «земле»

Для обозначения связи проводника с «землей» (GND) принято использовать стандартный символ, изображенный на рис. 4.10.

Символ «земли» в конце линии на схеме сообщает, что проводник физически подключен к контакту GND на плате Arduino.

Рис. 4.10. Символ, обозначающий связь с «землей»

Чтение принципиальных схем

Теперь, когда вы знаете, как на схемах обозначаются различные компоненты и их связи, давайте попробуем прочитать схему, нарисованную для проекта 1. Напомню, что там мы воспроизвели эффект световой волны на пяти светодиодах.

Сравните схему на рис. 4.11 и схему на рис. 3.13: согласитесь, принципиальная схема позволяет намного проще описать конструкцию устройства.

С этого момента в описаниях проектов будут использоваться принципиальные схемы, а обозначения новых компонентов я буду объяснять по мере знакомства с ними.

ПРИМЕЧАНИЕ

Если у вас появится желание рисовать собственные принципиальные схемы на компьютере, попробуйте воспользоваться приложением Fritzing, его бесплатно можно скачать по адресу http://www.fritzing.org/.

Конденсатор

Конденсатор — это устройство, способное сохранять электрический заряд. Он состоит из двух металлических пластин, разделенных слоем диэлектрика, который позволяет наращивать разность потенциалов между пластинами. После отключения конденсатора от электрической сети заряд на нем остается, но может и «стекать» (в этом случае говорят, что конденсатор *разряжается*), если найдет новый путь для дальнейшего движения.

Емкость конденсатора

Величина электрического заряда, которая может накапливаться в конденсаторе, определяется его *емкостью*, которая измеряется в *фарадах*. Стоит отметить, что один фарад — это очень большая емкость, поэтому обычно емкость конденсаторов измеряется в пикофарадах или микрофарадах.

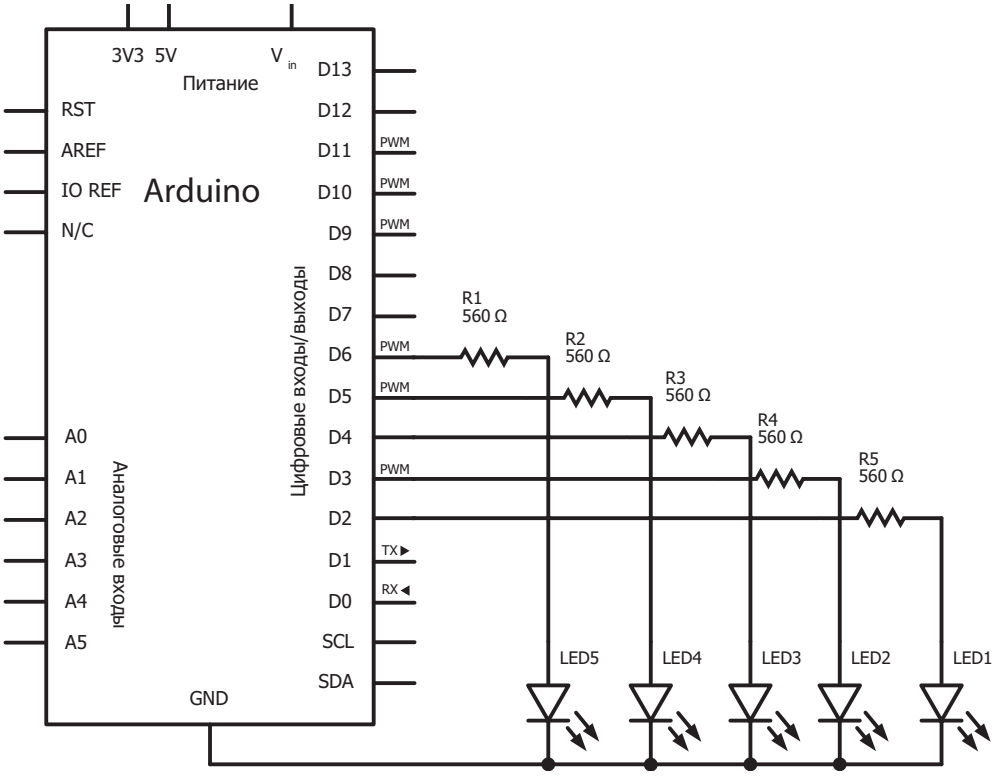


Рис. 4.11. Принципиальная электрическая схема для проекта 1

Один *пикофарад* (пФ, или pF) — это одна триллионная доля фарада, а один микрофарад (мкФ, или μF) — одна миллионная. При маркировке конденсаторов указывается также максимальное напряжение, которое способен выдерживать конденсатор. В этой книге мы будем работать только с низковольтными схемами, поэтому нам не понадобятся конденсаторы, рассчитанные на напряжение выше 10 В, но вообще в электронных устройствах принято использовать конденсаторы, рассчитанные на напряжение чуть выше, чем используется в схеме. Наиболее типичными номинальными напряжениями считаются 10, 16, 25 и 50 В.

Маркировка конденсаторов

Для чтения маркировки керамических конденсаторов требуется определенная практика, потому что номинальная емкость печатается на корпусе в виде кода. Первые две цифры представляют значение в пикофарадах, а третья определяет множитель в виде числа нулей. Например, на конденсатор, изображенный на рис. 4.12, нанесен код 104. Он означает число 10, за которым следует четыре нуля, что дает в результате 100 000 пикофарад (пФ), 100 нанофарад (нФ) или 0,1 микрофарада (мкФ).

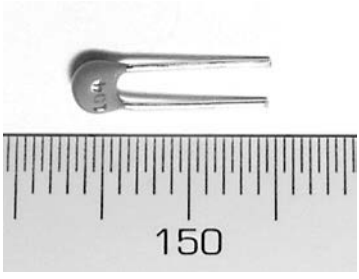


Рис. 4.12. Керамический конденсатор емкостью 0,1 мкФ

ПРИМЕЧАНИЕ

Преобразование единиц измерения может вызывать некоторые сложности, поэтому в случае затруднений обращайтесь к превосходной таблице по адресу http://www.justradios.com/uFnFpF.html.[1]

Типы конденсаторов

В наших проектах будут использоваться конденсаторы двух типов: керамические и электролитические.

Керамические конденсаторы

Керамические конденсаторы, например, изображенный на рис. 4.12, отличаются очень маленькими размерами и потому обладают очень небольшой емкостью. Они не имеют полярности и могут включаться в схему в любом положении. На схемах неполярный конденсатор изображается, как показано на рис. 4.13.

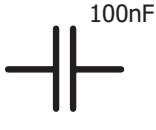


Рис. 4.13. Символ неполярного конденсатора с обозначением емкости справа вверху

Керамические конденсаторы прекрасно работают в высокочастотных цепях, потому что, благодаря своей малой емкости, способны заряжаться и разряжаться очень быстро.

[1] Отличный калькулятор перевода величин на русском языке можно найти по адресу http://www.translatorscafe.com/cafe/RU/units-converter/electrostatic-capacitance/. — *Примеч. пер.*

Электролитические конденсаторы

Электролитические конденсаторы, например, изображенный на рис. 4.14, имеют больший размер, чем керамические конденсаторы, большую емкость, и их выводы различаются полярностью. На корпусе отмечается либо положительный (+), либо отрицательный (–) вывод. На рис. 4.14 можно видеть полосу и маленький знак «минус» (–), отмечающий отрицательный вывод. Так же как резисторы, конденсаторы имеют диапазон отклонений значений относительно номинала. Конденсатор на рис. 4.14 имеет диапазон отклонений 20 % и емкость 100 мкФ.

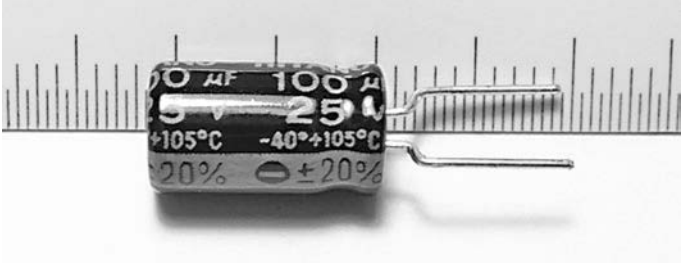


Рис. 4.14. Электролитический конденсатор

На схемах электролитический конденсатор изображается, как показано на рис 4.15; знак + обозначает полярность конденсатора.

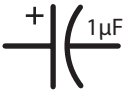


Рис. 4.15. Символ полярного конденсатора

Электролитические конденсаторы часто используются для накопления больших зарядов и сглаживания пульсаций напряжения. Подобно маленьким батарейкам, они могут демпфировать броски напряжения и стабилизировать его в цепях, где потребляемый ток быстро меняется, предупреждая нежелательные помехи в цепях. Номинал конденсаторов, к счастью, печатается на корпусе в понятной форме и не требует расшифровки.

Теперь, когда у вас есть опыт вывода простых сигналов с платы Arduino, пришло время научиться вводить сигналы извне и на их основе принимать решения.

Цифровые входы

В главе 3 мы использовали контакты цифровых входов/выходов в качестве выходов, чтобы включать и выключать светодиоды. Те же самые контакты можно использовать в качестве входов, принимающих сигналы от пользователя, например сигнал нажатия кнопки.

Подобно цифровым выходам, цифровые входы имеют два состояния: высокое и низкое. Простейшей формой входного сигнала является нажатие кнопки без фиксации, представленной на рис. 4.16. Ее можно установить непосредственно на макетной плате. *Кнопка без фиксации* пропускает электрический ток, пока она нажата, а цифровой вход используется для определения наличия напряжения на контакте и, соответственно, факта нажатия кнопки.

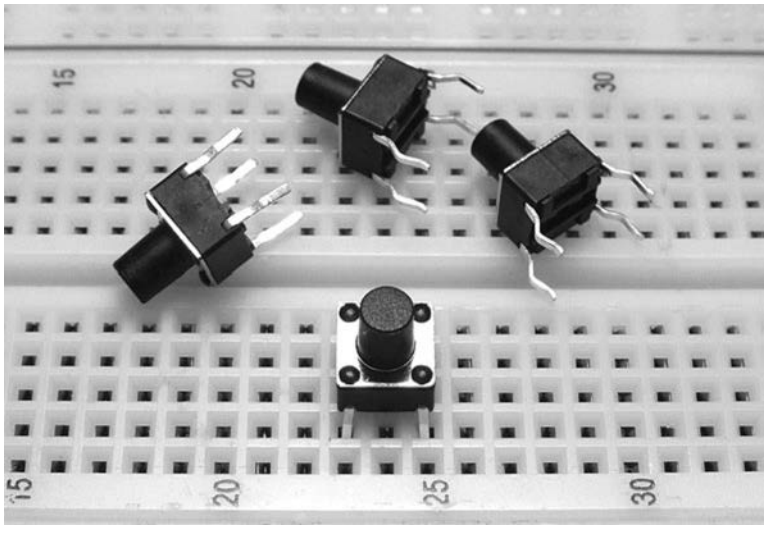


Рис. 4.16. Простые кнопки без фиксации на макетной плате

Обратите внимание, как контакты кнопки на рис. 4.16 вставлены в гнезда на макетной плате и соединяют ряды 23 и 25. Когда кнопка нажата, она соединяет эти два ряда. На принципиальных схемах данная конкретная кнопка изображается, как показано на рис. 4.17. Символ изображает две стороны кнопки и обозначается буквенно-цифровым кодом с префиксом S. Когда кнопка нажата, линия замыкает две половины и позволяет электрическому току течь через контакты.

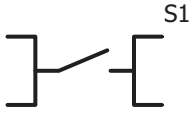


Рис. 4.17. Обозначение кнопки без фиксации

ИССЛЕДОВАНИЕ ЭФФЕКТА ДРЕБЕЗГА КОНТАКТОВ С ПОМОЩЬЮ ОСЦИЛЛОГРАФА С ЦИФРОВОЙ ПАМЯТЬЮ

Кнопки без фиксации порождают эффект под названием «дребезг контактов», или просто «дребезг», который проявляется в виде череды импульсов, сопровождающей единственное нажатие кнопки. Этот эффект возникает из-за того, что замыкаемые металлические контакты внутри кнопки настолько малы, что могут вибрировать после отпускания кнопки и очень быстро замыкаться и размыкаться несколько раз подряд.

Эффект дребезга контактов можно наблюдать с помощью осциллографа с цифровой памятью, устройства, которое отображает изменения напряжения за некоторый период времени. Например, взгляните на рис. 4.18, где показано, как выглядит эффект дребезга на экране осциллографа.

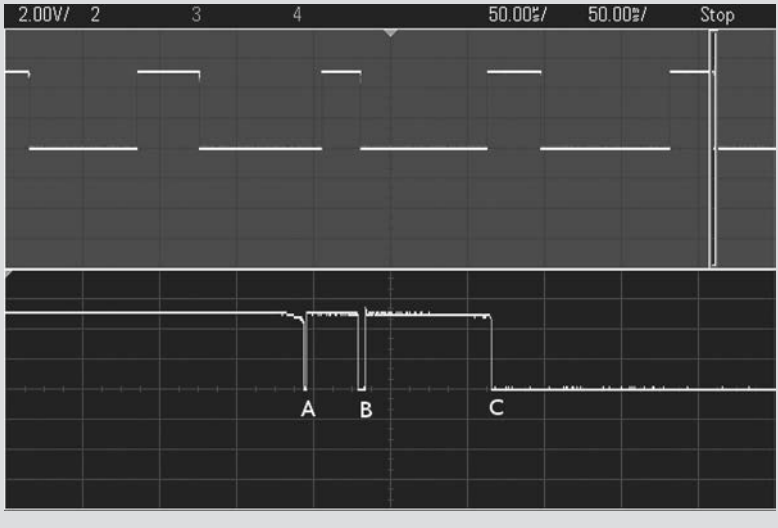


Рис. 4.18. Проявление эффекта дребезга контактов

В верхней половине рис. 4.18 показаны результаты нескольких нажатий кнопки. Когда линия, обозначающая напряжение и отмеченная на рис. 4.18 стрелкой, находится в верхнем положении (5 В), кнопка находится в состоянии «включено» и ток течет через нее. Под словом Stop двумя вертикальными линиями выделен отрезок времени сразу после отпускания кнопки. Этот отрезок времени приводится на нижней половине рис. 4.18 в увеличенном масштабе. В точке A пользователь отпустил кнопку, и уровень напряжения упал до 0 В. Однако дальше, из-за физических колебаний контактов, напряжение снова подскочило до 5 В и оставалось высоким до точки B, где контакты разомкнулись. После этого, из-за вибрации, контакты вновь замкнулись и разомкнулись уже в точке C, после чего кнопка перешла в устойчивое состояние «выключено». В результате вместо одного сигнала нажатия кнопки мы непреднамеренно послали три.

Проект № 4: Демонстрация работы цифрового входа

Цель этого проекта — реализовать включение светодиода на полсекунды в ответ на нажатие кнопки.

Алгоритм

Далее приводится алгоритм работы проекта:

1. Проверить, нажата ли кнопка.
2. Если кнопка нажата, включить светодиод на полсекунды и затем выключить его.
3. Если кнопка не нажата, ничего не делать.
4. Повторять до бесконечности.

Оборудование

Ниже перечислено оборудование, необходимое для реализации данного проекта:

- Одна кнопка.
- Один светодиод.
- Один резистор с номиналом 560 Ом.
- Один резистор с номиналом 10 кОм.
- Один конденсатор емкостью 100 нФ.
- Несколько отрезков провода разной длины.
- Одна макетная плата.
- Плата Arduino и кабель USB.

Схема

Сначала соберем на макетной плате схему, изображенную на рис. 4.19. Обратите внимание, что резистор с номиналом 10 кОм соединяет вывод GND и контакт 7. Этот резистор называется *согласующим*, или *подтягивающим вниз*, потому что в отсутствие сигнала он понижает напряжение на цифровом входе практически до нуля. Кроме того, дополнительный конденсатор емкостью 100 нФ, включенный параллельно резистору с номиналом 10 кОм, создает простую цепь, устраняющую эффект дребезга контактов, тем самым помогая отфильтровать паразитные импульсы. В момент нажатия кнопки на цифровой вход подается напряжение высокого

уровня. Когда кнопка отпускается, цифровой вход «подтягивается к земле» резистором с номиналом 10 кОм, а конденсатор емкостью 100 нФ создает небольшую задержку. Эта задержка имеет величину, большую, чем продолжительность дребезга контактов кнопки. Задержка замедляет падение напряжения до 0 В и тем самым устраняет большинство ложных срабатываний.

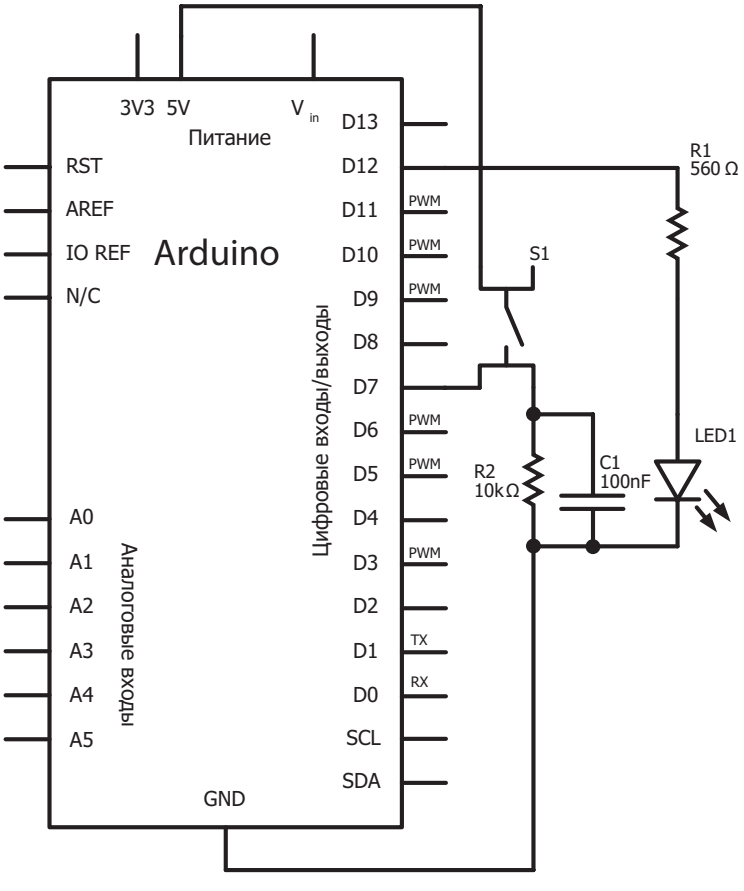


Рис. 4.19. Принципиальная схема проекта 4

Так как это первый проект, который собирается с применением принципиальной схемы, ниже приводятся пошаговые инструкции по ее сборке; это поможет понять, как соединять компоненты:

1. Вставьте кнопку в макетную плату, как показано на рис. 4.20.
2. Поверните макетную плату на 90 градусов против часовой стрелки и вставьте резистор с номиналом 10 кОм, короткий проводник и конденсатор, как показано на рис. 4.21.

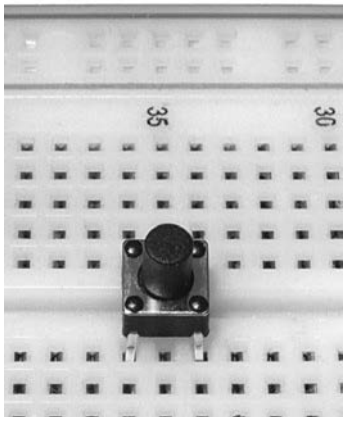


Рис. 4.20. Кнопка на макетной плате

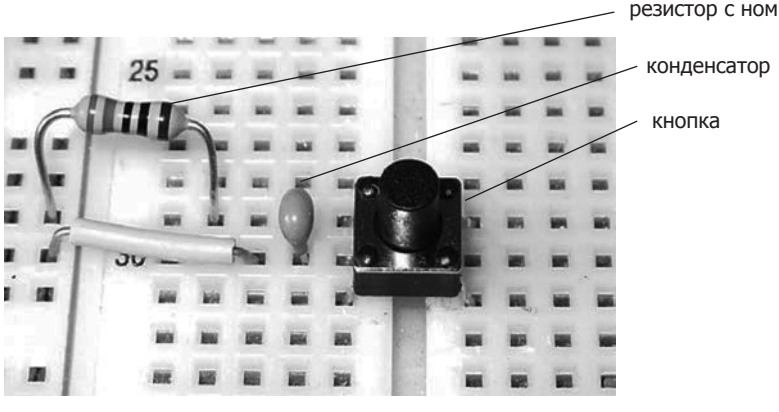


Рис. 4.21. Кнопка, конденсатор и резистор с номиналом 10 кОм

3. Соедините отрезком провода контакт 5 V на плате Arduino с самым левым вертикальным рядом на макетной плате. Соедините другим отрезком провода контакт GND на плате Arduino с вертикальным рядом на макетной плате, находящимся правее вертикального ряда, соединенного с контактом 5 V. Соедините горизонтальным проводником вертикальный ряд GND и левый нижний контакт кнопки. Полученная схема представлена на рис. 4.22.
4. Соедините проводом цифровой вход 7 на плате Arduino и правый верхний контакт кнопки, как показано на рис. 4.23.
5. Вставьте светодиод в макетную плату коротким выводом (катодом) в один из разъемов вертикального ряда GND, а длинным выводом (анодом) — в один из разъемов горизонтального ряда правее. Затем подсоедините резистор с номиналом 560 Ом справа от светодиода, как показано на рис. 4.24.

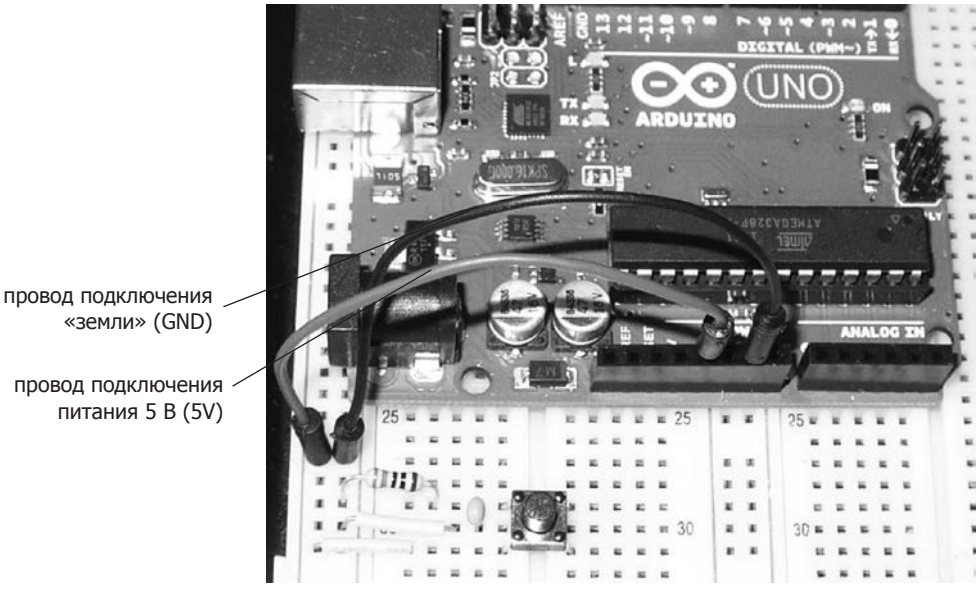


Рис. 4.22. Провода питания: 5 В (красный) и «земля» (черный)

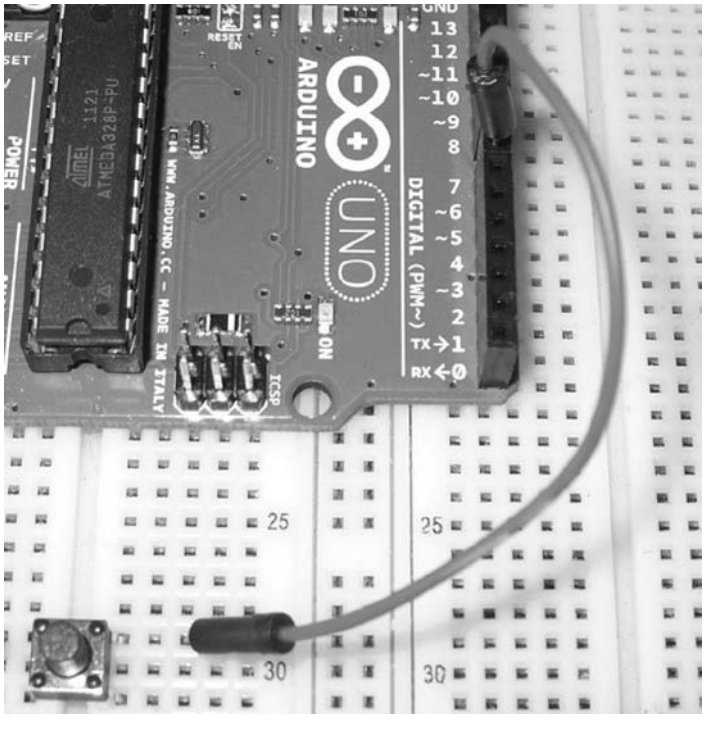


Рис. 4.23. Соединение кнопки с цифровым входом

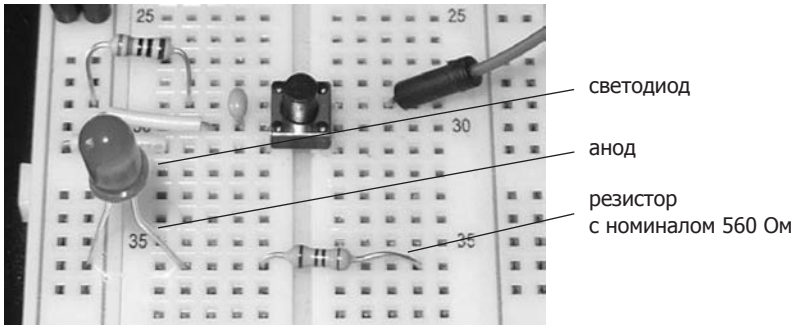


Рис. 4.24. Включение светодиода и резистора с номиналом 560 Ом

6. Соедините отрезком провода правый конец резистора с номиналом 560 Ом и цифровой контакт 12 на плате Arduino, как показано на рис. 4.25.

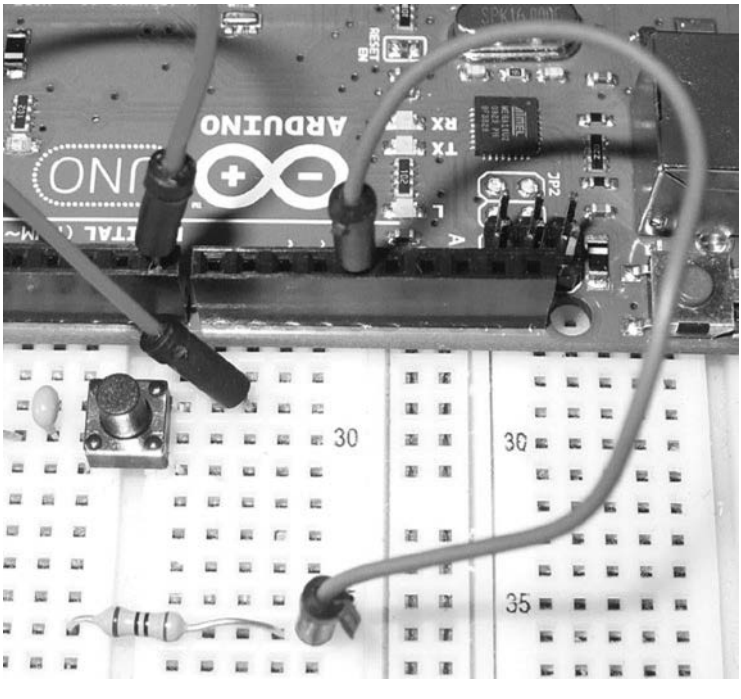


Рис. 4.25. Соединение светодиода с платой Arduino

Прежде чем продолжить, проверьте еще раз собранную цепь и убедитесь, что все компоненты соединены правильно. Сравните принципиальную схему с фактически собранной цепью.

Скетч

Введите скетч, представленный в листинге 4.1, и загрузите его в плату.

Листинг 4.1. Цифровой вход

```
// Проект 4 - Демонстрация работы цифрового входа
❶ #define LED 12
  #define BUTTON 7
  void setup()
  {
❷   pinMode(LED, OUTPUT);   // Выход для управления светодиодом
    pinMode(BUTTON, INPUT);  // Вход для кнопки
  }

  void loop()
  {
    if ( digitalRead(BUTTON) == HIGH )
    {
      digitalWrite(LED, HIGH); // включить светодиод
      delay(500);              // ждать 0.5 секунды
      digitalWrite(LED, LOW);  // выключить светодиод
    }
  }
```

Загрузив скетч, нажмите и сразу отпустите кнопку, при этом светодиод должен зажечься на полсекунды.

Изменение скетча

Добившись успеха, попробуйте изменить в скетче продолжительность интервала, в течение которого светодиод остается включенным, или добавьте управление включением светодиода кнопкой в проект 3. (Не разбирайте пока эту схему; она нам еще понадобится в следующем примере.)

Анализ скетча

Давайте исследуем новые элементы, появившиеся в скетче для проекта 4: инструкцию `#define`, определение режима работы цифрового входа и оператор `if-then`.

Определение констант с помощью #define

Перед функцией `void setup()` находятся инструкции `#define` (❶), определяющие фиксированные переменные: во время компиляции скетча IDE заменит все вхождения определяемых имен их числовыми значениями. Например, когда IDE встретит слово `LED` в строке ❷, она заменит его числом `12`.

Команду `#define` в основном мы будем использовать для обозначения в скетчах цифровых входов/выходов, к которым подключены светодиоды и кнопки. Обратите внимание, что после значения в команде `#define` отсутствует точка с запятой. Вообще говоря, номера контактов и другие фиксированные значения (такие, как время задержки) в скетчах предпочтительнее определять именно таким способом. Это удобно, потому что если значение используется в скетче несколько раз и позднее потребуется изменить его, вам не придется править его в разных местах. В данном примере константа `LED` используется трижды; чтобы исправить ее значение, достаточно всего лишь изменить определение в инструкции `#define`.

Чтение состояний цифровых входов

Чтобы получить возможность читать состояние кнопки, в функции `void setup()` контакт сначала настраивается на работу в режиме цифрового входа:

```
pinMode(BUTTON, INPUT); // Вход для кнопки
```

Затем, чтобы определить, соединяет ли кнопка цифровой вход с источником напряжения 5 В (то есть нажата ли кнопка), вызывается функция `digitalRead(pin)`, где `pin` — номер цифрового входа, состояние которого требуется прочитать. Функция возвращает значение `HIGH` (напряжение на контакте близко к 5 В) или `LOW` (напряжение на контакте близко к 0 В).

Принятие решений с помощью if

Оператор `if` позволяет принимать решения в скетчах и, в зависимости от принятого решения, выполнять различный код. Например, в скетче для проекта 4 использован код, представленный в листинге 4.2.

Листинг 4.2. Простой пример использования оператора if-then

```
// Листинг 4.2
if (digitalRead(BUTTON) == HIGH)
{
  digitalWrite(LED, HIGH); // включить светодиод
  delay(500);       // ждать 0,5 секунды
  digitalWrite(LED, LOW); // выключить светодиод
}
```

Первая строка в листинге 4.2 начинается с проверки условия в операторе `if`. Если условие истинно (то есть напряжение имеет величину `HIGH`), это означает, что кнопка нажата и можно выполнить код в фигурных скобках.

Чтобы определить факт нажатия кнопки (когда `digitalRead(BUTTON)` возвращает значение `HIGH`), мы использовали *оператор сравнения* (`==`). Если заменить `==` оператором `!=` (не равно), тогда светодиод будет выключаться в момент нажатия на кнопку. Попробуйте изменить скетч и убедитесь, что это действительно так.

ПРИМЕЧАНИЕ

Использование для сравнения единственного знака равенства (=), который означает «сделать равным», вместо двух знаков равенства (==), которые означают «проверить равенство», является типичной ошибкой. В этом случае вы можете не получить сообщения об ошибке, но оператор `if` будет работать неправильно!

Принятие альтернативных решений с помощью if-then-else

В оператор `if`, в ветвь `else`, можно добавить альтернативную последовательность действий. Например, если переписать листинг 4.1, добавив ветвь `else`, как показано в листинге 4.3, то светодиод будет включаться, *если* кнопка нажата, в противном случае он будет выключаться. Использование `else` вынуждает плату Arduino выполнить второй фрагмент кода, если проверка в операторе `if` вернула ложное значение.

Листинг 4.3. Дополнительная ветвь else

```
// Листинг 4.3
#define LED 12
#define BUTTON 7

void setup()
{
  pinMode(LED, OUTPUT);   // Выход для управления светодиодом
  pinMode(BUTTON, INPUT); // Вход для кнопки
}

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

Логические переменные

Иногда требуется запомнить некоторую характеристику, имеющую только два состояния, например включено/выключено или горячо/холодно. *Логическая (булева) переменная* — легендарный компьютерный «бит», который может принимать только два значения: ноль (0, ложь) или один (1, истина). Логические переменные как раз и пригодятся для хранения состояний таких характеристик: они могут хранить

только одно из двух значений: `true` или `false`. Подобно любым другим переменным, они должны объявляться перед использованием:

```
boolean raining = true; // создать переменную "raining" (дождь)
                        // и присвоить ей начальное значение true
```

Изменить состояние логической переменной в скетче можно простым присваиванием, например:

```
raining = false;
```

Логические переменные удобно использовать для принятия решений в операторе `if`. Так как логические величины могут сравниваться со значениями `true` и `false`, к ним допускается применять операторы сравнения `!=` и `==`. Например:

```
if ( raining == true )
{
  if ( summer != true )
  {
    // Идет дождь, но не летом
  }
}
```

Операторы сравнения

Для принятия решений на основе значений двух или более логических переменных или других состояний используются несколько дополнительных операторов. В их число входят операторы *НЕ* (`!`), *И* (`&&`) и *ИЛИ* (`||`).

Оператор НЕ

Оператор *НЕ* обозначается восклицательным знаком (`!`). Он используется как сокращенная форма проверки того, что некоторое состояние *НЕ ИСТИННО*. Например:

```
if ( !raining )
{
  // дождя нет (raining == false)
}
```

Оператор И

Логический оператор *И* обозначается как `&&`. Он помогает уменьшить число отдельных проверок с помощью оператора `if`. Например:

```
if (( raining == true ) && ( !summer ))
{
  // Идет дождь, но не летом (raining == true И summer == false)
}
```

Оператор ИЛИ

Логический оператор *ИЛИ* обозначается как ||. Использовать его очень просто, например:

```
if (( raining == true ) || ( summer == true ))
{
  // этот код выполняется в дождь или летом
}
```

Выполнение двух и более сравнений

В одном операторе `if` можно выполнить два или несколько сравнений. Например:

```
if ( snow == true && rain == true && !hot )
{
  // этот код выполняется, когда идет снег с дождем и не жарко
}
```

Для определения порядка выполнения операций используются круглые скобки. В следующем примере сначала проверяется истинность или ложность условия в круглых скобках, а затем выполняется последнее сравнение в инструкции `if-then`.

```
if (( snow == true || rain == true ) && hot == false))
{
  // этот код выполняется, когда идет снег или дождь и не жарко
}
```

Наконец, так же как в примерах с оператором *НЕ* (`!`), простые проверки на истинность или ложность могут выполняться без сравнения с помощью `== true` или `== false`. Следующий код действует точно так же, как в предыдущем примере:

```
if (( snow || rain ) && !hot )
{
  // этот код выполняется, когда идет снег или дождь и не жарко
  // ("идет снег" – истинно ИЛИ "идет дождь" - истинно) И НЕ жарко
}
```

Как видите, с помощью операторов сравнения и логических переменных можно организовать принятие решений, проверяя самые разные условия. Перейдя к более сложным проектам, вы по достоинству оцените эту возможность.

Проект № 5: Управление движением

Теперь применим вновь приобретенные знания для решения гипотетической проблемы. Вообразите себя градостроителем и представьте, что у вас есть мост через

реку, имеющий единственную полосу для движения автотранспорта. Каждую неделю по ночам происходит одно-два дорожно-транспортных происшествия, когда уставшие водители мчатся через мост, забывая остановиться и убедиться, что путь свободен. Вы предложили установить светофор, но чиновники хотят увидеть на макете, как он будет действовать, прежде чем подписать распоряжение о выделении средств. Можно было бы арендовать переносные светофоры, но они дороги. Поэтому вы решили сконструировать модель моста с действующим светофором, собранным на светодиодах и плате Arduino.

Цель

Наша цель — установить трехцветные светофоры на обоих концах моста. Светофоры должны разрешать движение по мосту только в одном направлении в каждый конкретный момент времени. Когда датчики на одном конце моста обнаруживают автомобиль, ожидающий включения зеленого сигнала, светофоры должны переключиться и разрешить движение.

Алгоритм

Для имитации датчиков обнаружения автомобилей на обоих концах моста мы используем две кнопки. Светофор с каждой стороны будет состоять из светодиодов красного, желтого и зеленого цвета. Первоначально система разрешает движение с запада на восток, поэтому на светофоре, обращенном на запад, должен гореть зеленый свет, а на светофоре, обращенном на восток, — красный.

Когда к мосту приближается автомобиль (моделируется нажатием кнопки) и на светофоре горит красный свет, система должна переключить свет на противоположном конце с зеленого на желтый, а затем на красный. После этого она должна выждать некоторое время, чтобы позволить автомобилям, уже находящимся на мосту, завершить его пересечение. Далее, на стороне с ожидающим автомобилем должен включиться желтый мигающий свет, означающий «приготовиться к движению», и затем желтый свет должен смениться зеленым. Зеленый свет должен гореть, пока на противоположной стороне не появится автомобиль, после чего процесс должен повториться в обратном направлении.

Оборудование

Ниже перечислено оборудование, которое потребуется для реализации этого проекта:

- Два красных светодиода (LED1 и LED2).
- Два желтых светодиода (LED3 и LED4).
- Два зеленых светодиода (LED5 и LED6).

- Шесть резисторов с номиналом 560 Ом (R1—R6).
- Два резистора с номиналом 10 кОм (R7 и R8).
- Два конденсатора емкостью 100 нФ (C1 и C2).
- Две кнопки без фиксации (S1 и S2).
- Одна макетная плата среднего размера.
- Одна плата Arduino и кабель USB.
- Несколько отрезков провода разной длины.

Схема

Так как предполагается управлять всего шестью светодиодами и принимать сигналы с двух кнопок, проект имеет достаточно простую схему, она изображена на рис. 4.26.

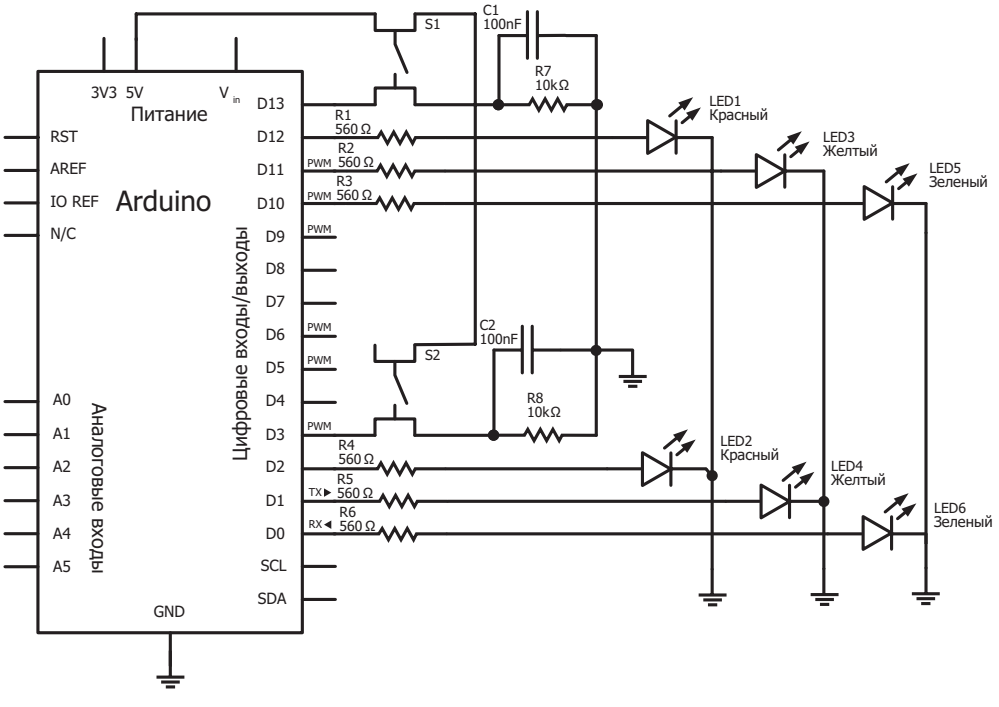


Рис. 4.26. Принципиальная схема для проекта 5

Эта схема сложнее версии с кнопкой и светодиодом из проекта 4 и имеет больше резисторов, больше светодиодов и еще одну кнопку.

Будьте внимательны при установке светодиодов, постарайтесь не перепутать полярность: резисторы должны подключаться к анодам светодиодов, а катоды светодиодов должны подключаться к контакту GND на плате Arduino, как показано на рис. 4.27.

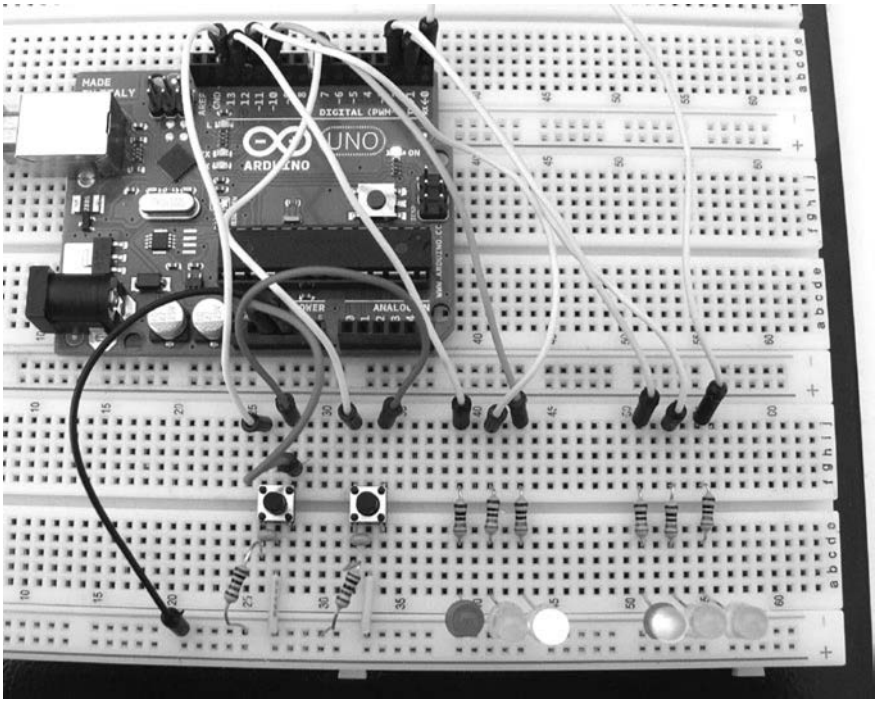


Рис. 4.27. Собранная схема

Скетч

А теперь перейдем к скетчу. Сможете ли вы самостоятельно проверить его соответствие алгоритму?

```
  // Проект 5 - Управление движением

  // определение контактов для подключения кнопок и светодиодов:
❶ #define westButton 3
  #define eastButton  13
  #define westRed     2
  #define westYellow  1
  #define westGreen   0
  #define eastRed     12
  #define eastYellow  11
  #define eastGreen   10
```

```
#define yellowBlinkTime 500  // период мигания желтого света 0,5 секунды

❷ boolean trafficWest = true; // запад = true, восток = false
❸ int flowTime = 10000;       // период ожидания, чтобы пропустить
                              // автомобили, уже находящиеся на мосту
❹ int changeDelay = 2000;     // задержка перед сменой цвета

void setup()
{
  // настройка цифровых входов/выходов
  pinMode(westButton, INPUT);
  pinMode(eastButton, INPUT);
  pinMode(westRed, OUTPUT);
  pinMode(westYellow, OUTPUT);
  pinMode(westGreen, OUTPUT);
  pinMode(eastRed, OUTPUT);
  pinMode(eastYellow, OUTPUT);
  pinMode(eastGreen, OUTPUT);

  // начальное состояние светофоров - зеленый на западной стороне
  digitalWrite(westRed, LOW);
  digitalWrite(westYellow, LOW);
  digitalWrite(westGreen, HIGH);
  digitalWrite(eastRed, HIGH);
  digitalWrite(eastYellow, LOW);
  digitalWrite(eastGreen, LOW);
}

void loop()
{
  // запрошено движение с запада на восток?
  if ( digitalRead(westButton) == HIGH )
  {
    // продолжать, только если движение меняется на
    // противоположное
    if ( trafficWest != true )
    {
      trafficWest = true; // изменить флаг направления запад->восток
      delay(flowTime);    // дать автомобилям время пересечь мост
      digitalWrite(eastGreen, LOW);   // на восточной стороне погасить
      digitalWrite(eastYellow, HIGH); // зеленый сигнал, зажечь желтый
      delay(changeDelay);             // и затем красный
      digitalWrite(eastYellow, LOW);
      digitalWrite(eastRed, HIGH);
      delay(changeDelay);
      for ( int a = 0; a < 5; a++ )  // воспроизвести мигающий желтый
      {
        digitalWrite(westYellow, LOW);
        delay(yellowBlinkTime);
        digitalWrite(westYellow, HIGH);
        delay(yellowBlinkTime);
      }
```

```
        digitalWrite(westYellow, LOW);
        digitalWrite(westRed, LOW);    // сменить сигнал на западной
        digitalWrite(westGreen, HIGH); // стороне с красного на зеленый
      }
    }

    // запрошено движение с запада на восток?
    if ( digitalRead(eastButton) == HIGH )
    {
      // продолжать, только если движение меняется на
      // противоположное
      if ( trafficWest == true )
      {
        trafficWest = false; // изменить флаг направления восток->запад
        delay(flowTime);     // дать автомобилям время пересечь мост
        digitalWrite(westGreen, LOW);

        // на восточной стороне сменить зеленый
        // сигнал на желтый и затем на красный
        digitalWrite(westYellow, HIGH);
        delay(changeDelay);
        digitalWrite(westYellow, LOW);
        digitalWrite(westRed, HIGH);
        delay(changeDelay);
        for ( int a = 0 ; a < 5 ; a++ ) // воспроизвести мигающий желтый
        {
          digitalWrite(eastYellow, LOW);
          delay(yellowBlinkTime);
          digitalWrite(eastYellow, HIGH);
          delay(yellowBlinkTime);
        }
        digitalWrite(eastYellow, LOW);
        digitalWrite(eastRed, LOW);    // сменить сигнал на восточной
        digitalWrite(eastGreen, HIGH); // стороне с красного на зеленый
      }
    }
  }
```

Скетч начинается с определения (❶) соответствий между номерами контактов цифровых входов/выходов и смысловыми именами светодиодов и двух кнопок. С каждой стороны моста у нас имеются красный, желтый и зеленый светодиоды и кнопка. Логическая переменная `trafficWest` (❷) хранит текущее направление движения по мосту — `true` (с запада на восток) и `false` (с востока на запад).

ПРИМЕЧАНИЕ

Обратите внимание, что `trafficWest` — единственная логическая переменная, определяющая направление движения как `true` или `false`. Наличие единственной переменной, как в данном скетче, вместо двух (одна для направления на восток, другая — на запад) гарантирует, что оба направления не могут быть выбраны одновременно. Это поможет избежать аварий на мосту.

Целочисленная переменная flowTime (❸) определяет минимальный период времени, который отводится автомобилям для завершения движения по мосту. Когда со стороны, где горит красный свет, появляется автомобиль, система выполняет задержку на этот период, чтобы дать время автомобилям, уже находящимся на мосту, закончить его пересечение. Целочисленная переменная changeDelay (❹) определяет период времени между переключениями светофора с зеленого на желтый и на красный.

Прежде чем скетч вызовет функцию `void loop()`, в функции `void setup()` устанавливается направление движения с запада на восток.

Запуск скетча

После запуска скетч ничего не делает, пока не будет нажата одна из кнопок. Когда же кнопка будет нажата на восточной стороне, следующая строка

```
if ( trafficWest == true )
```

разрешит изменение сигналов светофора, только если движение по мосту осуществляется в противоположном направлении. Остальной код в этом разделе реализует простую последовательность из задержек и команд включения/выключения разных светодиодов, имитирующую работу светофора.

Аналоговые и цифровые сигналы

В этом разделе вы узнаете, чем отличаются цифровые и аналоговые сигналы и как измерять величину аналоговых сигналов на контактах аналоговых входов.

До сих пор мы использовали в своих скетчах только цифровые электрические сигналы, имеющие два дискретных уровня. В частности, мы использовали `digitalWrite(pin, HIGH)` и `digitalWrite(pin, LOW)`, чтобы реализовать мигание светодиода, и `digitalRead()`, для определения наличия (`HIGH`) или отсутствия (`LOW`) напряжения на контакте. На рис. 4.28 изображено визуальное представление цифрового сигнала, изменяющегося между высоким и низким уровнями.

В отличие от цифровых, аналоговые сигналы могут изменяться с неопределенным шагом между высоким и низким уровнем. Например, на рис. 4.29 изображен аналоговый сигнал в виде синусоиды. Обратите внимание, что с течением времени напряжение плавно меняется между высоким и низким уровнем.

На плате Arduino высокий уровень близок к 5 В, низкий — к 0 В. Плата имеет шесть аналоговых входов (рис. 4.30), с помощью которых можно измерять значения аналоговых сигналов. Эти аналоговые входы позволяют безопасно измерять напряжение от 0 до 5 В.

Если вызвать функцию `analogRead()`, она вернет число в диапазоне от 0 до 1023, пропорциональное напряжению, приложенному к контакту аналогового входа.

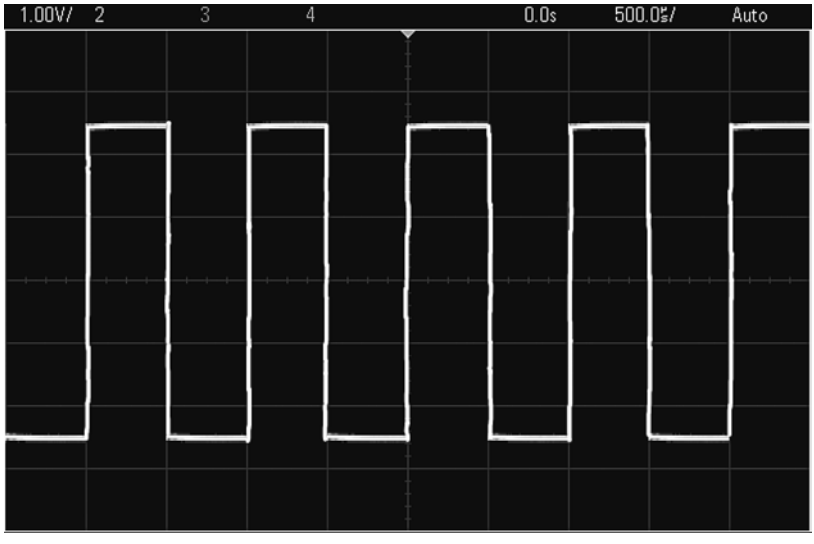


Рис. 4.28. Цифровой сигнал с уровнями HIGH, которые выглядят как горизонтальные отрезки в верхней части, и LOW — в нижней части

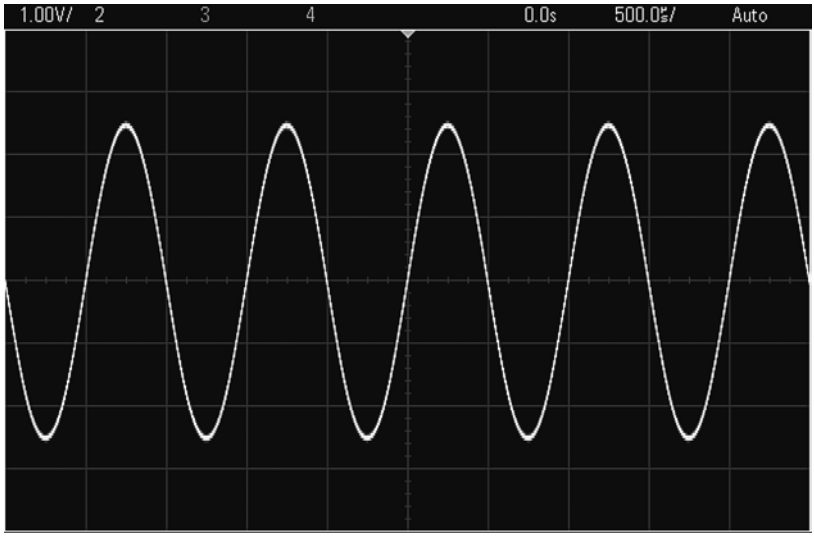


Рис. 4.29. Аналоговый сигнал в форме синусоиды

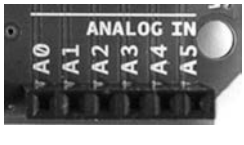


Рис. 4.30. Аналоговые входы на плате Arduino Uno

Например, с помощью `analogRead()` можно сохранить значение напряжения на аналоговом входе `A0` в целочисленной переменной `a`:

```
a = analogRead(0); // прочитать напряжение на аналоговом входе 0 (A0)
                   // вернет число в диапазоне от 0 до 1023, обычно
                   // соответствующем диапазону от 0,000 до 4,995 вольт
```

Проект № 6: Тестер для одноэлементных батареек

Несмотря на снижение спроса на одноэлементные батарейки, многие все еще пользуются бытовыми приборами и устройствами, питающимися от батареек типа AA, AAA, C или D, такими как пульты дистанционного управления, часы или детские игрушки. Подобные батарейки имеют напряжение намного меньше 5 В, поэтому можно измерить его с помощью Arduino и определить состояние элемента. В этом проекте мы создадим тестер для батареек.

Цель

Новые одноэлементные батарейки, например типа AA, обычно имеют напряжение около 1,6 В, уровень которого снижается по мере использования. В нашем проекте мы измерим напряжение на батарейке и отобразим степень ее пригодности при помощи светодиодов. Напряжение будет определяться путем чтения значения `analogRead()` и преобразования в вольты. Максимальное напряжение, которое можно подать на аналоговый вход, равно 5 В, поэтому, поделив 5 на 1024 (число возможных значений), мы получим число 0,0048, которое и будем использовать как коэффициент для преобразования прочитанного значения в вольты. То есть если `analogRead()` вернет число 512, то, умножив его на 0,0048, мы получим напряжение 2,4576 В.

Алгоритм

Рассмотрим алгоритм работы тестера батареек:

1. Прочитать значение с аналогового входа.
2. Умножить прочитанное значение на коэффициент 0,0048, чтобы получить величину напряжения в вольтах.
3. Если напряжение больше или равно 1,6 В, включить на короткий промежуток зеленый светодиод.
4. Если напряжение больше 1,4 В и меньше 1,6 В, включить на короткий промежуток желтый светодиод.
5. Если напряжение меньше 1,4 В, включить на короткий промежуток красный светодиод.
6. Повторять до бесконечности.

Оборудование

Ниже перечислено оборудование, которое потребуется для реализации этого проекта:

- Три резистора с номиналом 560 Ом (R1—R3).
- Один зеленый светодиод (LED1).
- Один желтый светодиод (LED2).
- Один красный светодиод (LED3).
- Одна макетная плата.
- Несколько отрезков провода разной длины.
- Одна плата Arduino и кабель USB.

Схема

На рис. 4.31 изображена принципиальная схема тестера для одноэлементных батареек. Обратите внимание на два контакта слева, подписанных как + и –. К этим

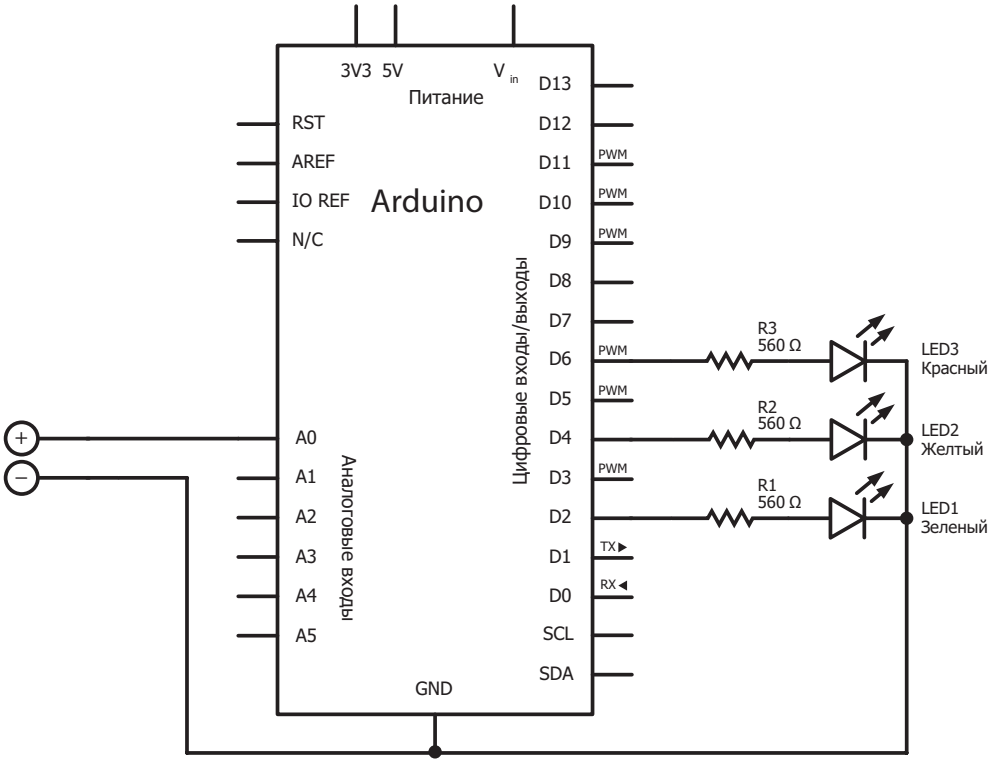


Рис. 4.31. Принципиальная схема для проекта 6

контактам должны подключаться, *с соблюдением полярности*, проверяемые батарейки. Положительный полюс батарейки должен подключаться к положительному контакту (+), а отрицательный — к отрицательному (–).

ВНИМАНИЕ

Ни при каких условиях не пытайтесь измерять напряжение выше 5 В и не подключайте положительный полюс батарейки к отрицательному контакту или наоборот, подобные действия выведут из строя вашу плату Arduino.

Скетч

А теперь скетч:

```
// Проект 6 - Тестер для одноэлементных батареек
#define newLED 2 // зеленый светодиод 'новая'
#define okLED 4  // желтый светодиод 'норма'
#define oldLED 6 // красный светодиод 'старая'

int analogValue = 0;
❶ float voltage = 0;
int ledDelay = 2000;

void setup()
{
  pinMode(newLED, OUTPUT);
  pinMode(okLED, OUTPUT);
  pinMode(oldLED, OUTPUT);
}

void loop()
{
❷  analogValue = analogRead(0);
❸  voltage = 0.0048*analogValue;
❹  if ( voltage >= 1.6 )
   {
     digitalWrite(newLED, HIGH);
     delay(ledDelay);
     digitalWrite(newLED, LOW);
   }
❺  else if ( voltage < 1.6 && voltage > 1.4 )
   {
     digitalWrite(okLED, HIGH);
     delay(ledDelay);
     digitalWrite(okLED, LOW);
   }
❻  else if ( voltage <= 1.4 )
   {
     digitalWrite(oldLED, HIGH);
     delay(ledDelay);
     digitalWrite(oldLED, LOW);
   }
}
```

Скетч для проекта 6 считывает значение с аналогового входа 0 (❷) и преобразует его в напряжение (❸). С новым типом переменных — `float` (❶) — вы познакомитесь в следующем разделе. В скетче находятся уже знакомые вам конструкции, такие как оператор `if-else`, и кое-что новое, например арифметические операции и операторы сравнения чисел, которые мы рассмотрим далее.

Выполнение арифметических операций в Arduino

Подобно карманному калькулятору, Arduino выполняет различные арифметические операции, такие как умножение, деление, сложение и вычитание. Ниже приводится несколько примеров:

```
a = 100;
b = a + 20;
c = b - 200;
d = c + 80; // d получит значение 0
```

Вещественные переменные

Когда требуется оперировать вещественными числами, в скетчах используются переменные типа `float`. В таких переменных можно хранить значения от $3{,}4028235 \times 10^{38}$ до $-3{,}4028235 \times 10^{38}$, точность которых ограничена шестью-семью десятичными знаками. В вычислениях допускается смешивать целые и вещественные числа. Например, можно сложить вещественное число `f` с целым числом `a` и сохранить результат в вещественной переменной `g`:

```
int a = 100;
float f;
float g;
f = a / 3; // f = 33.333333
g = a + f; // g = 133.333333
```

Операторы сравнения чисел

Мы уже использовали операторы сравнения `==` и `!=` с инструкциями `if` в проекте 5, где определяли уровень входных цифровых сигналов. В дополнение к ним существует еще несколько операторов, которые можно применять к числам или числовым переменным:

- `<` меньше;
- `>` больше;

- <= меньше или равно;
- >= больше или равно.

Мы использовали эти операторы в строках ❹, ❺ и ❻, в скетче для проекта 6, описанного выше.

Увеличение точности измерения аналоговых сигналов с помощью источника опорного напряжения

Как было показано в проекте 6, функция `analogRead()` возвращает значение, пропорциональное напряжению из диапазона от 0 до 5 В. Верхнее значение (5 В), его называют *опорным напряжением*, является максимально допустимым напряжением, которое можно подавать на аналоговые входы Arduino и для которого возвращается наивысшее значение 1023.

Чтобы увеличить точность измерения более низких напряжений, задействуем низковольтный источник опорного напряжения. Например, когда опорное напряжение равно 5 В, функция `analogRead()` представляет измеренное напряжение как значение от 0 до 1023. Однако если потребуется измерять напряжение (например) не выше 2 В, можно заставить Arduino представлять числами 0—1 023 диапазон от 0 до 2 В, чтобы повысить точность измерений. Для этого необходимо задействовать внутренний или внешний источник опорного напряжения, как описывается далее.

Использование внешнего источника опорного напряжения

Первый метод заключается в подаче опорного напряжения на контакт AREF (analog reference — опорный аналоговый сигнал), как показано на рис. 4.32.

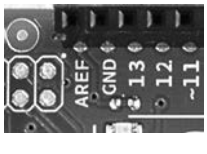


Рис. 4.32. Контакт AREF на плате Arduino Uno

Подать новое опорное напряжение можно, подключив положительный полюс внешнего источника питания к контакту AREF на плате Arduino, а отрицательный — к контакту GND. Помните, что в качестве опорного можно использовать только более низкое напряжение, но нельзя более высокое, потому что опорное напряжение, подводимое к плате Arduino Uno, не должно превышать 5 В. Самый простой способ

подвести более низкое опорное напряжение — собрать делитель напряжения на двух резисторах, как показано на рис. 4.33.

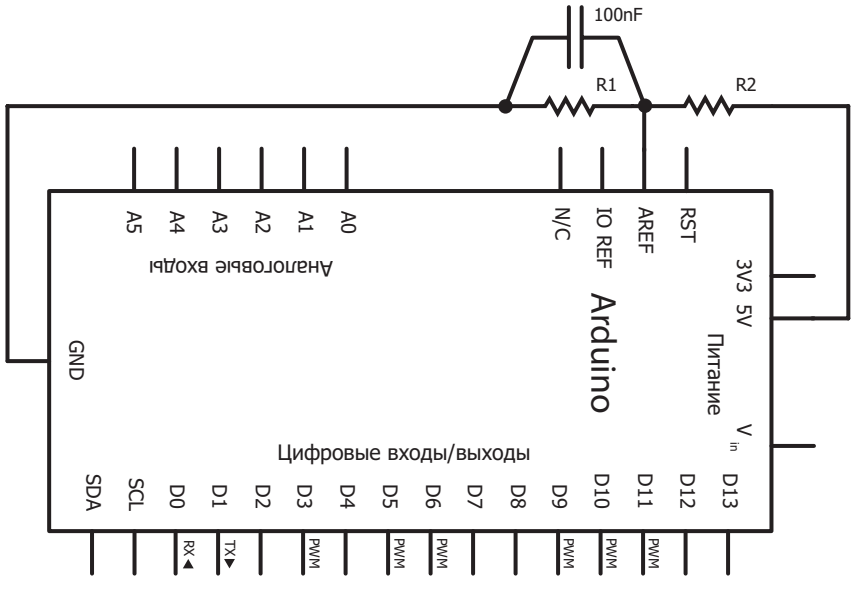


Рис. 4.33. Делитель напряжения

Величина опорного напряжения в этом случае определяется сопротивлением резисторов *R*1 и *R*2 по следующей формуле:

$$V_{out} = V_{in}\left(\frac{R2}{R1+R2}\right),$$

где V_{out} — величина опорного напряжения, а V_{in} — входное напряжение, в данном случае 5 В; *R*1 и *R*2 — сопротивление резисторов в омах.

Простейший способ поделить напряжение — разбить V_{in} пополам, выбрав резисторы *R*1 и *R*2 с одинаковым сопротивлением, например 10 кОм каждый. Для делителя лучше использовать резисторы с наименьшим допуском отклонений, какие только сможете найти, например 1 %; проверьте их истинное сопротивление при помощи мультиметра и используйте измеренные значения в вычислениях. Кроме того, нелишним будет добавить конденсатор емкостью 100 нФ между контактами AREF и GND, чтобы избавиться от шумов на контакте AREF и уменьшить нестабильность измерений.

При использовании внешнего источника опорного напряжения вставьте следующую строку в функцию `void setup()`:

```
analogReference(EXTERNAL); // использовать контакт AREF в качестве
                           // источника опорного напряжения
```

Использование внутреннего источника опорного напряжения

В Arduino Uno имеется также внутренний источник опорного напряжения 1,1 В. Если этот уровень соответствует вашим потребностям, вам не придется использовать дополнительные электронные компоненты. Просто добавьте следующую строку в функцию `void setup()`:

```
analogReference(INTERNAL); // выбрать внутренний источник
                           // опорного напряжения 1,1 В
```

Переменный резистор

Переменные резисторы, также известные как *потенциометры*, позволяют изменять их сопротивление от 0 Ом до соответствующего им номинального значения. На принципиальных схемах переменные резисторы обозначаются, как показано на рис. 4.34.

Переменные резисторы имеют три контакта: один центральный и по одному с каждого конца. По мере вращения штока переменного резистора сопротивление между одним крайним и центральным контактом увеличивается, а между другим крайним и центральным контактом — уменьшается.

Переменные резисторы могут быть *линейными* или *логарифмическими*. Сопротивление переменного резистора с линейной характеристикой изменяется прямо пропорционально углу поворота штока, а сопротивление резистора с логарифмической характеристикой изменяется сначала медленно, затем все быстрее и быстрее. Логарифмические потенциометры чаще используются в звукоусилительной аппаратуре, потому что точнее соответствуют кривой восприятия человеческого уха. В большинстве проектов для Arduino используются линейные переменные резисторы, такие как резистор, изображенный на рис. 4.35.

Рис. 4.34. Обозначение переменного резистора (потенциометра)

Рис. 4.35. Типичный линейный переменный резистор

Существуют также переменные резисторы в миниатюрном исполнении, они часто называются *подстроечными* (рис. 4.36). Благодаря небольшим размерам подстроечные резисторы удобно использовать в конструкциях для регулировки, а также для создания прототипов, потому что они отлично умещаются на макетной плате.

Рис. 4.36. Различные подстроечные резисторы

ПРИМЕЧАНИЕ

Приобретая подстроечные резисторы, обращайте внимание на их тип. Зачастую предпочтительнее использовать такие, которые легко регулируются при помощи отвертки, имеющейся под рукой. Кроме того, резисторы с закрытым корпусом, такие как, например, изображенные на рис. 4.36, служат дольше своих более дешевых аналогов с открытыми контактами.

Пьезоэлектрические зуммеры

Пьезоэлектрический зуммер (или просто пьезозуммер) — это маленькое устройство в цилиндрическом корпусе, которое можно использовать для подачи громких и раздражающих звуковых сигналов, например для предупреждения об аварии или для забавы. На рис. 4.37 изображен зуммер TDK PS1240 рядом с 25-центовой монетой США[1], чтобы вы могли получить представление о его размерах.

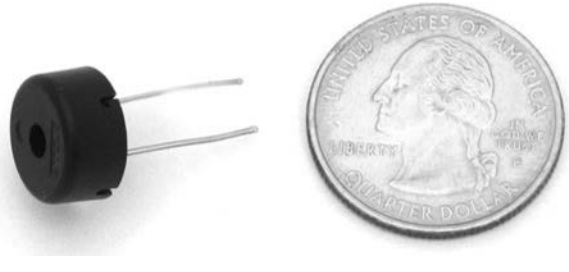


Рис. 4.37. Пьезоэлектрический зуммер TDK PS1240

[1] Диаметр 25-цетовой монеты (24,3 мм) практически совпадает с диаметром современной пятирублевой монеты (25 мм). — *Примеч. пер.*

Внутри зуммера находится очень тонкая пластина, которая изменяет форму при подаче напряжения. Когда ток подается на зуммер импульсами, пластина начинает вибрировать, генерируя звуковые волны определенной частоты.

Зуммер может пригодиться в наших проектах, потому что его можно быстро включать и выключать, как светодиод, имитируя импульсы. Пьезоэлементы не имеют полярности и могут подключаться как угодно.

Изображение пьезоэлектрических зуммеров на схемах

Значок, обозначающей зуммер на схеме, напоминает динамик (рис. 4.38), что упрощает его идентификацию.

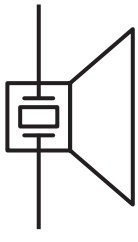


Рис. 4.38. Обозначение пьезоэлектрического зуммера на схемах

ПРИМЕЧАНИЕ

Приобретая зуммер для данного проекта, убедитесь, что выбранная модель *содержит только пьезоэлемент*; некоторые зуммеры похожи на изображенный на рис. 4.38, но имеют встроенную схему, генерирующую звук. Такой зуммер для данного проекта не подходит, потому что мы собираемся управлять высотой звука с помощью Arduino.

Проект № 7: Испытание пьезоэлектрического зуммера

Если у вас есть пьезозуммер и желание опробовать его, загрузите следующий демонстрационный скетч в свою плату Arduino:

```
 // Проект 7 - Испытание пьезоэлектрического зуммера
#define PIEZO 3 // контакт 3 может выводить сигналы ШИМ
                // для управления высотой звука
int del = 500;

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

```
  void loop()
  {
❶   analogWrite(PIEZO, 128); // сгенерировать импульсный сигнал ШИМ
                             // с коэффициентом заполнения 50%
    delay(del);
    digitalWrite(PIEZO, LOW); // выключить зуммер
    delay(del);
  }
```

Этот скетч выводит сигнал с широтно-импульсной модуляцией на третий контакт. Изменяя коэффициент заполнения в вызове функции `analogWrite()`, который сейчас равен 128, что составляет 50 % (❶), можно управлять громкостью звучания зуммера.

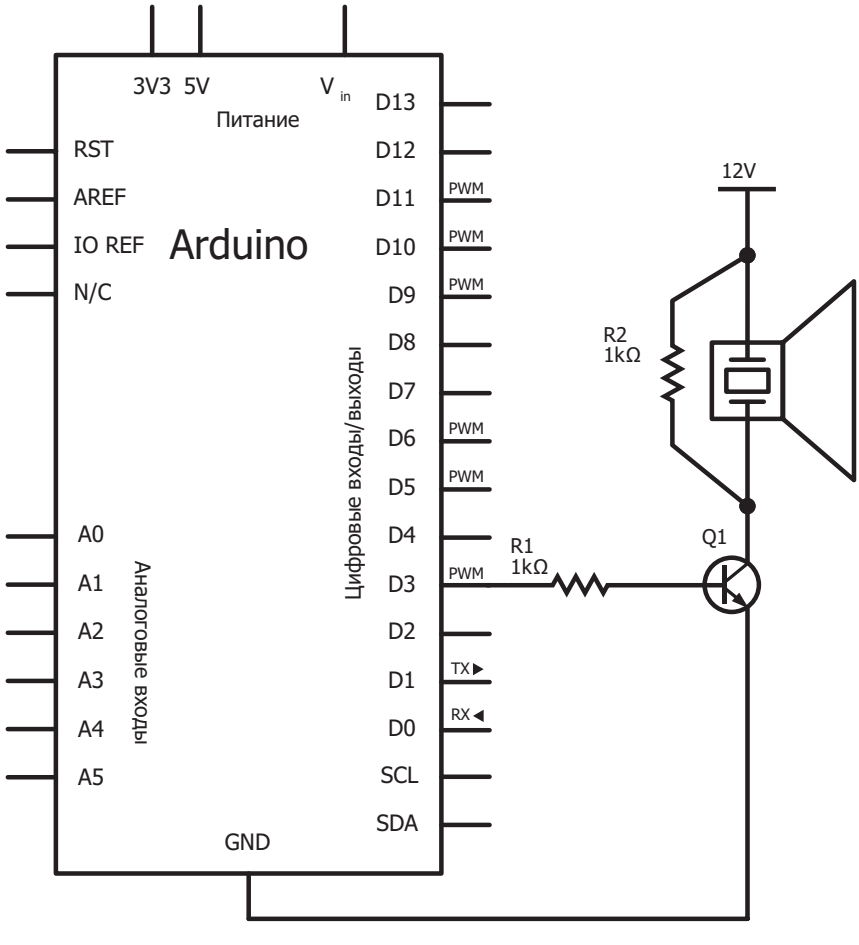


Рис. 4.39. Принципиальная схема для проекта 7

Чтобы увеличить громкость звучания, увеличьте напряжение, приложенное к зуммеру. В настоящее время максимальное напряжение ограничено 5 В, но зуммер будет издавать более громкий звук, если приложить к нему напряжение 9 или 12 В. Так как Arduino не имеет возможности выдавать более высокое напряжение, вам понадобится внешний источник питания, например 9-вольтовая батарея, и транзистор в качестве электронного переключателя. При этом мы будем использовать тот же скетч и схему, изображенную на рис. 4.39.

Элемент на схеме, обозначенный как 12 V, — это точка подключения положительного полюса внешнего источника более высокого напряжения, отрицательный полюс которого должен быть подключен к контакту GND на плате Arduino.

Проект № 8: Быстродействующий термометр

Температура может быть представлена аналоговым сигналом, например, генерируемым температурным датчиком TMP36 (рис. 4.40), который производится компанией Analog Devices (*http://www.analog.com/tmp36/*).

Рис. 4.40. Температурный датчик TMP36

Обратите внимание, что датчик TMP36 похож на транзистор BC548, который мы использовали для управления реле в главе 3. Датчик TMP36 выдает напряжение, пропорциональное температуре, благодаря чему текущую температуру можно определять простым преобразованием. Например, температуре 25 °C соответствует выходное напряжение 750 мВ, и каждое изменение температуры на 1 градус влечет изменение напряжения на 10 мВ. Датчик TMP36 может измерять температуры в диапазоне от –40 °C до 125 °C.

Функция `analogRead()` возвращает значение в диапазоне от 0 до 1023, который соответствует диапазону напряжения от 0 до (почти) 5000 мВ (5 В). Если умножить результат вызова `analogRead()` на (5000/1024), получится напряжение, фактически возвращаемое сенсором. Далее нужно вычесть 500 (смещение, используемое датчиком TMP36, чтобы обеспечить возможность измерения отрицательных температур), и разделить на 10. В результате получится температура в градусах Цельсия. Если вы предпочитаете использовать шкалу Фаренгейта, умножьте температуру в градусах Цельсия на 1,8 и прибавьте 32.

Цель

В этом проекте мы используем датчик TMP36 для создания быстродействующего термометра. Когда температура опустится ниже +20 °C, должен включиться синий светодиод. При температуре в диапазоне от +20 °C до +26 °C должен быть включен зеленый светодиод, а когда температура превысит +26 °C, должен включаться красный светодиод.

Оборудование

Ниже перечислено оборудование, необходимое для реализации данного проекта:

- Три резистора с номиналом 560 Ом (R1—R3).
- Одна кнопка.
- Один красный светодиод (LED1).
- Один зеленый светодиод (LED2).
- Один синий светодиод (LED3).
- Один датчик температуры TMP36.
- Одна макетная плата.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Устройство имеет очень простую схему. Если смотреть на датчик температуры TMP36 со стороны маркировки, вывод слева должен подключаться к источнику напряжения 5 В, центральный вывод является выходом напряжения, соответствующего температуре, и вывод справа должен подключаться к «земле» (GND), как показано на рис. 4.41.

Скетч

А теперь скетч:

```
// Проект 8 - Быстродействующий термометр

// определение контактов подключения светодиодов:
#define HOT    6
#define NORMAL 4
#define COLD   2
```

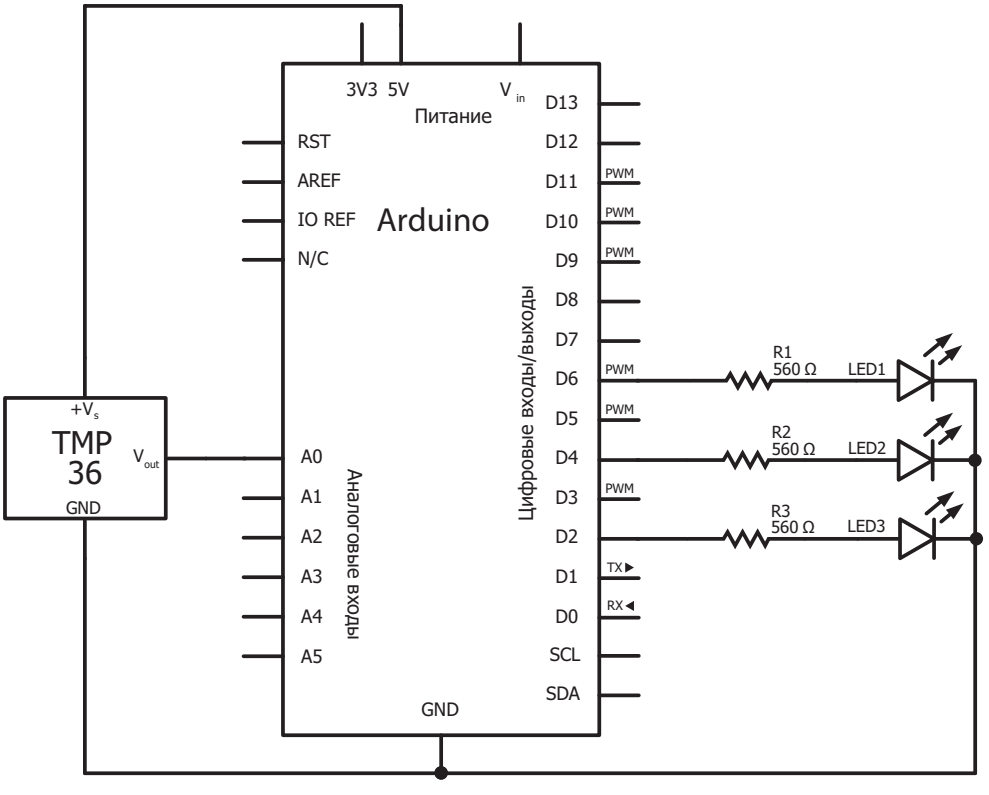


Рис. 4.41. Принципиальная схема для проекта 8

```
float voltage = 0;
 float celsius = 0;
 float hotTemp  = 26;
 float coldTemp = 20;
 float sensor = 0;

 void setup()
 {
   pinMode(HOT, OUTPUT);
   pinMode(NORMAL, OUTPUT);
   pinMode(COLD, OUTPUT);
 }

 void loop()
 {
   // прочитать напряжение с датчика
   // и преобразовать в градусы Цельсия
❶   sensor = analogRead(0);
   voltage = (sensor*5000)/1024; // преобразовать в милливольты
   voltage = voltage-500;        // учесть смещение
```

```
    celsius = voltage/10;         // преобразовать милливольты в градусы

    // выполнить действия для разных диапазонов температур
❷   if ( celsius < coldTemp )
    {
      digitalWrite(COLD, HIGH);
      delay(1000);
      digitalWrite(COLD, LOW);
    }
❸   else if ( celsius > coldTemp && celsius <= hotTemp )
    {
      digitalWrite(NORMAL, HIGH);
      delay(1000);
      digitalWrite(NORMAL, LOW);
    }
    else
    {
      // celsius > hotTemp
      digitalWrite(HOT, HIGH);
      delay(1000);
      digitalWrite(HOT, LOW);
    }
  }
```

Вначале скетч считывает напряжение, поступающее с датчика TMP36, и преобразует его в градусы Цельсия (❶). Далее с помощью оператора `if-else` (❷ и ❸) текущая температура сравнивается с границами холодного и жаркого диапазонов, и включается соответствующий светодиод. Инструкции `delay(1000)` предотвращают слишком быстрое переключение светодиодов, когда температура колеблется на границе двух диапазонов.

Доработка скетча

Несмотря на всю простоту скетча, его с успехом можно использовать в качестве основы для управления приборами на базе показаний разных датчиков. Например, можно добавить в схему модуль управления PowerSwitch Tail, изображенный на рис. 4.42.

Модуль PowerSwitch Tail позволяет безопасно управлять приборами, питающимися от обычной бытовой электросети, такими как обогреватели, лампы и т. д., с помощью цифрового выхода Arduino. (За дополнительной информацией обращайтесь по адресу *http://www.powerswitchtail.com/*.) Например, модуль PowerSwitch Tail пригодится для создания обогревателя или вентилятора, управляемого температурой, для управления освещением в гараже, чтобы оно автоматически выключалось через определенный интервал времени после включения, или удаленного управления новогодней иллюминацией на улице.

Рис. 4.42. Модуль PowerSwitch Tail, управляющий включением переменного напряжения до 120 В

Забегая вперед

Вот и четвертая глава подошла к концу. Теперь ваш арсенал пополнился дополнительными инструментами, такими как цифровые входы и выходы, новые типы переменных и разнообразные математические функции. В следующей главе вы продолжите развлекаться со светодиодами, научитесь создавать собственные функции, сконструируете компьютерную игру с электронным кубиком и многое другое.

5 Функции

В этой главе вы:

- ✓ научитесь создавать свои функции;
- ✓ узнаете, как принимать решения с помощью циклов `while` и `do-while`;
- ✓ освоите обмен данными между платой Arduino и окном монитора последовательного порта;
- ✓ познакомитесь с переменными типа `long`.

Далее вы узнаете о новых методах, облегчающих чтение скетчей для Arduino и упрощающих их проектирование за счет создания собственных функций, а также научитесь писать модульный код, поддерживающий возможность многократного использования, который поможет существенно сэкономить время. Вы узнаете, как принимать решения, управляющие выполнением блоков кода, и встретитесь с новым типом целочисленных переменных, именуемых `long`. Затем вы приступите к реализации очередного проекта — к созданию термометра нового типа с использованием собственных функций.

Функция состоит из последовательности инструкций, которую можно вызывать из любой точки в скетче. Несмотря на наличие в языке Arduino большого количества встроенных функций, не всегда удается найти функцию, отвечающую конкретным потребностям. А в некоторых случаях возникает необходимость в многократном использовании последовательности инструкций в разных частях скетча, что влечет неэффективное расходование памяти. В обоих случаях возникает желание иметь функцию, выполняющую необходимые действия. Должен сообщить, что такую функцию не нужно искать — вы можете написать ее самостоятельно.

Проект № 9: Функция для повторного выполнения действий

Итак, рассмотрим простую функцию, повторно выполняющую некоторые действия по мере необходимости. Например, следующая функция дважды включает (❶ и ❸) и выключает (❷ и ❹) встроенный светодиод.

```
  void blinkLED()
  {
❶   digitalWrite(13, HIGH);
    delay(1000);
❷   digitalWrite(13, LOW);
    delay(1000);
❸   digitalWrite(13, HIGH);
    delay(1000);
❹   digitalWrite(13, LOW);
    delay(1000);
}
```

А теперь используем эту функцию в законченном скетче, который можно загрузить в плату Arduino:

```
  // Проект 9 - Создание функции для повторного выполнения действий

  #define LED 13
  #define del 200

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

  void blinkLED()
  {
    digitalWrite(LED, HIGH);
    delay(del);
    digitalWrite(LED, LOW);
    delay(del);
    digitalWrite(LED, HIGH);
    delay(del);
    digitalWrite(LED, LOW);
    delay(del);
  }

  void loop()
  {
❶   blinkLED();
    delay(1000);
}
```

Когда внутри функции `void loop()` вызывается функция `blinkLED()` (❶), Arduino выполняет команды внутри `void blinkLED()`. Иными словами, вы создали собственную функцию и использовали ее по мере необходимости.

Проект № 10: Функция, изменяющая число миганий светодиода

Только что созданная нами функция довольно ограниченна. Представьте, что вам понадобилось изменить число миганий или задержку. Это легко реализовать — напишем функцию, позволяющую изменять значения, как показано далее:

```
void blinkLED(int cycles, int del)
{
  for ( int z = 0 ; z < cycles ; z++ )
  {
    digitalWrite(LED, HIGH);
    delay(del);
    digitalWrite(LED, LOW);
    delay(del);
  }
}
```

Наша новая функция `void blinkLED()` принимает два целочисленных значения: `cycles` (определяет, сколько раз должен мигнуть светодиод) и `del` (время задержки между включением и выключением светодиода). То есть если понадобится мигнуть светодиодом 12 раз с задержкой в 100 миллисекунд, вы выполните вызов `blinkLED(12, 100)`. Введите следующий скетч в IDE и поэкспериментируйте с функцией:

```
// Проект 10 – Функция, изменяющая количество миганий

#define LED 13

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

void blinkLED(int cycles, int del)
{
  for ( int z = 0 ; z < cycles ; z++ )
  {
    digitalWrite(LED, HIGH);
    delay(del);
    digitalWrite(LED, LOW);
    delay(del);
  }
```

```
}

void loop()
{
❶  blinkLED(12, 100);
  delay(1000);
}
```

В строке (❶) можно видеть, что нашей функции `blinkLED()` передаются значения `12` и `100` (число миганий и величина задержки соответственно), при этом переменная `cycles` получает значение `12`, а `del` — значение `100`. Как результат, светодиод мигнет 12 раз с задержкой 100 миллисекунд между вспышками.

Функция, возвращающая значения

Мы можем создавать функции, не только принимающие значения в виде параметров (подобно функции `void blinkLED()` в проекте 10), но и возвращающие значения, как это делает функция `analogRead()`, возвращающая число в диапазоне от 0 до 1023, пропорциональное напряжению на аналоговом входе. Эта функция демонстрировалась в проекте 8. Ключевое слово `void` перед началом функции означает, что она ничего не возвращает, то есть возвращает значение `void` (ничего). Давайте напишем какую-нибудь полезную функцию, возвращающую значение.

Взгляните на следующую функцию, которая преобразует градусы Цельсия в градусы Фаренгейта:

```
float convertTemp(float celsius)
{
  float fahrenheit = 0;
  fahrenheit = (1.8 * celsius) + 32;
  return fahrenheit;
}
```

В первой строке определяется имя функции (`convertTemp`), она возвращает значение (типа `float`) и принимает любые значения, которые вы хотите ей передать (`float celsius`). Чтобы получить значение функции, его нужно присвоить переменной. Например, если понадобится преобразовать 40 градусов Цельсия в градусы Фаренгейта и сохранить результат в переменной типа `float` с именем `tempf`, это можно сделать так:

```
tempf = convertTemp(40);
```

Эта инструкция поместит число `40` в переменную `celsius` внутри `convertTemp`, вызовет функцию для выполнения вычислений `fahrenheit = (1.8 * celsius) + 32` и сохранит в переменной `tempf` результат, возвращаемый из `convertTemp` строкой `return fahrenheit`.

Проект № 11: Быстродействующий термометр, сообщающий температуру миганием светодиода

Теперь, когда вы знаете, как писать свои функции, реализуем следующий проект — быстродействующий термометр на основе датчика температуры TMP36 из главы 4 и светодиода на плате Arduino. Если температура ниже +20 °C, светодиод должен мигнуть два раза и сделать паузу; если температура находится в диапазоне от +20 °C до +26 °C, светодиод должен мигнуть четыре раза и сделать паузу; если температура выше +26 °C, светодиод должен мигнуть шесть раз.

Мы сделаем наш скетч более модульным, разбив его на отдельные функции, благодаря чему его проще будет понять, а функции будут использоваться многократно. Наш термометр решает две основные задачи: измеряет и классифицирует температуру и мигает светодиодом определенное число раз (в зависимости от температуры).

Оборудование

Нам потребуется минимальный набор оборудования:

- Один температурный датчик TMP36.
- Одна макетная плата.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Схема очень проста, она показана на рис. 5.1.

Скетч

Для скетча требуется написать две функции. Первая будет считывать значение с датчика TMP36, преобразовывать его в градусы Цельсия и возвращать число 2, 4 или 6, соответствующее числу миганий светодиода. Для решения этой задачи примем за основу скетч из проекта 8.

Для решения второй задачи возьмем функцию `blinkLed()` из проекта 9. Наша функция `void loop` будет вызывать упомянутые функции и выполнять паузу в 2 секунды перед перезапуском.

ПРИМЕЧАНИЕ

Не забудьте сохранить измененные скетчи проектов в файлах с другими именами, чтобы случайно не потерять плоды своих трудов!

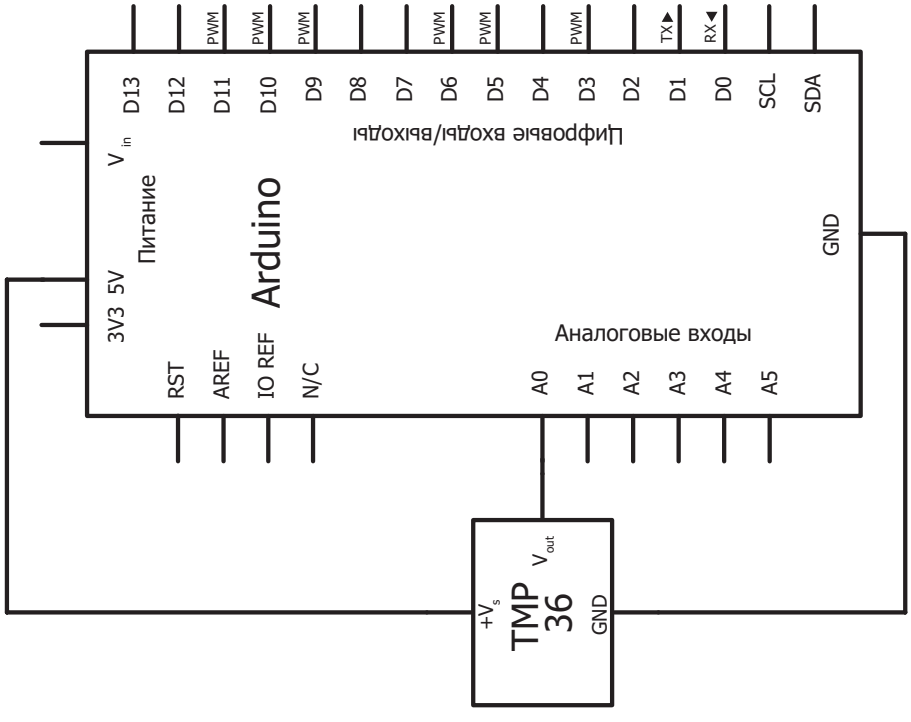


Рис. 5.1. Принципиальная электрическая схема для проекта 11

Введите следующий скетч в IDE:

```
// Проект 11 - Быстродействующий термометр,
//             сообщающий температуру миганием

#define LED 13

int blinks = 0;

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

int checkTemp()
{
  float voltage = 0;
  float celsius = 0;
  float hotTemp = 26;
  float coldTemp = 20;
  float sensor = 0;
  int result;
```

```
  // прочитать значение с датчика и преобразовать его в градусы Цельсия
  sensor = analogRead(0);
  voltage = (sensor*5000)/1024; // преобразовать в милливольты
  voltage = voltage-500;        // учесть смещение
  celsius = voltage/10;         // преобразовать милливольты в градусы

  // выполнить действия для разных диапазонов температур
  if (celsius < coldTemp)
  {
    result = 2;
  }
  else if (celsius >= coldTemp && celsius <= hotTemp)
  {
    result = 4;
  }
  else
  {
    result = 6; // (celsius > hotTemp)
  }
  return result;
}

void blinkLED(int cycles, int del)
{
  for ( int z = 0 ; z < cycles ; z++ )
  {
    digitalWrite(LED, HIGH);
    delay(del);
    digitalWrite(LED, LOW);
    delay(del);
  }
}

void loop()
{
  blinks = checkTemp();
  blinkLED(blinks, 500);
  delay(2000);
}
```

Поскольку основные задачи решаются нашими собственными функциями, внутри `void loop()` остается только вызвать их и выполнить паузу. Функция `checkTemp()` возвращает целое число, которое сохраняется в переменной `blinks`, затем `blinkLED()` мигает светодиодом `blinks` раз с задержкой 500 миллисекунд. Перед повторением скетч приостанавливается на 2 секунды.

Загрузите скетч и понаблюдайте за поведением светодиода. (Не разбирайте пока собранную схему, она пригодится нам в следующих примерах.)

Отображение данных из Arduino в окне монитора последовательного порта

До сих пор мы загружали скетчи в Arduino и для вывода информации (например, температуры или направления движения через мост) использовали светодиоды. Мигание светодиодами упрощает организацию обратной связи с Arduino, но вспышки света не слишком информативны. В этом разделе вы узнаете, как использовать проводное соединение Arduino с компьютером и окно монитора порта в IDE для вывода данных с платы и отправки данных в плату с клавиатуры.

Монитор последовательного порта

Чтобы открыть монитор порта, запустите IDE и щелкните на пиктограмме Serial Monitor (Монитор порта) на панели инструментов, изображенной на рис. 5.2. В ответ на это откроется окно монитора порта, как показано на рис. 5.3.

Рис. 5.2. Пиктограмма Serial Monitor (Монитор порта) на панели инструментов

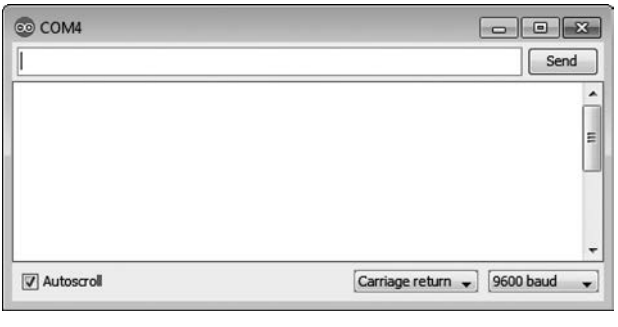


Рис. 5.3. Окно монитора последовательного порта

Как показано на рис. 5.3, вверху в окне монитора порта имеется однострочное поле ввода с кнопкой Send (Отправить) и область вывода под ним, где отображаются данные, полученные с платы Arduino. Если поставить флажок Autoscroll (Автопрокрутка), в области вывода будут отображаться самые свежие данные, а при ее переполнении старые данные будут «уходить» за границы области. Сняв флажок Autoscroll (Автопрокрутка), вы сможете вручную прокручивать область вывода во время исследования данных.

Инициализация обмена с монитором порта

Прежде чем использовать монитор порта, его нужно активировать, добавив следующий вызов в функцию `void setup()`:

```
Serial.begin(9600);
```

Число `9600` — это скорость передачи данных между компьютером и платой Arduino, измеряемая в *бодах*. Это число должно соответствовать значению, выбранному в раскрывающемся списке справа внизу в окне монитора порта (см. рис. 5.3).

Отправка текста в монитор порта

Послать текст для отображения в области вывода в мониторе порта можно вызовом функции `Serial.print`:

```
Serial.print("Arduino for Everyone!");
```

Она отправит в монитор порта текст, заключенный в кавычки.

Для вывода текста и принудительного перехода на новую строку воспользуйтесь функцией `Serial.println`:

```
Serial.println("Arduino for Everyone!");
```

Вывод содержимого переменных

Аналогично осуществляется вывод в монитор порта содержимого переменных. Например, далее показано, как вывести содержимое переменной `results`:

```
Serial.println(results);
```

Значения переменных типа `float` по умолчанию выводятся с двумя знаками после десятичной точки. Количество знаков можно изменить, передав число от 0 до 6 во втором параметре после имени переменной. Например, чтобы вывести значение вещественной переменной `results` с четырьмя десятичными знаками, выполните следующий вызов:

```
Serial.print(results,4);
```

Проект № 12: Отображение температуры в мониторе порта

Используя оборудование для проекта 8, реализуем вывод температуры в градусах Цельсия и Фаренгейта в окно монитора порта. Для этого мы создадим одну функ-

цию, определяющую температуру, и еще одну — отображающую температуру в окне монитора порта.

Введите следующий код в IDE:

```
// Проект 12 - Отображение температуры в мониторе порта

float celsius = 0;
float fahrenheit = 0;

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

❶ void findTemps()
{
  float voltage = 0;
  float sensor = 0;

  // прочитать величину с датчика и преобразовать
  // ее в градусы Цельсия и Фаренгейта
  sensor = analogRead(0);
  voltage = (sensor*5000)/1024; // преобразовать в милливольты
  voltage = voltage - 500;      // учесть смещение
  celsius = voltage / 10;       // преобразовать милливольты
                                // в градусы Цельсия
  fahrenheit = (1.8 * celsius) + 32; // преобразовать градусы Цельсия
                                     // в градусы Фаренгейта
}

❷ void displayTemps()
{
  Serial.print("Temperature is ");
  Serial.print(celsius, 2);
  Serial.print(" deg. C / ");
  Serial.print(fahrenheit, 2);
  Serial.println(" deg. F");
  // здесь используется .println, чтобы вывод следующего
  // замера начинался с новой строки
}

void loop()
{
  findTemps();
  displayTemps();
  delay(1000);
}
```

Этот скетч содержит много разных действий, но мы написали две функции, `findTemps()` (❶) и `displayTemps()` (❷), чтобы упростить его. Эти функции вызываются внутри `void loop()`, которая получилась очень простой. Этот пример наглядно демонстрирует одну из причин создания собственных функций: чтобы

сделать скетчи более простыми и понятными и чтобы сделать код более модульным и пригодным для многократного использования.

Загрузив скетч, подождите несколько секунд и затем откройте окно монитора последовательного порта. В области вывода появится список результатов измерения температуры, как показано на рис. 5.4.

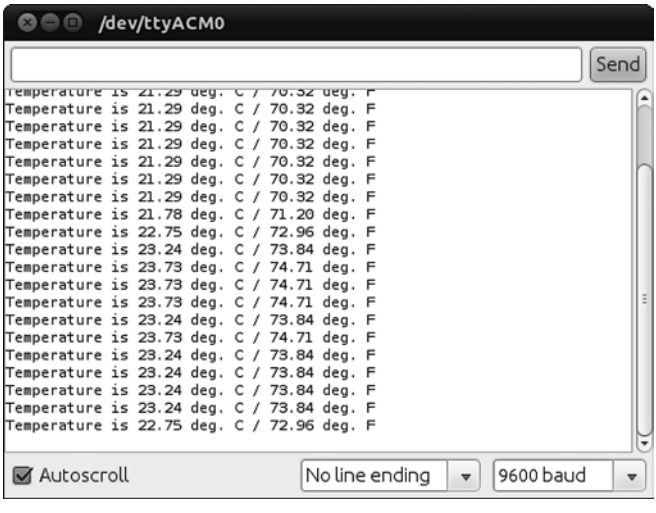


Рис. 5.4. Результаты работы проекта 12

Отладка при помощи монитора порта

Монитор порта можно с успехом использовать для *отладки* (поиска и исправления ошибок) скетчей. Если добавить в скетч несколько инструкций `Serial.println();` для вывода кратких примечаний об их местоположении в скетче, можно будет увидеть, когда Arduino выполняет каждую из добавленных инструкций. Например, следующую строку:

```
Serial.println("now in findTemps()");
```

можно вставить в функцию `findTemps()`, это позволит увидеть, когда Arduino выполняет данную конкретную функцию.

Принятие решений при помощи инструкций while

Инструкции `while()` используются в скетчах для многократного выполнения фрагментов кода, пока (*while*) заданное условие остается истинным. Условие в инструкции `while()` всегда проверяется *перед* выполнением фрагмента кода. На-

пример, `while ( temperature > 30 )` сначала убеждается, что значение `temperature` больше `30`. В условных выражениях внутри круглых скобок можно использовать любые операторы сравнения.

В следующем листинге Arduino сначала отсчитает 10 секунд, а затем продолжит выполнение оставшейся части программы:

```
int a = 0; // целое число
while ( a < 10 )
{
  a = a + 1;
  delay(1000);
}
```

Этот скетч сначала присваивает переменной `a` значение `0`. Затем прибавляет `1` к значению `a` (которое первоначально равно `0`), ждет 1 секунду (`delay(1000)`) и после этого повторяет процесс до тех пор, пока значение `a` не станет больше или равно `10` (`while ( a < 10 )`). Как только переменная `a` получит значение `10`, операция сравнения в инструкции `while` вернет `false`, и Arduino продолжит выполнение скетча, начиная с инструкции, следующей сразу за фигурной скобкой, закрывающей цикл `while`.

do-while

В отличие от `while`, конструкция `do-while()` выполняет проверку *после* выполнения фрагмента кода внутри инструкции `do-while`. Например:

```
int a = 0; // целое число
do
{
  delay(1000);
  a = a + 1;
} while ( a < 100 );
```

В данном случае код в фигурных скобках будет выполняться *перед* проверкой условия (`while ( a < 100 )`). То есть даже если условие не выполняется, цикл будет выполнен не менее одного раза. Вам самим придется решить, какую инструкцию выбрать — `while` или `do-while` — в зависимости от требований конкретного проекта.

Передача данных из монитора порта в Arduino

Чтобы послать данные из монитора порта в Arduino, нужно, чтобы Arduino извлекала полученные данные из *буфера последовательного порта* — составной части Arduino, которая осуществляет прием данных извне, полученных через контакты последовательного порта (цифровые входы/выходы 0 и 1), связанные через ин-

терфейс USB и кабель с компьютером. Буфер последовательного порта хранит входящие данные, введенные в поле ввода в окне монитора порта и отправленные пользователем.

Проект № 13: Умножение числа на два

Для исследования процесса передачи и приема данных посредством монитора порта рассмотрим следующий скетч. Он принимает одноразрядное число от пользователя, умножает его на 2 и отправляет результат в монитор порта.

```
// Проект 13 - Умножение числа на два

int number;

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

void loop()
{
  number = 0;     // обнулить переменную, подготовив
                  // ее к приему нового числа
  Serial.flush(); // очистить буфер порта от "мусора" перед ожиданием
❶  while (Serial.available() == 0)
  {
    // ничего не делать, пока что-то не появится в буфере порта
  }
❷  while (Serial.available() > 0)
  {
    number = Serial.read() - '0';
    // прочитать цифру из буфера порта,
    // вычесть из ASCII-кода цифры
    // код символа '0', чтобы получить число
  }

  // Вывести число!
  Serial.print("You entered: ");
  Serial.println(number);
  Serial.print(number);
  Serial.print(" multiplied by two is ");
  number = number * 2;
  Serial.println(number);
}
```

Функция `Serial.available()` в первой инструкции `while` (❶) возвращает 0, если пользователь ничего не ввел в монитор порта. Иными словами, она сообщает Arduino: «Ничего не делай, пока пользователь не введет что-нибудь». Следующая инструкция `while` (❷) извлекает цифру из буфера порта и преобразует код символа,

обозначающего цифру, в фактическое целое число. Затем Arduino выводит число, полученное из буфера порта, и результат его умножения на два.

Функция `Serial.flush()` в начале скетча очищает буфер порта от любых данных, которые могли там находиться, готовя скетч к приему последующих данных. На рис. 5.5 изображено окно монитора порта после запуска скетча.

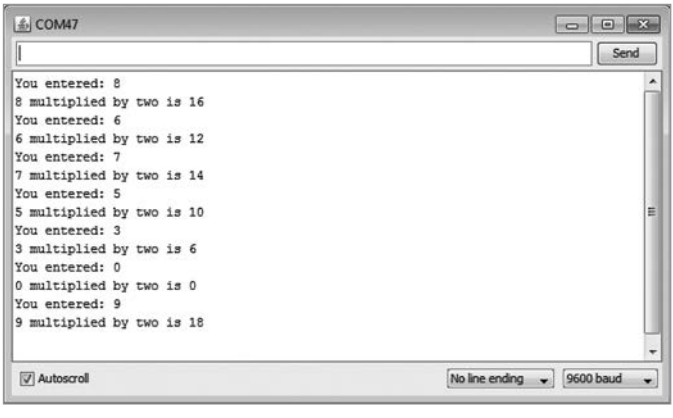


Рис. 5.5. Пример ввода и вывода в проекте 13

Несмотря на то что теперь вы можете вводить данные в мониторе порта для обработки платой Arduino, использование переменных типа `int` сильно ограничивает диапазон допустимых чисел. Чтобы расширить этот диапазон, следует использовать переменные типа `long`, о которых рассказывается далее.

Переменные типа long

Чтобы использовать монитор порта для ввода чисел, состоящих более чем из одной цифры, нужно немного расширить предыдущий скетч, как будет показано ниже. Однако когда приходится работать с большими числами, диапазон значений, представляемый типом `int`, может оказаться слишком узким, потому что переменные этого типа не способны хранить значения больше 32 767. К счастью, это ограничение легко преодолеть при помощи переменных типа `long`. Переменные типа `long` могут хранить целые числа в диапазоне от –2 147 483 648 до 2 147 483 647, что намного шире диапазона переменных типа `int` (от –32 768 до 32 767).

Проект № 14: Использование переменных типа long

В этом проекте мы используем монитор порта для передачи значений типа `long` и чисел, состоящих из нескольких цифр. Данный скетч принимает наше число, умножает его на два и выводит результат в монитор порта.

```
// Проект 14 - Использование переменных типа long

long number = 0;
long a = 0;

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

void loop()
{
  number = 0;     // обнулить переменную, подготовив
                  // ее к приему нового числа
  Serial.flush(); // очистить буфер порта от "мусора" перед ожиданием
  while (Serial.available() == 0)
  {
    // ничего не делать, пока что-то не появится в буфере порта,
    // когда что-то появится в буфере, Serial.available вернет
    // количество символов, ожидающих обработки в буфере
  }

  // как минимум один символ имеется в буфере,
  // начать вычисления
  while (Serial.available() > 0)
  {
    // сдвинуть предыдущие цифры на разряд влево;
    // иными словами, 1 превратится в 10, если в буфере имеются данные
    number = number * 10;

    // прочитать следующую цифру из буфера и вычесть из нее
    // код символа '0', чтобы превратить в фактическое целое число
    a = Serial.read() - '0';

    // прибавить это значение к накапливаемому значению
    number = number + a;

    // выполнить короткую задержку, чтобы дать возможность
    // следующим цифрам достичь буфера
    delay(5);
  }
  Serial.print("You entered: ");
  Serial.println(number);
  Serial.print(number);
  Serial.print(" multiplied by two is ");
  number = number * 2;
  Serial.println(number);
}
```

В этом примере два цикла `while` позволяют плате Arduino принять несколько цифр из монитора порта. Когда в буфер поступает первая цифра (самый левый разряд во введенном числе), она преобразуется в число и прибавляется к переменной `number`.

Если пользователь ввел всего одну цифру, скетч завершит цикл и продолжит выполнение дальше. Если была введена еще одна цифра (например, `2` в числе `42`), значение `number` будет умножено на 10, чтобы сдвинуть первую цифру на один разряд влево, и к значению `number` добавится новая цифра. Этот цикл продолжается, пока в `number` не будет добавлена самая правая цифра во введенном числе. Не забудьте выбрать пункт No line ending (Нет конца строки) в раскрывающемся списке (рис. 5.6) в нижней части окна монитора порта.

На рис. 5.6 изображено окно монитора порта после запуска скетча.

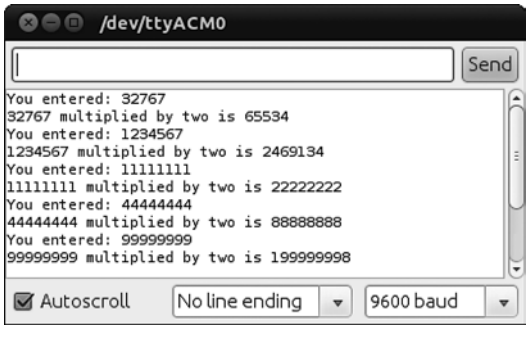


Рис. 5.6. Пример ввода и вывода в проекте 14

Забегая вперед

Возможно, эта глава показалась вам немного суховатой, однако умение писать собственные функции — очень важный навык, который помогает упростить скетчи и сэкономить время и силы. Достойное применение полученным здесь знаниям мы найдем в следующей главе.

6 Числа, переменные и арифметика

В этой главе вы:

- ✓ научитесь генерировать случайные числа;
- ✓ создадите электронный игровой кубик;
- ✓ познакомитесь с двоичной системой счисления;
- ✓ освоите работу с микросхемой сдвигового регистра, позволяющей увеличить количество цифровых входов;
- ✓ проверите свои знания двоичной арифметики в небольшой викторине;
- ✓ узнаете о массивах переменных;
- ✓ научитесь выводить цифры на семисегментный индикатор;
- ✓ познакомитесь с математической функцией деления по модулю;
- ✓ создадите цифровой термометр;
- ✓ освоите приемы поразрядной арифметики;
- ✓ создадите статические и динамические изображения на светодиодной матрице.

В этой главе вас ждет знакомство с новыми функциями, которые помогут создавать новые виды проектов, такими как генерация случайных чисел, новые математические функции и хранилища переменных в виде упорядоченных списков, которые называют *массивами*. Кроме того, вы узнаете, как использовать цифровые и матричные светодиодные модули для отображения данных и простых образов. Все полученные знания и умения мы используем на практике и создадим игру, цифровой термометр и другие устройства.

Случайные числа

Возможность генерировать случайные числа программным способом очень пригодится в проектах игр и для создания разных эффектов. Например, случайные числа находят применение в реализации игрового кубика или лотереи на основе Arduino для создания световых эффектов при помощи светодиодов или для создания звуковых и световых эффектов в игре-викторине. К сожалению, сама плата Arduino не может воспроизводить по-настоящему случайные числа. Но вы можете помочь ей, выбирая произвольное начальное число, которое затем будет использоваться в вычислениях других случайных чисел.

Использование электрического поля для генерации случайных чисел

Самый простой способ сгенерировать случайное число в Arduino — написать программу, которая будет читать напряжение со свободного (ни к чему не подключенного) аналогового входа (например, с нулевого) в функции `void setup()`:

```
randomSeed(analogRead(0));
```

Даже если к аналоговому входу Arduino ничего не подключено, на нем все равно присутствует некоторое измеримое статическое напряжение, создаваемое окружением. Величина этого напряжения изменяется хаотически. Мы можем сохранить результат измерения этого наведенного электрического потенциала в целочисленной переменной и использовать его в качестве начального значения для вычисления последовательности псевдослучайных чисел при помощи функции `random(lower, upper)`. Параметры `lower` и `upper` определяют нижнюю и верхнюю границы диапазона случайных чисел. Например, чтобы сгенерировать случайное число между 100 и 1000, используем следующий код:

```
int a = 0;
a = random(100, 1001);
```

Здесь в качестве верхней границы указано число 1001, а не 1000, потому что значение верхней границы не входит в диапазон.

С другой стороны, чтобы сгенерировать случайное число в диапазоне от 0 до некоторого значения, достаточно передать только верхнюю границу. Следующая инструкция сгенерирует случайное число в диапазоне от 0 до 6 (включительно):

```
a = random(7);
```

Пример скетча в листинге 6.1 генерирует случайные числа между 0 и 1000, а также между 10 и 50:

Листинг 6.1. Генератор случайных чисел

```
// Листинг 6.1
int r = 0;
void setup()
{
  randomSeed(analogRead(0));
  Serial.begin(9600);
}

void loop()
{
  Serial.print("Random number between zero and 1000 is: ");
  r = random(0, 1001);
  Serial.println(r);
  Serial.print("Random number between ten and fifty is: ");
  r = random(10, 51);
  Serial.println(r);
  delay(1000);
}
```

На рис. 6.1 показаны результаты, отображаемые в окне монитора последовательного порта.

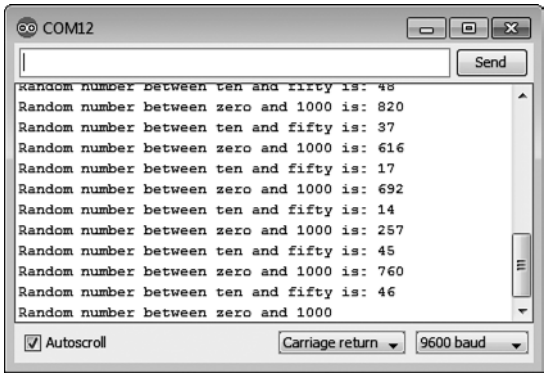


Рис. 6.1. Вывод скетча из листинга 6.1

Теперь, когда вы знаете, как генерировать случайные числа, применим это знание для создания электронного кубика.

Проект № 15: Электронный кубик

Наша цель: случайным образом выбрать и зажечь один из шести светодиодов, чтобы сымитировать бросок игрового кубика. Мы будем генерировать случайное число от 1 до 6 и включать соответствующий светодиод, чтобы показать результат.

Для этого необходимо создать функцию, случайно выбирающую один из шести светодиодов и оставляющую его включенным в течение некоторого времени. После включения или сброса платы Arduino со скетчем она должна начать быстро зажигать случайно выбранные светодиоды, создавая эффект паузы-размышления, оставлять их включенным в течение некоторого времени, затем постепенно уменьшить скорость выбора и, наконец, остановиться на одном из светодиодов. Выбранный светодиод останется включенным, пока плата не будет сброшена или выключена.

Оборудование

Для реализации кубика нам понадобится следующее оборудование:

- Шесть светодиодов любого цвета (LED1—LED6).
- Один резистор с номиналом 560 Ом (R1).
- Несколько отрезков провода разной длины.
- Одна макетная плата среднего размера.
- Плата Arduino и кабель USB.

Схема

Так как в каждый момент времени включаться будет только один светодиод, мы соединим катоды всех светодиодов с выводом GND через единственный ограничивающий резистор. Принципиальная схема кубика показана на рис. 6.2.

Скетч

Далее приводится исходный код скетча, реализующего работу кубика:

```
// Проект 15 - Электронный кубик

void setup()
{
  randomSeed(analogRead(0));      // начальное число для
                                  // генератора случайных чисел
  for ( int z = 1 ; z < 7 ; z++ ) // настроить контакты 1-6
  {                               // на работу в режиме выходов
    pinMode(z, OUTPUT);
  }
}

void randomLED(int del)
{
  int r;
  r = random(1, 7);      // получить случайное число от 1 до 6
```

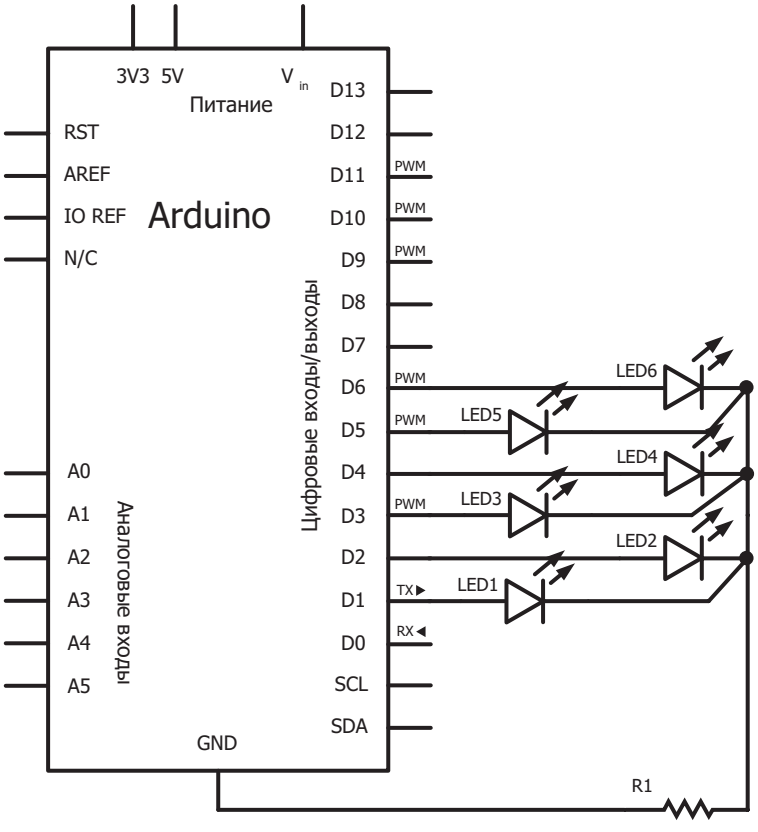


Рис. 6.2. Принципиальная схема проекта 15

```
    digitalWrite(r, HIGH); // включить соответствующий светодиод
    if (del > 0)
    {
❶       delay(del);         // оставить включенным в течение
    }                       // указанного времени
❷   else if (del == 0)
    {
      do                    // указана нулевая задержка, оставить
      {}                    // светодиод включенным "навсегда"
❸      while (1);
    }
    digitalWrite(r, LOW);  // выключить светодиод
  }

  void loop()
  {
    int a;
    // создать эффект выбора светодиода
    for ( a = 0 ; a < 100 ; a++ )
```

```
    {
      randomLED(50);
    }
    // постепенно замедлить эффект
❹   for ( a = 1 ; a <= 10 ; a++ )
    {
      randomLED(a * 100);
    }
    // и остановиться на случайно выбранном светодиоде
    randomLED(0);
  }
```

Здесь, в функции `void setup()`, для настройки контактов на работу в режиме цифровых выходов использован цикл. Функция `randomLED()` принимает целое число, используемое в вызове функции `delay()` (❶), чтобы оставить светодиод включенным на некоторое время. Если функция получит значение задержки `0` (❷), она оставит светодиод включенным «навсегда», войдя в бесконечный цикл (❸)

```
do {} while (1);
```

потому что 1 — всегда 1.

Чтобы «бросить кубик», необходимо выполнить сброс платы Arduino. Для создания эффекта размышлений с постепенным замедлением перед остановкой на окончательно выбранном светодиоде сначала 100 раз включается случайно выбранный светодиод с задержкой на 50 миллисекунд (❹). Затем скорость перебора уменьшается за счет увеличения периода, когда светодиод остается включенным, со 100 до 1000 миллисекунд и с шагом 100 миллисекунд. Цель этого решения — сымитировать замедление вращения кубика перед полной остановкой, после чего Arduino отображает результат выпадения кубика, включая один из светодиодов в последней строке:

```
randomLED(0);
```

Доработка скетча

Этот проект можно видоизменять «в разных направлениях». Например, можно добавить еще шесть светодиодов, чтобы сымитировать бросок сразу двух кубиков, или отображать результат при помощи единственного встроенного светодиода, чтобы он мигал выпавшее число раз. Поэкспериментируйте, используя свое воображение и полученные знания!

Краткое введение в двоичную систему счисления

Большинство детей учатся счету с использованием десятичной системы счисления, но компьютеры (и плата Arduino) производят вычисления с применением двоичной

системы. *Двоичные числа* состоят только из нулей и единиц — например, 10101010. Каждая цифра в двоичном числе, справа налево, представляет число 2 в степени, соответствующей порядковому (начиная с нуля) номеру разряда (счет разрядов ведется справа налево). Произведения в каждом разряде складываются, и сумма определяет величину значения числа.

Например, рассмотрим двоичное число 11111111, представленное в табл. 6.1. Чтобы преобразовать двоичное число 11111111 в десятичную систему счисления, сложим степени двойки во всех разрядах, указанные в последней строке табл. 6.1:

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$$

В результате получится десятичное число 255. Двоичное число с восемью разрядами (или *битами*) представляет 1 *байт* данных; 1 байт может иметь числовое значение в диапазоне от 0 до 255. Самый левый бит называют *старшим битом* (Most Significant Bit, MSB), а самый правый — *младшим битом* (Least Significant Bit, LSB).

Двоичные числа отлично подходят для хранения некоторых типов данных, таких как признаки включено/выключено для светодиодов, да/нет для параметров настройки и состояния цифровых выходов. Двоичные числа являются строительными блоками всех типов данных в компьютерах.

Таблица 6.1. Пример преобразования двоичного числа в десятичное

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	1	1	1	1	1	1	1	Двоичное
128	64	32	16	8	4	2	1	Десятичное

Переменные типа byte

Переменные типа `byte` — один из способов хранения двоичных чисел. Создать переменную типа `byte` можно следующим образом:

```
byte outputs = B11111111;
```

Буква `B` в начале числа сообщает, что число, следующее за ней, должно интерпретироваться как двоичное (в данном случае 11111111), а не десятичное. Это демонстрирует листинг 6.2.

Листинг 6.2. Демонстрация двоичного числа

```
// Листинг 6.2

  byte a;
```

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

void loop()
{
  for ( int count = 0 ; count < 256 ; count++ )
  {
    a = count;
    Serial.print("Base-10 = ");
❶    Serial.print(a, DEC);
    Serial.print(" Binary = ");
❷    Serial.println(a, BIN);
    delay(1000);
  }
}
```

Скетч выводит значение переменной типа `byte` вызовом функции `Serial.print()` в виде десятичного числа при использовании параметра `DEC` (❶) и в виде двоичного при использовании параметра `BIN` (❷). После загрузки скетча в окне монитора порта должен появиться вывод, как показано на рис. 6.3.

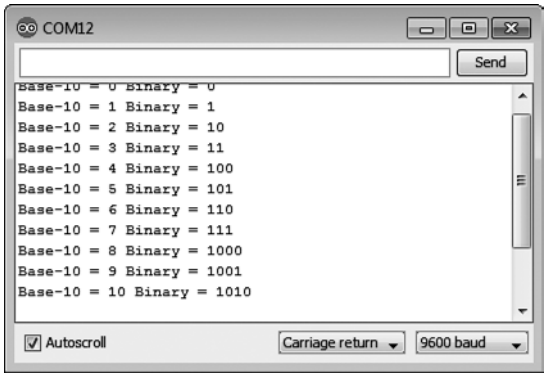


Рис. 6.3. Вывод скетча из листинга 6.2

Увеличение числа цифровых выходов с применением сдвигового регистра

Плата Arduino Uno имеет 13 контактов, которые можно использовать в роли цифровых выходов. Но иногда такого количества оказывается недостаточно. С помощью *сдвигового регистра* можно добавить дополнительные цифровые выходы и получить достаточное количество цифровых выходов для проекта. Сдвиговый регистр — это микросхема с восемью цифровыми выходами, которыми можно управ-

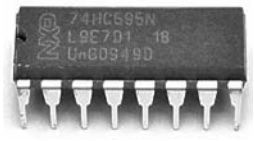


Рис. 6.4. Сдвиговый регистр 74HC595

лять, посылая в микросхему байты данных. В наших проектах мы используем сдвиговый регистр 74HC595, изображенный на рис. 6.4.

Сдвиговый регистр 74HC595 имеет восемь цифровых выходов, действующих подобно цифровым выходам на плате Arduino. Для управления сдвиговым регистром требуется задействовать три цифровых выхода платы, в итоге чистый выигрыш составит пять дополнительных цифровых выходов.

Принцип действия сдвигового регистра прост: мы посылаем в него 1 байт данных (8 бит), а он включает или выключает восемь своих выходов согласно состояниям разрядов в этом байте. Биты, составляющие байт данных, соответствуют цифровым выходам регистра в порядке от старшего к младшему. То есть самому левому биту данных соответствует выход 7 сдвигового регистра, а самому правому — выход 0. Например, если послать в сдвиговый регистр байт `B10000110`, он включит напряжение на выходах 7, 2 и 1 и выключит — на выходах 0 и 3–6. Эти уровни напряжения на выходах останутся, пока регистр не получит другой байт данных или пока с него не будет снято напряжение питания.

Одновременно к одним и тем же трем цифровым выходам на плате можно подключить несколько сдвиговых регистров и с каждым получить восемь дополнительных цифровых выходов; это делает сдвиговые регистры очень удобным инструментом, когда требуется управлять большим количеством светодиодов. Давайте теперь реализуем такое управление на практике, создав «дисплей» для вывода двоичных чисел.

Проект № 16: Светодиодный индикатор для двоичных чисел

В этом проекте мы реализуем отображение двоичных чисел от 0 до 255 с помощью восьми светодиодов. Скетч будет перебирать в цикле числа от 0 до 255 и посылать каждое из них в сдвиговый регистр, управляющий светодиодами для отображения двоичных эквивалентов.

Оборудование

Ниже перечислено необходимое оборудование:

- Один сдвиговый регистр 74HC595.
- Восемь светодиодов (LED1—LED8).
- Восемь резисторов с номиналом 560 Ом (R1—R8).
- Одна макетная плата.

- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Подключение микросхемы 74HC595

На рис. 6.5 показано, как изображается сдвиговый регистр 74HC595 на принципиальных схемах.

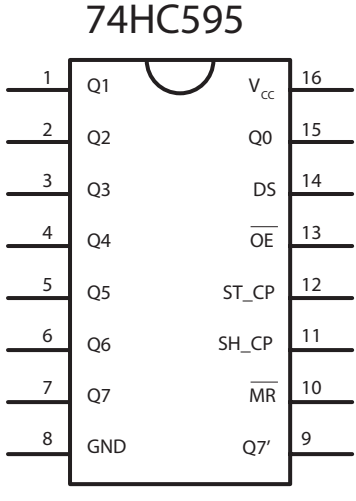


Рис. 6.5. Изображение сдвигового регистра 74HC595 на принципиальных схемах

Микросхема сдвигового регистра имеет 16 выводов:

- Выводы 15 и с 1-го по 7-й — это восемь цифровых выходов, которыми мы будем управлять (отмечены как Q0—Q7 соответственно).
- Вывод Q7 соответствует первому биту, посылаемому в сдвиговый регистр, а вывод Q0 — последнему.
- Вывод 8 подключается к «земле» (GND).
- Вывод 9 — «выходные данные», используется для передачи данных другому сдвиговому регистру, если таковой имеется.
- Вывод 10 всегда должен быть подключен к шине питания 5 В (например, к контакту 5 V на плате Arduino).
- Выводы 11 и 12 называются *входом для тактовых импульсов* и *защелкой*.
- Вывод 13 используется для переключения выходов между высокоомным и рабочим состоянием и обычно подключается к «земле» (GND).

- Вывод 14 — вход для битов данных, последовательно посылаемых платой Arduino.
- Вывод 16 — питание 5 В (подключается к контакту 5 V на плате Arduino).

Ориентация контактов определяется по полукруглой метке на корпусе сдвигового регистра, она показана слева на рис. 6.4 и находится между выводами 1 и 16.

Выводы нумеруются в направлении против часовой стрелки, как показано на рис. 6.6 с изображением принципиальной схемы светодиодного индикатора для двоичных чисел.

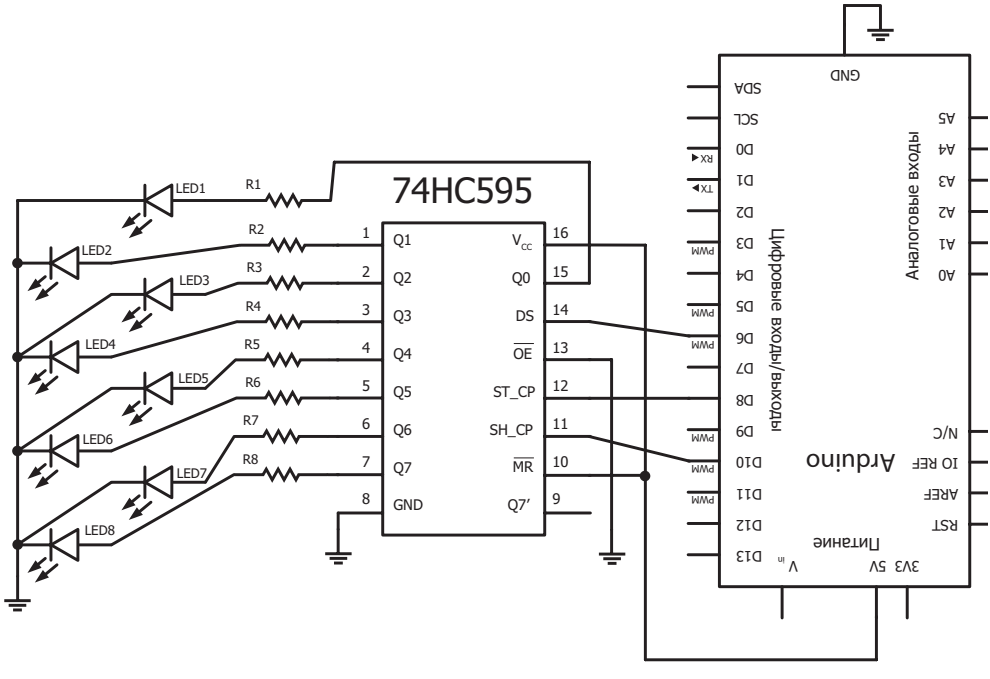


Рис. 6.6. Принципиальная схема для проекта 16

ПРИМЕЧАНИЕ

Закончив этот проект, не разбирайте собранную схему, она еще пригодится в следующем проекте.

Скетч

А теперь скетч:

```
// Проект 16 — Светодиодный индикатор для двоичных чисел

#define DATA 6  // цифровой выход 6 - к выводу 14 микросхемы 74HC595
#define LATCH 8 // цифровой выход 8 - к выводу 12 микросхемы 74HC595
#define CLOCK 10 // цифровой выход 10 - к выводу 11 микросхемы 74HC595
```

```
void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void loop()
{
  int i;
  for ( i = 0; i < 256; i++ )
  {
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, MSBFIRST, i);
    digitalWrite(LATCH, HIGH);
    delay(200);
  }
}
```

Три контакта на плате, к которым подключается сдвиговый регистр, мы настроили в функции `void setup()` на работу в режиме цифровых выходов и добавили в функцию `void loop()` цикл, который последовательно перебирает числа от 0 до 255. Все волшебство заключено в цикле. Когда цикл `for` выводит байт данных (например, 240, или B11110000) в сдвиговый регистр, происходит три события:

- На вывод 12 (защелка) микросхемы подается уровень `LOW` (то есть сигнал низкого уровня выводится на выход 8 платы Arduino). Это подготавливает сдвиговый регистр к подаче на вывод 12 микросхемы уровня `HIGH`, который «защелкнет» данные на ее выходах, как только `shiftOut` завершит свою работу.
- Байт данных (например, B11110000) посылается с цифрового выхода 6 платы Arduino в сдвиговый регистр, при этом функции `shiftOut` сообщается направление интерпретации байта данных. Например, если передать функции `shiftOut` параметр `LSBFIRST`, светодиоды с 1-го по 4-й включатся, а остальные — выключатся. Если передать параметр `MSBFIRST`, включатся светодиоды с 5-го по 8-й, а остальные — выключатся.
- Наконец, на вывод 12 микросхемы подается уровень `HIGH` (5 В). Таким способом сдвиговому регистру сообщается, что передача битов закончена. В этот момент сдвиговый регистр изменит состояния своих выходов в соответствии с полученными данными.

Проект № 17: Игра «Двоичная викторина»

В этом проекте мы создадим игру «Двоичная викторина», используя случайные числа, монитор последовательного порта и схему из проекта 16. Плата Arduino

будет отображать случайное двоичное число при помощи светодиодов, а вам будет предложено ввести это же число в десятичном виде в окне монитора порта. Затем монитор порта сообщит, правильное число вы ввели или нет, и игра отобразит на светодиодах новое число.

Алгоритм

Алгоритм можно поделить на три функции. Функция `displayNumber()` отобразит двоичное число с использованием светодиодов. Функция `getAnswer()` примет число из монитора порта и выведет его обратно. Наконец, функция `checkAnswer()` сравнит введенное число с отображаемым на светодиодах и сообщит, правильно или неправильно пользователь решил задачу.

Скетч

Скетч генерирует случайные числа в диапазоне от 0 до 255, выводит их в двоичном виде с использованием светодиодов, предлагает пользователю ввести это число в десятичном виде и отображает результаты в окне монитора порта. Вы уже знакомы с функциями, используемыми в скетче, поэтому, несмотря на большой объем, код не должен вызывать у вас затруднений при чтении. Исследовать его помогут комментарии в скетче и следующие за ним.

```
  // Проект 17 - Игра двоичной викторины
#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

int number = 0;
int answer = 0;

❶ void setup()
  {
    pinMode(LATCH, OUTPUT);    // подготовить 74HC595
    pinMode(CLOCK, OUTPUT);
    pinMode(DATA, OUTPUT);
    Serial.begin(9600);
    randomSeed(analogRead(0)); // инициализировать генератор
                               // случайных чисел
    displayNumber(0);          // выключить все светодиоды
  }
❷ void displayNumber(byte a)
  {
    // посылает байт для отображения на светодиодах
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, MSBFIRST, a);
    digitalWrite(LATCH, HIGH);
  }
❸ void getAnswer()
```

```
  {
    // принимает ответ игрока
    int z = 0;
    Serial.flush();
    while (Serial.available() == 0)
    {
// ничего не делать, пока что-то не появится в буфере порта
    }
// как минимум один символ имеется в буфере,
// начать вычисления
    while (Serial.available() > 0)
    {
// сдвинуть предыдущие цифры на разряд влево;
// иными словами, 1 превратится в 10, если в буфере имеются данные
      answer = answer * 10;

// прочитать следующую цифру из буфера и вычесть из нее
// код символа '0', чтобы превратить в фактическое целое число
      z = Serial.read() - '0';

// прибавить это значение к накапливаемому значению
      answer = answer + z;

// выполнить короткую задержку, чтобы дать возможность
// следующим цифрам достичь буфера
      delay(5);
    }
    Serial.print("You entered: ");
    Serial.println(answer);
  }

❹ void checkAnswer()
  {
    // проверяет ответ игрока и выводит результаты
    if (answer == number) // Верно!
    {
      Serial.print("Correct! ");
      Serial.print(answer, BIN);
      Serial.print(" equals ");
      Serial.println(number);
      Serial.println();
    }
    else // Неверно
    {
      Serial.print("Incorrect, ");
      Serial.print(number, BIN);
      Serial.print(" equals ");
      Serial.println(number);
      Serial.println();
    }
    answer = 0;
    delay(10000); // дать игроку время прочитать результаты
  }

  void loop()
```

```
{
  number = random(256);
  displayNumber(number);
  Serial.println("What is the binary number in base-10? ");
  getAnswer();
  checkAnswer();
}
```

Давайте разберемся, как действует этот скетч. Функция `void setup()` (❶) настраивает цифровые выходы, к которым подключен сдвиговый регистр, инициализирует обмен с монитором порта и устанавливает начальное число для генератора случайных чисел. Функция `displayNumber()` (❷) принимает байт данных и передает его в сдвиговый регистр, к которому подключены светодиоды для отображения байта в двоичной форме (как в проекте 16). Функция `getAnswer()` (❸) извлекает из монитора порта число, введенное пользователем (как в проекте 14), и выводит его, как показано на рис. 6.7.

Функция `checkAnswer()` (❹) сравнивает число, введенное пользователем и полученное с помощью `getAnswer()`, со случайным числом, сгенерированным прежде в функции `void loop()`. Затем игроку сообщается, правильно или неправильно он ответил на вопрос с выводом числа в двоичной и десятичной форме. Наконец, в основной функции `void loop()` генерируется случайное двоичное число для викторины, затем вызываются функции для его отображения с помощью светодиодов, и после этого принимается и проверяется ответ пользователя.

На рис. 6.7 изображено окно монитора порта в ходе игры.

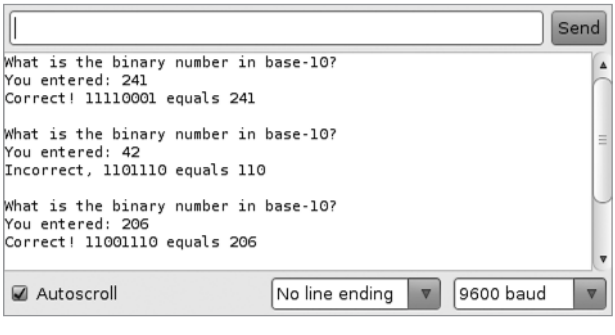


Рис. 6.7. Проект 17 в процессе игры

Массивы

Массив — это множество переменных, или значений, сгруппированных так, что к ним можно обращаться как к единому целому. При наличии большого количества однотипных связанных данных обычно предпочтительнее использовать массивы, это позволяет сохранить их организованными.

Определение массива

Составные части массива называются *элементами*. Например, представьте, что имеется шесть вещественных переменных, хранящих значения температуры за последние шесть часов: вместо того чтобы всем им дать уникальные имена, можно определить массив с именем `temperatures`:

```
float temperatures[6];
```

В определение массива допускается вставлять начальные значения его элементов. В этом случае размер массива можно не указывать. Например:

```
float temperatures[]={11.1, 12.2, 13.3, 14.4, 15.5, 16.6};
```

Обратите внимание, что в этом случае размер массива в квадратных скобках (`[]`) не указан — он определяется по числу элементов в фигурных скобках (`{}`).

Обращение к значениям в массиве

Нумерация элементов в массиве начинается слева и с 0; массив `temperatures[]` имеет элементы с порядковыми номерами от 0 до 5. Обращаться к отдельным элементам массива можно, добавляя порядковый номер элемента в квадратных скобках после имени массива. Например, присвоить первому элементу в массиве `temperatures[]` (имеет текущее значение 16,6) новое значение 12,34 можно так:

```
temperatures[0] = 12.34;
```

Запись в массивы и чтение из массивов

В листинге 6.3 демонстрируются приемы записи и чтения значений из массива с пятью элементами. Первый цикл `for` в скетче записывает случайные числа во все элементы массива, а второй цикл `for` извлекает элементы и выводит их в монитор порта.

Листинг 6.3. Демонстрация операций чтения/записи с массивом

```
// Листинг 6.3

void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

int array[5]; // определение массива с пятью элементами
void loop()
{
```

```
  int i;
  Serial.println();
  for ( i = 0 ; i < 5 ; i++ ) // запись в массив
  {
    array[i] = random(10);    // случайные числа от 0 до 9
  }
  for ( i = 0 ; i < 5 ; i++ ) // вывести содержимое массива
  {
    Serial.print("array[");
    Serial.print(i);
    Serial.print("] contains ");
    Serial.println(array[i]);
  }
  delay(5000);
}
```

На рис. 6.8 изображено окно монитора порта.

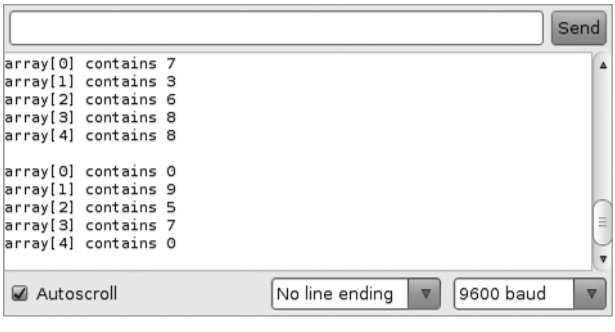


Рис. 6.8. Скетч из листинга 6.3 в действии

Теперь, когда вы знаете, как работать с двоичными числами, сдвиговыми регистрами и массивами, используем новые знания на практике. В следующем разделе мы подключим к плате несколько цифровых индикаторов.

Семисегментные светодиодные индикаторы

Работать со светодиодами очень интересно, но их можно применять для отображения весьма ограниченного набора данных. В этом разделе мы начнем использовать светодиодные цифровые семисегментные индикаторы, изображенные на рис. 6.9.

Эти индикаторы отлично подходят для отображения чисел и именно поэтому они используются при конструировании цифровых будильников, спидометров и других устройств, отображающих числовую информацию. Каждый индикатор имеет семисегментный дисплей с восемью светодиодами. Выпускаются индикаторы со

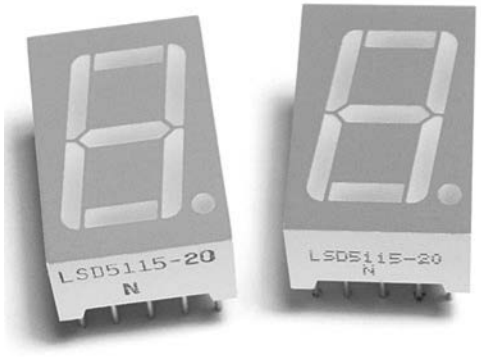


Рис. 6.9. Семисегментные индикаторы

светодиодами разного цвета. Чтобы уменьшить число выводов у индикаторов, все аноды или катоды светодиодов соединены с общим выводом, который называется *общим анодом* или *общим катодом* соответственно. В наших проектах будут использоваться индикаторы с общим катодом.

Светодиодные сегменты снабжены метками от *A* до *G* и *DP* (Decimal Point — десятичная точка). Каждому сегменту соответствует свой вывод, соединенный с анодом, а катоды подключены к общему выводу. Размещение сегментов всегда соответствует изображенному на рис. 6.10, где сегмент *A* находится вверху, *B* — справа и т. д. То есть, например, чтобы показать цифру 7, следует подать напряжение на сегменты *A*, *B* и *C*.

Порядок размещения выводов на корпусе светодиодного индикатора может отличаться у разных производителей, но они всегда соответствуют шаблону, изображенному на рис. 6.10. Планируя использовать такие индикаторы в своих проектах, обязательно потребуйте схему расположения выводов у продавца, чтобы не терять время на выяснение назначения разных выводов.

Для изображения семисегментных индикаторов на принципиальных схемах мы будем использовать значок, показанный на рис. 6.11.

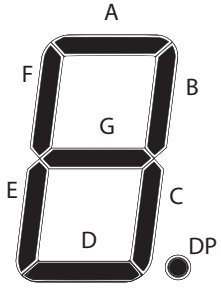


Рис. 6.10. Карта светодиодов типичного семисегментного индикатора

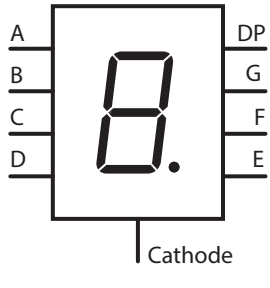


Рис. 6.11. Изображение семисегментного индикатора на принципиальных схемах

Управление сегментами

Для управления сегментами мы воспользуемся методом, описанным в проекте 17, подключив выводы с *A* по *DP* к выводам сдвигового регистра *Q0—Q7*. Какие сегменты включать и выключать для отображения цифр или букв, определяется по матрице в табл. 6.2.

Верхняя строка в матрице — это выводы сдвигового регистра, управляющие сегментами, названия которых перечислены во второй строке. Все остальные строки соответствуют отображаемым цифрам и содержат значения в двоичном и десятичном виде, которые требуется послать в сдвиговый регистр.

Таблица 6.2. Матрица для отображения цифр на семисегментном индикаторе

Выводы сдвигового регистра	**Q0**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	
Цифра	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**DP**	**Десятичное**
0	1	1	1	1	1	1	0	0	252
1	0	1	1	0	0	0	0	0	96
2	1	1	0	1	1	0	1	0	218
3	1	1	1	1	0	0	1	0	242
4	0	1	1	0	0	1	1	0	102
5	1	0	1	1	0	1	1	0	182
6	1	0	1	1	1	1	1	0	190
7	1	1	1	0	0	0	0	0	224
8	1	1	1	1	1	1	1	0	254
9	1	1	1	1	0	1	1	0	246
A	1	1	1	0	1	1	1	0	238
B	0	0	1	1	1	1	1	0	62
C	1	0	0	1	1	1	0	0	156
D	0	1	1	1	1	0	1	0	122
E	1	0	0	1	1	1	1	0	158
F	1	0	0	0	1	1	1	0	142

Например, чтобы отобразить цифру 7, как показано на рис. 6.12, нужно включить сегменты *A*, *B* и *C*, которым соответствуют выводы Q0, Q1 и Q2 сдвигового регистра. То есть чтобы установить высокий уровень на первых трех выводах, соответствующих перечисленным сегментам, мы должны послать в сдвиговый регистр байт B1110000 (вызовом `shiftOut` с параметром `LSBFIRST`).

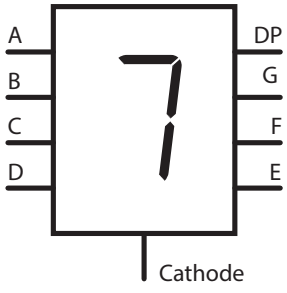


Рис. 6.12. Отображение цифры 7

В следующем примере соберем схему, которая последовательно будет отображать цифры от 0 до 9 и буквы от A до F. Затем цикл повторится, но с включенным сегментом десятичной точки.

Проект № 18: Дисплей с одной цифрой

В этом проекте мы соберем схему, использующую один цифровой индикатор.

Оборудование

Ниже перечислено необходимое оборудование:

- Один сдвиговый регистр 74HC595.
- Один семисегментный светодиодный индикатор с общим катодом.
- Восемь резисторов с номиналом 560 Ом (R1—R8).
- Одна большая макетная плата.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Схема для проекта изображена на рис. 6.13.

Подключая светодиодный индикатор к сдвиговому регистру, необходимо соединить выводы A—G с выводами Q0—Q6 соответственно, а вывод DP соединить с выводом Q7.

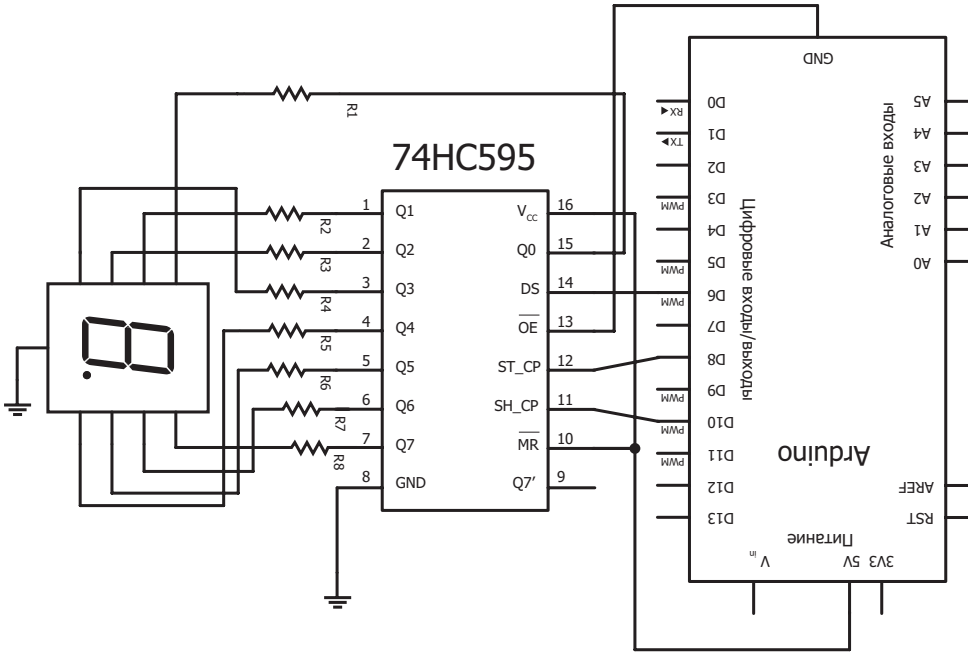


Рис. 6.13. Принципиальная схема проекта 18

Скетч

Скетч для проекта 18 хранит десятичные значения (см. табл. 6.2) в массиве `int digits[]`. Функция `void loop` сначала последовательно посылает эти значения в сдвиговый регистр (❶), а затем повторяет процесс с включенной десятичной точкой на индикаторе, добавляя 1 к значению, посылаемому в сдвиговый регистр (❷):

```
// Проект 18 - Дисплей с одной цифрой

#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

// Подготовить массив с комбинациями сегментов
// для цифр 0 - 9 и букв A - F (из табл. 6.2)
int digits[] = {252, 96, 218, 242, 102, 182, 190, 224,
                254, 246, 238, 62, 156, 122, 158, 142};

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void loop()
```

```
  {
    int i;
    for ( i = 0 ; i < 16 ; i++ ) // вывести символы 0-9, A-F
    {
      digitalWrite(LATCH, LOW);
❶      shiftOut(DATA, CLOCK, LSBFIRST, digits[i]);
      digitalWrite(LATCH, HIGH);
      delay(250);
    }
    for ( i = 0 ; i < 16 ; i++ ) // вывести символы 0-9, A-F с точкой
    {
      digitalWrite(LATCH, LOW);
❷      shiftOut(DATA, CLOCK, LSBFIRST, digits[i]+1); // +1, чтобы включить точку
      digitalWrite(LATCH, HIGH);
      delay(250);
    }
  }
```

Семисегментные индикаторы светятся ярко и легко читаются. Например, на рис. 6.14 изображен индикатор со светящейся цифрой 9 и десятичной точкой.

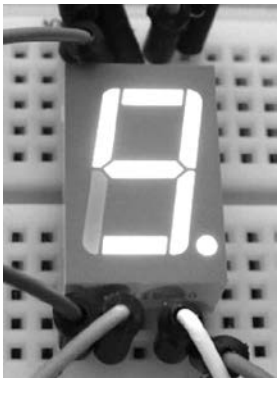


Рис. 6.14. Одна из цифр, отображаемых проектом 18

Отображение двух цифр

Чтобы использовать несколько сдвиговых регистров для управления дополнительными цифровыми выходами, соедините вывод 9 микросхемы 74HC595 (которая принимает данные с платы Arduino) с выводом 14 второго сдвигового регистра. После этого необходимо послать 2 байта данных: первый — для управления вторым сдвиговым регистром и второй — для управления первым. Например:

```
digitalWrite(LATCH, LOW);
shiftOut(DATA, CLOCK, MSBFIRST, 254); // данные для второй микросхемы 74HC595
shiftOut(DATA, CLOCK, MSBFIRST, 254); // данные для первой микросхемы 74HC595
digitalWrite(LATCH, HIGH);
```

Проект № 19: Управление двумя семисегментными индикаторами

В этом проекте вы узнаете, как управлять двумя семисегментными светодиодными индикаторами и отображать двузначные числа.

Оборудование

Ниже перечислено необходимое оборудование:

- Два сдвиговых регистра 74HC595.
- Два семисегментных светодиодных индикатора с общим катодом.
- Шестнадцать резисторов с номиналом 560 Ом (R1—R16).
- Одна большая макетная плата.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Схема для проекта с двумя индикаторами изображена на рис. 6.15.

Обратите внимание, что выводы «вход для тактовых импульсов» и «защелка» (11 и 12 соответственно) сдвиговых регистров попарно соединены между собой и с платой Arduino. Линия передачи данных из платы Arduino (контакт 6) идет к выводу 14 первого сдвигового регистра и далее, через вывод 9 первого регистра, подключается к выводу 14 второго сдвигового регистра.

Для отображения чисел в диапазоне от 0 до 99 требуется более сложный скетч. Если число меньше 10, достаточно просто послать само число и 0, чтобы на правом индикаторе отобразить цифру, а на левом — 0. Но если число больше или равно 10, то необходимо определить цифры, составляющие число, и послать каждую в свой сдвиговый регистр. Чтобы упростить процедуру, мы воспользуемся математической функцией деления по модулю.

Деление по модулю

Деление по модулю — это функция, возвращающая остаток от деления. Например, деление по модулю 10 на 7 даст в результате 3 — иными словами, остаток от деления 10 на 7 равен 3. Операцию деления по модулю в языке C выполняет оператор `%`. Ниже показано, как он используется в скетчах:

```
int a = 8;
int b = 3;
c = a % b;
```

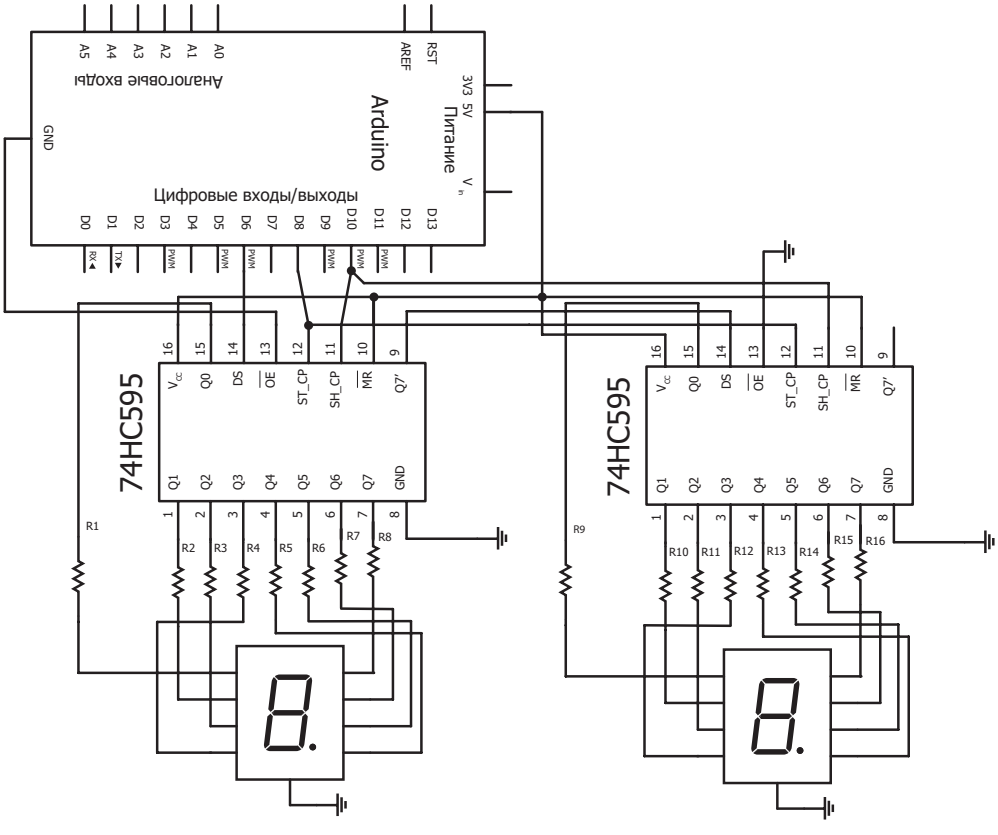


Рис. 6.15. Принципиальная схема проекта 19

В данном примере переменная `c` получит значение 2. То есть чтобы получить младшую цифру из двузначного числа, следует использовать операцию деления по модулю, возвращающую остаток от деления двух чисел.

Для автоматизации отображения одно- и двузначных чисел мы напишем функцию `displayNumber()`. В ней будет использоваться операция деления по модулю для определения цифр, составляющих двузначные числа. Например, чтобы отобразить число 23, сначала нужно отделить левую цифру, разделив 23 на 10, в результате вы получите 2 (дробную часть следует игнорировать). Чтобы отделить правую цифру, разделите по модулю 23 на 10, что даст в результате 3.

```
// Проект 19 - Управление двумя семисегментными индикаторами

  #define DATA 6   // к выводу 14 микросхемы 74HC595
  #define LATCH 8  // к выводу 12 микросхемы 74HC595
  #define CLOCK 10 // к выводу 11 микросхемы 74HC595

  // Подготовить массив с комбинациями сегментов
```

```
// для цифр 0 - 9 и букв A - F (из табл. 6.2)
int digits[] = {252, 96, 218, 242, 102, 182, 190, 224,
                254, 246, 238, 62, 156, 122, 158, 142};

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void displayNumber(int n)
{
  int left, right=0;
❶  if (n < 10)
  {
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[n]);
    shiftOut(DATA, CLOCK, LSBFIRST, 0);
    digitalWrite(LATCH, HIGH);
  }
  else if (n >= 10)
  {
❷    right = n % 10; // остаток от деления числа на 10
    left = n / 10;  // частное от деления числа на 10
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[right]);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[left]);
    digitalWrite(LATCH, HIGH);
  }
}

❸ void loop()
{
  int i;
  for ( i = 0 ; i < 100 ; i++ )
  {
    displayNumber(i);
    delay(100);
  }
}
```

Функция `displayNumber()` сначала сравнивает отображаемое число с 10 (❶). Если число меньше 10, функция посылает в сдвиговые регистры само число и цифру 0. Но если число больше или равно 10, функция использует операции деления по модулю и деления (❷) для получения цифр и посылает их в сдвиговые регистры по отдельности. Наконец, в функции `void loop()` (❸) осуществляется вывод чисел от 0 до 99 с помощью функции `displayNumber()`.

Проект № 20: Цифровой термометр

В этом проекте мы добавим в схему вывода двузначных чисел температурный датчик TMP36, использовавшийся в главе 4, и создадим цифровой термометр. Алгоритм работы такого термометра прост: скетч будет читать выходное напряжение датчика TMP36 (используя прием из проекта 12) и преобразовывать его в градусы Цельсия.

Оборудование

Ниже перечислено необходимое оборудование:

- Собранная схема с двумя сдвиговыми регистрами из проекта 19.
- Один температурный датчик TMP36.

Подключите центральный вывод датчика TMP36 к контакту аналогового входа A5, левый — к контакту 5 V и правый — к контакту GND.

Скетч

Ниже приводится скетч для данного проекта:

```
// Проект 20 - Цифровой термометр

#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

int temp = 0;
float voltage = 0;
float celsius = 0;
float sensor = 0;
int digits[]={252, 96, 218, 242, 102, 182, 190, 224,
              254, 246, 238, 62, 156, 122, 158, 142};

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void displayNumber(int n)
```

```
{
  int left, right = 0;
  if (n < 10)
  {
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[n]);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[0]);
    digitalWrite(LATCH, HIGH);
  }
  if (n >= 10)
  {
    right = n % 10;
    left = n / 10;
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[right]);
    shiftOut(DATA, CLOCK, LSBFIRST, digits[left]);
    digitalWrite(LATCH, HIGH);
  }
}

void loop()
{
  sensor = analogRead(5);
  voltage = (sensor * 5000) / 1024; // преобразовать в милливольты
  voltage = voltage - 500;   // учесть смещение
  celsius = voltage / 10;    // преобразовать милливольты в градусы
  temp = int(celsius);       // преобразовать вещественное значение
                             // температуры в целочисленное
  displayNumber(temp);
  delay(500);
}
```

Скетч очень прост и заимствует код из предыдущих проектов: `displayNumber()` из проекта 19 и вычисление температуры из проекта 12. Вызов `delay(500);` во второй с конца строке скетча препятствует слишком быстрой смене показаний, вызываемых флуктуациями температуры.

Матричные светодиодные индикаторы

Если вам понравилось экспериментировать с мигающими светодиодами, то вы просто обязаны влюбиться в матричные светодиодные индикаторы. *Матричный светодиодный индикатор* состоит из нескольких рядов и столбцов светодиодов, которыми можно управлять по отдельности или группами. В следующем проекте будет использоваться матричный индикатор (изображен на рис. 6.16), имеющий восемь рядов и восемь столбцов с красными светодиодами, всего 64 светодиода. Светодиоды в этой матрице соединены по схеме с общим катодом.

Рис. 6.16. Светодиодная матрица

В этом проекте мы соберем одну схему и затем будем использовать разные скетчи для создания разнообразных эффектов.

Схема светодиодной матрицы

На принципиальных схемах светодиодная матрица выглядит довольно сложно, это показано на рис. 6.17.

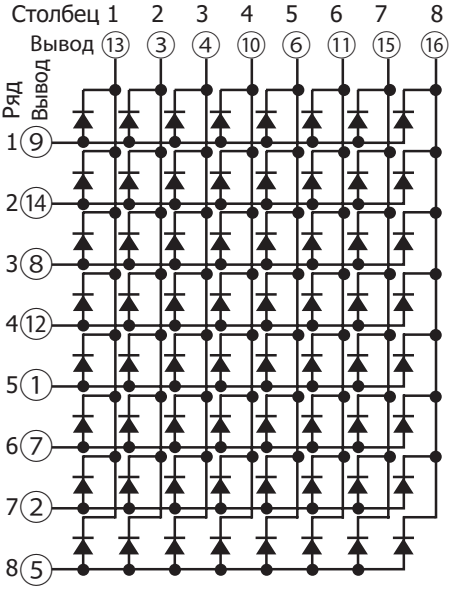


Рис. 6.17. Изображение светодиодных матриц на принципиальных схемах

Обратите внимание, что нумерация выводов матрицы, как показано на рис. 6.17, не соответствует нумерации рядов и столбцов. Если смотреть с обратной стороны матрицы, вывод 1 будет находиться справа внизу. Например, на рис. 6.18 можно видеть маленькую цифру 1 под выводом.

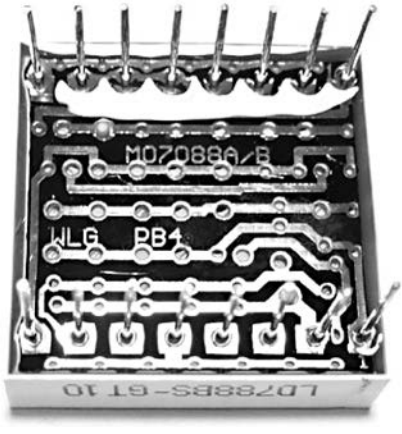


Рис. 6.18. Выводы светодиодной матрицы с маленькой меткой 1 под правым нижним выводом

Выводы светодиодной матрицы нумеруются в направлении по часовой стрелке (если смотреть со стороны выводов), то есть вывод 8 находится слева внизу, а вывод 16 — справа вверху. Мы будем управлять матрицей с помощью двух сдвиговых регистров 74HC595, подобно тому, как в проекте 19 осуществлялось управление семисегментными индикаторами.

Принципиальная схема проекта изображена на рис. 6.19. Резисторы R1—R8 имеют номинал 560 Ом каждый. Один сдвиговый регистр управляет рядами светодиодов, а второй — столбцами. Светодиодная матрица (здесь не показана) подключается к выходам этой схемы согласно табл. 6.3.

Таблица 6.3. Таблица подключения выводов матрицы к сдвиговым регистрам 74HC595

Вывод регистра, управляющего рядом	Вывод матрицы	Вывод регистра, управляющего столбцом	Вывод матрицы
15	9	15	13
1	14	1	3
2	8	2	4
3	12	3	10
4	1	4	6
5	7	5	11
6	2	6	15
7	5	7	16

Соединения

Соедините выводы сдвиговых регистров с выводами светодиодной матрицы в соответствии с табл. 6.3 (не забудьте добавить ограничивающие резисторы между выводами сдвигового регистра, управляющего рядами, и выводами матрицы).

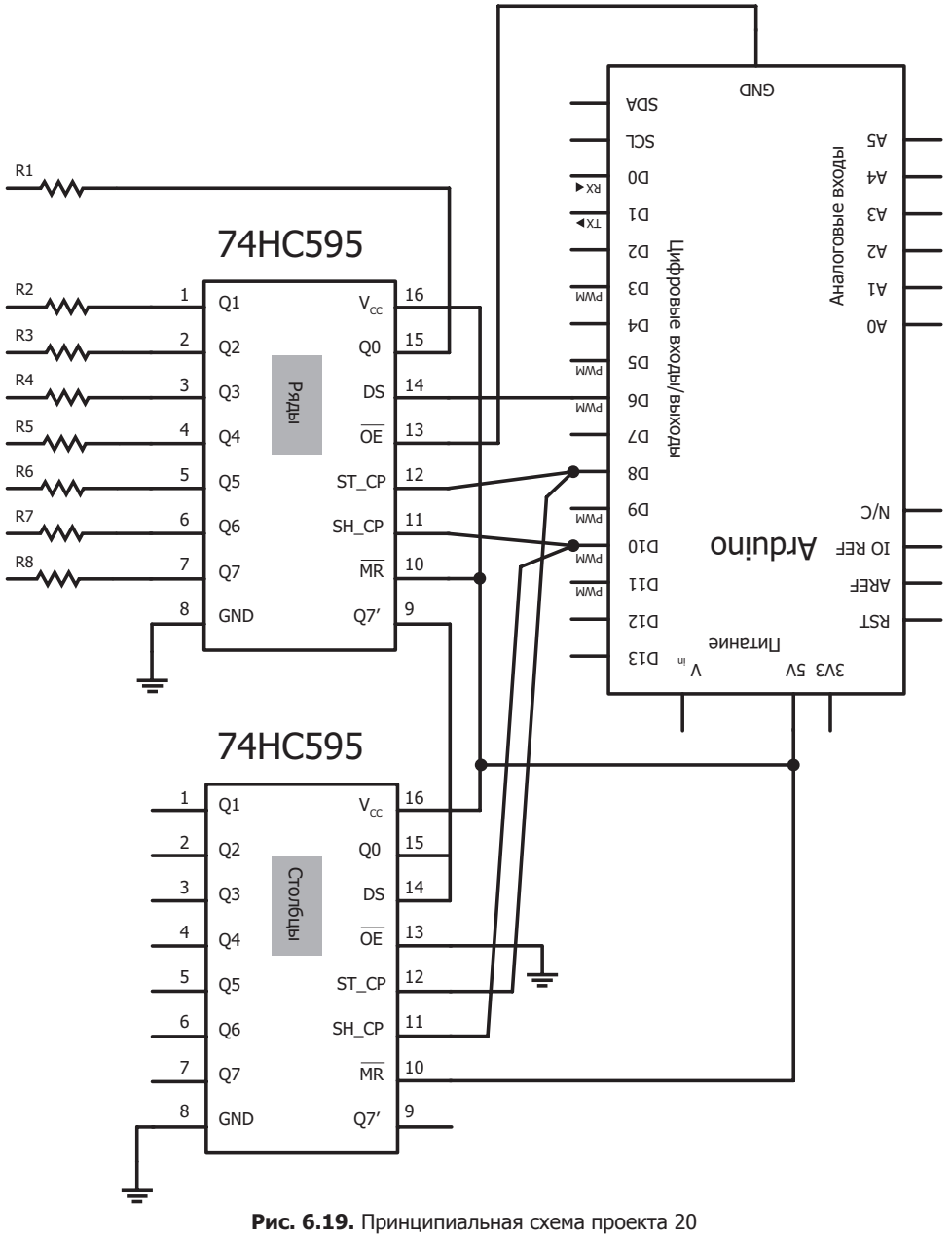


Рис. 6.19. Принципиальная схема проекта 20

Сдвиговый регистр, подписанный как «Ряды» на рис. 6.19, позволяет току течь через ряды матрицы, а сдвиговый регистр, подписанный как «Столбцы», позволяет току течь через столбцы матрицы к контакту GND. В следующем проекте мы воспользуемся простым скетчем для проверки схемы. Чтобы включить светодиод, необходимо подать напряжение на соответствующий ряд и столбец матрицы через сдвиговые регистры.

Поразрядная арифметика

Поразрядная арифметика позволяет манипулировать двоичным представлением значений переменных типов `int` и `byte`. Преимущество поразрядной арифметики перед десятичной состоит в том, что поразрядная арифметика помогает увеличить скорость управления цифровыми входами и выходами и способна осуществлять сравнение чисел в двоичной форме.

В арсенале поразрядной арифметики имеется шесть операторов: И (AND), ИЛИ (OR), ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR), НЕ (NOT), сдвиг влево и сдвиг вправо. Все они по очереди обсуждаются в следующих разделах главы. (Несмотря на то что в примерах используются двоичные числа, эти операторы могут применяться к переменным типов `int` и `byte`.)

Оператор поразрядного И (AND)

Оператор И (AND) — `&` — выполняет поразрядное сравнение двух двоичных чисел. Если биты в каком-то разряде обоих чисел равны 1, то бит в той же позиции результата тоже будет равен 1. Если же хотя бы в одном из чисел соответствующий бит равен 0, то и бит в той же позиции результата будет равен 0. Рассмотрим пример с тремя переменными типа `byte`:

```
byte a = B00110011;
byte b = B01010001;
byte c = 0;
```

Операция сравнения

```
c = a & b;
```

даст в результате 00010001. Дополним пример текстовыми комментариями, чтобы получить более полное представление:

```
byte a = B00110011;
//    ||||||||
byte b = B01010001;
//    ||||||||
//    00010001 c = a & b; // c получит результат a 'И' b
```

Оператор поразрядного ИЛИ (OR)

Оператор ИЛИ (OR) — | — выполняет поразрядное сравнение двух двоичных чисел, но, в отличие от поразрядного И (AND), возвращает 1, если хотя бы в одном из операндов бит в данной позиции равен 1. Если оба операнда имеют в некоторой позиции бит со значением 0, в результате бит в этой же позиции будет установлен в 0.

Воспользуемся для демонстрации теми же значениями операндов, что и в предыдущем примере:

```
byte a = B00110011;
byte b = B01010001;
byte c = 0;
```

Операция сравнения

```
c = a | b;
```

даст в результате 01110011.

Оператор поразрядного ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR)

Оператор ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR) — ^ — вернет 1, если биты в операндах отличаются, и 0, если они одинаковы.

Воспользуемся теми же значениями еще раз:

```
byte a = B00111100;
byte b = B01011010;
byte c = 0;
```

Операция сравнения

```
c = a ^ b;
```

даст в результате 01100110.

Оператор поразрядного НЕ (NOT)

Оператор НЕ (NOT) — ~ — просто обращает значения битов: нули превращаются в единицы, а единицы — в нули. Рассмотрим пример: если применить оператор поразрядного НЕ (NOT) к байту `a` и сохранить результат в переменной `b`, как показано ниже:

```
byte a = B00111100;
byte b = 0;
b = ~a;
```

в результате `b` получит значение 11000011.

Поразрядный сдвиг влево и вправо

Операторы поразрядного сдвига влево (<<) и вправо (>>) сдвигают биты влево или вправо на указанное число позиций. Например, если содержимое переменной `a` сдвинуть влево на четыре позиции, как показано ниже:

```
byte a = B00100101;
byte b = a << 4;
```

в результате `b` получит значение 01010000. Биты в `a` были сдвинуты влево на четыре позиции, а освободившиеся биты справа заполнились нулями.

Если выполнить сдвиг в противоположном направлении, как показано ниже:

```
byte a = B11110001;
byte b = a >> 4;
```

в результате `b` получит значение 00001111.

Проект № 21: Создание светодиодной матрицы

Цель этого проекта — показать, как пользоваться светодиодной матрицей; мы включим каждый второй столбец и каждый второй ряд в матрице, как показано на рис. 6.20.

Рис. 6.20. Модель шахматной доски

Чтобы создать шаблон, как на рис. 6.20, мы отправим значение `B10101010` в сдвиговый регистр, управляющий рядами, и `~B10101010` — в сдвиговый регистр, управляющий столбцами. Единицы и нули в каждом байте соответствуют рядам и столбцам матрицы.

ПРИМЕЧАНИЕ

Обратите внимание на применение оператора поразрядного НЕ (NOT) — ~ — к байту управления столбцами. Бит в сдвиговом регистре, управляющем столбцами, должен иметь значение 0, чтобы включить светодиод в столбце, соответствующем этому биту. Но бит в сдвиговом регистре, управляющем рядами, должен иметь значение 1, чтобы включить светодиод в ряду, соответствующем этому биту. Поэтому чтобы инвертировать байт данных, посылаемый во второй сдвиговый регистр, управляющий столбцами, к нему применяется оператор поразрядной арифметики ~.

Чтобы воссоздать эффект, изображенный на рис. 6.20, используем следующий скетч:

```
// Проект 21 - Создание светодиодной матрицы

#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void loop()
{
  digitalWrite(LATCH, LOW);
  shiftOut(DATA, CLOCK, MSBFIRST, ~B10101010); // столбцы
  shiftOut(DATA, CLOCK, MSBFIRST, B10101010);  // ряды
  digitalWrite(LATCH, HIGH);
  do {} while (1);                             // ничего не делать
}
```

Результат работы скетча изображен на рис. 6.21. Мы включили каждый второй светодиод в матрице, сформировав модель шахматной доски.

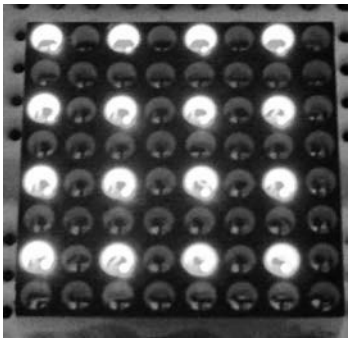


Рис. 6.21. Результат выполнения скетча из проекта 21

Проект № 22: Создание образов на светодиодной матрице

Для отображения произвольных образов на светодиодной матрице необходима функция, включающая только один светодиод в каждый момент времени. Но чтобы получилось изображение, следует воспользоваться *инерционностью зрительного восприятия* и включать и выключать светодиоды очень быстро. Благодаря инерционности зрения наш глаз продолжает «видеть» изображение еще в течение долей секунды уже после того, как оно исчезло. Мы можем воспользоваться этим эффектом, чтобы создавать произвольные изображения, быстро «сканируя» светодиодную матрицу по рядам. Этот прием пригодится для создания анимационных эффектов, отображения данных и создания других художественных эффектов.

Управление отдельными светодиодами будет продемонстрировано в двух следующих проектах. В скетче ниже функция `void setLED()` принимает номер ряда и столбца, а также продолжительность свечения светодиода. Он включает случайные светодиоды по одному.

```
// Проект 22 - Создание образов на светодиодной матрице

#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
  randomSeed(analogRead(0));
}

❶ int binary[] = {1, 2, 4, 8, 16, 32, 64, 128};
int r, c = 0;

void setLED(int row, int column, int del)
{
  digitalWrite(LATCH, LOW);
  shiftOut(DATA, CLOCK, MSBFIRST, ~binary[column]); // столбцы
  shiftOut(DATA, CLOCK, MSBFIRST, binary[row]);     // ряды
  digitalWrite(LATCH, HIGH);
  delay(del);
}

void loop()
{
  r = random(8);
  c = random(8);
  // включить светодиод в случайно выбранном ряду и столбце
  // на 1 миллисекунду
  setLED(r, c, 1);
}
```

Для управления светодиодами вместо непосредственной передачи двоичных чисел в функцию `shiftOut()` мы использовали справочную таблицу в форме массива `int binary[]` (❶). Эта таблица содержит десятичные эквиваленты всех битов в байте для отправки в сдвиговый регистр. Например, чтобы включить светодиод в четвертом ряду и четвертом столбце, необходимо в оба сдвиговых регистра послать значение `binary[3]` (число 8, или `B00001000`), при этом перед отправкой в регистр управления столбцами к значению из таблицы должна применяться операция поразрядного НЕ (~). Это очень удобный способ преобразования номеров рядов и столбцов в форму, пригодную для записи в сдвиговые регистры; нет необходимости задумываться о двоичном представлении, достаточно лишь указать номер ряда и столбца.

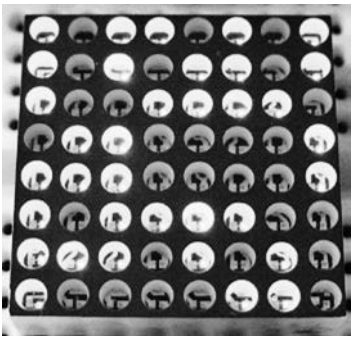


Рис. 6.22. Проект 22 во время работы

Благодаря задержке в 1 миллисекунду светодиоды в проекте 22 включаются и выключаются так быстро, что наш инерционный глаз «видит» включенными сразу несколько светодиодов; этот проект наглядно демонстрирует эффект инерционности зрительного восприятия: создается иллюзия, что одновременно включено сразу несколько светодиодов, хотя фактически в каждый момент времени включен только один из них. Пример такой иллюзии изображен на рис. 6.22.

Проект № 23: Отображение образа на светодиодной матрице

В этом проекте мы отобразим на светодиодной матрице образ, представленный на рис. 6.23.

Рис. 6.23. Образ для примера, использующего инерционность зрительного восприятия

Каждый ряд можно определить как двоичное число и поместить эти числа в массив:

```
byte smile[] = {B00000000,
                B00000110,
                B01000110,
                B00000000,
                B00011000,
                B00000000,
                B11000011,
                B01111110};
```

Обратите внимание, что единицы в этом массиве соответствуют включенным светодиодам в образе, изображенном на рис. 6.23. Использование двоичных чисел упрощает определение образов в программном коде. С помощью смены параметров `MSBFIRST` и `LSBFIRST` в вызове функции `shiftOut()` реализуется зеркальное отражение каждой строки. Для поочередного отображения рядов в следующем скетче используется цикл `for` (❶):

```
// Проект 23 - Отображение образа на светодиодной матрице

#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

byte smile[] = {B00000000, B00000110, B01000110, B00000000,
                B00011000, B00000000, B11000011, B01111110};
int binary[] = {1, 2, 4, 8, 16, 32, 64, 128};

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void loop()
{
  int i;
❶  for ( i = 0 ; i < 8 ; i++ )
  {
    digitalWrite(LATCH, LOW);
    shiftOut(DATA, CLOCK, MSBFIRST, ~smile[i]); // столбцы
    shiftOut(DATA, CLOCK, LSBFIRST, binary[i]); // ряды
    digitalWrite(LATCH, HIGH);
    delay(1);
  }
}
```

В результате выполнения скетча на светодиодной матрице получается изображение подмигивающего смайлика, как показано на рис. 6.24.

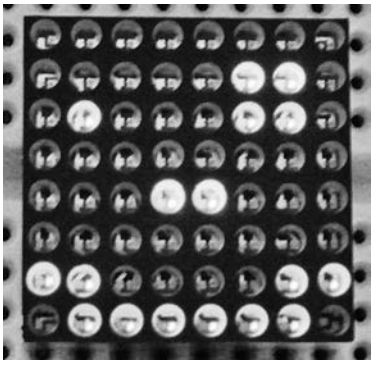


Рис. 6.24. Результат выполнения скетча из проекта 23

Проект № 24: Анимация на светодиодной матрице

С помощью поразрядной арифметики можно создать эффект прокрутки образа из проекта 23 в любом направлении. Например, реализовать прокрутку смайлика вправо за границы матрицы можно так: отобразить образ, при помощи оператора >> сдвинуть каждый ряд вправо и затем вновь отобразить образ.

Скетч

Ниже приводится скетч для данного проекта:

```
// Проект 24 - Анимация на светодиодной матрице

#define DATA 6   // к выводу 14 микросхемы 74HC595
#define LATCH 8  // к выводу 12 микросхемы 74HC595
#define CLOCK 10 // к выводу 11 микросхемы 74HC595

byte smile[] = {B00000000, B00000110, B01000110, B00000000,
                B00011000, B00000000, B11000011, B01111110};
int binary[] = {1, 2, 4, 8, 16, 32, 64, 128};

void setup()
{
  pinMode(LATCH, OUTPUT);
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
}

void loop()
{
  int a, hold, shift;
```

```
❶   for ( shift = 0 ; shift < 9 ; shift++ )
    {
❷     for ( hold = 0 ; hold < 25 ; hold++ )
      {
        for ( a = 0 ; a < 8 ; a++ )
        {
          digitalWrite(LATCH, LOW);
❸          shiftOut(DATA, CLOCK, MSBFIRST, ~smile[a]>>shift); // столбцы
          shiftOut(DATA, CLOCK, LSBFIRST, binary[a]);        // ряды
          digitalWrite(LATCH, HIGH);
          delay(1);
        }
      }
    }
  }
```

Скетч удерживает образ на светодиодной матрице в течение 25 итераций цикла `for` (❷). Переменная `shift` определяет количество разрядов для операции сдвига вправо. После каждой итерации цикла переменная `shift` увеличивается на 1 (❶). Затем циклы отображения повторяются. В результате возникает эффект смещения образа вправо на 1 столбец. Меняя параметры `MSBFIRST` и `LSBFIRST` в третьем цикле `for` (❸), можно менять направление прокрутки.

Забегая вперед

В этой главе вы овладели множеством базовых навыков, которые пригодятся при реализации будущих проектов. Светодиодные индикаторы весьма надежны, поэтому не бойтесь пытаться воспроизвести на них разные визуальные эффекты. Однако их изобразительные возможности имеют предел, поэтому в следующей главе мы воспользуемся более мощными средствами для отображения текста и графики.

7 Жидкокристаллические индикаторы

В этой главе вы:

- ✓ научитесь пользоваться символьными жидкокристаллическими индикаторами для отображения текста и числовых данных;
- ✓ узнаете, как создавать свои символы для отображения на жидкокристаллических индикаторах;
- ✓ освоите отображение текста и данных на больших жидкокристаллических индикаторах;
- ✓ создадите цифровой термометр с памятью, отображающий историю изменения температуры в виде графика.

В некоторых проектах вам потребуется отображать информацию на чем-то ином, кроме монитора настольного компьютера. Проще всего для этой цели использовать жидкокристаллические индикаторы (ЖКИ) вместе с платой Arduino. Они позволяют отображать текст, нестандартные символы, числовые данные и графику (графические ЖКИ).

Символьные жидкокристаллические индикаторы

Жидкокристаллические индикаторы, отображающие символьную информацию, например текст и числа, являются самыми недорогими и простыми в использовании среди всех ЖКИ. Они имеют разные размеры, измеряемые числом и длиной отображаемых строк. Некоторые имеют подсветку и позволяют выбирать цвет символов и фона. Любой жидкокристаллический индикатор с HD44780- или KS0066-совместимым интерфейсом и напряжением питания подсветки 5 В «умеет» работать с платой Arduino. Для начала мы попробуем использовать ЖКИ с подсветкой, способный отображать 2 строки по 16 символов в каждой, изображенный на рис. 7.1.

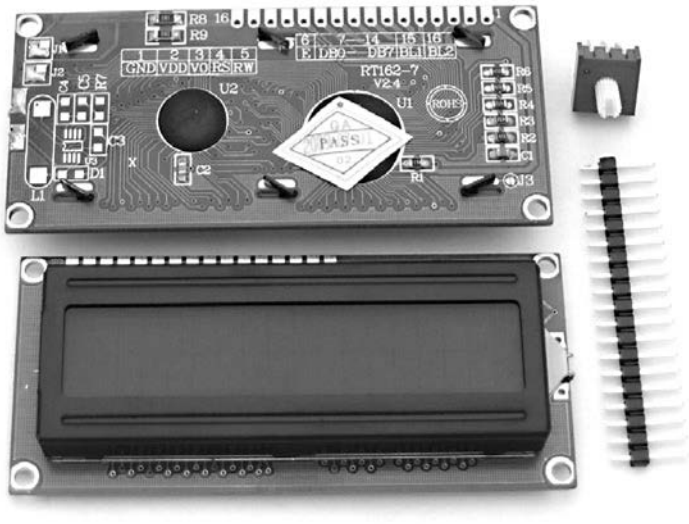


Рис. 7.1. Жидкокристаллический индикатор с подстроечным резистором и колодкой контактов

Подстроечный резистор (переменный резистор для ЖКИ) имеет сопротивление 10 кОм и используется для регулировки контрастности индикатора. Колодка контактов впаивается в ряд отверстий в верхней части платы с ЖКИ, чтобы можно было подключать индикатор непосредственно к макетной плате. Отверстия вдоль верхнего края платы с ЖКИ пронумерованы от 1 до 16. Отверстие с номером 1 расположено ближе к углу платы и на схеме, изображенной на рис. 7.2, подписано как VSS (подключается к «земле»). Если ваш ЖКИ имеет напряжение питания подсветки 4.2 В, подключите диод 1N4004 между контактом 5V на плате Arduino и контактом LED+ на плате ЖКИ. Мы будем ссылаться на эту схему во всех примерах в книге, где будут использоваться жидкокристаллические индикаторы.

Использование символьного ЖКИ в скетче

Чтобы задействовать символьный жидкокристаллический индикатор, показанный на рис. 7.1, прежде всего необходимо познакомиться с некоторыми функциями и понять, как они работают. Для этого рассмотрим несколько простых примеров. Введите и загрузите скетч, представленный в листинге 7.1.

Листинг 7.1. Скетч, демонстрирующий работу с ЖКИ

```
// Листинг 7.1
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9); // к выводам RS, E, DB4, DB5, DB6, DB7

void setup()
```

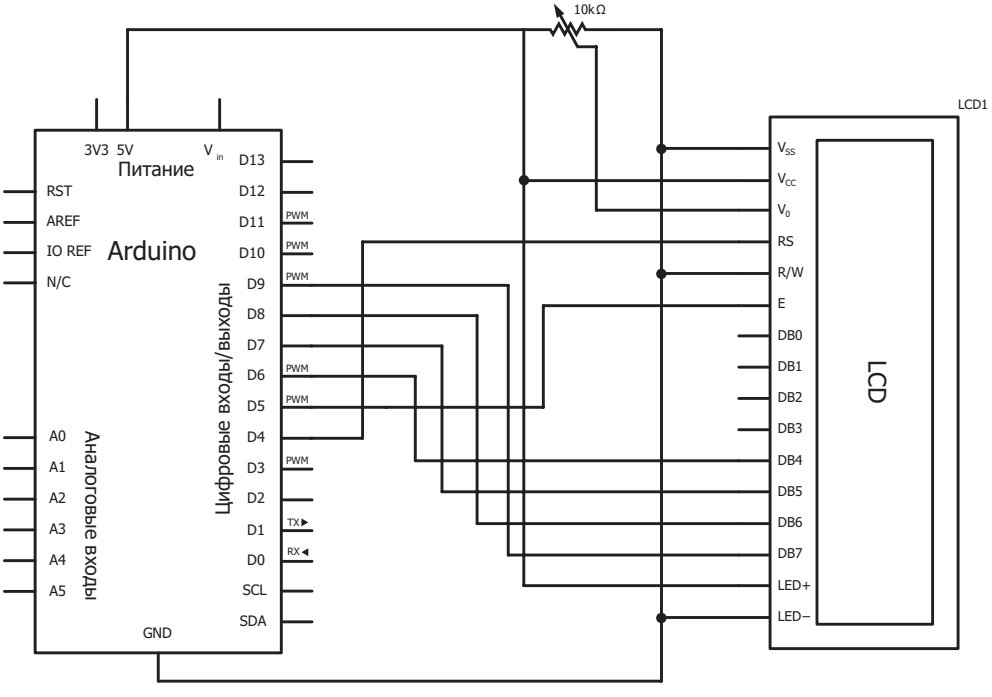


Рис. 7.2. Схема подключения жидкокристаллического индикатора

```
{
  lcd.begin(16, 2);
  lcd.clear();
}

void loop()
{
  lcd.setCursor(0, 5);
  lcd.print("Hello");
  lcd.setCursor(1, 6);
  lcd.print("world!");
  delay(10000);
}
```

На рис. 7.3 показан результат работы скетча из листинга 7.1.

Рис. 7.3. Демонстрация ЖКИ: «Hello world!»

Теперь «разложим по полочкам» работу скетча из листинга 7.1. Прежде всего, скетч должен включать две начальные строки. Их цель — подключить библиотеку для работы с жидкокристаллическими индикаторами (она автоматически устанавливается в составе Arduino IDE) и сообщить ей, к каким контактам на плате Arduino подключен ЖКИ. Этой цели служат следующие две строки перед функцией `void setup()`:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9); // к выводам RS, E, DB4, DB5, DB6, DB7
```

Если вам потребуется использовать другие контакты на плате Arduino, измените номера контактов во второй строке.

Далее, в функции `void setup()`, библиотеке сообщается размер экрана ЖКИ в столбцах и строках. Например, в этом скетче мы сообщаем, что экран ЖКИ имеет две строки по 16 символов в каждой:

```
lcd.begin(16, 2);
```

Отображение текста

После настройки ЖКИ в следующей строке производится очистка экрана:

```
lcd.clear();
```

Затем курсор устанавливается в позицию начала вывода текста вызовом функции:

```
lcd.setCursor(x, y);
```

Здесь `x` — это номер столбца в строке (от 0 до 15), а `y` — номер строки (0 или 1). После этого производится вывод текста вызовом функции `lcd.print()`, например, чтобы вывести слово «text», скетч должен выполнить команду:

```
lcd.print("text");
```

Теперь, когда вы знаете, как позиционировать курсор и выводить текст, перейдем к отображению содержимого переменных.

Отображение переменных или чисел

Чтобы вывести содержимое переменной на экран ЖКИ, используйте вызов:

```
lcd.print(variable);
```

При выводе переменной типа `float` укажите количество десятичных знаков для вывода. Например, `lcd.print(pi, 3)` в следующем примере отобразит значение `pi` с тремя десятичными знаками, как показано на рис. 7.4:

```
float pi = 3.141592654;
lcd.print("pi: ");
lcd.print(pi, 3);
```

Рис. 7.4. Отображение значения вещественной переменной на экран ЖКИ

Содержимое целочисленной переменой можно отобразить на экране ЖКИ не только в десятичном, но также в двоичном и шестнадцатеричном виде, как показано в листинге 7.2.

Листинг 7.2. Функции для отображения целых чисел в двоичном и шестнадцатеричном форматах

```
// Листинг 7.2
int zz = 170;
lcd.setCursor(0, 0);
lcd.print("Binary: ");
lcd.print(zz, BIN); // вывести число 170 в двоичном виде
lcd.setCursor(0, 1);
lcd.print("Hexadecimal: ");
lcd.print(zz, HX); // вывести число 170 в шестнадцатеричном виде
```

Этот фрагмент выведет на экран ЖКИ текст, как показано на рис. 7.5.

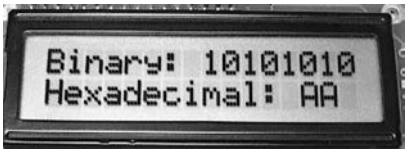


Рис. 7.5. Результаты выполнения кода из листинга 7.2

Проект № 25: Определение собственных символов

Помимо стандартных букв, цифр и других символов из стандартного набора в каждом скетче можно определять и собственные символы. Обратите внимание, что на экране ЖКИ каждый символ отображается в матрице, содержащей восемь рядов по пять точек, или *пикселей*. На рис. 7.6 эти матрицы показаны крупным планом.

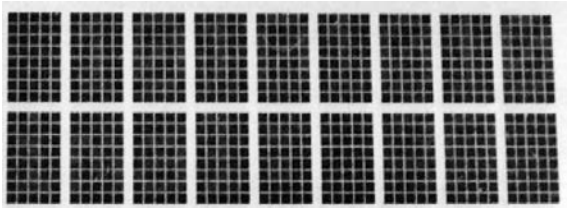


Рис. 7.6. Каждый символ состоит из пяти рядов по пять пикселей

Чтобы отобразить собственный символ, его сначала нужно определить в виде массива. Например, символ-смайлик определяется следующим образом:

```
byte a[8] = { B00000,
              B01010,
              B01010,
              B00000,
              B00100,
              B10001,
              B01110,
              B00000 };
```

Каждая цифра в этом массиве соответствует пикселю: 0 — выключенному, 1 — включенному. Элементы массива представляют строки пикселей на экране; первый элемент соответствует верхней строке, следующий элемент — второй строке и т. д.

ПРИМЕЧАНИЕ

Планируя использование собственных символов, попробуйте сначала нарисовать их на разлинованной бумаге. Каждый закрашенный квадрат на ней будет соответствовать 1 в массиве, а каждый пустой квадрат — 0.

В этом примере первый элемент массива имеет значение `B00000`, поэтому все пиксели в верхнем ряду будут выключены. Во втором ряду (ему соответствует элемент `B01010`) будет включен каждый второй пиксель — они образуют верхние края глаз. Каждый следующий ряд продолжает заполнять символ.

Далее мы сохраним массив (определяющий новый символ) в первый из восьми слотов, предназначенных для нестандартных символов, в функции `void setup()`:

```
lcd.createChar(0, a); // сохранить массив a[8] в слот 0
```

Наконец, чтобы отобразить символ, добавим следующий вызов в `void loop()`:

```
lcd.write(byte(0));
```

Используем следующий код для отображения нестандартных символов:

```
// Проект 25 - Определение собственных символов
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9); // к выводам RS, E, DB4, DB5, DB6, DB7
byte a[8] = { B00000,
              B01010,
              B01010,
              B00000,
              B00100,
              B10001,
              B01110,
              B00000 };

void setup()
{
  lcd.begin(16, 2);
  lcd.createChar(0, a);
}

void loop()
{
  lcd.write(byte(0)); // вывести нестандартный символ из слота 0
                      // в следующую позицию курсора
}
```

На рис. 7.7 показаны два ряда смайликов на экране ЖКИ.

Рис. 7.7. Результат выполнения проекта 25

Символьные жидкокристаллические индикаторы просты в использовании и довольно универсальны. Например, с помощью такого индикатора можно создать цифровой термометр, объединив его с измерительной частью проекта 20. Однако если нужно будет отображать много данных или графические элементы, вам потребуется *графический ЖКИ*.

Графические жидкокристаллические индикаторы

Графические жидкокристаллические индикаторы больше и дороже символьных индикаторов, но они обладают большей гибкостью. Графические ЖКИ позволяют не только отображать текст, но и рисовать линии, точки, окружности и многое другое

Рис. 7.8. Графический ЖКИ

для создания визуальных эффектов. В проектах этой книги будет использоваться KS0108B-совместимый графический ЖКИ с экраном 128 на 64 пикселя, изображенный на рис. 7.8.

Подстроечный резистор для графического ЖКИ, как и для символьного, имеет номинал 10 кОм и используется для регулировки контрастности индикатора. Колодка контактов впаивается в ряд отверстий в нижней части платы с ЖКИ, что позволяет подключать индикатор непосредственно к макетной плате. Отверстия вдоль нижнего края платы с ЖКИ пронумерованы от 1 до 20. Отверстие с номером 1 расположено ближе к углу платы.

Подключение графического ЖКИ

Прежде чем использовать графический ЖКИ, соедините его 20 проводами с платой Arduino, как описано в табл. 7.1.

Таблица 7.1. Карта соединений ЖКИ с платой Arduino

Контакт на плате ЖКИ	Контакт на плате Arduino	Назначение контакта на плате ЖКИ
1	5 V	Напряжение питания
2	GND	Земля
3	Центральный вывод подстроечного резистора с номиналом 10 кОм	Входное напряжение; регулировка контраста
4	D8	Бит 0 данных
5	D9	Бит 1 данных
6	D10	Бит 2 данных
7	D11	Бит 3 данных
8	D4	Бит 4 данных
9	D5	Бит 5 данных
10	D6	Бит 6 данных
11	D7	Бит 7 данных
12	A0	Выбор контроллера 2 экрана
13	A1	Выбор контроллера 1 экрана
14	RST	Сброс
15	A2	Чтение/запись

Контакт на плате ЖКИ	Контакт на плате Arduino	Назначение контакта на плате ЖКИ
16	A3	Данные/инструкции
17	A4	Разрешено
18	Внешний вывод подстроечного резистора с номиналом 10 кОм (другой внешний вывод подключается к контакту 5 V)	Выходное напряжение; регулировка контраста
19	Напряжение 5 В через резистор с сопротивлением 22 Ом	Анод светодиода подсветки (+)
20	GND	Катод светодиода подсветки (-)

Использование ЖКИ

После подключения ЖКИ можно загрузить и установить библиотеку поддержки графических ЖКИ для Arduino. Последняя версия библиотеки находится по адресу *http://code.google.com/p/glcd-arduino/downloads/list/*, и ее можно установить, прочитав описание в разделе «Расширение возможностей скетчей с помощью библиотек» в главе 8.

Теперь, чтобы задействовать ЖКИ, вставьте следующие строки перед функцией `void setup()`:

```
#include <glcd.h>                // подключить поддержку графических ЖКИ
#include "fonts/SystemFont5x7.h" // подключить стандартные шрифты
```

Затем добавьте следующие строки в функцию `void setup()`, чтобы настроить индикатор:

```
GLCD.Init(NON_INVERTED);    // используйте INVERTED, чтобы инвертировать
                            // включенные/выключенные пиксели
GLCD.SelectFont(System5x7); // выбрать шрифт для отображения текста
GLCD.ClearScreen();         // очистить экран ЖКИ
```

Управление индикатором

На экране ЖКИ умещается до восьми текстовых строк, по 20 символов в каждой. Позиционирование текстового курсора осуществляется следующей командой (замените `x` и `y` фактическими координатами):

```
GLCD.CursorTo(x, y);
```

Отображение текста осуществляется следующей командой (замените `text` желаемым текстом):

```
GLCD.Puts("text");
```

Отображение целых чисел осуществляется следующей командой (замените `number` требуемым числом):

```
GLCD.PrintNumber(number);
```

Проект № 26: Опробование текстовых функций в действии

В следующем скетче демонстрируется применение текстовых функций:

```
// Проект 26 - Опробование текстовых функций в действии
#include <glcd.h>                 // подключить поддержку графических ЖКИ
#include "fonts/SystemFont5x7.h" // подключить стандартные шрифты

int j = 7;

void setup()
{
  GLCD.Init(NON_INVERTED);
  GLCD.ClearScreen();
  GLCD.SelectFont(System5x7);
}

void loop()
{
  GLCD.ClearScreen();
  GLCD.CursorTo(1, 1);
  GLCD.Puts("Hello, world.");
  GLCD.CursorTo(1, 2);
  GLCD.Puts("I hope you are ");
  GLCD.CursorTo(1, 3);
  GLCD.Puts("enjoying this");
  GLCD.CursorTo(1, 4);
  GLCD.Puts("book. ");
  GLCD.CursorTo(1, 5);
  GLCD.Puts("This is from ");
  GLCD.CursorTo(1, 6);
  GLCD.Puts("chapter ");
  GLCD.PrintNumber(j);
  GLCD.Puts(".");
  do {} while (1);
}
```

Результатом выполнения скетча должен стать вывод, показанный на рис. 7.9.

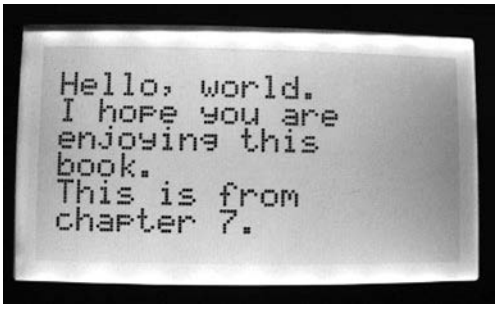


Рис. 7.9. Результаты работы проекта 26

Создание более сложных изобразительных эффектов

Теперь рассмотрим несколько функций для создания различных изобразительных эффектов. Имейте в виду, что экран графического ЖКИ имеет разрешение 128 × 64 пикселя, но в различных функциях пиксели нумеруются от 0 до 127 по горизонтали и от 0 до 63 по вертикали.

Графическая функция `SetDot()` включает единственный пиксель в позиции `x`, `y`, окрашивая его в цвет `BLACK` (черный), или выключает, окрашивая в цвет `WHITE` (белый). В параметре `color` можно передать только черный или белый цвет.

```
GLCD.SetDot(x, y, color); // color может иметь значение BLACK или WHITE
```

Следующая функция рисует прямоугольник шириной `w` и высотой `h`, верхний левый угол которого находится в точке с координатами `x`, `y`. Получившийся прямоугольник будет иметь черные границы и белый фон.

```
GLCD.DrawRect(x, y, w, h, color);
```

Следующая функция рисует закрашенный прямоугольник с теми же параметрами:

```
GLCD.FillRect(x, y, w, h, color);
```

Функция ниже рисует прямоугольник с теми же параметрами, но с закругленными углами, с радиусом `r`.

```
GLCD.DrawRoundRect(x, y, w, h, r, color);
```

Следующая функция рисует окружность с центром в точке `x`, `y` и с радиусом `r` пикселей:

```
GLCD.DrawCircle(x, y, r, color);
```

Следующая функция рисует вертикальную линию от точки `x`, `y` длиной `l` пикселей.

```
GLCD.DrawVertLine(x, y, l, color);
```

А следующая — горизонтальную линию от точки `x`, `y` длиной `l` пикселей.

```
GLCD.DrawHoriLine(x, y, l, color);
```

Вооружившись функциями, представленными выше, и собственным воображением, вы сможете создавать самые разные изобразительные эффекты или отображать данные в графическом представлении. В следующем разделе мы сконструируем быстродействующий термометр, использующий графический ЖКИ и некоторые из этих функций.

Проект № 27: Цифровой термометр с памятью

Наша цель в этом проекте — измерять температуру каждые 20 минут и отображать последние 100 замеров в виде графика. Каждый замер будет представлен пикселем с координатой Y, соответствующей температуре, и с координатой X, соответствующей времени. Самые свежие замеры отобразятся слева, и сам график будет прокручиваться слева направо с каждым новым замером. Текущее значение температуры будет также отображаться в числовом виде.

Алгоритм

Несмотря на кажущуюся сложность, этот проект довольно прост в реализации и требует всего двух функций. Первая принимает показания с датчика температуры TMP36 и сохраняет в массиве со 100 значениями. Каждый раз после выполнения нового замера результаты предыдущих 99 замеров смещаются в массиве вниз, чтобы освободить место для новых данных, и самые старые значения затираются следующими за ними. Вторая функция создаст изображение на экране ЖКИ. Ее задача — отобразить текущую температуру, определить масштаб для графика и вывести каждый замер в виде пикселя на экране, чтобы получить график изменения температуры во времени.

Оборудование

Ниже перечислено оборудование, которое понадобится для этого проекта:

- Один графический жидкокристаллический индикатор с размером экрана 128 × 64 пикселя, совместимый с интерфейсом KS0108B и имеющий колодку контактов для монтирования на макетной плате.
- Один подстроечный резистор с номиналом 10 кОм.
- Один датчик температуры TMP36.
- Несколько отрезков провода разной длины.

- Одна макетная плата.
- Плата Arduino и кабель USB.

Подключите графический ЖКИ в соответствии с табл. 7.1 и соедините датчик TMP36 с контактами 5 V, A5 и GND на плате Arduino.

Скетч

Введите и загрузите следующий скетч, предусмотрительно снабженный комментариями, описывающими его работу.

```
// Проект 27 - Цифровой термометр с памятью

#include <glcd.h>                // подключить поддержку графических ЖКИ
#include "fonts/SystemFont5x7.h" // подключить стандартные шрифты

int tcurrent;
int tempArray[100];

void setup()
{
  GLCD.Init(NON_INVERTED); // Настроить объект GLCD
  GLCD.ClearScreen();      // выключить все пиксели на экране
  GLCD.SelectFont(System5x7);
}

// Читает температуру с датчика TMP36
void getTemp()
{
  float sum = 0;
  float voltage = 0;
  float sensor = 0;
  float celsius;

  // прочитать значение с датчика и преобразовать
  // его в градусы Цельсия
  sensor   = analogRead(5);
  voltage  = (sensor * 5000) / 1024;
  voltage  = voltage - 500;
  celsius  = voltage / 10;
  tcurrent = int(celsius);

  // вставить новый замер в начало массива
  for (int a = 99 ; a >= 0 ; --a )
  {
    tempArray[a] = tempArray[a-1];
  }
  tempArray[0] = tcurrent;
}

// Выводит изображение на экран ЖКИ
```

```
void drawScreen()
{
  int q;
  GLCD.ClearScreen();
  GLCD.CursorTo(5, 0);
  GLCD.Puts("Current:");
  GLCD.PrintNumber(tcurrent);
  GLCD.CursorTo(0, 1);
  GLCD.PrintNumber(40);
  GLCD.CursorTo(0, 2);
  GLCD.PrintNumber(32);
  GLCD.CursorTo(0, 3);
  GLCD.PrintNumber(24);
  GLCD.CursorTo(0, 4);
  GLCD.PrintNumber(16);
  GLCD.CursorTo(1, 5);
  GLCD.PrintNumber(8);
  GLCD.CursorTo(1, 6);
  GLCD.PrintNumber(0);
  for (int a = 28 ; a < 127 ; a++)
  {
    q = (55 - tempArray[a-28]);
    GLCD.SetDot(a, q, BLACK);
  }
}

void loop()
{
  getTemp();
  drawScreen();
  for (int a = 0 ; a < 20 ; a++) // ждать 20 минут до следующего замера
  {
    delay(60000); // ждать 1 минуту
  }
}
```

Результат

В результате работы скетча на экране должно сформироваться изображение, напоминающее показанное на рис. 7.10.

Доработка скетча

Некоторые из нас лучше воспринимают информацию, если она представлена в графическом, а не в числовом виде. Проект такого вида можно использовать для отображения самых разных данных, таких как уровни напряжения от различных датчиков, измеряемые на контактах аналоговых входов. Также можно добавить еще один датчик температуры и отображать сразу два графика. Практически все, что имеет числовое выражение, можно отобразить на экране графического ЖКИ.

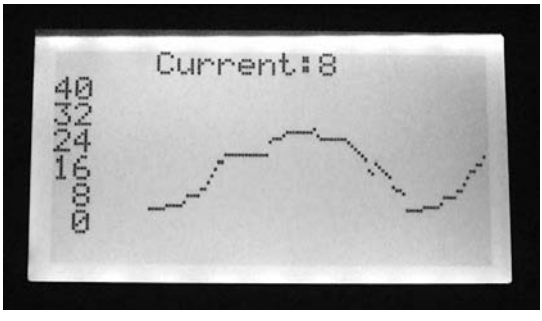


Рис. 7.10. Результаты работы проекта 27

Забегая вперед

Теперь, после знакомства с жидкокристаллическими индикаторами, более наглядным стал тот факт, что плата Arduino является маленьким компьютером: она может принимать и обрабатывать данные и отображать результаты. Но это только начало. В следующей главе мы создадим собственную макетную плату расширения для Arduino, научимся сохранять данные на карте памяти microSD и познакомимся с библиотеками и функциями Arduino для хронометража.

8 Расширение Arduino

В этой главе вы:

- ✓ познакомитесь с большим разнообразием плат расширения для Arduino;
- ✓ создадите собственную макетную плату расширения на основе платы ProtoShield;
- ✓ узнаете, как библиотеки для Arduino могут расширять набор доступных функций;
- ✓ научитесь пользоваться платой расширения с картой памяти microSD для записи данных, которые затем можно проанализировать с помощью электронной таблицы;
- ✓ сконструируете устройство регистрации температуры;
- ✓ узнаете, как реализовать секундомер с использованием функций `micros()` и `millis()`;
- ✓ познакомитесь с прерываниями в Arduino и научитесь использовать их.

В этой главе мы продолжим исследование способов расширения возможностей платы Arduino (с использованием разнообразных плат расширения), и вы на практике узнаете, как создать свою плату расширения. Со временем, обретя опыт в экспериментах с электроникой и Arduino, вы научитесь создавать более долговечные конструкции, собирая их на *ProtoShield* — пустой макетной печатной плате, предназначенной для создания собственной схемы.

Одной из наиболее полезных плат расширения является плата с устройством для чтения/записи карт памяти *microSD*. На ее основе мы в этой главе создадим устройство регистрации температуры, записывающее замеры температуры на карту памяти в течение длительного времени; данные, записанные с помощью платы расширения, в дальнейшем можно перенести куда угодно для последующего анализа.

Вы познакомитесь с функциями `micros()` и `millis()` — очень удобными инструментами измерения интервалов времени, которые будут использованы в проекте реализации секундомера. В заключение мы займемся исследованием прерываний.

Платы расширения

Расширить функциональные возможности платы Arduino можно с помощью дополнительных плат расширения. Плата расширения — это печатная плата, подключаемая контактами к разъемам, расположенным по краям платы Arduino. На рынке существуют сотни таких плат. Один из популярных проектов, например, объединяет плату расширения GPS (определения географических координат) с платой расширения, содержащей карту памяти microSD, в устройство, регистрирующее и сохраняющее историю изменения координат с течением времени, например путь, пройденный автомобилем или пешим путешественником. Из других проектов следует упомянуть сетевые адаптеры Ethernet, позволяющие подключить Arduino к Интернету (рис. 8.1).

Рис. 8.1. Плата расширений Ethernet, подключенная к Arduino Uno

Приемники сигналов от спутников GPS позволяют определять местоположение Arduino (рис. 8.2). Интерфейсы с картами памяти microSD дают возможность сохранять данные на карте памяти (рис. 8.3).

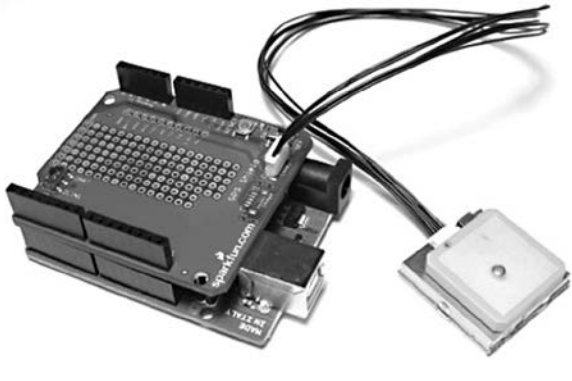


Рис. 8.2. Плата приемника GPS (с отдельным модулем GPS), подключенная к Arduino Uno

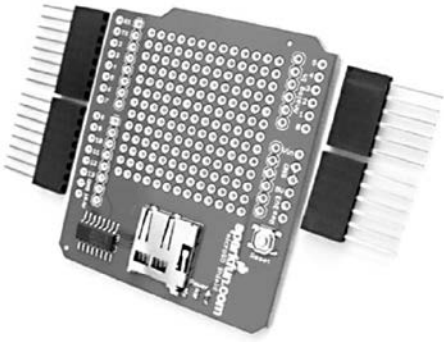


Рис. 8.3. Плата с держателем для карт памяти microSD

Платы расширения для Arduino могут соединяться в стек друг с другом и обычно поддерживают совместную работу с другими платами расширения. Например, на рис. 8.4 показана плата Arduino Uno с подключенной платой расширения microSD, предназначенной для хранения данных, платой расширения Ethernet для доступа к Интернету и платой расширения с жидкокристаллическим индикатором для отображения информации.

Рис. 8.4. Три платы расширения, подключенные к Arduino Uno стопкой

ВНИМАНИЕ

При соединении плат расширения в стек убедитесь, что они не используют одни и те же цифровые или аналоговые входы/выходы. Попытка использовать одни и те же входы/выходы для разных функций может повредить всю конструкцию, поэтому будьте осторожны. Производители плат расширения обычно сопровождают свои изделия информацией об используемых ими входах/выходах. Исчерпывающий перечень плат расширения и используемых ими входов/выходов, поддерживаемый сообществом, можно найти по адресу *http://www.shieldlist.org/*.

Макетные платы ProtoShield

Платы расширения можно приобрести в интернет-магазинах (например, в *http://www.shieldlist.org/*) или создать самим на основе макетной платы ProtoShield. Плата ProtoShield продается в собранном виде или в виде конструктора, как показано на рис. 8.5. Кроме того, ProtoShield может служить отличной основой для макетирования без паяльника, она позволяет размещать маленькие схемы в физических границах Arduino (например, быстродействующий термометр, как показано на рис. 8.6). С помощью клея-геля Blu-Tack или двусторонней клейкой ленты к печатной плате можно прикрепить небольшой модуль для временного монтажа, умещающийся между рядами разъемов. Платы ProtoShield также используются в качестве основы для сборки постоянных схем, предварительно опробованных на макетной плате.

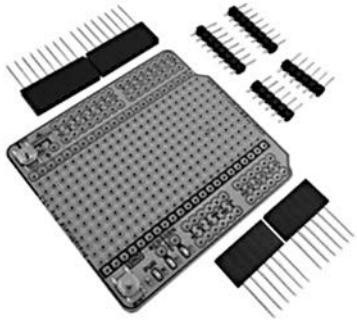


Рис. 8.5. Конструктор ProtoShield

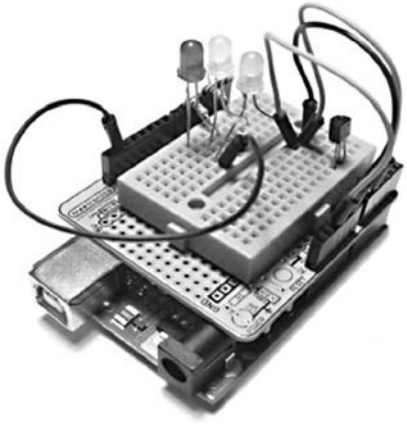


Рис. 8.6. Быстродействующий термометр из проекта 8

Конструирование устройств на плате ProtoShield включает этап предварительного и специального планирования, в том числе проектирование принципиальной схемы и схемы размещения компонентов на плате. Наконец, законченное устройство можно спаять на плате, но перед этим обязательно следует проверить его на макетной плате, чтобы убедиться, что оно работает. Для некоторых плат ProtoShield имеются специальные файлы PDF с рисунком платы, которые можно загрузить, напечатать и использовать для рисования схем к своим проектам.

Проект № 28: Собственная плата расширения с восемью светодиодами

В этом проекте мы создадим собственную плату расширения с восемью светодиодами и ограничивающими резисторами. Эта плата расширения предназначена для упрощения экспериментов со светодиодами, подключенными к цифровым выходам.

Оборудование

Ниже перечислено, что понадобится для этого проекта:

- Одна пустая плата Arduino ProtoShield.
- Восемь светодиодов любого цвета.
- Восемь резисторов с номиналом 560 Ом (R1—R8).
- Две колодки с 6 контактами в каждой.
- Две колодки с 8 контактами в каждой.

Схема

Принципиальная схема устройства изображена на рис. 8.7.

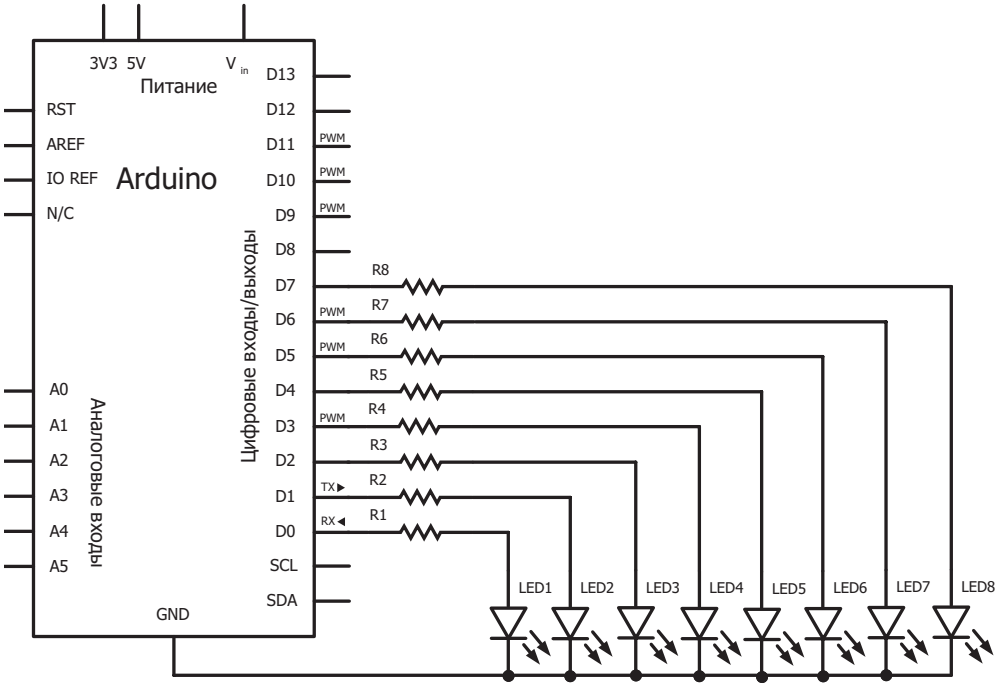


Рис. 8.7. Принципиальная схема для проекта 28

Топология макетной платы ProtoShield

Следующий наш шаг — знакомство с топологией отверстий на плате ProtoShield. Ряды и столбцы отверстий на плате ProtoShield должны совпадать с отверстиями на макетной плате для навесного монтажа. На пустой плате ProtoShield, изо-

браженной на рис. 8.8, проектировщики окружили черными линиями отверстия, электрически соединенные между собой.

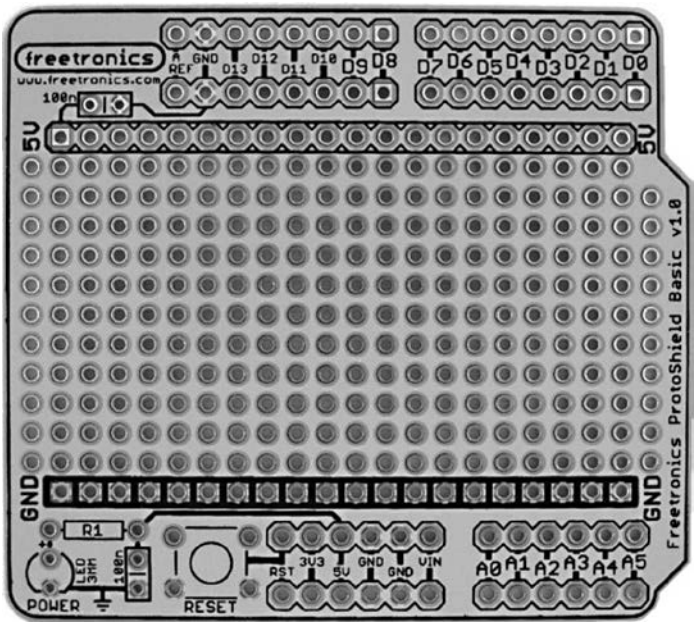


Рис. 8.8. Пустая плата Blank ProtoShield, вид сверху

Длинная горизонтальная линия ниже ряда контактов цифровых входов/выходов соединяется с разъемом питания 5V, а горизонтальная линия выше ряда контактов аналоговых входов/выходов и контактов питания соединяется с разъемом GND. Отверстия между этими двумя линиями электрически изолированы друг от друга. Наконец, на плате имеется еще два ряда отверстий: один расположен вдоль верхнего края, а другой — вдоль нижнего. Верхний ряд разбит на две группы по восемь отверстий в каждой, а нижний — на две группы по шесть отверстий. В эти отверстия впаиваются колодки контактов для подключения к плате Arduino.

Проектирование

Нарисуйте план размещения компонентов на линованной бумаге, как показано на рис. 8.9.

Когда план будет готов, проверьте, можно ли разместить фактические электронные компоненты на плате ProtoShield, как вы задумали, и не получилась ли компоновка чрезмерно плотной. Если на плате ProtoShield останется место для кнопки сброса, обязательно добавьте ее, потому что после подключения платы расширения к разъемам на плате Arduino доступ к стандартной кнопке сброса будет закрыт.

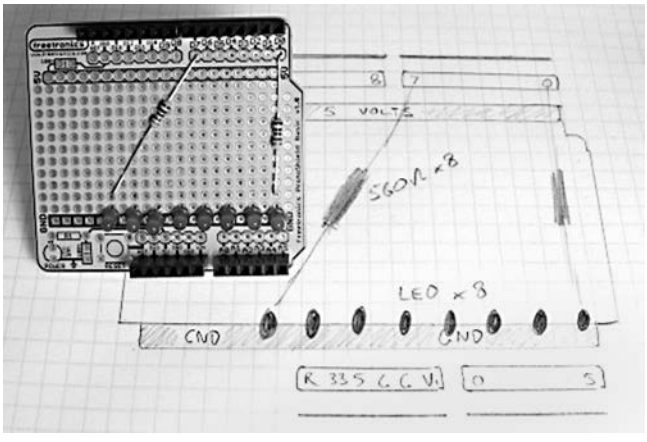


Рис. 8.9. Планирование будущей платы расширения

Пайка компонентов

После того как удовлетворяющее вас расположение компонентов на плате ProtoShield будет найдено и проверка работоспособности схемы проведена, приступайте к пайке компонентов. Пользоваться паяльником совсем несложно, и не стремитесь покупать дорогостоящие паяльные станции — для наших проектов вполне достаточно простого паяльника мощностью 25–40 Вт, такого как на рис. 8.10.

Рис. 8.10. Паяльник

ПРИМЕЧАНИЕ

Если прежде вам не приходилось пользоваться паяльником, к вашим услугам иллюстрированная инструкция: *http://mightyohm.com/soldercomic/*.[1]

[1] Перевод на русский язык можно найти по адресу *https://geektimes.ru/post/256784/*. — *Примеч. пер.*

В процессе пайки компонентов иногда приходится соединять контакты небольшой каплей припоя, как показано на рис. 8.11. Как можно видеть на принципиальной схеме, один конец каждого резистора соединяется с анодом соответствующего светодиода.

Проверяйте каждый спаянный контакт по мере продвижения вперед, потому что ошибки проще найти и исправить до того, как проект будет закончен. Когда наступит черед пайки четырех колодок с контактами, зафиксируйте их в разъемах другой платы расширения, чтобы избежать смещений, как показано на рис. 8.12.

Рис. 8.11. Соединение контактов каплей припоя

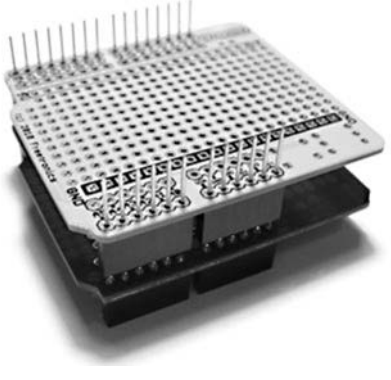


Рис. 8.12. Пайка колодок с контактами

На рис. 8.13 показано готовое устройство: наша собственная плата расширения для Arduino с восемью светодиодами.

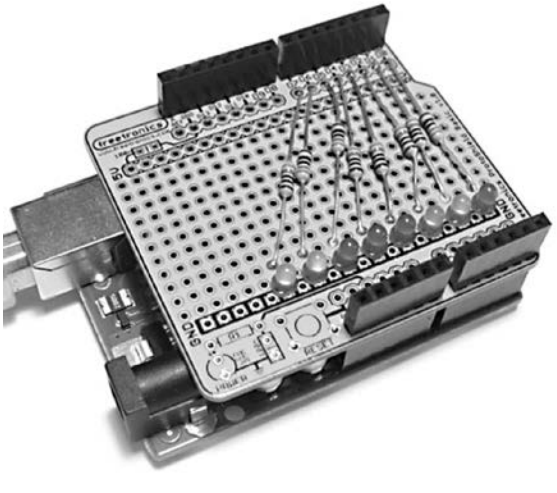


Рис. 8.13. Законченная плата расширения!

Доработка платы расширения

Эту простую плату можно использовать и для создания более сложного проекта — например, для мониторинга состояния цифровых контактов с 0 по 7. Если добавить еще шесть резисторов и светодиодов, мы сможем осуществлять мониторинг всего набора цифровых выходов. Существует масса вариантов использования этой платы расширения. Просто включите воображение!

Расширение возможностей скетчей с помощью библиотек

Так же как платы расширения Arduino расширяют функциональные возможности оборудования, *библиотеки* могут добавлять полезные функции для решения определенных задач в скетчах или для использования дополнительного оборудования, устанавливаемого разными производителями. Любой желающий может создать свою библиотеку, как это часто делают производители различных плат расширения Arduino для поддержки своего оборудования.

Среда разработки Arduino IDE уже включает множество предустановленных библиотек. Чтобы подключить их к своим скетчам, достаточно выбрать в меню пункт Sketch ▸ Import Library (Скетч ▸ Подключить библиотеку), и вы увидите список предустановленных библиотек с такими именами, как Ethernet, LiquidCrystal, Servo и др. Многие имена говорят сами за себя. (Далее, если для работы по проекту потребуется некоторая библиотека, она будет описана более подробно.)

Импортирование библиотек поддержки плат расширения

После приобретения платы расширения обычно требуется загрузить и установить библиотеку для ее поддержки с сайта производителя или по указанной ссылке.

Чтобы показать, как это делается, давайте загрузим библиотеку, необходимую для использования платы расширения microSD, изображенной на рис. 8.3.

1. Откройте страницу *https://github.com/greiman/SdFat/* и щелкните на кнопке Download ZIP (Загрузить ZIP-архив). На рис. 8.14 показано, как выглядит эта страница.
2. Через несколько мгновений в папке *Downloads* (Загрузки) появится файл *SdFat-master.zip*. Выполните двойной щелчок на нем, чтобы распаковать архивную папку, содержащую файлы, как показано на рис. 8.15.
3. Распаковав библиотеку, следуйте инструкциям, приведенным ниже, чтобы установить ее в своей операционной системе.

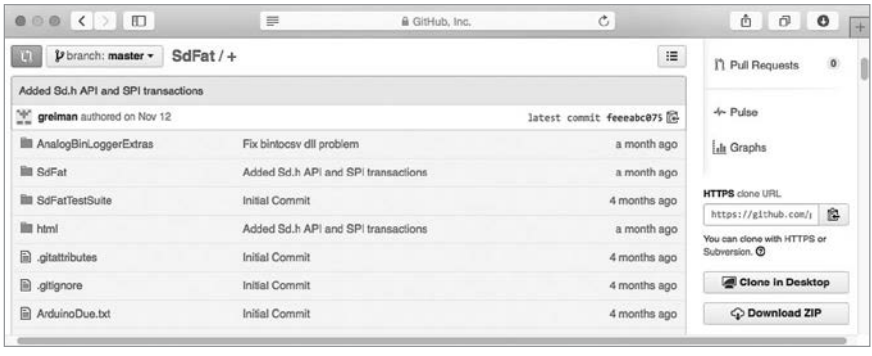


Рис. 8.14. Страница загрузки библиотеки

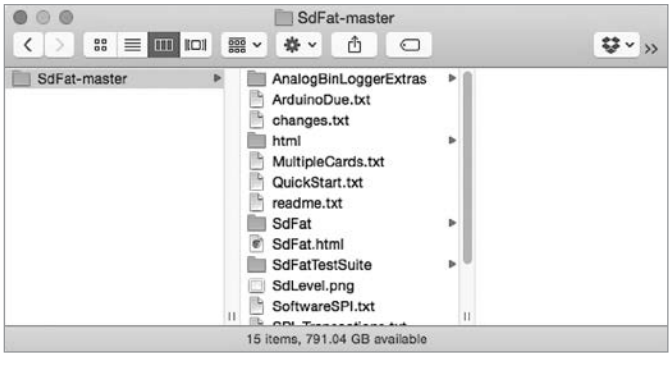


Рис. 8.15. Папка с библиотекой

Установка библиотеки в Mac OS X

Если вы пользуетесь операционной системой Mac OS X, следуйте приведенным ниже инструкциям:

1. Откройте папку *Downloads* (Загрузки) и найдите папку, извлеченную из загруженного файла, как показано на рис. 8.16.

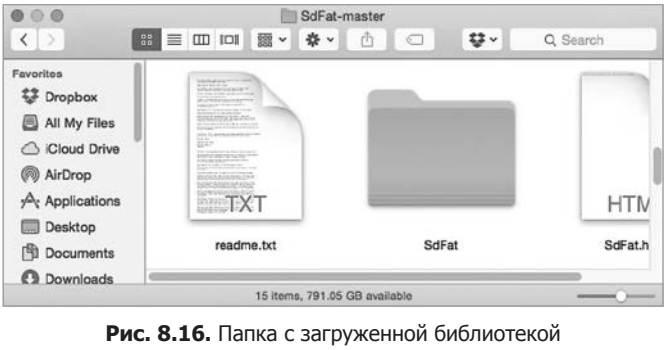


Рис. 8.16. Папка с загруженной библиотекой

2. Откройте Arduino IDE и выберите пункт меню Sketch ▸ Import Library ▸ Add Library (Скетч ▸ Подключить библиотеку ▸ Добавить библиотеку), как показано на рис. 8.17.

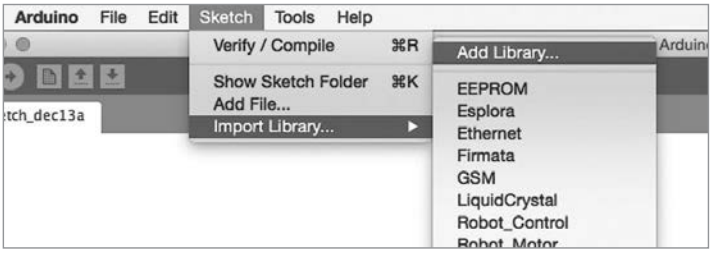


Рис. 8.17. Импортирование библиотеки

3. Когда появится диалог, изображенный на рис. 8.18, выберите папку *SdFat-master*, находящуюся в папке *Downloads* (Загрузки), и щелкните на кнопке Choose (Выбрать).

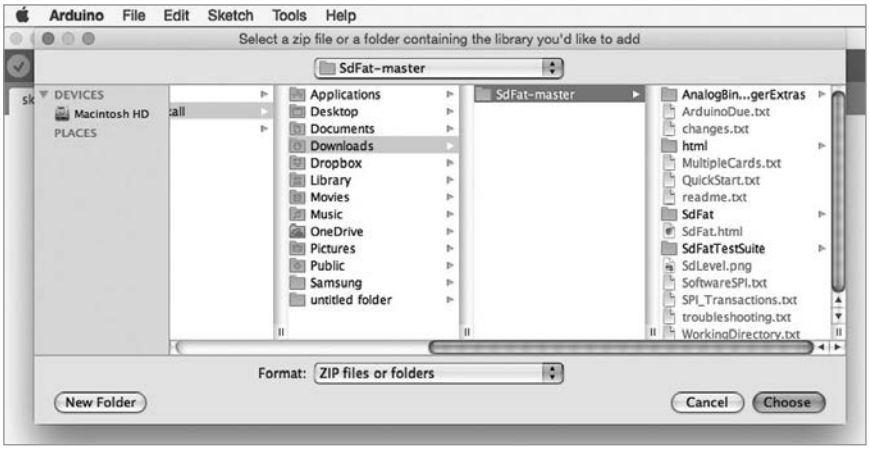


Рис. 8.18. Выбор папки библиотеки

4. Проверьте доступность библиотеки *SdFat* после установки, перезапустив IDE и выбрав пункт меню Sketch ▸ Import Library (Скетч ▸ Подключить библиотеку). В списке должен появиться пункт SdFat. (Если этого не произошло, попробуйте повторить процедуру установки.)

Если все прошло благополучно, переходите к разделу «Карты памяти microSD», далее в этой главе.

Установка библиотеки в Windows XP и более поздних версиях

Если вы пользуетесь операционной системой Windows XP или более поздней версией, следуйте приведенным ниже инструкциям:

1. Когда загрузка завершится, откройте загруженный архив и найдите в нем папку *SdFat*, как показано на рис. 8.19.

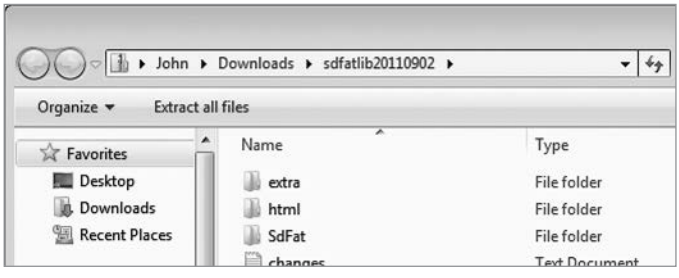


Рис. 8.19. Папка с загруженной библиотекой

2. Скопируйте папку *SdFat* из окна, изображенного на рис. 8.19, в папку *Arduino/libraries*, как показано на рис. 8.20.

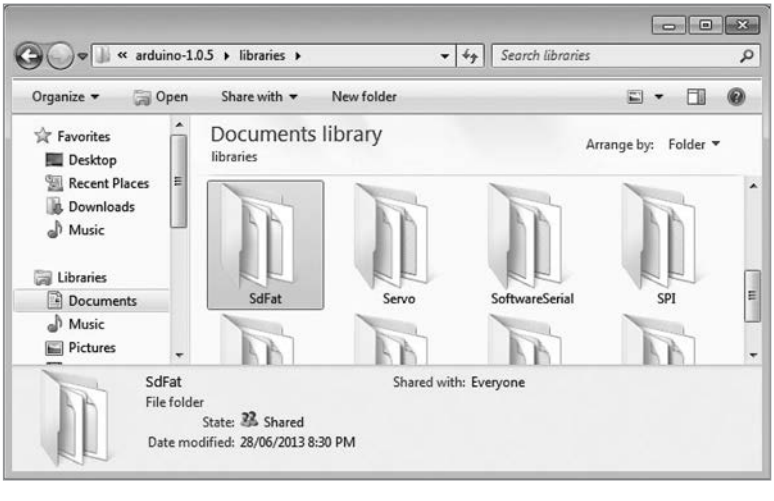


Рис. 8.20. Завершение установки библиотеки

3. Проверьте доступность библиотеки *SdFat* после установки, перезапустив IDE и выбрав пункт меню Sketch ▸ Import Library (Скетч ▸ Подключить библиотеку). В списке должен появиться пункт SdFat. (Если этого не произошло, попробуйте повторить процедуру установки.)

Если все прошло благополучно, переходите к разделу «Карты памяти microSD», далее в этой главе.

Установка библиотеки в Ubuntu Linux 11.04 и выше

Если вы пользуетесь операционной системой Ubuntu Linux 11.04 или выше, следуйте приведенным ниже инструкциям:

1. Найдите загруженный файл и выполните двойной щелчок на нем. В результате должно появиться окно Archive Manager (Диспетчер архивов) с папкой *SdFat*, как показано на рис. 8.21.

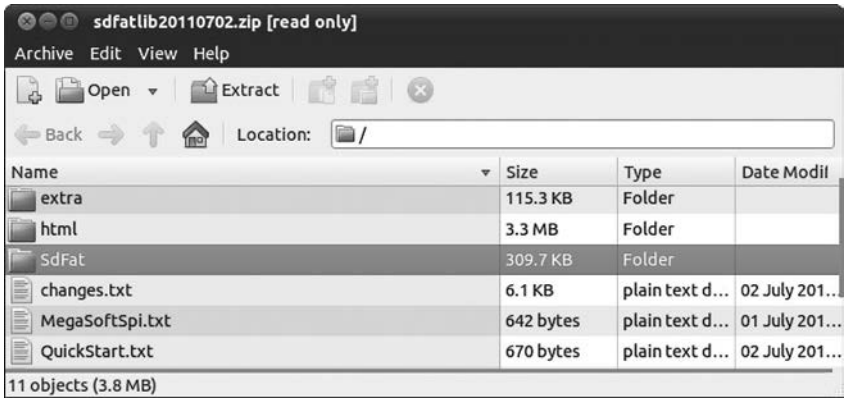


Рис. 8.21. Папка с загруженной библиотекой

2. Щелкните правой кнопкой на папке *SdFat* в окне, изображенном на рис. 8.21, и затем на кнопке Extract (Извлечь), чтобы извлечь папку в ваш каталог */libraries* (рис. 8.22).

Рис. 8.22. Выбор места назначения для извлечения папки с библиотекой

3. Проверьте доступность библиотеки *SdFat* после установки, перезапустив IDE и выбрав пункт меню Sketch ▸ Import Library (Скетч ▸ Подключить библиотеку). В списке должен появиться пункт SdFat. (Если этого не произошло, попробуйте повторить процедуру установки.)

Затем переходите к следующему разделу.

Карты памяти microSD

Используя карты памяти microSD с платой Arduino, можно организовать сохранение данных, поступающих из множества источников, таких как температурный датчик TMP36 из главы 4. На карте памяти microSD можно также хранить данные для веб-сервера или любые другие файлы, используемые проектом. Для записи и хранения данных можно использовать плату расширения с картой памяти microSD, изображенной на рис. 8.23. Плата расширения поддерживает карты памяти microSD (не microSDHC) емкостью до 2 ГБ.

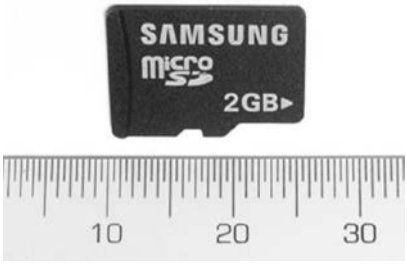


Рис. 8.23. Карта microSD емкостью 2 ГБ

Платы расширения с картами памяти выпускают многие известные производители, такие как SparkFun, Adafruit Industries и Snootlab. В этой книге будет использоваться плата (изображена на рис. 8.3) от компании SparkFun (артикул DEV-09802 и PRT-10007).

ПРИМЕЧАНИЕ

Прежде чем использовать карту памяти с платой расширения, ее необходимо отформатировать. Для этого подключите карту памяти к компьютеру и следуйте инструкциям к вашей операционной системе по форматированию карт памяти. Используйте формат FAT16. Кроме того, в зависимости от модели платы расширения может потребоваться впаять в плату колодку с контактами, в этом случае следуйте процедуре, описанной в разделе «Пайка компонентов», приведенном выше.

Тестирование карты microSD

Завершив форматирование карты и сборку платы расширения, проверьте ее работоспособность. Для этого выполните следующие шаги:

1. Подключите плату расширения к плате Arduino, вставьте карту памяти в разъем и подключите Arduino к компьютеру кабелем USB.
2. Запустите IDE и выберите пункт меню File ▸ Examples ▸ SdFat ▸ SdInfo (Файл ▸ Примеры ▸ SdFat ▸ SdInfo). Затем загрузите скетч *SdInfo*, открытый в IDE.
3. Откройте окно Serial Monitor (Монитор порта), установите скорость обмена 9600 бод, нажмите любую клавишу на клавиатуре и затем нажмите ENTER. Спустя мгновение в области вывода появится информация о карте microSD, как показано на рис. 8.24.

Если в мониторе порта ничего не появилось, то выполните следующие действия, — скорее всего, они решат вашу проблему:

- Отключите кабель USB от платы Arduino, извлеките и вновь вставьте карту microSD в разъем.
- Проверьте, правильно ли припаяна колодка с контактами и нет ли замыканий между контактами.
- Проверьте, установлена ли скорость 9600 бод в окне монитора порта и убедитесь, что используемая вами плата Arduino совместима с обычной Arduino Uno. Модели Mega и некоторые другие имеют иное расположение контактов интерфейса SPI, посредством которого осуществляется обмен данными с платами расширения.
- Отформатируйте карту microSD еще раз.

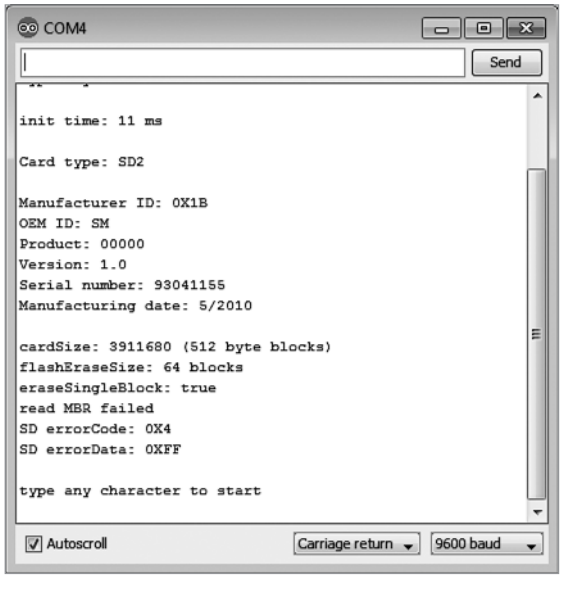


Рис. 8.24. Результаты успешного тестирования карты microSD

Проект № 29: Запись данных на карту памяти

Для записи данных на карту памяти подключите плату расширения, вставьте карту microSD в разъем, введите и загрузите следующий скетч:

```
// Проект 29 - Запись данных на карту памяти

int b = 0;

#include <SD.h>

void setup()
{
  Serial.begin(9600);
  Serial.print("Initializing SD card...");
  pinMode(10, OUTPUT);

  // проверить наличие и работоспособность карты microSD
  if (!SD.begin(8))
  {
    Serial.println("Card failed, or not present");
    // остановить скетч
    return;
  }
  Serial.println("microSD card is ready");
}

void loop()
{
❶ // создать файл для записи
  File dataFile = SD.open("DATA.TXT", FILE_WRITE);
  // если файл готов, записать в него данные:
  if (dataFile)
❷ {
    for ( int a = 0 ; a < 11 ; a++ )
    {
      dataFile.print(a);
      dataFile.print(" multiplied by two is ");
      b = a * 2;
❸      dataFile.println(b, DEC);
❹    }
❺    dataFile.close(); // закрыть файл, как только система
                       //завершит запись (обязательно)
  }
  // если файл не готов, сообщить об ошибке:
  else
  {
    Serial.println("error opening DATA.TXT");
❻  }
  Serial.println("finished");
  do {} while (1);
}
```

Скетч создаст на карте памяти microSD текстовый файл с именем *DATA.txt*, содержимое которого показано на рис. 8.25.

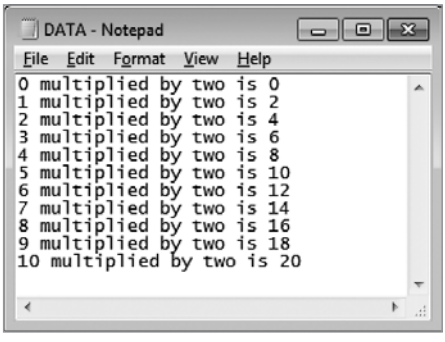


Рис. 8.25. Результаты работы проекта 29

Давайте исследуем функцию `void loop()` в скетче и посмотрим, как она создает текстовый файл. Код в функции `void loop()` между ❶ и ❷ и между ❹ и ❻ создает файл и открывает его для записи. Запись в текстовый файл осуществляется вызовом `dataFile.print()` или `dataFile.println()`.

Эти функции можно использовать по аналогии, например, с `Serial.println()`, то есть запись в файл осуществляется точно так же, как запись в монитор порта. В ❶ определяется имя создаваемого файла, длиной не более восьми символов, за которым должна следовать точка и три символа расширения, как в имени *DATA.txt*.

В ❸ передается второй параметр `DEC`. Он сообщает, что переменная `b` хранит десятичное число, которое должно быть записано в текстовый файл. При записи вещественной переменной (типа `float`) можно указать число десятичных знаков для записи (не более шести).

По завершении записи данных в файл (❺) вызывается `dataFile.close()`, чтобы закрыть файл. Если этот шаг опустить, компьютер не сможет прочитать созданный текстовый файл.

Проект № 30: Устройство регистрации температуры

Теперь, когда вы знаете, как записывать данные, попробуем с помощью датчика TMP36, представленного в главе 4, измерять температуру каждую минуту в течение 8 часов и записывать результаты на карту microSD. Для этого объединим функции записи на карту microSD из проекта 29 с функцией измерения температуры из проекта 27.

Оборудование

Для этого проекта понадобится:

- один температурный датчик TMP36;
- одна макетная плата;
- несколько отрезков провода разной длины;
- плата расширения с картой памяти microSD;
- плата Arduino и кабель USB.

Вставьте карту microSD в разъем на плате расширения, а затем подключите плату расширения к плате Arduino. Соедините левый вывод датчика TMP36 с контактом 5V на плате Arduino, средний вывод — с аналоговым входом и правый вывод — с контактом GND.

Скетч

Введите и загрузите следующий скетч:

```
// Проект 30 - Устройство регистрации температуры

#include <SD.h>

float sensor, voltage, celsius;

void setup()
{
  Serial.begin(9600);
  Serial.print("Initializing SD card...");
  pinMode(10, OUTPUT);

  // проверить наличие и работоспособность карты microSD
  if (!SD.begin(8))
  {
    Serial.println("Card failed, or not present");
    // остановить скетч
    return;
  }
  Serial.println("microSD card is ready");
}

void loop()
{
  // создать файл для записи
  File dataFile = SD.open("DATA.TXT", FILE_WRITE);
  // если файл готов, записать в него данные:
  if (dataFile)
  {
```

```
    for ( int a = 0 ; a < 481 ; a++ ) // 480 минут, или 8 часов
    {
      sensor = analogRead(0);
      voltage = (sensor * 5000) / 1024; // преобразовать в милливольты
      voltage = voltage - 500;
      celsius = voltage / 10;
      dataFile.print(" Log: ");
      dataFile.print(a, DEC);
      dataFile.print(" Temperature: ");
      dataFile.print(celsius, 2);
      dataFile.println(" degrees C");
      delay(599900); // ждать примерно одну минуту
    }
    dataFile.close(); // обязательно
    Serial.println("Finished!");
    do {} while (1);
  }
}
```

До завершения работы скетча потребуется чуть больше 8 часов, но вы можете изменить продолжительность периода, уменьшив значение в вызове `delay(599900)`.

После завершения скетча извлеките карту microSD и откройте файл в текстовом редакторе, как показано на рис. 8.26.

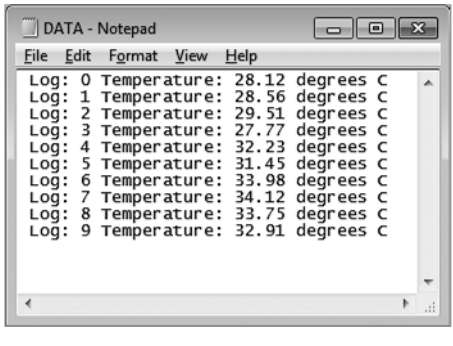
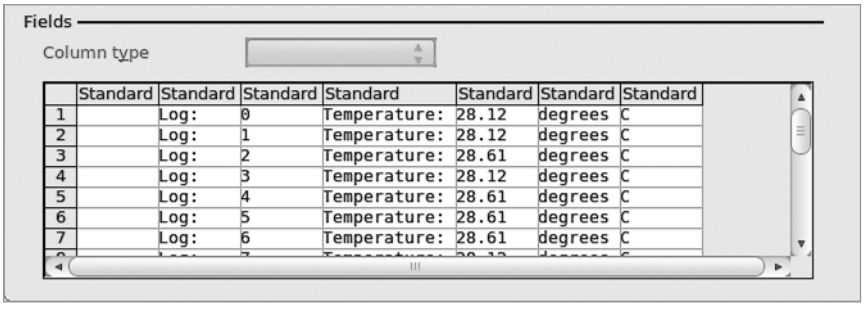


Рис. 8.26. Результаты работы проекта 30

Для более серьезного анализа сохраненных данных разделите текстовые строки пробелами или двоеточиями, чтобы упростить импортирование файла в электронную таблицу. Например, файл можно импортировать в OpenOffice Calc, выбрав пункт меню Insert ▸ Sheet From File (Вставка ▸ Лист из файла), в результате вы получите электронную таблицу, как показано на рис. 8.27, с помощью которой затем можно проанализировать данные, как показано на рис. 8.28.

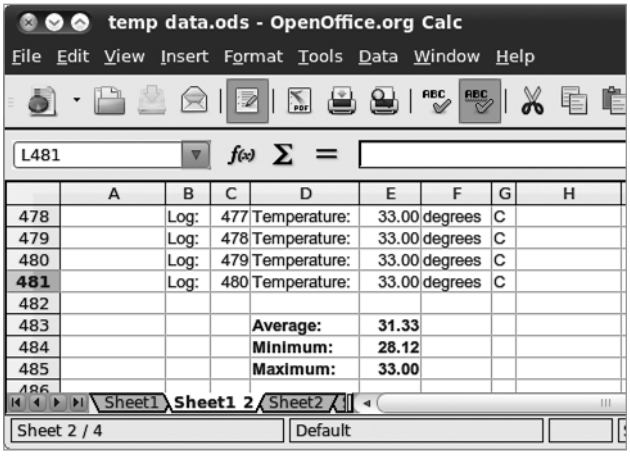
Примеры с измерением температуры можно приспособить для собственных проектов анализа данных. Рассмотренные принципы применимы для записи любых данных, которые только способна сгенерировать система Arduino.


Fields

Column type

	Standard	Standard	Standard	Standard	Standard	Standard	Standard
1		Log:	0	Temperature:	28.12	degrees	C
2		Log:	1	Temperature:	28.12	degrees	C
3		Log:	2	Temperature:	28.61	degrees	C
4		Log:	3	Temperature:	28.12	degrees	C
5		Log:	4	Temperature:	28.61	degrees	C
6		Log:	5	Temperature:	28.61	degrees	C
7		Log:	6	Temperature:	28.61	degrees	C

Рис. 8.27. Данные, импортированные в электронную таблицу


temp data.ods - OpenOffice.org Calc

File Edit View Insert Format Tools Data Window Help

L481

	A	B	C	D	E	F	G	H
478		Log:	477	Temperature:	33.00	degrees	C	
479		Log:	478	Temperature:	33.00	degrees	C	
480		Log:	479	Temperature:	33.00	degrees	C	
481		Log:	480	Temperature:	33.00	degrees	C	
482								
483				Average:	31.33			
484				Minimum:	28.12			
485				Maximum:	33.00			

Sheet1 Sheet1 2 Sheet2

Sheet 2 / 4 Default

Рис. 8.28. Анализ результатов измерения температуры

Хронометраж с применением millis() и micros()

Каждый раз, когда Arduino запускает скетч, она также начинает фиксировать ход времени в миллисекундах и микросекундах. Миллисекунда — это одна тысячная (0,001) секунды, а микросекунда — одна миллионная (0,000001) секунды. Эти значения можно использовать в скетчах для измерения интервалов времени.

Следующие функции возвращают значения времени, хранящиеся в переменных типа `unsigned long`:

```
unsigned long a,b;
a = micros();
b = millis();
```

Из-за ограничений типа `unsigned long` значение сбрасывается в 0 по достижении 4 294 967 295, что составляет примерно 50 дней для `millis()` и 70 минут

для `micros()`. Кроме того, из-за ограничений микропроцессора на плате Arduino `micros()` всегда возвращает значения, кратные четырем.

Давайте воспользуемся этими функциями для определения, сколько времени требуется плате Arduino, чтобы перевести цифровой выход из состояния `LOW` в состояние `HIGH` и обратно. Для этого мы прочитаем результат `micros()` до и после вызова функции `digitalWrite()`, определим разность и выведем ее в монитор порта. Для этого эксперимента нам потребуется только плата Arduino и кабель USB.

Введите и загрузите скетч из листинга 8.1.

Листинг 8.1. Хронометраж изменения состояния цифрового выхода с помощью micros()

```
// Листинг 8.1

unsigned long start, finished, elapsed;

void setup()
{
  Serial.begin(9600);
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
}

void loop()
{
❶   start = micros();
    digitalWrite(3, HIGH);
❷   finished = micros();
❸   elapsed = finished – start;
    Serial.print("LOW to HIGH: ");
    Serial.print(elapsed);
    Serial.println(" microseconds");
    delay(1000);

❹   start = micros();
    digitalWrite(3, LOW);
    finished = micros();
    elapsed = finished - start;
    Serial.print("HIGH to LOW: ");
    Serial.print(elapsed);
    Serial.println(" microseconds");
    delay(1000);
}
```

Скетч получает отметки времени с помощью `micros()` до и после вызова функции `digitalWrite(HIGH)` (❶ и ❷), вычисляет разность и выводит ее в монитор порта (❸). Затем операции повторяются для измерения времени переключения цифрового выхода в прежнее состояние (❹).

Теперь откройте окно монитора порта, чтобы увидеть результаты, как показано на рис. 8.29.

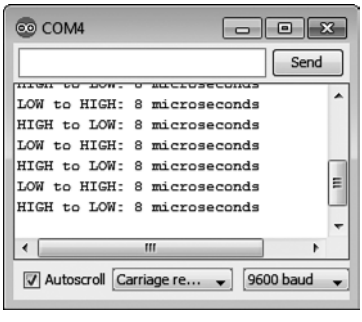


Рис. 8.29. Результаты работы скетча из листинга 8.1

Поскольку точность измерения составляет 4 микросекунды, полученная продолжительность 8 микросекунд означает, что реальная продолжительность больше 4 микросекунд и меньше или равна 8.

Проект № 31: Секундомер

Теперь, когда мы научились измерять интервалы времени между двумя событиями, сконструируем на основе Arduino простой секундомер. В нашем секундомере будут использоваться две кнопки: одна запускает отсчет (то есть сбрасывает счетчик), а другая останавливает отсчет и выводит прошедшее время. Скетч будет постоянно проверять состояние двух кнопок. В момент нажатия на кнопку запуска он сохранит значение `millis()`, а в момент нажатия кнопки останова — сохранит новое значение `millis()`. Наша собственная функция `displayResult()` преобразует измеренный интервал времени из миллисекунд в часы, минуты и секунды. Наконец, преобразованное время отобразится в окне монитора порта.

Оборудование

Ниже перечислено, что понадобится для этого проекта:

- Одна макетная плата.
- Две кнопки без фиксации (S1 и S2).
- Два резистора с номиналом 10 кОм (R1 и R2).
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Принципиальная схема для проекта изображена на рис. 8.30.

ПРИМЕЧАНИЕ

Эта схема потребуется нам в следующем проекте, поэтому не разбирайте ее, завершив эксперименты с этим проектом.

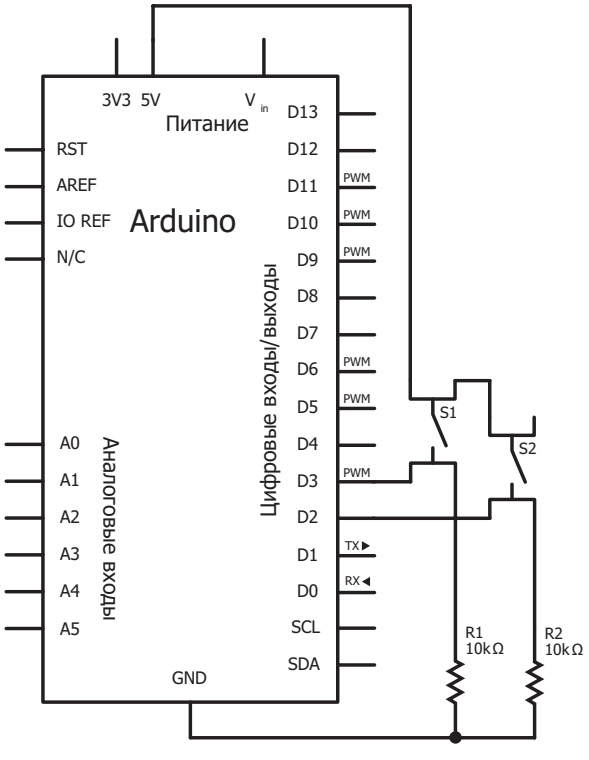


Рис. 8.30. Принципиальная схема для проекта 31

Скетч

Введите и загрузите следующий скетч:

```
// Проект 31 – Секундомер

unsigned long start, finished, elapsed;

void setup()
{
```

```
    Serial.begin(9600);
❶   pinMode(2, INPUT); // кнопка запуска
    pinMode(3, INPUT); // кнопка останова
    Serial.println("Press 1 for Start/reset, 2 for elapsed time");
  }

  void displayResult()
  {
    float h, m, s, ms;
    unsigned long over;

❷   elapsed = finished - start;

    h    = int(elapsed / 3600000);
    over = elapsed % 3600000;
    m    = int(over / 60000);
    over = over % 60000;
    s    = int(over / 1000);
    ms   = over % 1000;

    Serial.print("Raw elapsed time: ");
    Serial.println(elapsed);
    Serial.print("Elapsed time: ");
    Serial.print(h, 0);
    Serial.print("h ");
    Serial.print(m, 0);
    Serial.print("m ");
    Serial.print(s, 0);
    Serial.print("s ");
    Serial.print(ms, 0);
    Serial.println("ms");
    Serial.println();
  }

  void loop()
  {
❸   if (digitalRead(2) == HIGH)
    {
      start = millis();
      delay(200); // защита от дребезга контактов
      Serial.println("Started...");
    }
❹   if (digitalRead(3) == HIGH)
    {
      finished = millis();
      delay(200); // защита от дребезга контактов
      displayResult();
    }
  }
```

Наш секундомер имеет простую реализацию. В ❶ выполняется настройка контактов на работу в режиме цифровых входов, к которым подключены кнопки запуска

и останова. В ❷, если нажата кнопка запуска, Arduino запоминает значение, возвращаемое функцией `millis()`, чтобы использовать его в последующих вычислениях после нажатия кнопки останова (❸). Когда будет нажата кнопка останова, вызывается функция `displayResult()` (❹), которая вычисляет интервал времени и выводит результаты в монитор порта.

Результаты в окне монитора порта представлены на рис. 8.31.

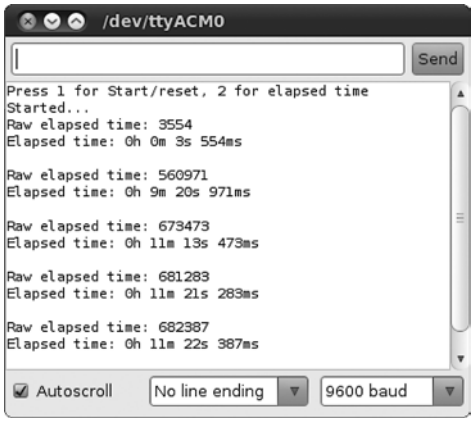


Рис. 8.31. Вывод проекта 31

Прерывания

Прерывание в мире Arduino — это по сути своей сигнал, позволяющий вызвать функцию в скетче в любой момент времени, например, когда изменяется состояние цифрового входа или когда истекает время таймера. Прерывания отлично подходят для приостановки нормальной работы скетча и вызова функции, например, обрабатывающей факт нажатия кнопки.

Если возникает прерывание, нормальная работа программы в `void loop()` временно приостанавливается, вызывается функция обработки прерывания и затем, когда эта функция завершится, работа программы возобновляется с того места, где она была прервана.

Имейте в виду, что функции обработки прерываний должны быть максимально короткими и простыми. Их «назначение» — быстро выполнять свою работу и, если функция обработки прерываний выполняет операцию, подобную той, что выполняется в главном цикле программы, она временно приостанавливает выполнение этой операции в главном цикле. Например, если главный цикл регулярно посылает строку «Hello» в последовательный порт, а функция обработки прерываний посылает последовательность символов «---», то в окне монитора последовательного

порта можно увидеть любой из следующих вариантов вывода: *H---ello*, *He---llo*, *Hel---lo*, *Hell---o* или *Hello---*.

Плата Arduino Uno поддерживает два прерывания, связанные с цифровыми контактами 2 и 3. При правильной настройке аппаратура Arduino следит за уровнем напряжения, приложенного к контактам. Когда напряжение изменяется определенным способом (например, в результате нажатия кнопки), генерируется прерывание, вызывающее соответствующую функцию, которая, например, выводит текст «Stop Pressing Me!».

Режимы прерываний

Различаются четыре вида изменений (*режимов*), вызывающих прерывания:

- `LOW`: к цифровому входу приложен низкий уровень напряжения.
- `CHANGE`: уровень напряжения на цифровом входе изменился — либо с высокого на низкий, либо с низкого на высокий.
- `RISING`: уровень напряжения на цифровом входе изменился с низкого на высокий.
- `FALLING`: уровень напряжения на цифровом входе изменился с высокого на низкий.

Например, чтобы определить факт нажатия кнопки, подключенной к контакту с поддержкой прерывания, используется режим `RISING`. Или, например, если вокруг вашего сада протянут тонкий провод (соединяющий контакт `5V` с контактом цифрового входа), используется режим `FALLING` для определения момента, когда кто-то порвет провод.

ПРИМЕЧАНИЕ

Функции `delay()` и `Serial.available()` не будут работать в функциях, вызываемых по прерываниям.

Настройка прерываний

Чтобы настроить прерывания, добавьте следующие инструкции в функцию `void setup()`:

```
attachInterrupt(0, function, mode);
attachInterrupt(1, function, mode);
```

Здесь 0 соответствует цифровому контакту 2, 1 — цифровому контакту 3, `function` — имя функции, вызываемой по прерываниям, и `mode` — один из четырех режимов прерываний.

Включение и выключение прерываний

Иногда в скетче нужно отключить прерывания. Для этого воспользуйтесь следующей функцией:

```
noInterrupts(); // выключить прерывания
```

А чтобы включить их вновь:

```
interrupts(); // включить прерывания
```

Прерывания весьма чувствительны и работают очень быстро, что делает их ценным инструментом для выполнения срочных операций, например, в момент нажатия кнопки «экстренная остановка».

Проект № 32: Использование прерываний

Для демонстрации прерываний мы вернемся к схеме, собранной в проекте 31. Данный пример включает и выключает встроенный светодиод каждые 500 миллисекунд, одновременно проверяя оба контакта с поддержкой прерываний. В момент нажатия кнопки, связанной с прерыванием 0, в монитор порта выводится значение `micros()`, а в момент нажатия кнопки, связанной с прерыванием 1, — значение `millis()`.

Скетч

Введите и загрузите следующий скетч:

```
  // Проект 32 – Использование прерываний

  #define LED 13

  void setup()
  {
    Serial.begin(9600);
    pinMode(13, OUTPUT);
    attachInterrupt(0, displayMicros, RISING);
    attachInterrupt(1, displayMillis, RISING);
  }

❶ void displayMicros()
  {
    Serial.write("micros() = ");
    Serial.println(micros());
  }
```

```
❷ void displayMillis()
  {
    Serial.write("millis() = ");
    Serial.println(millis());
  }

❸ void loop()
  {
    digitalWrite(LED, HIGH);
    delay(500);
    digitalWrite(LED, LOW);
    delay(500);
  }
```

Скетч включает и выключает встроенный светодиод на плате, как можно видеть в функции `void loop()` (❸). Когда возникает прерывание 0, вызывается функция `displayMicros()` (❶); а когда возникает прерывание 1, вызывается функция `displayMillis()` (❷). По завершении любой из функций скетч продолжает выполнение функции `void loop()`.

Откройте окно монитора порта и понажимайте обе кнопки. В результате в области вывода должны появиться значения `millis()` и `micros()`, как показано на рис. 8.32.

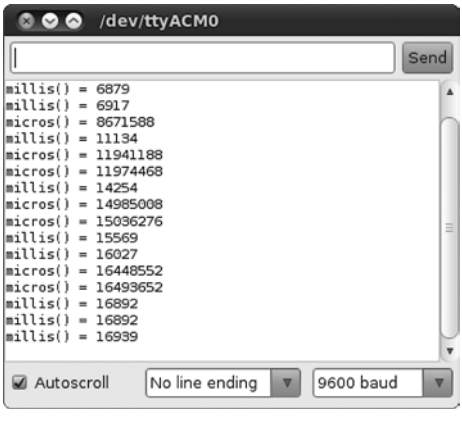


Рис. 8.32. Вывод проекта 32

Забегая вперед

В этой главе вы познакомились с несколькими инструментами и возможностями, которые можно использовать для создания и улучшения своих проектов. В последующих главах нам встретятся другие платы расширения для Arduino, а нашу плату расширения с картой памяти microSD мы используем в проектах для регистрации данных.

9 Цифровые клавиатуры

В этой главе вы:

- ✓ узнаете, как подключить цифровую клавиатуру к плате Arduino;
- ✓ научитесь читать в скетчах значения, набираемые на клавиатуре;
- ✓ освоите прием принятия решений с применением инструкции `switch-case`;
- ✓ сконструируете кодовый замок.

Цифровая клавиатура

Рис. 9.1. Цифровая клавиатура

В некоторых сложных проектах, когда плата Arduino не подключена к компьютеру, необходимо иметь возможность ввода чисел, например секретного кода, чтобы что-то включить или выключить. Для этого можно было бы подключить к цифровым входам 10 или более кнопок без фиксации (соответствующих цифрам от 0 до 9), но намного проще использовать цифровую клавиатуру, подобную той, что изображена на рис. 9.1.

Такая клавиатура обладает следующими преимуществами: она использует всего 7 контактов для 12 кнопок, для работы с ней имеется проверенная библиотека для Arduino и нет необходимости добавлять согласующие резисторы для борьбы с дребезгом контактов, как описывалось в главе 4. Упомянутую библиотеку с именем *Keypad* можно загрузить со страницы *http://playground.arduino.cc/Code/Keypad/*.

Подключение клавиатуры

Подключение цифровой клавиатуры к плате Arduino осуществляется просто. С обратной стороны клавиатуры имеется семь контактов (рис. 9.2).

Рис. 9.2. Контакты на цифровой клавиатуре

Контакты пронумерованы от 1 до 7, слева направо. Во всех проектах с цифровой клавиатурой в этой книге мы будем осуществлять подключение так, как показано в табл. 9.1.

Таблица 9.1. Подключение цифровой клавиатуры к плате Arduino

Номер контакта на цифровой клавиатуре	**Номер контакта на плате Arduino**
1	Цифровой вход 5
2	Цифровой вход 4
3	Цифровой вход 3
4	Цифровой вход 2
5	Цифровой вход 8
6	Цифровой вход 7
7	Цифровой вход 6

Программная обработка клавиатуры

В скетч для проекта, использующего цифровую клавиатуру, необходимо включить несколько строк, добавляющих поддержку клавиатуры, как отмечено в следующем примере скетча. Обязательный код начинается со строки ❶ и заканчивается строкой ❺.

Прежде чем двинуться дальше, проверим работоспособность клавиатуры. Для этого введите и выгрузите скетч из листинга 9.1:

Листинг 9.1. Скетч, демонстрирующий работу с цифровой клавиатурой

```
  // Листинг 9.1
❶ // Начало обязательного кода

  #include "Keypad.h"
  const byte ROWS = 4; // четыре ряда кнопок
  const byte COLS = 3; // по три в каждом ряду
  char keys[ROWS][COLS] =
    {{'1','2','3'},
     {'4','5','6'},
     {'7','8','9'},
❷    {'*','0','#'}};
❸ byte rowPins[ROWS] = {5, 4, 3, 2};
❹ byte colPins[COLS] = {8, 7, 6};
  Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

❺ // Конец обязательного кода

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

  void loop()
  {
    char key = keypad.getKey();
    if (key != NO_KEY)
    {
      Serial.print(key);
    }
  }
```

В строке ❷ объявляется переменная-массив типа `char` с символами — буквами, цифрами или знаками, которые могут генерироваться клавиатурой. В данном случае она содержит цифры и знаки. Строки кода ❸ и ❹ определяют, какие контакты на плате Arduino будут задействованы. При желании значения в этих строках можно изменить, чтобы подключить клавиатуру иначе, чем описано в табл. 9.1.

Тестирование скетча

Выгрузив скетч, откройте окно монитора порта и попробуйте понажимать клавиши на клавиатуре. Символы, соответствующие нажимаемым клавишам, должны появиться в окне монитора порта, как показано на рис. 9.3.

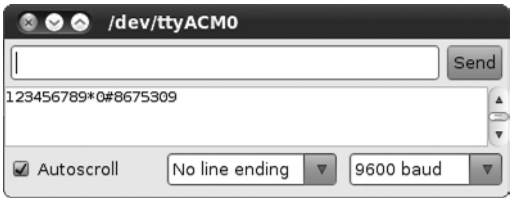


Рис. 9.3. Результаты нажатий клавиш на цифровой клавиатуре

Принятие решений с помощью switch-case

Если необходимо сравнить переменную с двумя или более значениями, зачастую проще всего это сделать с помощью инструкций `switch-case` вместо `if-then`, потому что инструкция `switch-case` способна выполнять произвольное количество сравнений и выполнять код, соответствующий обнаруженному совпадению. Например, если целочисленная переменная `xx` может принимать значения 1, 2 или 3 и, в зависимости от конкретного значения, требуется выполнить определенный фрагмент кода, это можно реализовать, как показано ниже, вместо использования нескольких инструкций `if-then`:

```
switch(xx)
{
  case 1:
    // выполнить операции, если xx имеет значение 1
    break; // завершить и продолжить выполнение скетча со строки,
           // следующей за инструкцией switch-case
  case 2:
    // выполнить операции, если xx имеет значение 2
    break;
  case 3:
    // выполнить операции, если xx имеет значение 3
    break;
  default:
    // выполнить операции, если xx имеет какое-то другое значение,
    // отличное от 1, 2 и 3
    // инструкцию default можно опустить
}
```

Необязательный раздел `default:` в конце этого фрагмента позволяет выполнить некоторые операции, если не нашлось совпадений со значениями, указанными выше в инструкции `switch-case`.

Проект № 33: Кодовый замок

В этом проекте мы создадим программную часть кодового замка, управляемого цифровой клавиатурой. Для этого воспользуемся настройками из скетча, представленного в листинге 9.1, а также добавим обработку секретного кода, который надо ввести с цифровой клавиатуры. С помощью монитора последовательного порта мы будем сообщать пользователю, правильный он ввел код или нет.

В зависимости от результатов проверки введенного шестизначного кода скетч будет вызывать разные функции. Код хранится в скетче, но он не выводится в монитор порта. Чтобы активировать или деактивировать замок, пользователь должен нажать *, затем ввести код и нажать #.

Скетч

Введите и выгрузите следующий скетч:

```
// Проект 33 - Кодовый замок

// Начало обязательного кода

#include "Keypad.h"
const byte ROWS = 4; // четыре ряда кнопок
const byte COLS = 3; // по три в каждом ряду
char keys[ROWS][COLS] =
  {{'1','2','3'},
   {'4','5','6'},
   {'7','8','9'},
   {'*','0','#'}};
byte rowPins[ROWS] = {5, 4, 3, 2};
byte colPins[COLS] = {8, 7, 6};
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

// Конец обязательного кода

❶ char PIN[6]={'1','2','3','4','5','6'}; // секретный код
char attempt[6]={0,0,0,0,0,0};
int z=0;

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

void correctPIN() // вызывается, если введен верный код
{
  Serial.println("Correct PIN entered...");
}

void incorrectPIN() // вызывается, если введен неверный код
```

```
  {
    Serial.println("Incorrect PIN entered!");
  }

  void checkPIN()
  {
    int correct=0;
    int i;
❷   for ( i = 0; i < 6 ; i++ )
    {
      if (attempt[i]==PIN[i])
      {
        correct++;
      }
    }
    if (correct==6)
    {
❸     correctPIN();
    } else
    {
❹     incorrectPIN();
    }
    for (int zz=0; zz<6; zz++) // удалить код, введенный в предыдущей попытке
    {
      attempt[zz]=0;
    }
  }

  void readKeypad()
  {
    char key = keypad.getKey();
    if (key != NO_KEY)
    {
❺     switch(key)
      {
        case '*':
          z=0;
          break;
        case '#':
          delay(100); // устранить вероятность дребезга
          checkPIN();
          break;
        default:
          attempt[z]=key;
          z++;
      }
    }
  }

  void loop()
  {
❻   readKeypad();
  }
```

Принцип действия

После обычной процедуры настройки (описанной в листинге 9.1) скетч начинает непрерывно «сканировать» клавиатуру, вызывая функцию `readKeypad()` (❻). Обнаружив факт нажатия клавиши, функция исследует ее с помощью инструкции `switch-case` (❺). Числовые значения, соответствующие цифровым клавишам, записываются в массив `attempt[]`, и когда пользователь нажимает #, вызывается функция `checkPin()`.

В ❷ значение нажатой клавиши сравнивается с секретным кодом в массиве `PIN[]` (❶). Если введена верная последовательность цифр, вызывается функция `correctPin()` (❸), куда можно добавить свой код, управляющий, например, замком; но если введена неверная последовательность, вызывается функция `incorrectPin()` (❹). Наконец, после проверки ввода пользователя массив `attempt[]` с введенным кодом очищается и подготавливается к следующей попытке ввода.

Тестирование скетча

Выгрузив скетч в плату Arduino, откройте окно монитора последовательного порта, нажмите клавишу со звездочкой (*) на цифровой клавиатуре, введите секретный код и затем нажмите клавишу с решеткой (#). Попробуйте ввести верный и неверный код. Вы должны увидеть результаты, аналогичные представленным на рис. 9.4.

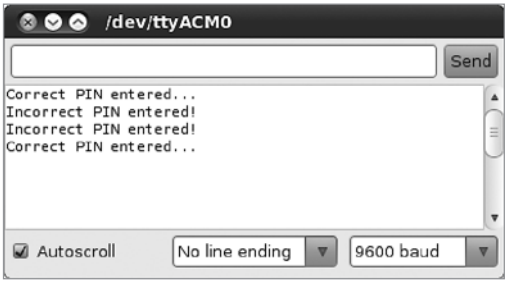


Рис. 9.4. Результаты ввода верного и неверного кода

Этот пример послужит отличной основой для реализации проектов устройств, активируемых вводом кода, таких как кодовый замок, сигнализация или что-то другое. Просто добавьте необходимый код в функции `correctPIN()` и `incorrectPIN()`, который должен выполняться в случае ввода верного или неверного кода.

Забегая вперед

В этой главе вы познакомились с еще одним способом ввода информации в плату Arduino, получили представление об использовании цифровой клавиатуры в скетче и возможности ограничения доступа ко всему, чем может управлять плата Arduino. Попутно вы встретились с очень полезной инструкцией `switch-case`. В следующей главе вы узнаете о еще одной форме ввода: сенсорном экране.

10 Сенсорные экраны

В этой главе вы:

- ✓ узнаете, как подключить резистивный сенсорный экран к плате Arduino;
- ✓ научитесь читать значения, которые может возвращать сенсорный экран;
- ✓ сконструируете простой выключатель, срабатывающий от прикосновения;
- ✓ сконструируете выключатель с функцией регулировки света.

Сенсорные экраны окружают нас со всех сторон: смартфоны, планшетные компьютеры и карманные игровые устройства. Так почему бы и нам не использовать сенсорный экран для взаимодействий с пользователем?

Сенсорные экраны

Сенсорные экраны могут стоить очень дорого, но мы будем использовать недорогую модель, выпускаемую компанией SparkFun (артикул LCD-08977 и BOB-09170), первоначально разработанную для игровой консоли Nintendo DS.

Этот сенсорный экран имеет размеры 5 × 7 см и изображен на рис. 10.1, где он смонтирован на макетной плате.

Обратите внимание, что сенсорный экран соединен шлейфом с маленькой печатной платой в правом верхнем углу (обведена на рис. 10.1). Этот *адаптер* используется для соединения сенсор-

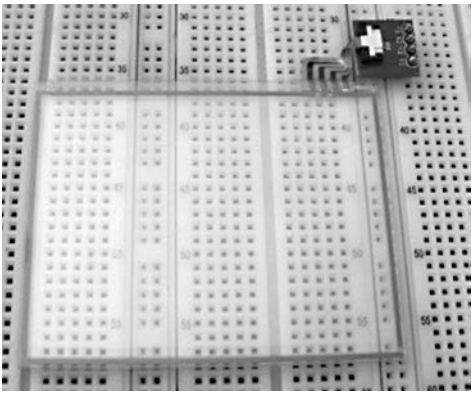


Рис. 10.1. Сенсорный экран, смонтированный на макетной плате

ного экрана с макетной платой и Arduino; на рис. 10.2 показано увеличенное изображение адаптера.

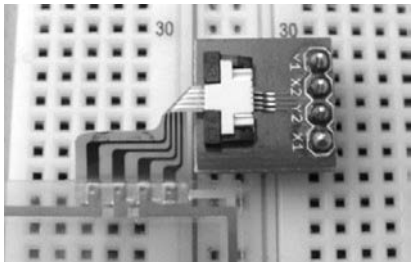


Рис. 10.2. Адаптер сенсорного экрана

Подключение сенсорного экрана

Адаптер сенсорного экрана подключается к плате Arduino, как показано в табл. 10.1.

Таблица 10.1. Подключение адаптера сенсорного экрана к плате Arduino

Контакт на адаптере	Контакт на плате Arduino
Y1	A0
X1	A1
Y2	A2
X2	A3

Проект № 34: Определение области касания на сенсорном экране

Резистивный сенсорный экран состоит из стеклянной панели и гибкой пластиковой мембраны, между которыми расположены два слоя резистивного покрытия. Одно резистивное покрытие действует как ось X, а другое — как ось Y. Сопротивление резистивного покрытия изменяется в зависимости от точки касания, то есть, измеряя напряжение на каждом слое, можно определять координаты X и Y области касания.

В этом проекте мы с помощью платы Arduino определим напряжение на каждом слое и преобразуем его в целочисленные координаты точки касания.

Оборудование

Ниже перечислено оборудование, которое понадобится для этого проекта:

- Сенсорный экран с адаптером.
- Один подстроечный резистор с номиналом 10 кОм.

- Один жидкокристаллический индикатор с размером экрана 16 × 2 символа.
- Несколько отрезков провода разной длины.
- Одна макетная плата.
- Плата Arduino и кабель USB.

Подключите сенсорный экран в соответствии с табл. 10.1 и жидкокристаллический индикатор, как показано на рис. 7.2.

Скетч

Введите и загрузите следующий скетч. Наиболее важные участки скетча снабжены комментариями:

```
  // Проект 34 - Определение области касания на сенсорном экране

  #include <LiquidCrystal.h>
  LiquidCrystal lcd(4,5,6,7,8,9);

  int x,y = 0;

  void setup()
  {
    lcd.begin(16,2);
    lcd.clear();
  }

❶ int readX() // возвращает координату X на сенсорном экране
  {
    int xr=0;
    pinMode(A0, INPUT);
    pinMode(A1, OUTPUT);
    pinMode(A2, INPUT);
    pinMode(A3, OUTPUT);
    digitalWrite(A1, LOW);  // подать низкий уровень на A1
    digitalWrite(A3, HIGH); // подать высокий уровень на A3
    delay(5);
    xr=analogRead(0);       // сохранить координату X
    return xr;
}

❷ int readY() // возвращает координату Y на сенсорном экране
  {
    int yr=0;
    pinMode(A0, OUTPUT);    // A0
    pinMode(A1, INPUT);     // A1
    pinMode(A2, OUTPUT);    // A2
    pinMode(A3, INPUT);     // A3
    digitalWrite(14, LOW);  // подать низкий уровень на A0
```

```
    digitalWrite(16, HIGH); // подать высокий уровень на A2
    delay(5);
    yr=analogRead(1);       // сохранить координату Y
    return yr;
  }

  void loop()
  {
    lcd.setCursor(0,0);
❸  lcd.print(" x = ");
    x=readX();
    lcd.print(x);
    y=readY();
    lcd.setCursor(0,1);
❹   lcd.print(" y = ");
    lcd.print(y);
    delay (200);
}
```

Функции `readX()` и `readY()` (❶ и ❷) читают напряжение на резистивных покрытиях сенсорного экрана с помощью `analogRead()` и возвращают полученное значение. Скетч постоянно вызывает эти две функции для обеспечения определения координат области касания сенсорного экрана в режиме реального времени и их вывода на экран ЖКИ (❸ и ❹). (Задержка `delay(5)` в каждой функции необходима, она дает время аналоговым входам/выходам изменить свое состояние.)

Тестирование скетча

В процессе тестирования скетча понаблюдайте за изменением показаний на экране ЖКИ, выполняя касания сенсорного экрана, и обратите внимание, как изменяются значения X и Y в зависимости от позиции точки касания. Обратите также внимание, какие значения отображаются на экране ЖКИ, когда вы не касаетесь экрана (рис. 10.3).

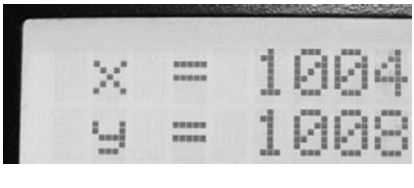


Рис. 10.3. Значения, которые появляются в отсутствие прикосновений к сенсорному экрану

Запомните эти значения — они пригодятся вам в том случае, если в скетче вы будете определять факт отсутствия прикосновения к экрану.

Калибровка сенсорного экрана

Коснувшись сенсорного экрана в углах, как показано на рис. 10.4, и записав полученные значения, вы фактически откалибруете его. В простейшем случае координат углов экрана будет вполне достаточно. Имея эти значения, вы сможете поделить сенсорный экран на небольшие области и использовать их для управления.

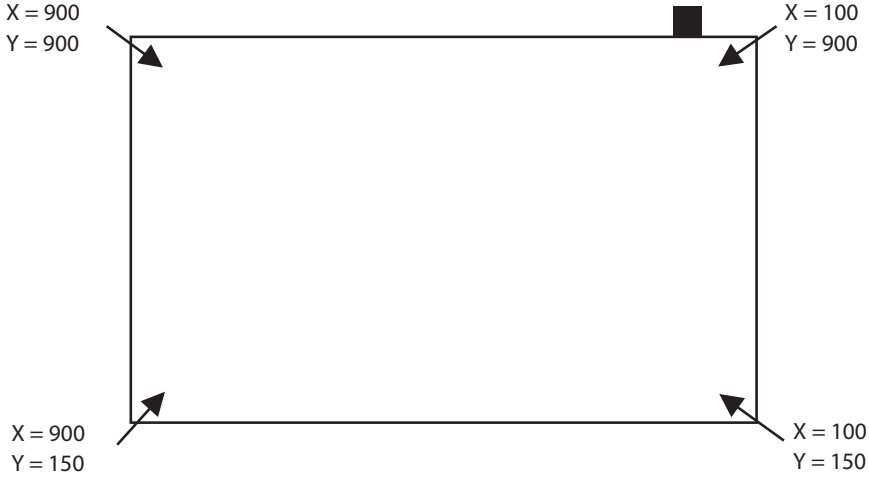


Рис. 10.4. Калибровка сенсорного экрана

Откалибровав сенсорный экран и поделив его на небольшие области, вы сможете с помощью функций `readX()` и `readY()` определять факт прикосновения к разным управляющим областям на экране и затем использовать инструкции `if-then` для выполнения определенных операций, как показано в проекте 35.

Проект № 35: Двухзонный выключатель

В этом проекте мы создадим на основе сенсорного экрана простой выключатель. Для начала поделим экран на две зоны по горизонтали, как показано на рис. 10.5: слева находится зона «включить», а справа — «выключить».

Сравнивая координаты точки касания с границами зон, плата Arduino будет определять зону, в которой произошло касание. После определения зоны можно установить высокий или низкий уровень напряжения на цифровом выходе, но в данном скетче мы просто выведем в монитор порта название зоны.

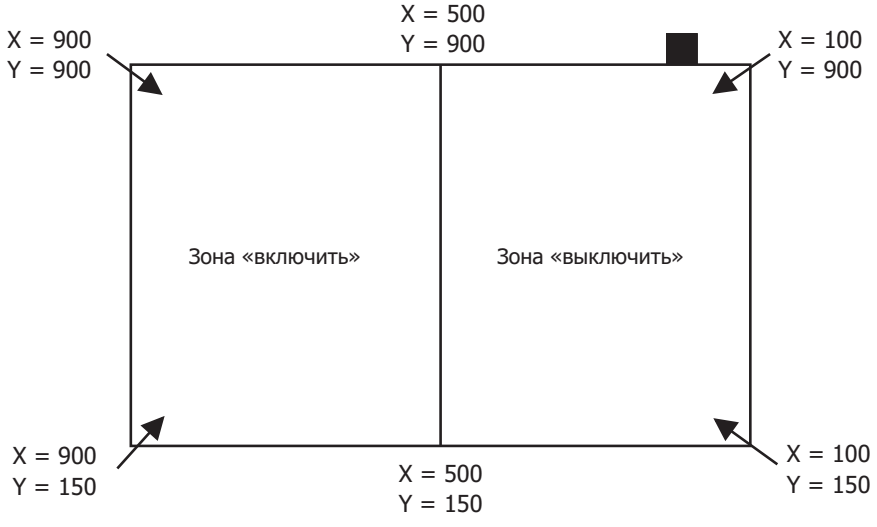


Рис. 10.5. Разметка двухзонного выключателя

Скетч

Введите и загрузите следующий скетч:

```
// Проект 35 - Двухзонный выключатель

int x,y = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(10, OUTPUT);
}

void switchOn()
{
  digitalWrite(10, HIGH);
  Serial.print("Turned ON at X = ");
  Serial.print(x);
  Serial.print(" Y = ");
  Serial.println(y);
  delay(200);
}

void switchOff()
{
  digitalWrite(10, LOW);
  Serial.print("Turned OFF at X = ");
  Serial.print(x);
  Serial.print(" Y = ");
  Serial.println(y);
  delay(200);
}
```

```
int readX() // возвращает координату X на сенсорном экране
{
  int xr=0;
  pinMode(A0, INPUT);
  pinMode(A1, OUTPUT);
  pinMode(A2, INPUT);
  pinMode(A3, OUTPUT);
  digitalWrite(A1, LOW);  // подать низкий уровень на A1
  digitalWrite(A3, HIGH); // подать высокий уровень на A3
  delay(5);
  xr=analogRead(0);
  return xr;
}

int readY() // возвращает координату Y на сенсорном экране
{
  int yr=0;
  pinMode(A0, OUTPUT);
  pinMode(A1, INPUT);
  pinMode(A2, OUTPUT);
  pinMode(A3, INPUT);
  digitalWrite(A0, LOW);  // подать низкий уровень на A0
  digitalWrite(A2, HIGH); // подать высокий уровень на A2
  delay(5);
  yr=analogRead(1);
  return yr;
}

void loop()
{
  x=readX();
  y=readY();
❶  // Касание в зоне "включить"?
  if (x<=900 && x>=500)
  {
    switchOn();
  }
❷  // Касание в зоне "выключить"?
  if (x<500 && x>=100)
  {
    switchOff();
  }
}
```

Принцип действия

Две инструкции `if` в функции `void loop()` определяют зону, в которой произошло касание. Если точка касания находится в левой зоне, то прикосновение интерпретируется как команда «включить» ❶. Если точка касания находится в правой зоне, то прикосновение интерпретируется как команда «выключить» ❷.

ПРИМЕЧАНИЕ

Координата Y в этом проекте игнорируется, потому что сенсорный экран условно разбит вертикальной границей на две зоны по горизонтали. Если бы мы определили еще и горизонтальные границы, тогда необходимо было бы проверять и координату Y, как мы увидим в проекте 36.

Тестирование скетча

На рис. 10.6 показаны результаты, полученные в процессе тестирования. Для каждого касания выводится состояние выключателя и координаты.

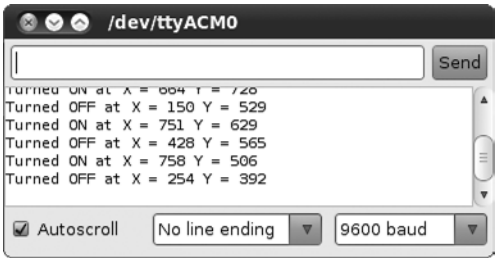


Рис. 10.6. Вывод скетча из проекта 35

Проект № 36: Трехзонный выключатель

В этом проекте мы реализуем трехзонный сенсорный выключатель для светодиода, подключенного к цифровому выходу 3, который будет включать, выключать и регулировать яркость светодиода, используя для этого методику ШИМ (см. главу 3).

Разметка сенсорного экрана

На рис. 10.7 показана разметка сенсорного экрана для этого проекта.

Сенсорный экран поделен на три зоны: «включить», «выключить» и «яркость». По значениям, возвращаемым сенсорным экраном, мы определим, в какой области произошло касание, и выполним соответствующие действия.

Скетч

Введите и загрузите следующий скетч:

```
// Проект 36 - Трехзонный выключатель

int x,y = 0;

void setup()
```

Рис. 10.7. Разметка сенсорного экрана для проекта 36

```
{
  pinMode(3, OUTPUT);
}

void switchOn()
{
  digitalWrite(3, HIGH);
  delay(200);
}

void switchOff()
{
  digitalWrite(3, LOW);
  delay(200);
}

void setBrightness()
{
  int xx, bright;
  float br;
  xx=x-100;
❶  br=(800-xx)/255;
  bright=int(br);
  analogWrite(3, bright);
}

int readX() // возвращает координату X на сенсорном экране
{
  int xr=0;
  pinMode(A0, INPUT);
  pinMode(A1, OUTPUT);
```

```
    pinMode(A2, INPUT);
    pinMode(A3, OUTPUT);
    digitalWrite(A1, LOW);  // подать низкий уровень на A1
    digitalWrite(A3, HIGH); // подать высокий уровень на A3
    delay(5);
    xr=analogRead(0);
    return xr;
  }

  int readY() // возвращает координату Y на сенсорном экране
  {
    int yr=0;
    pinMode(A0, OUTPUT); // A0
    pinMode(A1, INPUT);  // A1
    pinMode(A2, OUTPUT); // A2
    pinMode(A3, INPUT);  // A3
    digitalWrite(A0, LOW);  // подать низкий уровень на A0
    digitalWrite(A2, HIGH); // подать высокий уровень на A2
    delay(5);
    yr=analogRead(1);
    return yr;
  }

  void loop()
  {
    x=readX();
    y=readY();

    // Касание в зоне "включить"?
❷   if (x<=500 && x>=100 && y>= 150 && y<375)
    {
      switchOn();
    }

    // Касание в зоне "выключить"?
❸   if (x>500 && x<=900 && y>= 150 && y<375)
    {
      switchOff();
    }

    // Касание в зоне "яркость"?
❹   if (y>=375 && y<=900)
    {
      setBrightness();
    }
}
```

Принцип действия

Так же как и скетч двухзонного выключателя, этот скетч определяет, где произошло касание — в области «включить» (❷) или «выключить» (❸), которые в этом

проекте уменьшились в размерах, или в области «яркость» (❹) над горизонтальным разделителем, — эта область используется для регулировки яркости. Если касание произошло в области «яркость», координата X преобразуется в функции `setBrightness()` в относительное значение для аппаратуры PWM (❶) и яркость светодиода устанавливается в соответствии с этим значением.

Те же самые базовые функции можно использовать для реализации самых разных выключателей и регуляторов на основе этого простого и недорогого сенсорного экрана.

Забегая вперед

В этой главе был представлен еще один способ взаимодействия с пользователем и управления платой Arduino. В следующей главе мы сосредоточимся на самой плате Arduino, познакомимся с разными ее модификациями и создадим собственную версию Arduino на макетной плате для навесного монтажа.

11 Семейство плат Arduino

В этой главе вы:

- ✓ узнаете, как собрать собственную версию Arduino на макетной плате;
- ✓ исследуете преимущества и особенности широкого диапазона плат, совместимых с Arduino;
- ✓ познакомитесь с открытым аппаратным обеспечением.

В этой главе мы исследуем устройство платы Arduino, а затем создадим свою версию, собрав ее на макетной плате. Эти знания помогут сэкономить деньги, особенно тем, кто активно занимается изменением проектов и созданием прототипов. Здесь вы также познакомитесь с некоторыми новыми компонентами и схемами. Затем исследуете способы загрузки скетчей в самодельную плату Arduino без применения дополнительного оборудования. В заключение мы рассмотрим наиболее типичные альтернативы плате Arduino Uno и их отличия.

Проект № 37: Создание собственной платы Arduino

С увеличением количества или сложности проектов и экспериментов затраты на приобретение плат Arduino легко могут выйти из-под контроля, особенно если вам нравится одновременно работать над несколькими проектами. В таком случае проще и дешевле интегрировать плату Arduino в проекты, собирая ее схему на макетной плате, где затем можно разместить дополнительные компоненты, необходимые для реализации конкретного проекта. Стоимость всех компонентов, необходимых для воспроизведения простейшей версии Arduino на макетной плате (которая обычно допускает многократное использование при аккуратном обращении), не должна превышать 10 долларов США. Если проект предусматривает использование большого количества внешних компонентов, проще собрать свою версию Arduino, потому что в этом случае не придется тянуть массу проводов от платы Arduino к макетной плате.

Оборудование

Для сборки минимальной версии Arduino понадобится следующее оборудование:

- Одна макетная плата.
- Несколько отрезков провода разной длины.
- Один стабилизатор напряжения 7805.
- Один кварцевый резонатор с частотой 16 МГц (например, производимый компанией Newark, артикул 16C8140).
- Один микроконтроллер ATmega328P-PU с загрузчиком Arduino.
- Один электролитический конденсатор с емкостью 1 мкФ и рабочим напряжением 25 В (C1).
- Один электролитический конденсатор с емкостью 100 мкФ и рабочим напряжением 25 В (C2).
- Два керамических конденсатора с емкостью 22 пФ и рабочим напряжением 50 В (C3 и C4).
- Два керамических конденсатора с емкостью 100 нФ и рабочим напряжением 50 В (C5).
- Два резистора с номиналом 560 Ом (R1 и R2).
- Один резистор с номиналом 10 кОм (R3).
- Два светодиода по вашему выбору (LED1 и LED2).
- Одна кнопка без фиксации (S1).
- Одна колодка с шестью контактами.
- Один разъем для подключения батареи типа PP3.
- Одна батарея типа PP3 с напряжением 9 В.

Некоторые из этих компонентов могут быть вам незнакомы. В следующих разделах вы поближе познакомитесь с ними, увидите их фотографии и узнаете, как они обозначаются на принципиальных схемах.

Стабилизатор напряжения 7805 Linear Voltage Regulator

Стабилизатор содержит простую схему, преобразующую одно напряжение в другое. В список необходимого оборудования включен стабилизатор 7805, который может преобразовывать напряжение от 7 до 30 вольт в фиксированное напряжение 5 вольт с силой тока до 1 ампера, этого вполне достаточно для работы самодельной платы Arduino. На рис. 11.1 можно видеть типичного представителя стабилизаторов из серии 7805 в корпусе TO-220.

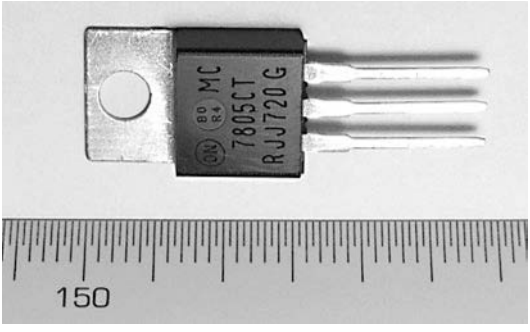


Рис. 11.1. Стабилизатор напряжения 7805

На рис. 11.2 показано, как стабилизатор 7805 изображается на принципиальных схемах. Если смотреть со стороны маркировки, левый вывод (IN) служит для подачи входного напряжения, центральный вывод (GND) подключается к «земле» и правый вывод (OUT) отдает выходное напряжение 5 В. Металлический язычок сверху с круглым отверстием позволяет смонтировать стабилизатор на массивной металлической основе, которую часто называют теплоотводом или радиатором. Наличие радиатора обязательно, если ток нагрузки близок к максимальному (1 А), потому что в таком режиме работы стабилизатор 7805 очень сильно нагревается. Металлический язычок соединен также с выводом GND. В данном проекте нам потребуется один стабилизатор напряжения 7805.

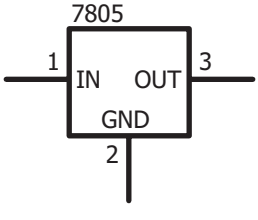


Рис. 11.2. Обозначение стабилизатора напряжения 7805

Кварцевый резонатор с частотой 16 МГц

Кварцевый резонатор часто называют просто *кварцем*. Он помогает поддерживать частоту электрических колебаний с очень высокой точностью. В данном случае частота равна 16 МГц. Кварц, используемый в этом проекте, представлен на рис. 11.3.

Сравните это изображение с кварцем на вашей плате Arduino. Они должны быть идентичны по форме и размеру.

Кварц не имеет полярности. На принципиальных схемах он изображается, как показано на рис. 11.4.

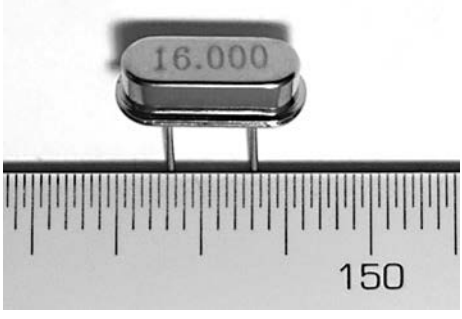


Рис. 11.3. Кварцевый резонатор

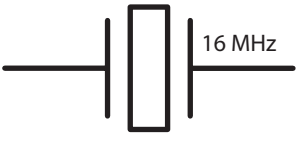


Рис. 11.4. Обозначение кварцевого резонатора

Кварц определяет быстродействие микроконтроллера. Например, микроконтроллер, который мы будем использовать, работает с тактовой частотой 16 МГц, то есть он выполняет 16 миллионов элементарных инструкций в секунду. Однако это не означает, что строки в скетче будут выполняться с такой скоростью, потому что каждая строка исходного кода преобразуется во время компиляции в множество элементарных инструкций.

Микроконтроллер Atmel ATmega328P-PU

Микроконтроллер — это маленький компьютер, содержащий процессор, выполняющий инструкции, память нескольких видов для хранения данных и инструкций, составляющих скетч, и поддерживающий несколько способов приема и передачи данных. Как рассказывалось в главе 2, микроконтроллер — это мозг платы Arduino. На рис. 11.5 можно увидеть, как выглядит микроконтроллер ATmega328P-PU. Обратите внимание, что вывод микроконтроллера с номером 1 находится на фотографии слева внизу и отмечен маленькой точкой.

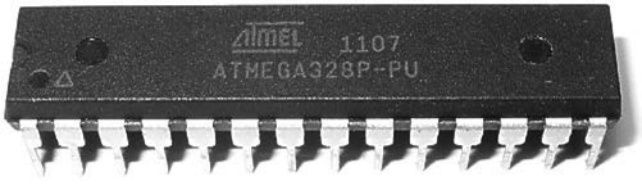


Рис. 11.5. Микроконтроллер ATmega328P-PU

На принципиальных схемах микроконтроллер изображается, как показано на рис. 11.6.

Не все микроконтроллеры содержат *загрузчик* Arduino — программу, которая загружает и запускает скетчи, написанные для Arduino. Выбирая микроконтроллер для сборки самодельной платы Arduino, проверьте, включает ли он загрузчик.

Обычно такие микроконтроллеры можно приобрести у тех же продавцов, что продают платы Arduino, таких как Adafruit, Freetronics и SparkFun.

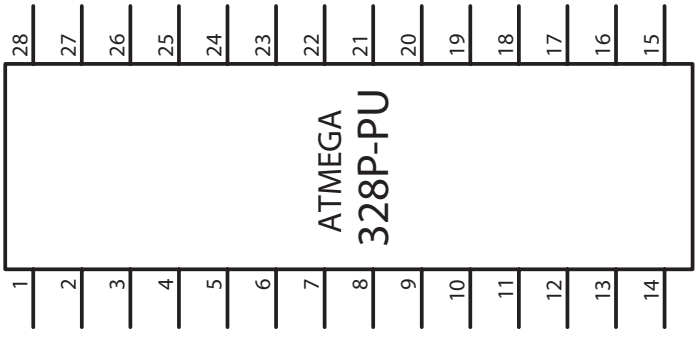


Рис. 11.6. Обозначение микроконтроллера

Схема

На рис. 11.7 изображена принципиальная схема нашей версии платы Arduino.

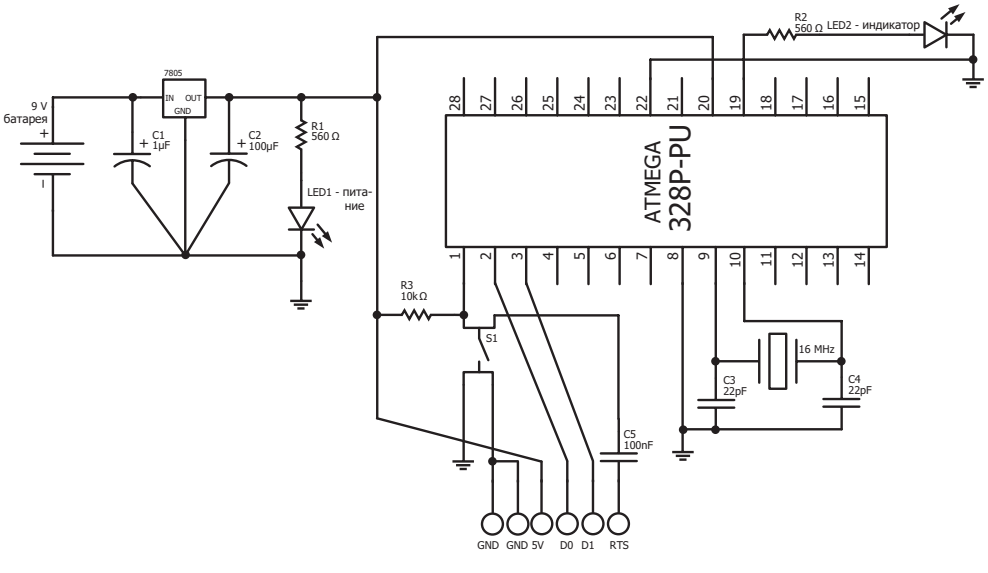


Рис. 11.7. Принципиальная схема минимальной платы Arduino

Схема имеет два раздела. Первый, слева, — это источник питания, который понижает входное напряжение до стабильных 5 В. Когда на плату подается питание, зажигается светодиод LED1. Второй раздел, справа, включает микроконтроллер,

кнопку сброса, контакты последовательного интерфейса для программирования и еще один светодиод. Этот светодиод подключен к выводу ATmega328P-PU, который на обычной плате Arduino связан с контактом 13. Соедините компоненты, как показано на схеме. Не забудьте протянуть провода к колодке с шестью контактами (показана на рис. 11.8), которая внизу на принципиальной схеме изображена как шесть кружочков. Далее мы будем использовать эти контакты для загрузки скетча в самодельную плату Arduino.

Схема запитывается от простой батареи с напряжением 9 В, подключаемой с помощью простого разъема, изображенного на рис. 11.9. Подключите красный вывод разъема к положительному (+) контакту, а черный — к отрицательному (–).

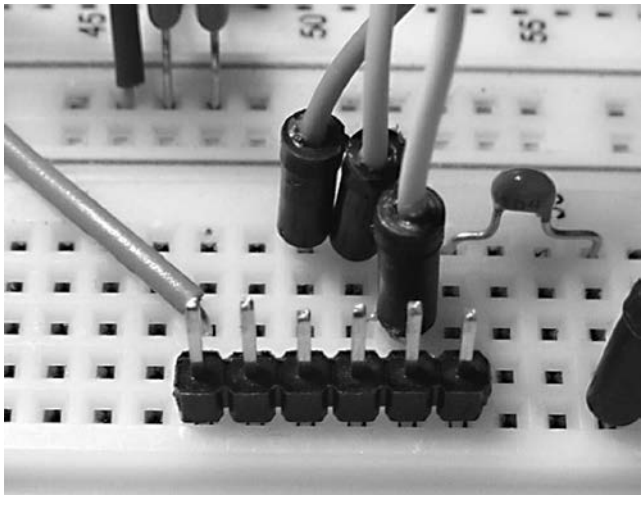


Рис. 11.8. Колодка с шестью контактами

Рис. 11.9. Батарея 9 В и разъем для ее подключения

Идентификация контактов Arduino

Где же на нашей самодельной плате Arduino расположены все остальные контакты? Эти контакты — аналоговые, цифровые и другие, имеющиеся на обычной плате Arduino, — есть и на нашей самодельной версии. Если они вам понадобятся, подключайтесь непосредственно к выводам микроконтроллера.

Резистор R2 и светодиод LED2 на самодельной плате Arduino подключены к цифровому контакту 13. В табл. 11.1 перечислены контакты на плате Arduino (слева) и соответствующие им выводы микроконтроллера ATmega328P-PU (справа).

Таблица 11.1. Назначение выводов ATmega328P-PU

Контакт на плате Arduino	**Вывод микроконтроллера ATmega328P-PU**
RST	1
RX/D0	2
TX/D1	3
D2	4
D3	5
D4	6
5V	7
GND	8
D5	11
D6	12
D7	13
D8	14
D9	15
D10	16
D11	17
D12	18
D13	19
5V	20
AREF	21
GND	22
A0	23
A1	24
A2	25
A3	26
A4	27
A5	28

Чтобы избежать путаницы, такие продавцы, как Adafruit и Freetronics, наносят маркировку на микроконтроллер, как показано на рис. 11.10 (*http://www.adafruit.com/products/554/*).

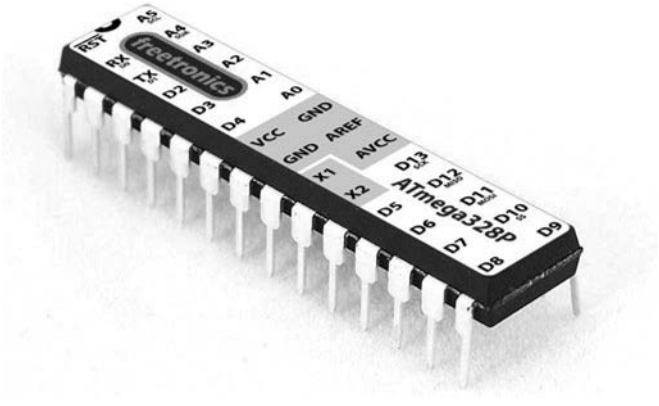


Рис. 11.10. Маркировка выводов микроконтроллера

Запуск проверочного скетча

Теперь пришло время загрузить скетч. Для начала загрузим скетч, который просто мигает светодиодом:

```
// Проект 37 - Создание собственной платы Arduino

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

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

Скетч можно загрузить одним из трех способов.

Метод замены микроконтроллера

Самый простой и наименее затратный способ — извлечь микроконтроллер из имеющейся платы Arduino, вставить микроконтроллер, предназначенный для

самодельной платы, загрузить скетч и вернуть запрограммированный микроконтроллер на место.

Для безопасного извлечения микроконтроллера из платы Arduino используйте специальный экстрактор, изображенный на рис. 11.11.

Рис. 11.11. Используйте экстрактор для извлечения микроконтроллера

Извлекая микроконтроллер, делайте это аккуратно и не торопясь, чтобы все выводы извлекались из панельки *равномерно*! Может потребоваться некоторое усилие, но в конце концов микроконтроллер выйдет из гнезд.

Чтобы вставить микроконтроллер в панельку на плате Arduino, может потребоваться подогнуть его выводы, чтобы они точно входили в гнезда. Для этого уприте микроконтроллер одной стороной в плоскую поверхность и осторожно надавите сверху вниз; затем повторите процедуру с другой стороной, как показано на рис. 11.12.

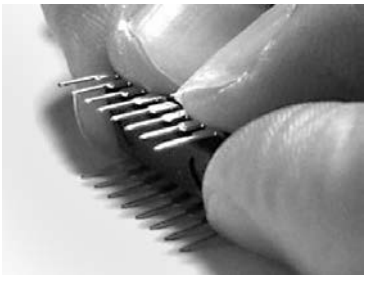


Рис. 11.12. Подгибание выводов микроконтроллера

Наконец, когда вы будете вставлять микроконтроллер в плату Arduino, не забудьте, что метка на корпусе должна находиться справа, как показано на рис. 11.13.

Рис. 11.13. Правильная ориентация микроконтроллера на плате Arduino

Подключение к имеющейся плате Arduino

Для загрузки скетча в микроконтроллер на самодельной плате можно использовать интерфейс USB на плате Arduino Uno. Этот метод позволяет уменьшить износ панельки для микроконтроллера на плате и сэкономить деньги, потому что не требует приобретения отдельного кабеля USB.

Давайте рассмотрим, как загрузить скетч в микроконтроллер с использованием интерфейса USB:

1. Отключите кабель USB от платы Arduino Uno и извлеките из нее микроконтроллер.
2. Выключите питание самодельной платы (если включено).
3. Соедините проводами цифровой контакт 0 на плате Arduino с выводом 2 микроконтроллера ATmega328P-PU на самодельной плате и цифровой контакт 1 на плате Arduino с выводом 3 микроконтроллера ATmega328P-PU на самодельной плате.
4. Соедините контакты 5V и GND платы Uno с соответствующими контактами на самодельной плате.
5. Соедините проводом контакт RST на плате Arduino RST с выводом 1 микроконтроллера ATmega328P-PU на самодельной плате.
6. Подключите кабель USB к плате Arduino Uno.

После этого вся система должна действовать, как если бы она состояла из единственной платы Arduino Uno. В результате у вас появится возможность загрузить скетч в микроконтроллер на самодельной плате и использовать монитор последовательного порта, если понадобится.

С использованием кабеля FTDI

Последний метод — самый простой, но он требует приобретения кабеля USB, известного как *кабель FTDI* (такое название объясняется тем, что внутри кабель

содержит микросхему интерфейса USB, разработанную компанией FTDI). Приобретая кабель FTDI, выбирайте тот, что предназначен для моделей с питанием 5 В, потому что кабель для моделей с питанием 3,3 В не подходит для нашего случая. Этот кабель (изображен на рис. 11.14) имеет разъем USB с одного конца и плоский разъем с шестью контактами — с другого. Разъем USB содержит внутри микросхему, эквивалентную микросхеме интерфейса USB на плате Arduino Uno. Плоский разъем с шестью контактами подключается к колодке на плате, изображенной на рис. 11.7 и 11.8.

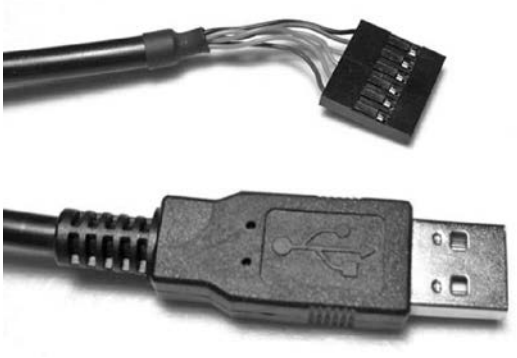


Рис. 11.14. Кабель FTDI

При подключении плоского разъема к колодке на плате черный провод должен соединяться с контактом GND на колодке. После подключения кабеля по нему будет так же подаваться питание, как в случае с обычной платой Arduino.

Прежде чем загружать скетч и использовать монитор последовательного порта, измените тип платы на Arduino Duemilanove или Nano w/ ATmega328 в меню Tools ▸ Board (Инструменты ▸ Плата), как показано на рис. 11.15.

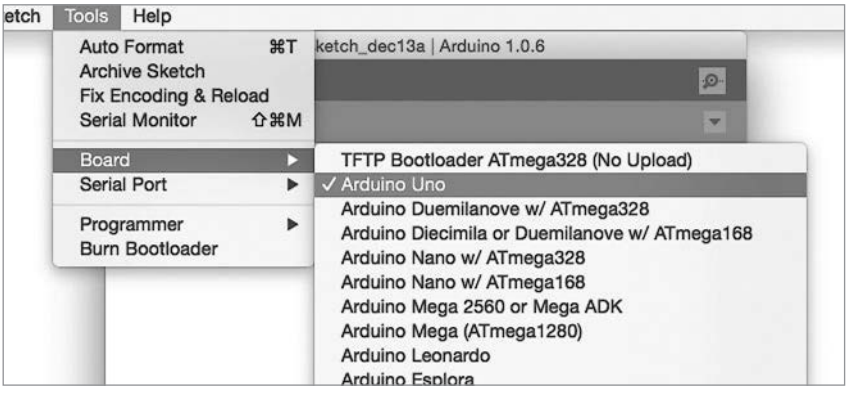


Рис. 11.15. Выбор типа платы в IDE

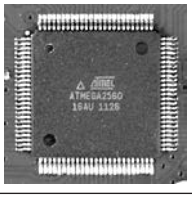
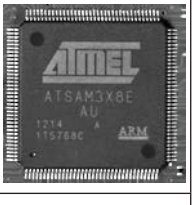
Выбрав способ по своему вкусу, проверьте его, попытавшись загрузить скетч из проекта 37. После этого можно приступать к разработке более сложных схем, используя единственную макетную плату, которая позволит вам создать большее количество проектов за меньшие деньги. Вы сможете даже создавать постоянные устройства с нуля, если научитесь самостоятельно делать печатные платы.

Обширное семейство плат Arduino

Во всех проектах в этой книге используется исключительно плата Arduino Uno, однако существует огромное множество альтернативных вариантов. Они отличаются размерами, количеством входов и выходов, объемом памяти для скетчей и ценой.

Одним из существенных отличий между моделями Arduino является используемый в них микроконтроллер. В настоящее время наиболее широкое распространение получили микроконтроллеры ATmega328 и ATmega2560, но в модели Due используется другой, более мощный микроконтроллер. Основные отличия между ними (включая обе версии ATmega328) перечислены в табл. 11.2.

Таблица 11.2. Сравнительные характеристики микроконтроллеров

	ATmega 328P-PU	**ATmega328P SMD**	**ATmega2560**	**SAM3X8E**
				
Извлекаемый	Да	Нет	Нет	Нет
Тактовая частота	16 МГц	16 МГц	16 МГц	84 МГц
Рабочее напряжение	5 В	5 В	5 В	3.3 В
Количество цифровых контактов	14 (6 с поддержкой ШИМ)	14 (6 с поддержкой ШИМ)	54 (14 с поддержкой ШИМ)	54 (12 с поддержкой ШИМ)
Количество аналоговых контактов	6	8	16	12
Ток на один контакт входа/выхода	40 мА	40 мА	40 мА	от 3 до 15 мА
Объем флеш-памяти	31.5 кБ	31.5 кБ	248 кБ	512 кБ
Объем ЭСППЗУ	1 кБ	1 кБ	4 кБ	Нет
Объем ОЗУ	2 кБ	2 кБ	8 кБ	96 кБ

Обычно при сравнении моделей Arduino учитываются типы имеющейся памяти и объем памяти каждого типа. Всего имеется три типа памяти:

- *Флеш-память* — это память, предназначенная для хранения скетча после его компиляции и загрузки из IDE.
- *ЭСППЗУ* (*электрически-стираемое перепрограммируемое постоянное запоминающее устройство*) — память небольшого объема для хранения постоянных данных, с которой вы познакомитесь в главе 16.
- *ОЗУ* — оперативная память, предназначенная для хранения переменных, используемых программами.

ПРИМЕЧАНИЕ

Помимо Arduino Uno существует множество других моделей, и некоторые, описываемые здесь, — лишь вершина айсберга. Когда вы приступите к планированию больших или сложных проектов, не бойтесь обращаться к более мощным моделям Mega. Если же вам необходимо лишь несколько входных и выходных контактов для постоянного проекта, обратите внимание на модель Nano или даже LilyPad.

А теперь исследуем линейку доступных плат.

Arduino Uno

Модель Uno в настоящее время считается стандартом Arduino. Все когда-либо производившиеся платы расширения для Arduino должны быть совместимы с моделью Uno. Плата Arduino Uno считается самой простой в использовании благодаря наличию встроенного интерфейса USB и извлекаемому микроконтроллеру.

Freetronics Eleven

На рынке существует множество плат, имитирующих работу Arduino Uno, а некоторые даже отличаются улучшенной конструкцией. Одной из таких плат является Freetronics Eleven, изображенная на рис. 11.16.

Несмотря на полную совместимость с Arduino Uno, плата Eleven имеет несколько усовершенствований, придающих ей особую привлекательность. Первое усовершенствование — наличие достаточно большой площадки для макетирования ниже ряда контактов цифровых входов/выходов. Наличие этой площадки позволяет собирать собственные схемы непосредственно на основной плате, экономя деньги и пространство, потому что вам не приходится приобретать отдельную макетную плату.

Второе усовершенствование: контакты для подключения линии приема/передачи данных (TX/RX) питания и встроенный светодиод, подключенный к контакту D13, размещены близко к правому краю платы; такая компоновка обеспечивает их видимую доступность даже при подключенной плате расширения. Наконец, на ней имеется разъем микро-USB, который намного меньше стандартного разъема USB, устанавливаемого на модели Uno. Это упрощает проектирование плат расширения, так как избавляет от необходимости волноваться о компоновке плат расширения,

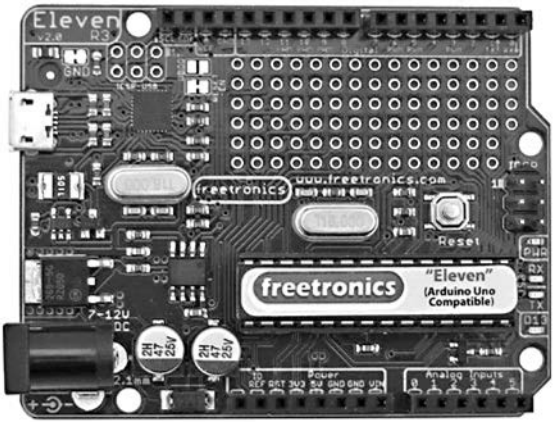


Рис. 11.16. Freetronics Eleven

чтобы избежать замыкания проводников на корпус разъема USB. Эта модель доступна по адресу *http://www.freetronics.com/products/eleven/*.

Freeduino

Плата Freeduino родилась как открытый проект. Материалы этого проекта опубликованы; пользуясь ими, любой желающий сможет конструировать свои платы, совместимые с Arduino. Одним из наиболее популярных вариантов является комплект деталей для создания совместимой с Duemilanove платы, изображенной на рис. 11.17.

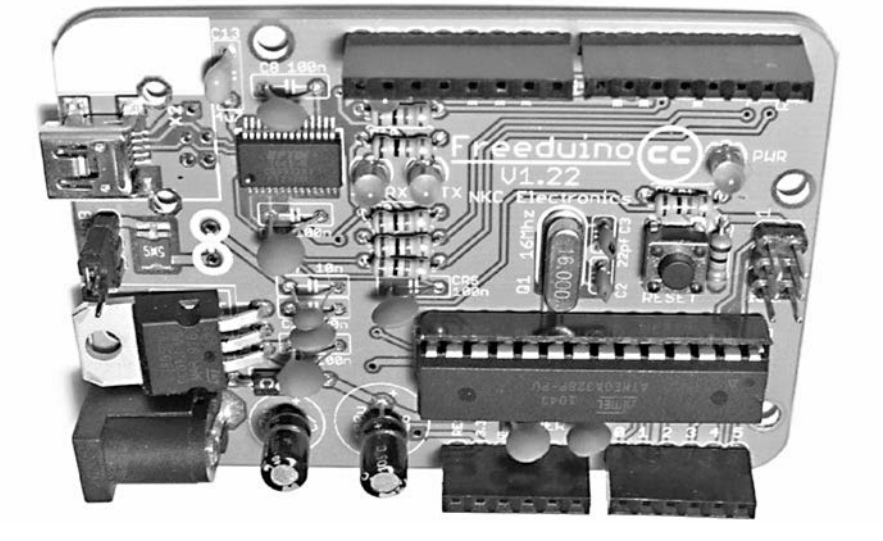


Рис. 11.17. Собранная плата Freeduino, совместимая с Duemilanove

Пользуясь платой Freeduino, можно опробовать все примеры из этой книги. Двумя основными преимуществами Freeduino являются ее дешевизна и возможность сборки вручную. Комплект для сборки Freeduino можно приобрести по адресу *http://www.seeedstudio.com/*.

Pro Trinket

Плата Pro Trinket (рис. 11.18) — это миниатюрная версия Arduino Uno, предназначенная для создания карманных электронных устройств, использования с макетными платами и в проектах, где необходима плата миниатюрных размеров.

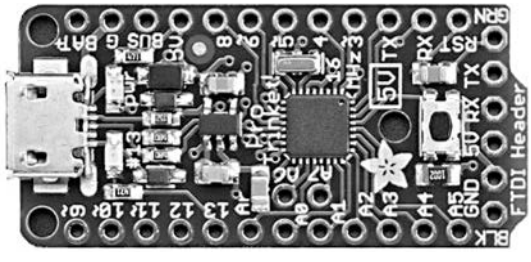


Рис. 11.18. Adafruit Pro Trinket

Она несколько отличается от Arduino Uno (например, отсутствует возможность связи по последовательному интерфейсу, кроме как посредством внешнего кабеля FTDI); однако невысокая цена делает эту плату очень и очень привлекательной. Приобрести Pro Trinket можно по адресу *http://www.adafruit.com/*.

Arduino Nano

Если компактность играет решающую роль, отличным выбором может стать готовая Arduino-совместимая плата Nano. Эта маленькая, но мощная плата Arduino (рис. 11.19) предназначена для использования вместе с макетными платами.

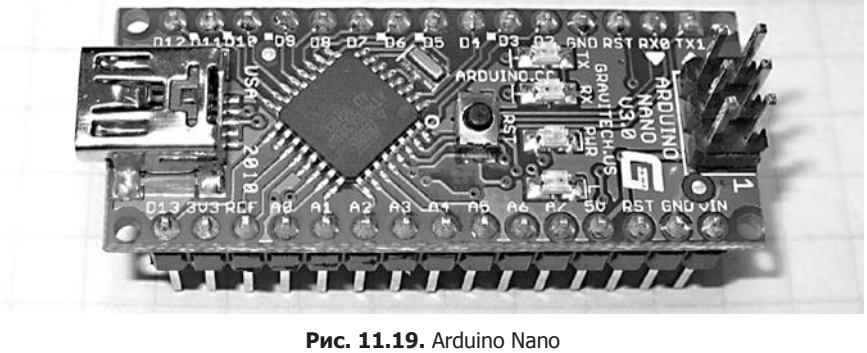


Рис. 11.19. Arduino Nano

Плата Nano имеет размеры 1,8 × 4,3 см, но обладает всеми функциональными возможностями Freeduino. Кроме того, в ней используется микроконтроллер ATmega328P в планарном корпусе, имеющий два дополнительных контакта аналоговых выходов (A6 и A7). Приобрести Nano можно по адресу *http://www.gravitech.us/arna30wiatp.html*.

Arduino LilyPad

Модель LilyPad разрабатывалась для встраивания в нестандартные проекты, такие как карманные электронные устройства. Она не боится влаги и мягкодействующих моющих средств, что делает LilyPad идеальной, например, для управления светящимися элементами одежды. Плата имеет уникальную конструкцию, как можно видеть на рис. 11.20.

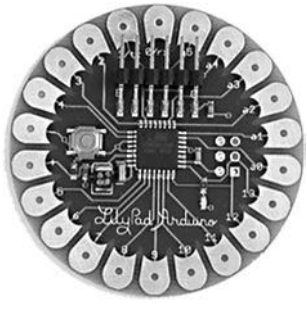


Рис. 11.20. Arduino LilyPad

Контакты на плате LilyPad выполнены в виде площадок, поэтому провода к ним должны припаиваться, из-за чего LilyPad лучше подходит для создания постоянных, неразборных устройств. Как следствие минимализма, на плате отсутствует стабилизатор напряжения, поэтому пользователь должен обеспечить источник питания с напряжением от 2,7 до 5,5 В. На плате LilyPad также отсутствует интерфейс USB, поэтому для загрузки скетчей необходимо применять 5-вольтовый кабель FTDI. Приобрести плату Arduino LilyPad можно практически у любого продавца Arduino.

Arduino Mega 2560

Если контактов входов/выходов на стандартной Arduino Uno оказывается недостаточно или необходим больший объем памяти для скетчей, обратите внимание на модель Mega 2560. Она изображена на рис. 11.21 и имеет большие размеры, чем Arduino — 11 × 5,3 см.

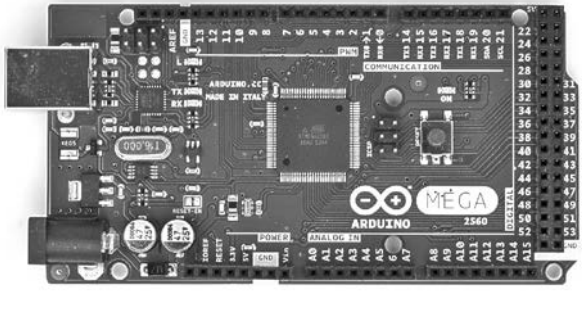


Рис. 11.21. Arduino Mega 2560

Несмотря на то что Mega 2560 больше стандартной модели Uno, она допускает возможность подключения к ней большинства плат расширения для Arduino, а для больших проектов выпускаются макетные платы расширения с размерами, соответствующими размерам модели Mega. В плате Mega используется микроконтроллер ATmega2560, имеющий больший объем памяти и количество входов/выходов (как описывается в табл. 11.2) в сравнении с Uno. Кроме того, наличие четырех отдельных линий последовательной шины расширяет ее возможности приема и передачи данных. Приобрести плату Mega 2560 можно практически у любого продавца Arduino.

Freetronics EtherMega

Если для проекта вам потребуется плата Arduino Mega 2560, плата расширения с картой памяти microSD и плата расширения с интерфейсом Ethernet для подключения к Интернету, лучшей альтернативой может стать модель EtherMega (рис. 11.22), которая обладает всеми этими функциями, но обойдется дешевле, чем перечисленные три компонента. Приобрести плату EtherMega можно по адресу *http://www.freetronics.com/em/*.

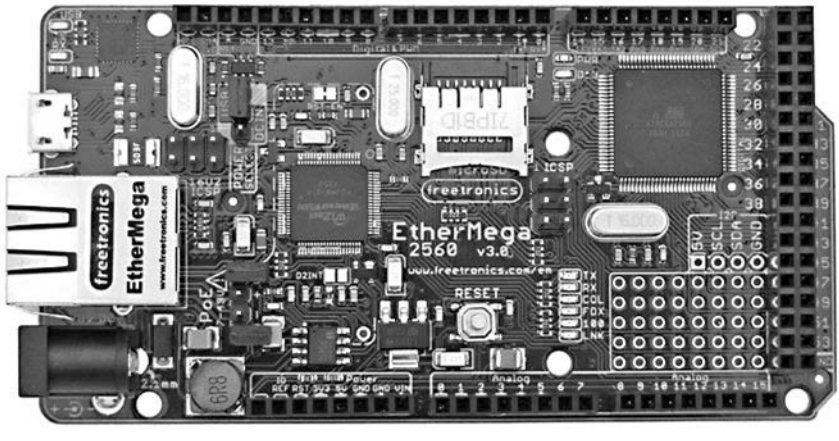


Рис. 11.22. Freetronics EtherMega

Arduino Due

Это самая мощная плата Arduino из когда-либо выпускавшихся. Ее процессор с тактовой частотой 84 МГц способен выполнять скетчи намного быстрее. Как можно видеть на рис. 11.23, эта плата очень похожа на модель Arduino Mega 2560, но в отличие от нее имеет дополнительный порт USB для подключения внешних устройств и иную маркировку контактов.

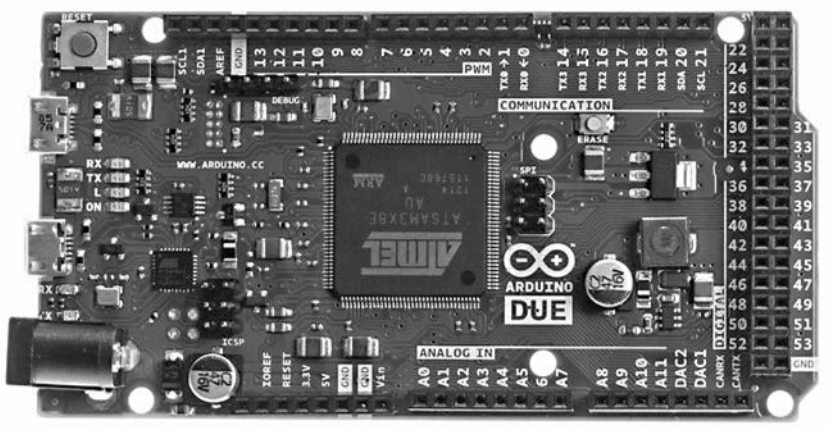


Рис. 11.23. Arduino Due

Кроме того, Due имеет объем памяти, в 16 раз превышающий объем памяти на плате Uno, что позволяет создавать по-настоящему сложные скетчи. Однако Due использует рабочее напряжение всего 3,3 В, поэтому любые конструкции, платы расширения и другие устройства, подключаемые к аналоговым или цифровым контактам, не должны иметь напряжение выше 3,3 В. Но, несмотря на эти ограничения, преимущества Due часто оправдывают необходимость внесения изменений в аппаратную часть проектов.

ПРИМЕЧАНИЕ

Принимая решение о приобретении своей следующей платы Arduino или аксессуаров, обращайтесь к продавцу, пользующемуся хорошей репутацией, который предоставляет поддержку и гарантии. Интернет переполнен предложениями недорогих вариантов, однако их производители часто упрощают конструкцию, чтобы максимально уменьшить цену, и у вас может не оказаться возможности вернуть неисправный или не отвечающий требованиям товар.

ОТКРЫТОЕ АППАРАТНОЕ ОБЕСПЕЧЕНИЕ

Плата Arduino имеет открытую конструкцию, описание которой опубликовано для общего доступа, поэтому любой желающий может производить, изменять, распространять и использовать эти платы как посчитает нужным. Такой способ распространения подпадает под определение «открытое аппаратное обеспечение» — недавно появившееся движение, созданное в противовес идеям защиты авторских прав и интеллектуальной собственности. Разработчики Arduino решили раскрыть свою конструкцию, чтобы способствовать развитию сообщества любителей электроники и для общего блага.

Следуя духу открытого аппаратного обеспечения, многие организации, производящие аксессуары или модифицированные версии плат Arduino, публикуют конструкторскую документацию для своих устройств под той же открытой лицензией. Это обеспечивает намного более быстрое развитие продуктов, чем могла бы обеспечить организация, занимающаяся разработкой в одиночестве.

Забегая вперед

В этой главе вы получили представление о линейке доступных устройств, включая самодельную плату Arduino, собранную на макетной плате, познакомились с компонентами, составляющими Arduino, и узнали, как собрать свою плату Arduino на макетной плате для навесного монтажа. Теперь вы знаете, как сконструировать несколько прототипов на основе Arduino, не приобретая множество плат. Вы также познакомились с разнообразием моделей Arduino, имеющихся на рынке, и теперь сможете выбрать модель, полностью соответствующую вашим потребностям. Наконец, вы узнали о существовании движения за открытое аппаратное обеспечение.

В следующей главе вы научитесь пользоваться разными электродвигателями и приступите к созданию своего танка с электродвигателем, управляемого платой Arduino!

12 Электродвигатели и движение

В этой главе вы:

- ✓ узнаете, как использовать сервопривод для создания аналогового термометра;
- ✓ научитесь управлять направлением и скоростью вращения электродвигателей;
- ✓ познакомитесь с платой расширения для Arduino, управляющей электродвигателями;
- ✓ приступите к созданию роботизированного танка с электродвигателем;
- ✓ узнаете, как с помощью микровыключателей избегать столкновений;
- ✓ освоите применение инфракрасных и ультразвуковых датчиков для предотвращения столкновений.

Реализация небольших перемещений с помощью сервоприводов

Сервопривод состоит из электродвигателя, который по команде может выполнять поворот вала на определенный угол. Сервопривод можно использовать, например, для дистанционного управления автомобилем, если присоединить его с помощью *рычага* к рулевому колесу. Примером такого рычага может служить стрелка аналоговых часов. На рис. 12.1 изображен сервопривод с тремя типами рычагов.

Выбор сервопривода

При выборе сервопривода необходимо учитывать следующие параметры:

- ❑ **Скорость** — время, необходимое сервоприводу для поворота вала; обычно измеряется в секундах на угловой градус.

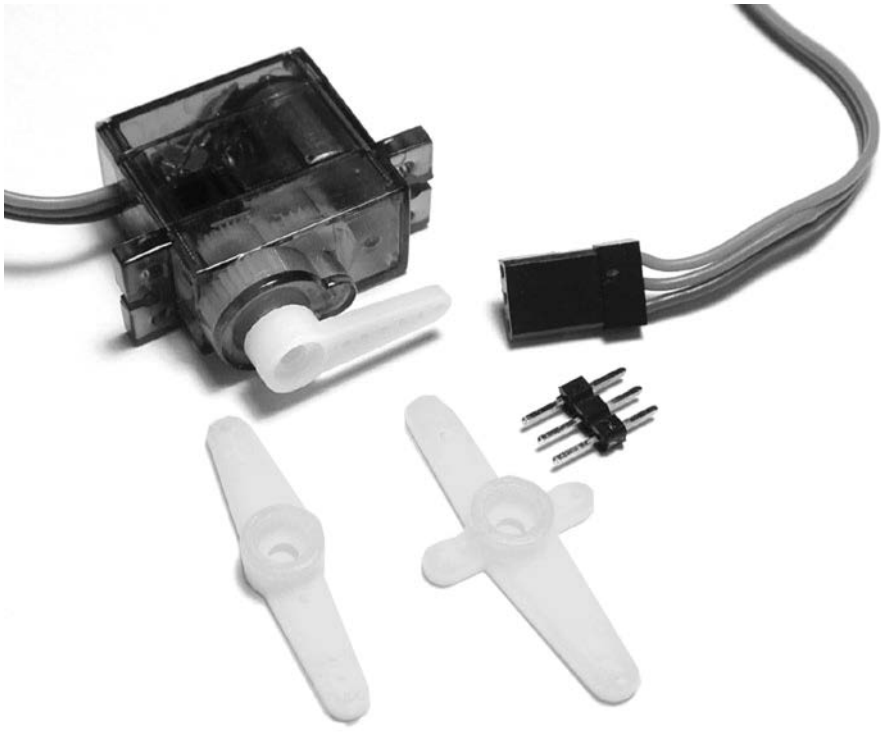


Рис. 12.1. Сервопривод и несколько видов рычагов

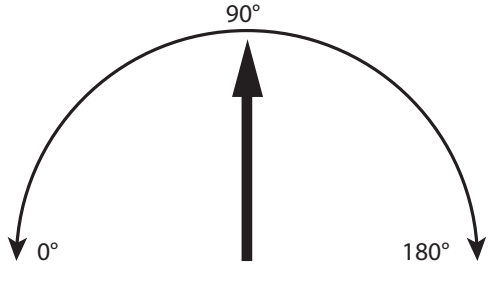


Рис. 12.2. Пример диапазона поворота сервопривода

- **Диапазон поворота** — угол поворота сервопривода; например, 180 градусов (половина полного оборота) или 360 градусов (один полный оборот).
- **Ток** — сила тока, потребляемого сервоприводом. При организации управления сервоприводами с помощью платы Arduino может потребоваться использовать внешний источник для питания сервопривода.

- **Крутящий момент** — усилие, которое способен приложить сервопривод. Чем больше крутящий момент, тем более тяжелыми конструкциями может управлять сервопривод. Вообще, крутящий момент прямо пропорционален силе потребляемого тока.

На рис. 12.1 изображен сервопривод hexTronik HXT900. Это недорогой сервопривод с диапазоном поворота 180 градусов, как показано на рис. 12.2.

Подключение сервопривода

Сервопривод достаточно просто подключается к плате Arduino, потому что имеет всего три вывода. Если вы используете HXT900, самый темный провод (рис. 12.1) должен подключаться к контакту GND, центральный провод — к напряжению питания 5 В, а самый светлый провод (*управляющий провод*) — к контакту цифрового выхода на плате. Если у вас другой сервопривод, схему его подключения ищите в сопроводительной документации.

Управление сервоприводом

Теперь попробуем управлять сервоприводом. Следующий скетч выполняет поворот сервопривода, используя весь доступный диапазон. Подключите сервопривод к плате Arduino, как описано выше, при этом управляющий провод соедините с цифровым контактом 4, а затем введите и загрузите скетч из листинга 12.1.

Листинг 12.1. Скетч, демонстрирующий работу сервопривода

```
// Листинг 12.1

#include <Servo.h>
Servo myservo;

void setup()
{
  myservo.attach(4);
}

void loop()
{
  myservo.write(180);
  delay(1000);
  myservo.write(90);
  delay(1000);
  myservo.write(0);
  delay(1000);
}
```

Этот скетч использует библиотеку *Servo*, поставляемую в комплекте Arduino IDE, и создает экземпляр класса *Servo*:

```
#include <Servo.h>
Servo myservo;
```

Затем, в функции `void setup()`, этому экземпляру сообщается, какой контакт на плате Arduino должен использоваться для управления сервоприводом:

```
myservo.attach(4); // контакт 4 управляет сервоприводом
```

Теперь сервопривод поворачивается простой командой:

```
myservo.write(x);
```

где `x` — целое число в диапазоне от 0 до 180 градусов — угол, на который должен быть повернут вал сервопривода. В ходе работы скетча из листинга 12.1 сервопривод будет поворачивать вал в одну и в другую сторону, приостанавливаясь в крайних позициях (0 градусов и 180 градусов) и в середине (90 градусов). Наблюдая за работой сервопривода, обратите внимание, что позиция, соответствующая углу поворота 180 градусов, находится слева, позиция 0 градусов — справа.

Помимо перемещения объектов, сервоприводы можно также использовать для сообщения информации, как это делают аналоговые индикаторы. Например, на основе сервопривода можно сконструировать аналоговый термометр, как показано в проекте 38.

Проект № 38: Аналоговый термометр

На основе сервопривода и температурного датчика TMP36, знакомого нам по предыдущим главам, мы сконструируем аналоговый термометр. Этот термометр будет измерять температуру и преобразовывать ее в угол поворота от 0 до 180 градусов, соответствующий шкале температуры от 0 °C до +30 °C. Сервопривод будет поворачиваться на угол, соответствующий текущей температуре.

Оборудование

Набор требуемого оборудования минимален:

- Один температурный датчик TMP36.
- Одна макетная плата.
- Один небольшой сервопривод.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Схема устройства также очень проста, это отображено на рис. 12.3.

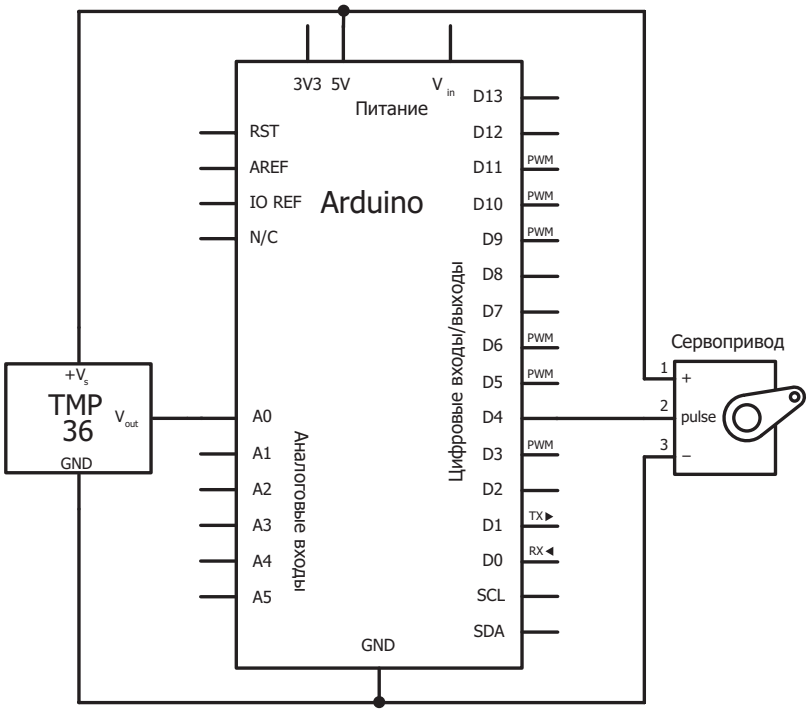


Рис. 12.3. Принципиальная схема для проекта 38

Скетч

Скетч определяет температуру, используя тот же прием, что был представлен в проекте 8. Затем значение температуры преобразуется в угол поворота сервопривода.

Введите и загрузите следующий скетч:

```
// Проект 38 - Аналоговый термометр

float voltage = 0;
float  sensor = 0;
float currentC = 0;
int   angle = 0;

#include <Servo.h>
Servo myservo;
```

```
void setup()
{
  myservo.attach(4);
}

int calculateservo(float temperature)
{
  float resulta;
  int resultb;
  resulta = -6 * temperature;
  resulta = resulta + 180;
  resultb = int(resulta);
  return resultb;
}

void loop()
{
  // прочитать текущую температуру
  sensor = analogRead(0);
  voltage = (sensor*5000)/1024;
  voltage = voltage-500;
  currentC = voltage/10;

  // преобразовать температуру в угол поворота
  angle = calculateservo(currentC);

  // выполнить поворот сервопривода
  if (angle >= 0 && angle <= 180)
  {
    myservo.write(angle); // повернуть на угол angle
    delay(1000);
  }
}
```

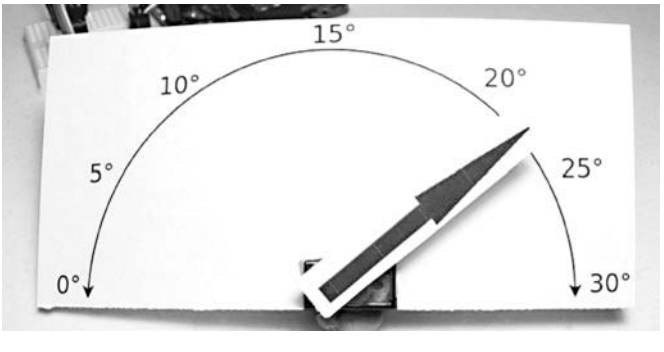


Рис. 12.4. Шкала, помогающая определить температуру, отображаемую термометром

Большая часть скетча должна быть понятна, единственное новое здесь — это функция `calculateservo()`. Данная функция преобразует температуру в угол поворота сервопривода по следующей формуле:

$$\text{угол} = (-6 \times \text{температура}) + 180$$

Возможно, вы захотите добавить шкалу с диапазоном температур и маленькую стрелку для наглядности. Пример такой шкалы показан на рис. 12.4.

Электродвигатели

Следующий шаг в нашем путешествии — управление электродвигателями. Маленькие электродвигатели имеют широкую область применения, от небольших вентиляторов до игрушечных автомобилей и моделей железной дороги. Так же как и в случае с сервоприводами, при выборе электродвигателя необходимо учитывать несколько параметров:

- **Рабочее напряжение** — может изменяться в диапазоне от 3 В до более чем 12 В.
- **Ток без нагрузки** — ток, потребляемый электродвигателем при рабочем напряжении, когда вал вращается свободно, в отсутствие любых механизмов, связанных с ним.
- **Пусковой ток** — ток, необходимый, чтобы провернуть вал двигателя при его запуске.
- **Скорость вращения при рабочем напряжении** — скорость вращения вала в оборотах в минуту (об/мин).

В нашем примере будет использоваться маленький и недорогой электродвигатель со скоростью вращения 8540 об/мин при рабочем напряжении 3 В, подобный тому, что изображен на рис. 12.5.

Рис. 12.5. Маленький электродвигатель

Управление электродвигателем будет осуществляться с помощью транзистора, как описывалось в главе 3. Так как электродвигатель потребляет ток до 0,7 А (больше, чем может пропустить транзистор BC548), для данного проекта будет использоваться составной транзистор, который называют транзистором Дарлингтона, или парой Дарлингтона.

Транзистор Дарлингтона TIP120

Транзистор Дарлингтона (пара Дарлингтона) может пропускать большой ток с высоким напряжением. Транзистор Дарлингтона TIP120 способен пропускать ток до 5 А с напряжением 60 В, этого более чем достаточно для управления нашим маленьким электродвигателем. Для обозначения транзистора TIP120 на принципиальных схемах используется значок, напоминающий обозначение транзистора BC548, как показано на рис. 12.6, но TIP120 имеет большие размеры, чем BC548.

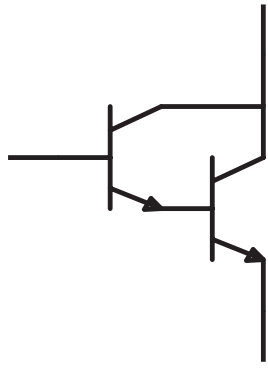


Рис. 12.6. Обозначение транзистора TIP120

Транзистор TIP120 выпускается в корпусе TO-220, как показано на рис. 12.7.

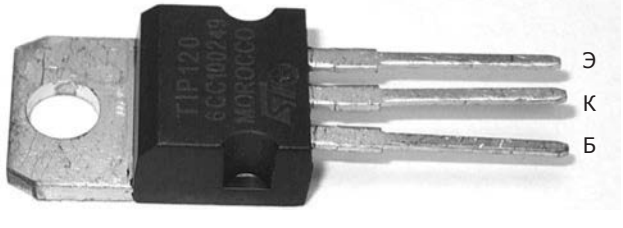


Рис. 12.7. Транзистор TIP120

Если смотреть на TIP120 со стороны маркировки, его выводы будут иметь следующее назначение (слева направо): база (Б), коллектор (К) и эмиттер (Э). Металлический язычок, предназначенный для крепления транзистора на радиаторе, соединен с коллектором.

Проект № 39: Управление электродвигателем

В этом проекте мы научимся регулировать скорость вращения электродвигателя.

Оборудование

Ниже перечислено оборудование, которое понадобится для этого проекта:

- Один маленький электродвигатель с рабочим напряжением 3 В.
- Один резистор с номиналом 1 кОм (R1).
- Одна макетная плата.
- Один диод 1N4004.
- Один транзистор Дарлингтона TIP120.
- Отдельный источник питания с напряжением 3 В.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Для работы с электродвигателями необходим отдельный источник питания, потому что плата Arduino не способна отдавать большие токи. Если электродвигатель остановится, для повторного запуска он может потребовать *пускового тока* силой более 1 А. Это больше, чем способна отдать плата Arduino, а если попытаться получить с платы такой ток, она может выйти из строя.

Простейшее решение — использование батарейного контейнера. Для питания электродвигателя с рабочим напряжением 3 В достаточно контейнера на две батарейки типа AA, такой контейнер изображен на рис. 12.8.

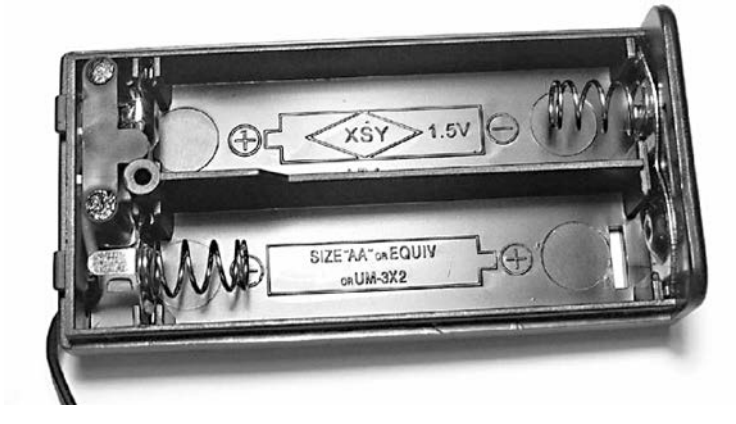


Рис. 12.8. Батарейный контейнер на две батарейки типа AA

Схема

Соберите схему, изображенную на рис. 12.9.

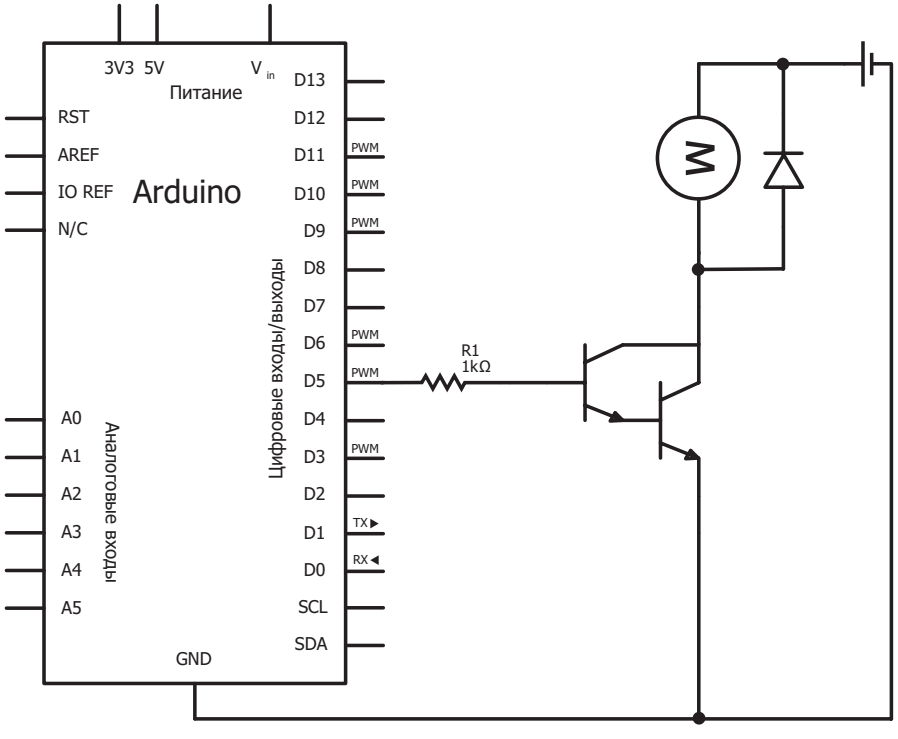


Рис. 12.9. Принципиальная схема для проекта 39

Скетч

В этом проекте мы будем регулировать скорость вращения электродвигателя от нуля (полная остановка) до максимального числа оборотов в минуту и затем обратно до полной остановки. Введите и загрузите следующий скетч:

```
// Проект 39 - Управление электродвигателем

void setup()
{
  pinMode(5, OUTPUT);
}

void loop()
{
```

```
❶   for (int a=0; a<256; a++)
    {
      analogWrite(5, a);
❷    delay(100);
    }
❸   delay(5000);
❹   for (int a=255; a>=0; a--)
    {
      analogWrite(5,a);
      delay(100);
    }
    delay(5000);
  }
```

Управление скоростью вращения двигателя осуществляется с применением технологии широтно-импульсной модуляции (как описывалось в проекте 3). Напомню, что эта технология поддерживается только цифровыми контактами 3, 5, 6, 9, 10 и 11. При использовании такой технологии ток будет подаваться на электродвигатель короткими импульсами: чем длиннее импульсы, тем выше скорость вращения, поскольку в единицу времени электродвигатель дольше будет находиться во включенном состоянии. В начале первого цикла `for` (❶) электродвигатель выключается, а затем напряжение, подаваемое на него, начинает постепенно увеличиваться; ускорением управляет задержка в ❷. В точке ❸ скорость вращения электродвигателя достигает максимума и удерживается на таком уровне в течение 5 секунд. В следующем цикле `for` (❹) выполняется обратный процесс, и электродвигатель останавливается.

ПРИМЕЧАНИЕ

Перед началом вращения электродвигателя вы можете услышать гудение, исходящее от него и напоминающее звук, издаваемый трамваем или электровозом, когда он трогается с места. Это нормально и не является поводом для волнения.

Диод на этой схеме используется с той же целью, что и на схеме подключения управляющего реле (рис. 3.19), — для защиты схемы. Когда с электродвигателя снимается напряжение, возникает кратковременный всплеск паразитного тока высокого напряжения на его обмотке, который должен быть куда-то направлен. Диод пропускает паразитный ток по кругу через обмотку электродвигателя, пока он не будет рассеян в виде небольшого количества тепла.

Проект № 40: Роботизированный танк и управление им

Умение управлять скоростью вращения одного электродвигателя может пригодиться при реализации проектов, но я предлагаю заняться кое-чем поинтереснее — управлением скоростью *и* направлением вращения сразу двух электродвигателей.

Наша цель — описать конструкцию роботизированного танка, над которым мы продолжим работать в нескольких следующих главах. Здесь будет описана конструкция и основы управления танком.

Наш танк имеет два электродвигателя, каждый из которых управляет своей гусеницей. Он может преодолевать небольшие препятствия, разворачиваться на месте и не врезаться в препятствия, встречающиеся на пути. Вы научитесь управлять скоростью и направлением движения, а также узнаете, как добавить в его конструкцию новые компоненты, предотвращающие столкновения и обеспечивающие возможность дистанционного управления. Опробовав все проекты из этой книги, вы получите знания и навыки, необходимые для создания собственных версий и воплощения новых идей.

Оборудование

Ниже перечислено оборудование, необходимое для этого проекта:

- Одно шасси Pololu RP5 для модели танка.
- Одна монтажная плата для шасси Pololu RP5.
- Шесть алкалиновых батареек типа AA.
- Одна 9-вольтовая батарея с разъемом для подключения к разъему питания на плате Arduino.
- Плата расширения DFRobot 2A Motor Shield для управления электродвигателями.
- Плата Arduino и кабель USB.

Шасси

Основой любого робота является шасси с электродвигателями, трансмиссией и источником электропитания. Робот, управляемый платой Arduino, также должен иметь место для размещения платы и различных дополнительных компонентов.

На рынке имеется большое разнообразие шасси для моделирования, но мы будем использовать шасси для модели танка из серии Pololu RP5, изображенное на рис. 12.10 и включающее два электродвигателя. Вы можете также использовать шасси Dagu Rover 5 от компании Pololu или других производителей.

Два источника питания

В комплект с шасси Pololu входит контейнер на шесть батарей типа AA, который мы будем использовать в качестве источника питания для электродвигателей, как показано на рис. 12.11. Контейнер для батарей располагается на дне шасси между электродвигателями и обеспечивает низкое расположение центра тяжести робота.

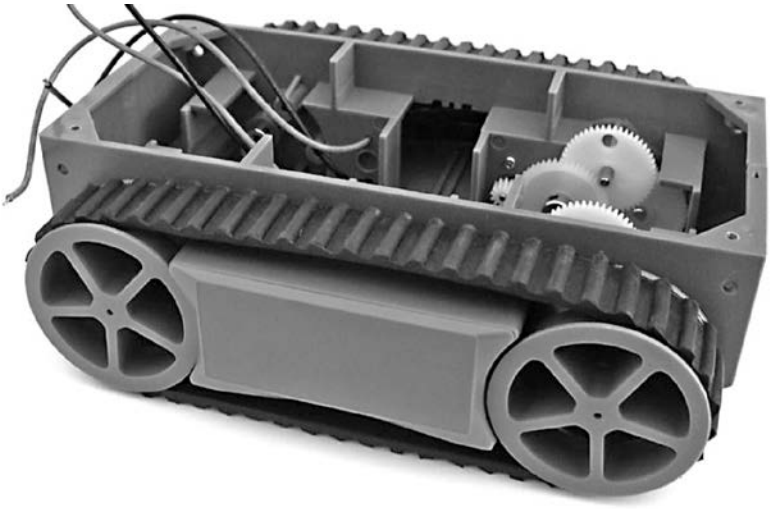


Рис. 12.10. Шасси для нашего танка

Рис. 12.11. Контейнер с шестью батареями типа AA

Несмотря на то что источник питания на рис. 12.11 достаточно мощный, для платы Arduino необходимо предусмотреть отдельный источник питания, потому что это позволит скетчу продолжить выполнение даже после того, как сядут батареи, питающие электродвигатели. В этом проекте плата Arduino будет питаться от 9-вольтовой батареи, подключенной к разъему питания на плате с помощью кабеля, изображенного на рис. 12.12.

Монтажная плата

Последний компонент, входящий в комплект шасси, — *монтажная плата*, изображенная на рис. 12.13.

Монтажная плата укладывается сверху на шасси и служит для монтирования компонентов с использованием стоек и *винтов* M3. (Винты, стойки и гайки можно приобрести в хозяйственных магазинах или в магазинах для моделистов-конструкторов.) На рис. 12.14 показана монтажная плата с уже смонтированной на ней платой Arduino.

Рис. 12.12. Кабель для подключения батареи к плате Arduino

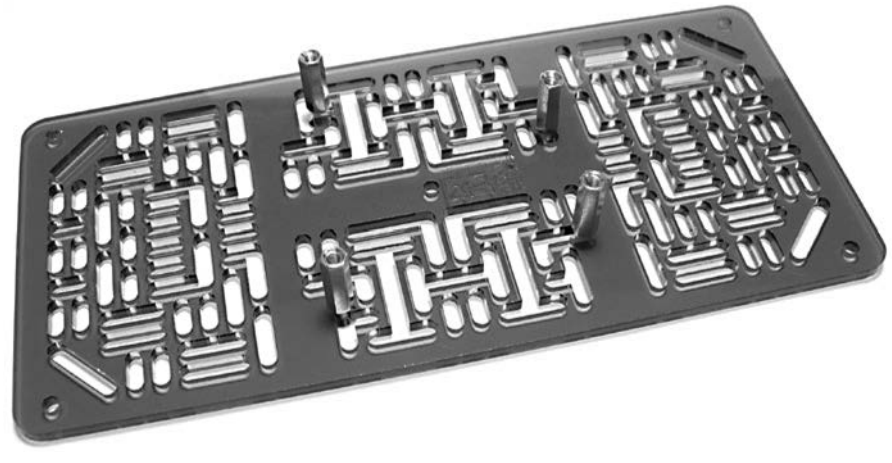


Рис. 12.13. Монтажная плата

Схема

Наконец, необходимо собрать схему управления двумя электродвигателями на шасси. Можно было бы для каждого электродвигателя использовать схему, изображенную на рис. 12.9, но она не дает возможности управлять направлением вращения электродвигателей и к тому же несколько неудобна в сборке. Поэтому мы воспользуемся специальной *платой расширения, предназначенной для управления электродвигателями*. Плата расширения содержит все необходимое для подачи большого тока на электродвигатели, а кроме того, «умеет» принимать от платы Arduino команды управления скоростью и направлением вращения обоих

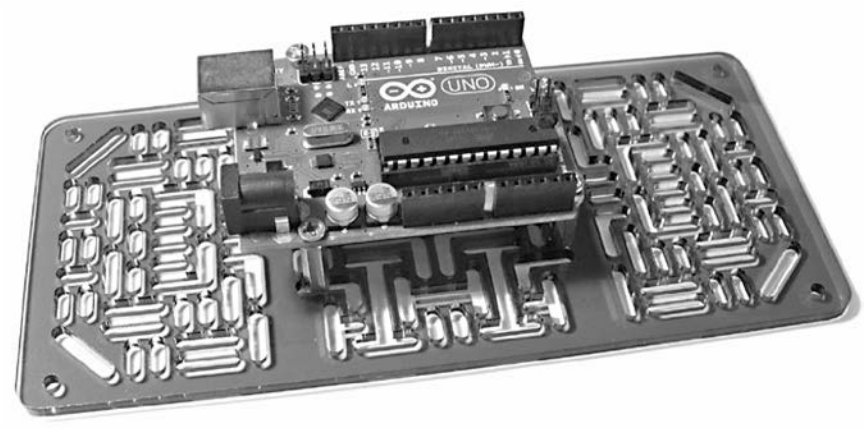


Рис. 12.14. Плата Arduino смонтирована на монтажной плате

электродвигателей. В нашей модели танка будет использоваться плата 2A Motor Shield для Arduino от компании DFRobot (*http://www.dfrobot.com/*), изображенная на рис. 12.15.

Рис. 12.15. Плата управления электродвигателями компании DFRobot

Подключение платы управления электродвигателями

Плата управления электродвигателями подключается просто: подключите провода, идущие от контейнера с батареями, к блоку контактов, расположенному на плате слева внизу, как показано на рис. 12.16. Черный провод (минус) должен подключаться к правому контакту, а красный (плюс) — к левому.

Затем подключите две пары проводов, идущие от электродвигателей. Обратите внимание, что цвет проводов должен соответствовать контактам, как показано на рис. 12.17.

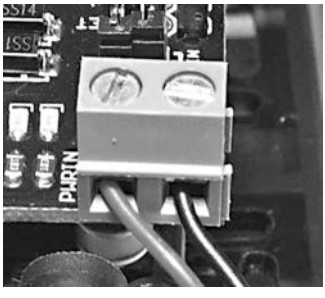


Рис. 12.16. Подключение проводов питания

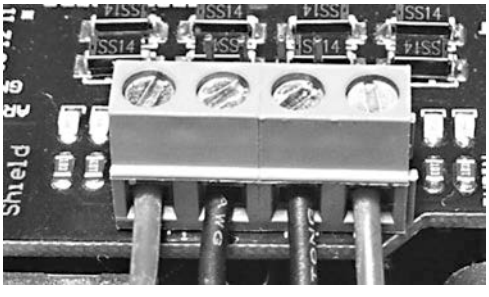


Рис. 12.17. Подключение электродвигателей

Установка перемычек

Последним шагом подготовки платы управления электродвигателями является установка перемычек. Внимательно посмотрите на плату: между контактами питания и нижним рядом контактов с меткой Power (Питание) находится колодка с шестью контактами и двумя черными перемычками. Вставьте перемычки горизонтально, так, чтобы они замыкали четыре контакта слева, как показано на рис. 12.18. Наконец, установите четыре перемычки вертикально, соединив контакты с меткой PWM (ШИМ), как показано на рис. 12.19.

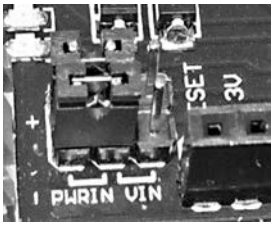


Рис. 12.18. Установка перемычек на шине питания

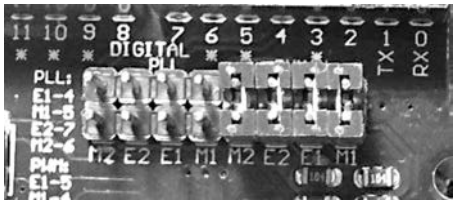


Рис. 12.19. Установка перемычек на шине управления электродвигателями

Если провода, идущие от электродвигателей, одного цвета, вам может понадобиться поменять их местами в контактах на плате, если после первой попытки двигатели вращаются не в том направлении, в котором нужно.

После подключения проводов и установки перемычек вставьте батарейки в контейнер, смонтируйте плату Arduino и плату расширения на монтажной плате и закрепите всю эту конструкцию на шасси. Ваш танк должен выглядеть, как показано на рис. 12.20.

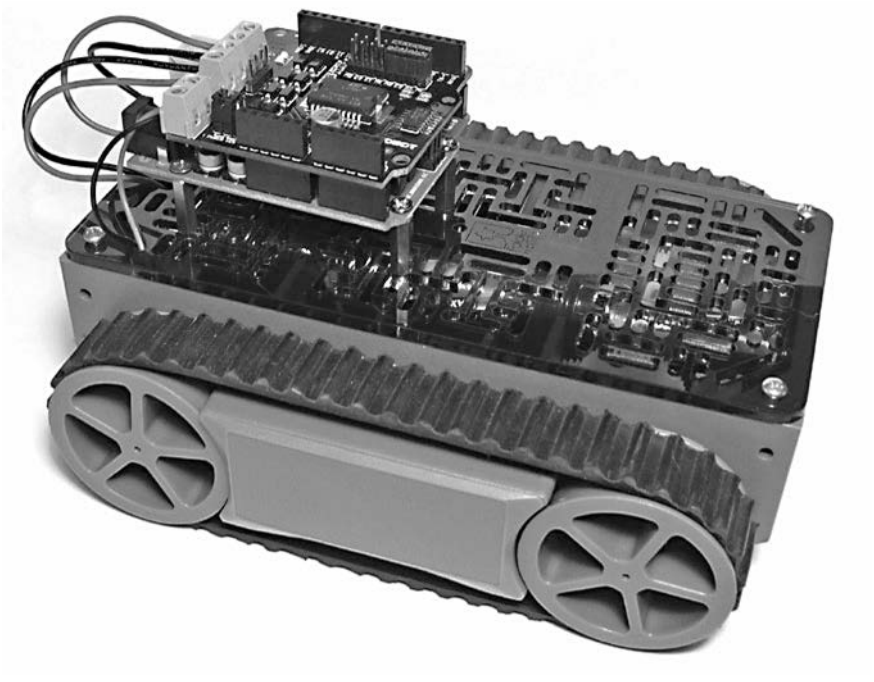


Рис. 12.20. Танк готов к испытаниям!

Скетч

Теперь заставим танк двигаться. Сначала создадим несколько функций, которые потом упростят управление движением. Так как танк приводится в движение двумя электродвигателями, нам потребуется реализовать четыре вида движений:

- движение вперед;
- движение назад;
- поворот по часовой стрелке;
- поворот против часовой стрелки.

Наша плата расширения управляет каждым двигателем с помощью двух цифровых выходов: один регулирует скорость вращения с применением ШИМ (как демонстрировалось в проекте 39), а другой определяет направление вращения.

Четырем видам движения в скетче соответствуют четыре функции: `goForward()`, `goBackward()`, `rotateLeft()` и `rotateRight()`. Каждая принимает значение в миллисекундах, определяющее время, в течение которого будут включены электродвигатели, и скорость в виде значения ШИМ от 0 до 255. Например, чтобы заставить танк двигаться вперед на полной скорости в течение 2 секунд, нужно выполнить вызов `goForward(2000, 255)`.

Введите и сохраните следующий скетч (но пока не загружайте его):

```
// Проект 40 - Роботизированный танк и управление им
int m1speed=6; // цифровые выходы, управляющие скоростью
int m2speed=5;
int m1direction=7; // цифровые выходы, управляющие направлением
int m2direction=4;

void setup()
{
  pinMode(m1direction, OUTPUT);
  pinMode(m2direction, OUTPUT);
  delay(5000);
}

void goForward(int duration, int pwm)
{
❶  digitalWrite(m1direction,HIGH); // вперед
  digitalWrite(m2direction,HIGH); // вперед
  analogWrite(m1speed, pwm);      // скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);        // скорость
  analogWrite(m2speed, 0);
}

void goBackward(int duration, int pwm)
{
  digitalWrite(m1direction,LOW); // назад
❷  digitalWrite(m2direction,LOW); // назад
  analogWrite(m1speed, pwm);     // скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);       // скорость
  analogWrite(m2speed, 0);
}

void rotateRight(int duration, int pwm)
{
❸  digitalWrite(m1direction,HIGH); // вперед
  digitalWrite(m2direction,LOW);  // назад
  analogWrite(m1speed, pwm);      // скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);        // скорость
  analogWrite(m2speed, 0);
}

void rotateLeft(int duration, int pwm)
{
❹  digitalWrite(m1direction,LOW);  // назад
  digitalWrite(m2direction,HIGH); // вперед
  analogWrite(m1speed, pwm);      // скорость
  analogWrite(m2speed, pwm);
```

```
    delay(duration);
    analogWrite(m1speed, 0);        // скорость
    analogWrite(m2speed, 0);
  }

  void loop()
  {
    goForward(1000, 255);
    rotateLeft(1000, 255);
    goForward(1000, 255);
    rotateRight(1000, 255);
    goForward(1000, 255);
    goBackward(2000, 255);
    delay(2000);
}
```

Направление вращения каждого электродвигателя устанавливается вызовом:

```
digitalWrite(m1direction,direction);
```

Значение `HIGH` в параметре `direction` соответствует вращению в прямом направлении, а `LOW` — в обратном. То есть, чтобы танк двигался вперед, необходимо установить одно и то же направление для обоих электродвигателей, что и делается в строках ❶ и ❷. Скорость вращения устанавливается вызовом:

```
analogWrite(m1speed, pwm);
```

Параметр `pwm` определяет скорость и может принимать значения от 0 до 255. Чтобы танк повернул влево или вправо, электродвигатели должны вращаться в противоположных направлениях, как это делается в строках ❸ и ❹.

ВНИМАНИЕ

Когда вы будете готовы загрузить скетч, поднимите танк над поверхностью стола, чтобы его гусеницы свободно вращались в воздухе; если этого не сделать, то после загрузки скетча танк рванется вперед и может упасть со стола!

Загрузите скетч, отсоедините кабель USB и включите кабель от 9-вольтовой батареи в гнездо питания на плате Arduino. Затем поместите танк на ковер или другую поверхность и отпустите его. Поэкспериментируйте с функциями движения в проекте 40; это поможет вам лучше понять суть задержек и их влияние на величину пройденного расстояния.

Определение столкновений

Теперь, научив наш танк двигаться, мы добавим дополнительные средства, такие как датчики определения, которые сообщат танку о произошедшем столкновении или измерят расстояние между танком и объектом на его пути, чтобы танк мог свернуть

и избежать столкновения. Мы будем использовать три способа предотвращения столкновений: микровыключатели, инфракрасные и ультразвуковые датчики.

Проект № 41: Определение столкновений с помощью микровыключателя

Микровыключатель действует как обычная кнопка без фиксации, которую мы использовали в главе 4; отличие в том, что микровыключатель имеет большие размеры и длинную металлическую пластину, играющую роль рычага (рис. 12.21).

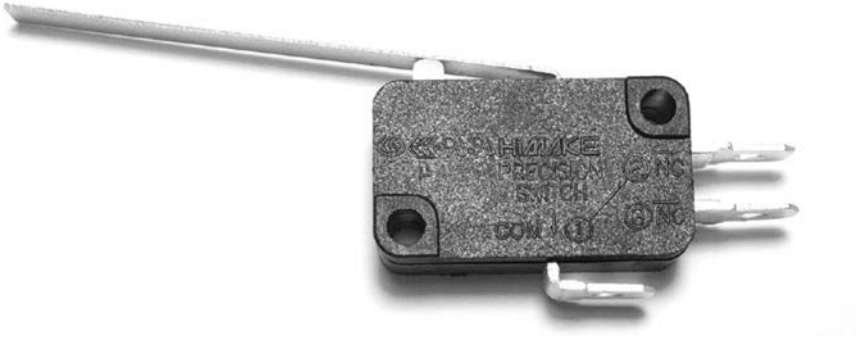


Рис. 12.21. Микровыключатель

Обычно при использовании микровыключателя один провод подключается к нижнему выводу, а другой — к выводу с меткой NO (normally open — нормально разомкнутый), чтобы ток тек только при нажатом рычаге. Мы смонтируем микровыключатель на передней панели танка, и когда танк достигнет какого-то препятствия, рычаг замкнет микровыключатель, ток потечет через его выводы и заставит танк изменить направление движения.

Схема

Подсоедините микровыключатель как обычную кнопку, как показано на рис. 12.22.

Скетч

Мы подключили микровыключатель к цифровому контакту 2, поддерживающему прерывания. Может показаться, что в функции, вызываемой по прерываниям, мы должны заставить танк двигаться в обратном направлении в течение какого-то времени, однако это невозможно, потому что функция `delay()` не работает внутри обработчиков прерываний. В данном случае необходимо найти какое-то другое решение.

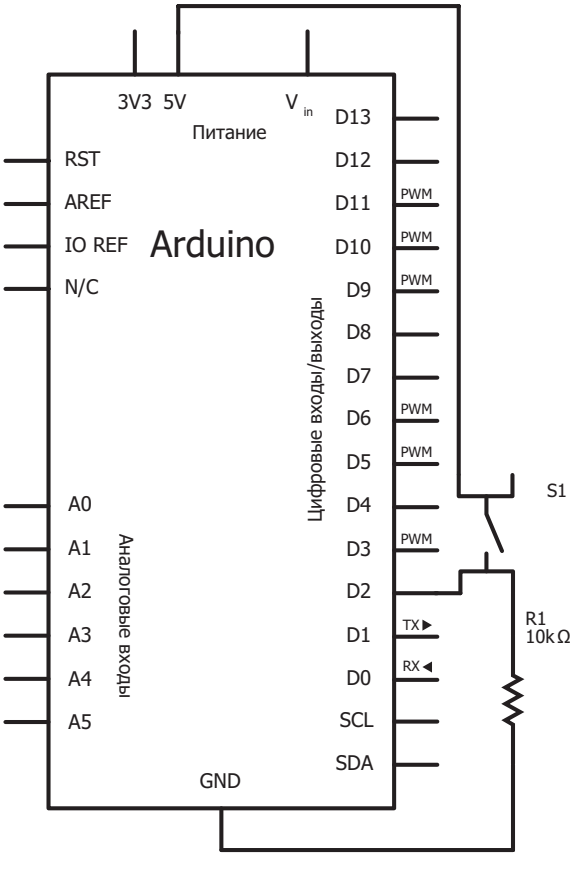


Рис. 12.22. Схема подключения микровыключателя для определения столкновений

Это решение заключается в следующем: функция `goForward()` будет включать электродвигатели только при соблюдении двух условий, задаваемых логическими переменными `crash` и `move`. Если `crash` имеет значение `true`, электродвигатели будут на 2 секунды включаться в обратном направлении с низкой скоростью, чтобы «выйти» из соприкосновения с препятствием.

Мы не можем вызывать функцию `delay()` из-за использования прерывания, поэтому время работы электродвигателей будет измеряться следующим образом: перед включением будет вызываться `millis()`, чтобы засечь момент включения двигателей, и затем в цикле каждое новое ее значение будет сравниваться с начальным. Когда разность сравняется или превысит требуемую продолжительность, функция присвоит переменной `move` значение `false` и остановит электродвигатели.

Введите и загрузите следующий скетч:

```
// Проект 41 – Определение столкновений с помощью микровыключателя

int m1speed=6;   // цифровые выходы, управляющие скоростью
int m2speed=5;
int m1direction=7; // цифровые выходы, управляющие направлением
int m2direction=4;
boolean crash=false;

void setup()
{
  pinMode(m1direction, OUTPUT);
  pinMode(m2direction, OUTPUT);
  attachInterrupt(0, backOut, RISING);
  delay(5000);
}

❶ void backOut()
{
  crash=true;
}

❷ void backUp()
{
  digitalWrite(m1direction,LOW); // назад
  digitalWrite(m2direction,LOW); // назад
  analogWrite(m1speed, 200);     // скорость
  analogWrite(m2speed, 200);
  delay(2000);
  analogWrite(m1speed, 0);       // скорость
  analogWrite(m2speed, 0);
}

void goForward(int duration, int pwm)
{
  long a,b;
  boolean move=true;

❸  a=millis();
  do
  {
    if (crash==false)
    {
      digitalWrite(m1direction,HIGH); // вперед
      digitalWrite(m2direction,HIGH); // вперед
      analogWrite(m1speed, pwm);      // скорость
      analogWrite(m2speed, pwm);
    }
    if (crash==true)
    {
      backUp();
      crash=false;
    }
```

```
❹     b=millis()-a;
      if (b>=duration)
      {
        move=false;
      }
    } while (move!=false);

    // остановить электродвигатели
    analogWrite(m1speed, 0);
    analogWrite(m2speed, 0);
  }

  void loop()
  {
    goForward(5000, 255);
    delay(2000);
  }
```

В этом скетче реализован усовершенствованный алгоритм движения вперед: управление движением осуществляется с применением двух переменных. Первая — логическая переменная `crash`. Если танк наткнулся на какое-то препятствие и включил микровыключатель, возникает прерывание, для обработки которого вызывается функция `backOut()` (❶). Она присваивает переменной `crash` значение `true`. Вторая переменная, управляющая движением, — логическая переменная `move`. Функция `goForward()` вызывает `millis()` (❸), чтобы засечь время начала движения, по которому потом будет определяться момент, когда время движения танка истекло (устанавливается параметром `duration`).

В строке ❹ функция вычисляет время, прошедшее с начала движения, и если оно меньше заданной продолжительности, в переменной `move` сохраняется значение `true`. То есть танку разрешается продолжить движение вперед, только если он не столкнулся с препятствием и время движения не истекло. Если обнаружено столкновение, вызывается функция `backUp()` (❷), и танк начинает медленно двигаться в обратном направлении в течение 2 секунд, а затем продолжает движение как обычно.

ПРИМЕЧАНИЕ

Чтобы расширить или изменить этот пример, попробуйте добавить в него другие функции движения из проекта 40.

Инфракрасный датчик расстояния

Наш следующий метод предотвращения столкновений основан на применении инфракрасного (ИК) датчика расстояния. Этот датчик фиксирует инфракрасное излучение, отраженное от поверхности перед ним, и возвращает напряжение, пропорциональное расстоянию между датчиком и поверхностью. Инфракрасные датчики расстояния удобно использовать для определения столкновений, потому

что они недороги, но они не годятся для точного измерения расстояний. В следующем проекте мы будем использовать аналоговый датчик Sharp GP2Y0A21YK0F, изображенный на рис. 12.23.

Рис. 12.23. ИК-датчик Sharp

Подключение

Чтобы подключить датчик к плате Arduino, соедините красный и черный провода, идущие от датчика, к контактам 5V и GND соответственно, а белый провод — к аналоговому входу. Для измерения напряжения, возвращаемого датчиком, мы будем использовать функцию `analogRead()`. График на рис. 12.24 показывает зависимость между расстоянием и выходным напряжением.

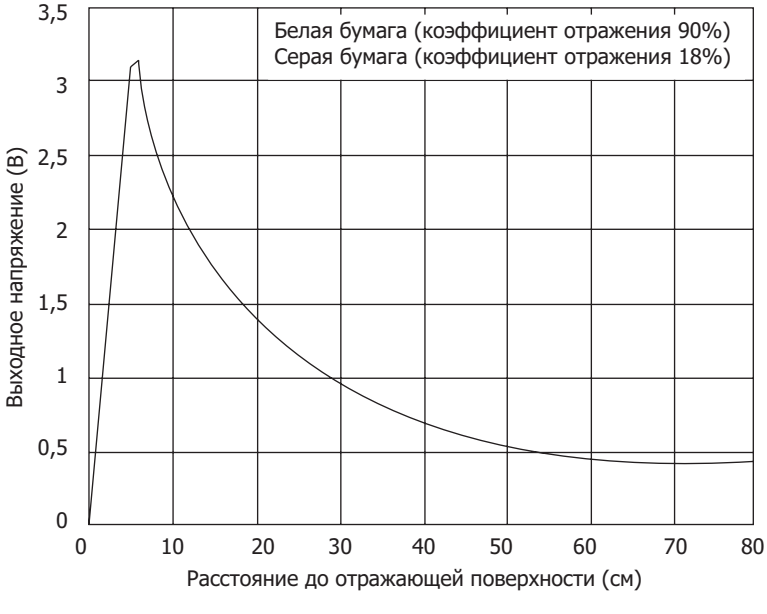


Рис. 12.24. График зависимости выходного напряжения на ИК-датчике от расстояния (линии для белой и серой бумаги совпадают)

Тестирование ИК-датчика расстояния

Поскольку зависимость между расстоянием и выходным напряжением трудно представить в виде уравнения, мы разобьем график на 5-сантиметровые отрезки. Рассмотрим, как это делается, на простом примере. Подключите белый провод инфракрасного датчика к аналоговому контакту 0, красный провод — к контакту 5V и черный провод — к контакту GND, а затем введите и загрузите скетч, представленный в листинге 12.2.

Листинг 12.2. Скетч, демонстрирующий использование ИК-датчика

```
// Листинг 12.2

float sensor = 0;
int cm = 0;

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

void loop()
{
❶  sensor = analogRead(0);
❷  if (sensor<=90)
   {
     Serial.println("Infinite distance!");
   } else if (sensor<100) // 80 см
   {
     cm = 80;
   } else if (sensor<110) // 70 см
   {
     cm = 70;
   } else if (sensor<118) // 60 см
   {
     cm = 60;
   } else if (sensor<147) // 50 см
   {
     cm = 50;
   } else if (sensor<188) // 40 см
   {
     cm = 40;
   } else if (sensor<230) // 30 см
   {
     cm = 30;
   } else if (sensor<302) // 25 см
   {
     cm = 25;
   } else if (sensor<360) // 20 см
   {
     cm = 20;
   } else if (sensor<505) // 15 см
   {
```

```
    cm = 15;
  } else if (sensor<510) // 10 см
  {
    cm = 10;
  } else if (sensor>=510) // слишком близко!
  {
    Serial.println("Too close!");
  }
  Serial.print("Distance: ");
  Serial.print(cm);
  Serial.println(" cm");
  delay(250);
}
```

Скетч читает напряжение с ИК-датчика (❶) и затем с помощью последовательности инструкций `if` (❷) определяет примерное расстояние. Расстояние вычисляется по напряжению, возвращаемому датчиком, с учетом зависимости напряжения от расстояния, изображенной на рис. 12.24, и знания того факта (из проекта 6), что `analogRead()` возвращает значение между 0 и 1023, представляющее напряжение в диапазоне от 0 до 5 В.

Загрузив скетч, откройте окно монитора порта и поэкспериментируйте, приближая руку или лист бумаги к датчику или отдаляя от него. В окне монитора порта должна при этом выводиться оценка расстояния, как показано на рис. 12.25.

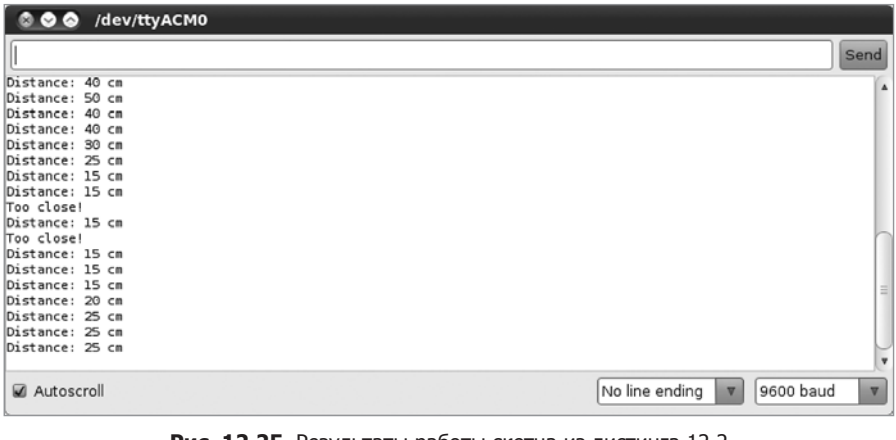


Рис. 12.25. Результаты работы скетча из листинга 12.2

Проект № 42: Определение столкновений с помощью ИК-датчика расстояния

Теперь заменим микровыключатель инфракрасным датчиком расстояния. Далее будем использовать немного измененную версию проекта 41. Вместо функции

обработки прерывания добавим в скетч функцию `checkDistance()`, которая будет присваивать переменой `crash` значение `true`, если расстояние до препятствия меньше 20 см. Эта функция вызывается в `goForward()`, в цикле `do... while`.

Смонтируйте ИК-датчик на танке и подключите его, затем введите и загрузите следующий скетч:

```
// Проект 42 - Определение столкновений с помощью ИК-датчика расстояния

int m1speed=6; // цифровые выходы, управляющие скоростью
int m2speed=5;
int m1direction=7; // цифровые выходы, управляющие направлением
int m2direction=4;
boolean crash=false;

void setup()
{
  pinMode(m1direction, OUTPUT);
  pinMode(m2direction, OUTPUT);
  delay(5000);
}

void backUp()
{
  digitalWrite(m1direction,LOW); // назад
  digitalWrite(m2direction,LOW); // назад
  analogWrite(m1speed, 200);     // скорость
  analogWrite(m2speed, 200);
  delay(2000);
  analogWrite(m1speed, 0);       // скорость
  analogWrite(m2speed, 0);
}

void checkDistance()
{
❶  if (analogRead(0)>360)
  {
    crash=true;
  }
}

void goForward(int duration, int pwm)
{
  long a,b;
  boolean move=true;
  a=millis();
  do
  {
    checkDistance();
    if (crash==false)
    {
```

```
      digitalWrite(m1direction,HIGH); // вперед
      digitalWrite(m2direction,HIGH); // вперед
      analogWrite(m1speed, pwm);      // скорость
      analogWrite(m2speed, pwm);
    }
    if (crash==true)
    {
      backUp();
      crash=false;
    }
    b=millis()-a;
    if (b>=duration)
    {
      move=false;
    }
  } while (move!=false);
  // остановить электродвигатели
  analogWrite(m1speed, 0);
  analogWrite(m2speed, 0);
}

void loop()
{
  goForward(5000, 255);
  delay(2000);
}
```

В этом скетче используется тот же алгоритм, что и в скетче из проекта 41, за исключением того, что данная версия постоянно проверяет расстояние (❶) и присваивает переменной `crash` значение `true`, если расстояние между ИК-датчиком и препятствием меньше примерно 20 см.

Запустив танк с установленным датчиком, вы наверняка обратили внимание на преимущество использования бесконтактного датчика столкновений. Теперь для вас не станет проблемой установка на танк дополнительных датчиков, например, спереди и сзади или по углам корпуса, а также добавление кода, проверяющего датчики по очереди и принимающего решения на основе информации о расстоянии до препятствия.

Ультразвуковой датчик расстояния

Заключительный способ предотвращения столкновений, который мы здесь рассмотрим, — применение *ультразвукового датчика расстояния*. Этот датчик посылает ультразвуковые импульсы (неслышимые человеческому уху) и измеряет время, за которое импульс вернется назад. В этом проекте будет использован ультразвуковой датчик расстояния Parallax Ping))), изображенный на рис. 12.26, потому что он стоит относительно недорого и имеет точность измерений до 1 см.

Указанная точность измерений обеспечивается ультразвуковым датчиком на расстоянии от 2 до 300 см. Однако из-за того, что звуковая волна должна отражаться от препятствий так, чтобы вернуться в датчик, угол между направлением движения и направлением датчика не должен превышать 45 градусов.

Рис. 12.26. Ультразвуковой датчик расстояния Parallax Ping)))

Подключение ультразвукового датчика

Присоедините выводы `5V` и `GND` датчика к одноименным контактам на плате Arduino, а вывод `SIG` (сокращенно от слова «signal» — сигнал) — к любому цифровому контакту Arduino.

Использование ультразвукового датчика

Ультразвуковой датчик выполняет измерения только по запросу. Чтобы произвести измерение, необходимо подать на контакт `SIG` короткий положительный импульс длительностью 5 микросекунд (мкс). Через мгновение датчик должен вернуть сигнал `HIGH`, длительность которого равна интервалу времени, потребовавшемуся ультразвуковому импульсу для достижения препятствия и возвращения обратно. Это значение следует разделить на два, чтобы получить фактическое расстояние между датчиком и препятствием.

Мы используем один и тот же цифровой контакт как вход и как выход и напишем две новые функции:

- `delayMicroseconds(mS)` — приостанавливает выполнение скетча на заданное число микросекунд (`mS`);
- `pulseDuration(pin, HIGH)` — измеряет длительность положительного импульса на цифровом входе `pin` и возвращает время в микросекундах.

После измерения продолжительность входящего импульса нужно преобразовать в сантиметры делением на 29,412 (потому что скорость звука составляет примерно 340 м/с, или 34 см/мс).

Тестирование ультразвукового датчика расстояния

Чтобы упростить использование датчика, мы будем использовать функцию `getDistance()` из листинга 12.3. Подключите выход SIG ультразвукового датчика к цифровому контакту 3, а затем введите и загрузите следующий скетч.

Листинг 12.3. Демонстрация ультразвукового датчика

```
// Листинг 12.3

int signal=3;

void setup()
{
  pinMode(signal, OUTPUT);
  Serial.begin(9600);
}

int getDistance()
// возвращает расстояние в см
// от датчика Ping))) до препятствия
{
  int distance;
  unsigned long pulseduration=0;

  // получить замер с датчика Ping)))
  // настроить контакт на работу в режиме выхода,
  // чтобы послать импульс
❶  pinMode(signal, OUTPUT);

  // установить низкий уровень напряжения
  digitalWrite(signal, LOW);
  delayMicroseconds(5);

❷  // послать положительный импульс длительностью 5 мкс,
  // чтобы активировать Ping)))
  digitalWrite(signal, HIGH);
  delayMicroseconds(5);
  digitalWrite(signal, LOW);

❸  // изменить режим работы цифрового контакта,
  // чтобы прочитать входящий импульс
  pinMode(signal, INPUT);

  // определить длительность входящего импульса
  pulseduration=pulseIn(signal, HIGH);

❹  // разделить длительность импульса пополам
  pulseduration=pulseduration/2;

❺  // преобразовать частное в сантиметры
```

```
    distance = int(pulseduration/29);
    return distance;
  }

  void loop()
  {
    Serial.print(getDistance());
    Serial.println(" cm ");
    delay(500);
}
```

Функция `int getDistance()` возвращает расстояние. В строках с ❶ по ❺ можно видеть, как на датчик посылается положительный импульс и затем измеряется продолжительность возвращаемого импульса, которая затем используется для вычисления расстояния.

Загрузив скетч, откройте окно монитора порта и поэкспериментируйте, приближая и отдаляя какой-нибудь объект к датчику или от него. В окне монитора порта должна при этом выводиться оценка расстояния в сантиметрах, как показано на рис. 12.27.

Рис. 12.27. Результаты работы скетча из листинга 12.3

Проект № 43: Определение столкновений с помощью ультразвукового датчика расстояния

Теперь, когда вы знаете, как действует датчик, попробуем использовать его на нашем танке.

Скетч

Для предотвращения столкновений мы применим функцию из листинга 12.3. Следующий скетч измеряет расстояние до препятствия и, если оно оказывается меньше 10 см, включает задний ход. Введите и загрузите скетч, чтобы убедиться, что он работает именно так, как описано:

```
// Проект 43 - Определение столкновений с помощью
// ультразвукового датчика расстояния

int m1speed=6; // цифровые выходы, управляющие скоростью
int m2speed=5;
int m1direction=7; // цифровые выходы, управляющие направлением
int m2direction=4;
int signal=3;
boolean crash=false;

void setup()
{
  pinMode(m1direction, OUTPUT);
  pinMode(m2direction, OUTPUT);
  pinMode(signal, OUTPUT);
  delay(5000);
  Serial.begin(9600);
}

int getDistance()
// возвращает расстояние в см
// от датчика Ping))) до препятствия
{
  int distance;
  unsigned long pulseduration=0;

  // получить замер с датчика Ping)))
  // настроить контакт на работу в режиме выхода,
  // чтобы послать импульс
  pinMode(signal, OUTPUT);

  // установить низкий уровень напряжения
  digitalWrite(signal, LOW);
  delayMicroseconds(5);

  // послать положительный импульс длительностью 5 мкс,
  // чтобы активировать Ping)))
  digitalWrite(signal, HIGH);
  delayMicroseconds(5);
  digitalWrite(signal, LOW);

  // изменить режим работы цифрового контакта,
  // чтобы прочитать входящий импульс
  pinMode(signal, INPUT);
```

```
  // определить длительность входящего импульса
  pulseduration=pulseIn(signal, HIGH);

  // разделить длительность импульса пополам
  pulseduration=pulseduration/2;

  // преобразовать частное в сантиметры
  distance = int(pulseduration/29);
  return distance;
}

void backUp()
{
  digitalWrite(m1direction,LOW);  // назад
  digitalWrite(m2direction,LOW);
  delay(2000);
  digitalWrite(m1direction,HIGH); // влево
  digitalWrite(m2direction,LOW);
  analogWrite(m1speed, 200);      // скорость
  analogWrite(m2speed, 200);
  delay(2000);
  analogWrite(m1speed, 0);        // скорость
  analogWrite(m2speed, 0);
}

void goForward(int duration, int pwm)
{
  long a,b;
  int dist=0;
  boolean move=true;
  a=millis();
  do
  {
    dist=getDistance();
    Serial.println(dist);
❶    if (dist<10) // если меньше 10 см до объекта
    {
      crash=true;
    }
    if (crash==false)
    {
      digitalWrite(m1direction,HIGH); // вперед
      digitalWrite(m2direction,HIGH); // вперед
      analogWrite(m1speed, pwm);      // скорость
      analogWrite(m2speed, pwm);
    }
    if (crash==true)
    {
      backUp();
      crash=false;
    }
    b=millis()-a;
```

```
      if (b>=duration)
      {
        move=false;
      }
    } while (move!=false);
    // остановить электродвигатели
    analogWrite(m1speed, 0);
    analogWrite(m2speed, 0);
}

void loop()
{
  goForward(1000, 255);
}
```

И вновь мы постоянно измеряем расстояние до препятствия (❶) и затем присваиваем переменной `crash` значение `true`, если расстояние между ультразвуковым датчиком и препятствием оказывается меньше 10 см. Наблюдение за тем, как танк, будто по волшебству, объезжает препятствия или состязается в сообразительности с домашними питомцами, может оказаться весьма увлекательным занятием.

Забегая вперед

В этой главе вы узнали, как добавить в проекты на основе Arduino возможность перемещаться в окружающем мире. С помощью одного или пары простых электродвигателей с платой управления вы сможете создавать устройства, способные перемещаться самостоятельно и даже уклоняться от столкновений. Для демонстрации диапазона возможностей мы использовали три типа датчиков с разной стоимостью и точностью измерений, полученные знания позволят вам принимать обоснованные решения исходя из потребностей и бюджета проекта.

Надеюсь, что теперь вы обрели достаточный опыт и получаете удовольствие от возможности проектировать и конструировать разнообразные устройства. Но это еще не конец. В следующей главе мы выйдем на улицу и испытаем возможность навигации по спутникам.

13 Arduino и GPS

В этой главе вы:

- ✓ узнаете, как подключить плату расширения GPS;
- ✓ создадите простой дисплей для отображения координат GPS;
- ✓ научитесь определять местоположение на карте по координатам GPS;
- ✓ создадите часы высокой точности;
- ✓ научитесь записывать последовательность координат объекта в процессе перемещения.

В этой главе вы узнаете, как с помощью недорогой платы расширения GPS определять местоположение, как создать часы высокой точности и как сохранить траекторию движения в карте памяти microSD, которую затем можно наложить на карту.

Что такое GPS?

Глобальная система определения местоположения (Global Positioning System, GPS) — это спутниковая навигационная система, принимающая данные со спутников на околоземной орбите с помощью приемника GPS на Земле, которые затем можно использовать для определения текущих координат в любой точке планеты. Возможно, вы уже знакомы с навигаторами GPS, используемыми в автомобилях и сотовых телефонах.

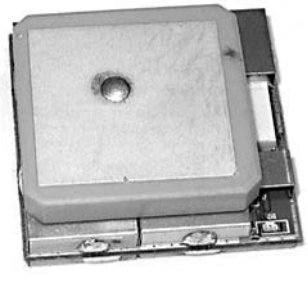


Рис. 13.1. GPS-приемник EM506

Несмотря на то что с применением Arduino нельзя создать навигационную систему с подробной картой, мы все же можем использовать модуль GPS и с его помощью определять географические координаты, время и примерную скорость перемещения (во время движения). Для этого мы воспользуемся GPS-приемником EM506, изображенным на рис. 13.1.

Чтобы задействовать этот приемник, нам потребуется конструктор платы расширения GPS от компании SparkFun (артикул KIT-13199), изображенный на рис. 13.2. В состав конструктора входит приемник, плата расширения, совместимая с Arduino, и соединительный кабель.

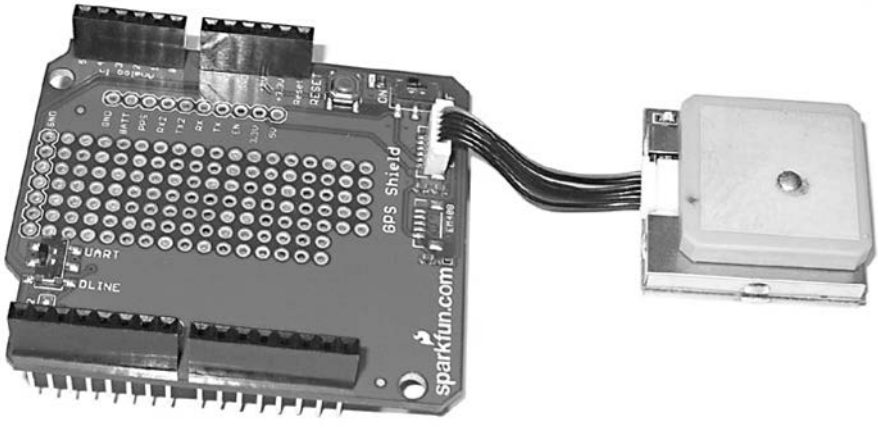


Рис. 13.2. Собранная плата расширения

Вам также нужно приобрести 30-сантиметровый кабель (производства SparkFun, артикул GPS-09123), изображенный на рис. 13.3, который упростит размещение приемника.

Рис. 13.3. Удлиненный кабель для подключения GPS-приемника к плате расширения

Тестирование платы расширения GPS

После покупки конструктора GPS убедитесь, что все детали исправны и обеспечивают прием сигналов GPS. Приемник GPS должен находиться на открытом пространстве, чтобы получать сигналы со спутников, которые, впрочем, могут проходить и через стекло. Тестирование лучше всего производить за пределами помещения или хотя бы у ничем не загороженного окна. Чтобы проверить прием сигналов, введите и загрузите простой скетч, отображающий принимаемые данные в необработанном виде.

Для тестирования подключите GPS-приемник посредством кабеля к плате расширения, а затем вставьте собранную конструкцию в разъемы на плате Arduino. Обратите внимание на маленькую перемычку на плате расширения GPS, изображенную на рис. 13.4.

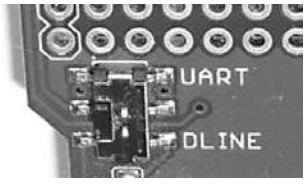


Рис. 13.4. Перемычка передачи данных на плате расширения GPS

Когда придет время загрузить скетч в плату расширения GPS, переместите перемычку в положение DLINE, а затем верните в положение UART и не забудьте включить питание приемника.

Введите и загрузите скетч из листинга 13.1.

Листинг 13.1. Простой скетч для тестирования

```
// Листинг 13.1

  void setup()
  {
❶   Serial.begin(4800);
  }

  void loop()
  {
    byte a;
❷   if ( Serial.available() > 0 )
    {
      a = Serial.read(); // извлечь байт данных из приемника GPS
❸     Serial.write(a);
    }
  }
```

Этот скетч «прослушивает» последовательный порт (❷) и, когда модуль присылает байт данных, выводит его в монитор порта (❸). (Обратите внимание, что последовательный порт настраивается на использование скорости 4800 бод (❶), которая совпадает со скоростью передачи данных приемником GPS.)

Загрузив скетч, верните перемычку в положение UART. Теперь взгляните на светодиод, находящийся на приемнике GPS. Если светодиод горит непрерывно, значит, приемник GPS пытается поймать сигналы от спутников. Спустя примерно 30 с светодиод должен начать мигать — это указывает на то, что приемник начал принимать данные. После того как светодиод начнет мигать, откройте окно последовательного порта в IDE и установите скорость обмена данными 4800 бод. После этого в окне отобразится непрерывный поток данных, как показано на рис. 13.5.

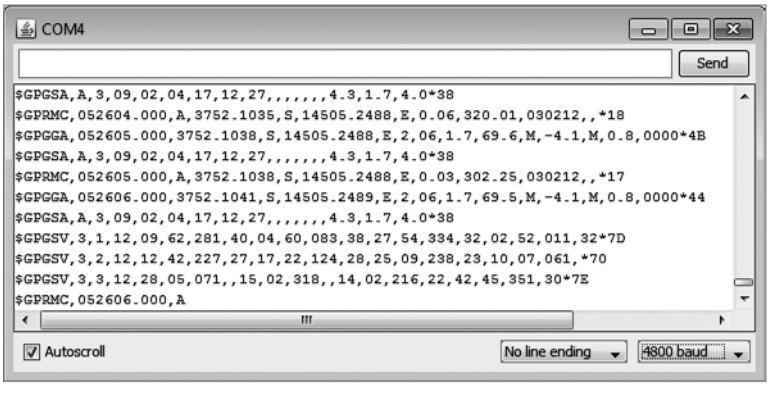


Рис. 13.5. Необработанные данные, полученные от приемника GPS

Приемник GPS посылает данные в плату Arduino по одному символу, которые тут же передаются в монитор порта. Но от таких необработанных данных мало проку, поэтому мы задействуем новую библиотеку и с ее помощью приведем эти данные в пригодный для использования вид. Для этого загрузите и установите библиотеку TinyGPS, доступную по адресу *http://www.arduiniana.org/libraries/tinygps/*, следуя процедуре, описанной в разделе «Расширение возможностей скетчей с помощью библиотек» в главе 8.

Проект № 44: Простой приемник GPS

Теперь создадим простой приемник GPS. Но сначала, так как устройства GPS обычно используются на открытом пространстве, а также для упрощения работы, добавим модуль ЖКИ для отображения данных, подобный тому, что изображен на рис. 13.6.

Рис. 13.6. Плата расширения с жидкокристаллическим индикатором и клавиатурой от компании Freetronics

ПРИМЕЧАНИЕ

В наших примерах будет использоваться плата расширения с жидкокристаллическим индикатором и клавиатурой от компании Freetronics. За дополнительной информацией об этой плате обращайтесь по адресу *http://www.freetronics.com.au/collections/display/products/lcd-keypad-shield/*. Если вы решите использовать другой модуль отображения, не забудьте подставить соответствующие значения в функцию `LiquidCrystal`.

Для отображения на экране ЖКИ текущих координат, получаемых от приемника GPS, сконструируем очень простое переносное устройство GPS, которое будет питаться от батареи с напряжением 9 В.

Оборудование

Для создания устройства требуется минимальное количество оборудования:

- Плата Arduino и кабель USB.
- Модуль ЖКИ или плата расширения с жидкокристаллическим индикатором от Freetronics (упоминавшаяся выше).
- Одна батарея с напряжением 9 В и кабель с разъемом.
- Один комплект для сборки приемника GPS от SparkFun.

Скетч

Введите и загрузите следующий скетч:

```
  // Проект 44 - Простой приемник GPS
❶ #include <TinyGPS.h>
  #include <LiquidCrystal.h>
  LiquidCrystal lcd( 8, 9, 4, 5, 6, 7 );

  // Создать экземпляр объекта TinyGPS
  TinyGPS gps;

❷ void getgps(TinyGPS &gps);
```

```
  void setup()
  {
    Serial.begin(4800);
    lcd.begin(16, 2);
  }

  void getgps(TinyGPS &gps)
  // Функция getgps отображает указанные данные на экране ЖКИ
  {
    float latitude, longitude;
    // извлечь и отобразить координаты
❸    gps.f_get_position(&latitude, &longitude);
    lcd.setCursor(0,0);
    lcd.print("Lat:");
    lcd.print(latitude,5);
    lcd.print("  ");
    lcd.setCursor(0,1);
    lcd.print("Long:");
    lcd.print(longitude,5);
    lcd.print("  ");
    delay(3000); // ждать 3 секунды
    lcd.clear();
  }

  void loop()
  {
    byte a;
    if ( Serial.available() > 0 ) // если получены данные
    {
      a = Serial.read(); // извлечь байт
      if(gps.encode(a))  // если приняты допустимые данные GPS...
      {
❹       getgps(gps);    // извлечь координаты и вывести на ЖКИ
      }
    }
  }
```

В строках с ❶ по ❷ скетч подключает необходимые библиотеки для работы с жидкокристаллическим индикатором и обработки данных GPS. В `void loop` (❹) символы, полученные от приемника GPS, передаются функции `getgps()` (❸), которая с помощью `gps.f_get_position()` извлекает координаты в переменные `&latitude` и `&longitude` (для последующего вывода на экран ЖКИ).

Отображение координат на экране ЖКИ

После того как скетч будет загружен и приемник GPS начнет принимать данные, на экране ЖКИ появятся широта и долгота вашего текущего местоположения, как показано на рис. 13.7.

Рис. 13.7. Широта и долгота, отображаемые скетчем из проекта 44

Но где находится эта точка на поверхности земли? Мы можем это узнать с помощью Google Maps (*http://maps.google.com/*). Откройте страницу Google Maps в веббраузере, введите в поле поиска широту и долготу, разделив их запятой и пробелом, и Google Maps покажет карту с отмеченной на ней точкой. Например, для координат, показанных на рис. 13.7, Google Maps выведет карту, как показано на рис. 13.8.

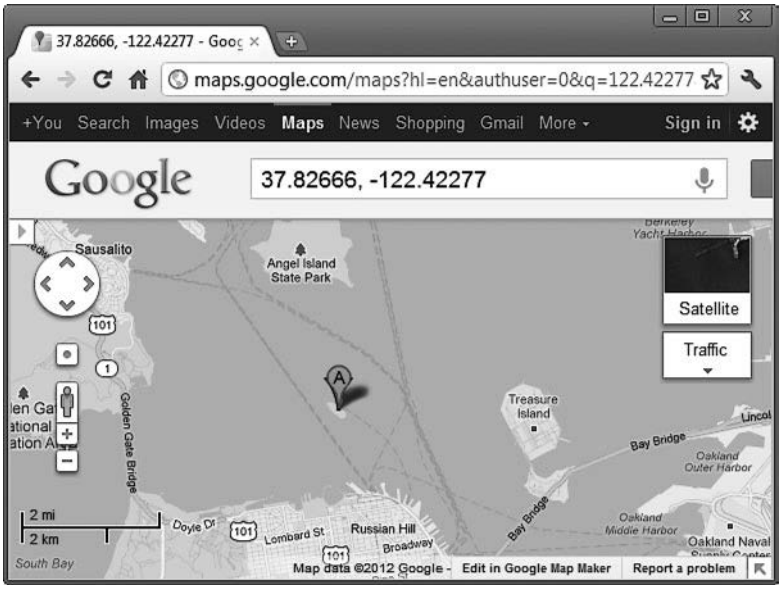


Рис. 13.8. Точка на карте, соответствующая координатам на рис. 13.7

Проект № 45: Часы высокой точности на основе GPS

Технология GPS позволяет не только определять местоположение, но и получать информацию о времени, на основе которой можно создать очень точные часы.

Оборудование

Для этого проекта мы используем то же оборудование, что и в проекте 44.

Скетч

Введите следующий скетч, извлекающий время из данных GPS:

```
// Проект 45 - Часы высокой точности на основе GPS

#include <TinyGPS.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd( 8, 9, 4, 5, 6, 7 );

// Создать экземпляр объекта TinyGPS
TinyGPS gps;

void getgps(TinyGPS &gps);

void setup()
{
  Serial.begin(4800);
  lcd.begin(16, 2);
}

void getgps(TinyGPS &gps)
{
  int year,a,t;
  byte month, day, hour, minute, second, hundredths;
❶  gps.crack_datetime(&year,&month,&day,
                      &hour,&minute,&second,&hundredths);
❷  hour=hour+10; // исправьте для вашего часового пояса
  if (hour>23)
  {
    hour=hour-24;
  }
  lcd.setCursor(0,0); // вывести дату и время
❸  lcd.print("Current time: ");
  lcd.setCursor(4,1);
  if (hour<10)
  {
    lcd.print("0");
  }
  lcd.print(hour, DEC);
  lcd.print(":");
  if (minute<10)
  {
    lcd.print("0");
  }
  lcd.print(minute, DEC);
```

```
  lcd.print(":");
  if (second<10)
  {
    lcd.print("0");
  }
  lcd.print(second, DEC);
}

void loop()
{
  byte a;
  if ( Serial.available() > 0 ) // если получены данные
  {
    a = Serial.read(); // извлечь байт
    if(gps.encode(a))  // если приняты допустимые данные GPS...
    {
      getgps(gps);     // извлечь время и вывести на ЖКИ
    }
  }
}
```

Этот скетч работает точно так же, как скетч из проекта 44, только извлекает время, а не координаты — всегда время по Гринвичу, более известное как всемирное координированное время (❶). В строке ❷ вы можете либо добавить, либо вычесть требуемое количество часов, чтобы привести время к своему часовому поясу. Затем время форматируется и отображается на экране ЖКИ (❸). Пример вывода времени показан на рис. 13.9.

Рис. 13.9. Проект 45 во время работы

Проект № 46: Запись координат перемещающегося объекта с течением времени

Теперь, зная как принимать координаты GPS и преобразовывать их в нормальный числовой вид, сконструируем GPS-логгер и организуем запись этой информации в плату расширения с картой памяти microSD из главы 8. Наш логгер будет определять и сохранять свои координаты. Благодаря использованию карты памяти microSD можно записать траекторию движения автомобиля, катера или любого

другого перемещающегося объекта, с которого возможен прием сигналов GPS, и потом просмотреть эту информацию на компьютере.

Оборудование

В этом проекте будет использоваться то же оборудование, что и в предыдущих примерах, с той лишь разницей, что плату расширения с жидкокристаллическим индикатором необходимо заменить платой расширения с картой microSD из главы 8 и использовать внешний источник питания. В данном примере будут сохраняться время, координаты и примерная скорость движения.

Скетч

После сборки устройства введите и загрузите следующий скетч:

```
// Проект 46 - Запись координат перемещающегося объекта
//             с течением времени

#include <SD.h>
#include <TinyGPS.h>

// Создать экземпляр объекта TinyGPS
TinyGPS gps;

void getgps(TinyGPS &gps);

void setup()
{
  pinMode(10, OUTPUT);
  Serial.begin(9600);

  // проверить наличие карты памяти microSD
  // и возможность ее использования
  if (!SD.begin(8)) {
    Serial.println("Card failed, or not present");
    // остановить скетч
    return;
  }
  Serial.println("microSD card is ready");
}

void getgps(TinyGPS &gps)
{
  float latitude, longitude;
  int year;
  byte month, day, hour, minute, second, hundredths;

  // декодировать и записать координаты
  gps.f_get_position(&latitude, &longitude);
  File dataFile = SD.open("DATA.TXT", FILE_WRITE);

  // если файл готов, записать в него данные
```

```
❶  if (dataFile)
   {
❷    dataFile.print("Lat: ");
     dataFile.print(latitude,5);
     dataFile.print(" ");
     dataFile.print("Long: ");
     dataFile.print(longitude,5);
     dataFile.print(" ");

     // декодировать и записать время
     gps.crack_datetime(&year,&month,&day,
                        &hour,&minute,&second,&hundredths);

     // исправьте для вашего часового пояса, как в проекте 45
     hour=hour+11;
     if (hour>23)
     {
       hour=hour-24;
     }
     if (hour<10)
     {
       dataFile.print("0");
     }
     dataFile.print(hour, DEC);
     dataFile.print(":");
     if (minute<10)
     {
       dataFile.print("0");
     }
     dataFile.print(minute, DEC);
     dataFile.print(":");
     if (second<10)
     {
       dataFile.print("0");
     }
     dataFile.print(second, DEC);
     dataFile.print(" ");
     dataFile.print(gps.f_speed_kmph());
❸    dataFile.println("km/h");
     dataFile.close();
❹    delay(30000); // выполнять запись раз в 30 с
   }
 }

 void loop()
 {
   byte a;

   if ( Serial.available() > 0 ) // если получены данные
   {
     a = Serial.read(); // извлечь байт
     if(gps.encode(a))  // если приняты допустимые данные GPS...
     {
```

```
❺       getgps(gps);    // извлечь данные и записать в карту памяти
      }
    }
  }
```

Этот скетч использует в функции `void loop()` тот же код, что в проектах 44 и 45, извлекающий данные из приемника GPS и передающий их другим функциям. В строке ❺ текст, извлеченный из приемника GPS, передается библиотеке *TinyGPS* для декодирования данных в нормальное числовое представление. В строке ❶ проверяется наличие карты памяти microSD и возможность записи на нее, а в строках с ❷ по ❸ извлеченные данные GPS записываются в текстовый файл на карте microSD. Так как после каждой записи файл закрывается, вы можете отключить питание платы Arduino в любой момент, не опасаясь потерять данные, и более того, вы должны это делать перед вставкой или извлечением карты microSD. Наконец, в строке ❹ можно установить интервал между записями, изменив значение аргумента в вызове функции `delay()`.

Отображение траектории на карте

После опробования GPS-логгера содержимое получившегося текстового файла должно выглядеть так, как показано на рис. 13.10.

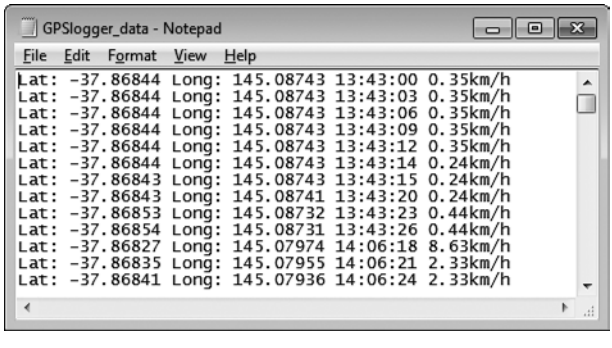


Рис. 13.10. Результаты работы проекта 46

Если полученные данные вручную ввести в Google Maps, мы получим траекторию движения на карте, точку за точкой. Но гораздо интереснее наложить на карту сразу всю траекторию. Для этого откройте текстовый файл в электронной таблице, отделите координаты и добавьте строку заголовка, как показано на рис. 13.11. Затем сохраните результат в файле с расширением *.csv*.

Теперь откройте в веб-браузере сайт GPS Visualizer (*http://www.gpsvisualizer.com/*). В разделе Get Started Now (Приступить прямо сейчас) щелкните на кнопке Browse (Обзор) в поле Upload a GPS file (Загрузить файл с данными GPS) и выберите свой файл с данными. В поле Choose an output format (Выходной формат) выберите

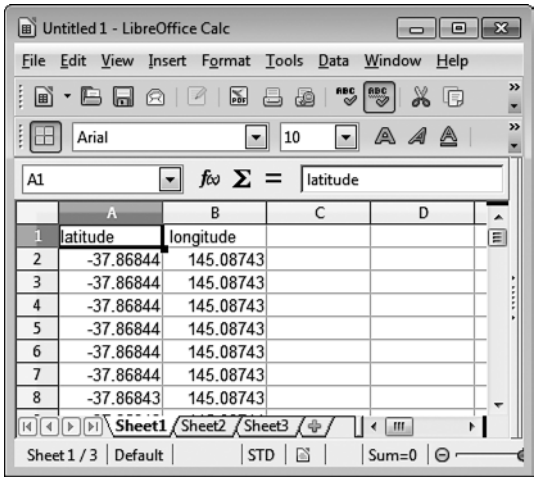


Рис. 13.11. Извлеченные координаты

пункт Google Maps и щелкните на кнопке Map It (Отобразить). Данные, зафиксированные вашим GPS-логгером, должны отобразиться на карте в виде траектории (рис. 13.12), которую затем можно отмасштабировать и исследовать.

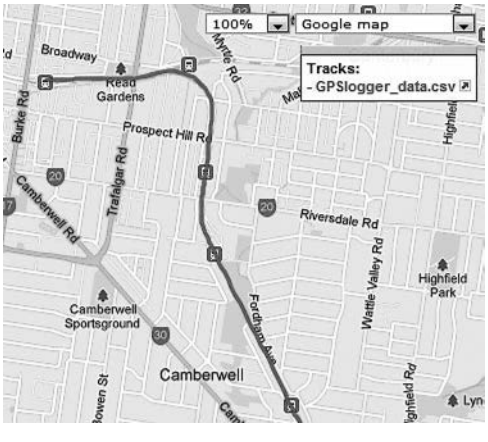


Рис. 13.12. Траектория, извлеченная из GPS-логгера и наложенная на карту

Забегая вперед

Как видите, даже то, что раньше казалось слишком сложным, как, например, работа с приемниками GPS, на самом деле оказывается простым благодаря Arduino. Продолжим эту тему: в следующей главе вы узнаете, как создать собственный беспроводной канал передачи данных и как организовать дистанционное управление.

14 Беспроводная передача информации

В этой главе вы узнаете, как посылать и принимать инструкции и данные, используя различное оборудование для беспроводной связи. Вы научитесь:

- ✓ посылать цифровые сигналы с использованием недорогих модулей беспроводной связи;
- ✓ создавать простую и недорогую систему дистанционного управления;
- ✓ использовать беспроводные приемопередатчики XBee;
- ✓ создавать термометр с дистанционным управлением.

Применение недорогих модулей беспроводной связи

Передачу текстовой информации в одном направлении легко можно организовать с помощью беспроводного канала, связывающего две системы на основе Arduino с недорогими модулями, которые работают в радиочастотном диапазоне, такими как радиопередатчик на рис. 14.1. Обычно эти модули продаются парами, и их часто называют модулями, или конструкторами, *RF Link* (радиосвязи). Отличными примерами служат модуль WLS107B4B компании Seeed Studio и модуль WRL-10534 компании SparkFun. В наших проектах мы будем использовать наиболее распространенные типы модулей, работающие в радиодиапазоне 433 МГц.

Выводы в нижней части модуля на рис. 14.1 имеют следующее назначение (слева направо): питание 5 В, «земля», ввод данных и внешняя антенна. Роль антенны способен выполнять кусок провода, но можно обойтись и без антенны, если передача осуществляется на короткие расстояния. (Конструкция разных моделей может немного отличаться, поэтому обязательно загляните в документацию, чтобы определить назначение выводов модуля, прежде чем выполнять его подключение.)

На рис. 14.2 изображен модуль приемника, он немного больше модуля передатчика.

Подключение приемника выполняется просто: контакты V+ и V– следует подключить к контактам 5V и GND соответственно, а контакт DATA приемника — к контакту на плате Arduino, предназначенному для приема данных.

Рис. 14.1. Модуль передатчика в радиодиапазоне

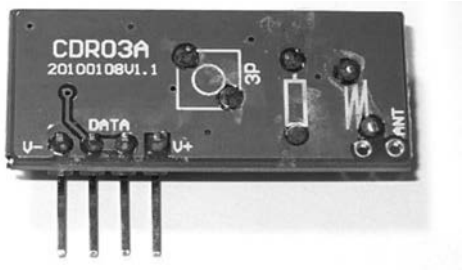


Рис. 14.2. Модуль приемника в радиодиапазоне

Прежде чем приступить к работе с этими модулями, необходимо загрузить и установить последнюю версию библиотеки *VirtualWire*, доступную по адресу *http://www.airspayce.com/mikem/arduino/VirtualWire/*. После ее установки можно переходить к следующему разделу.

ПРИМЕЧАНИЕ

Модули радиосвязи стоят недорого и просты в использовании, но в них отсутствуют средства проверки и исправления ошибок, которые гарантировали бы достоверность принимаемых данных. Поэтому я советую использовать их только для решения простых задач, таких как несложное дистанционное управление. Если вам потребуется высокая надежность передачи данных, используйте, например, модули XBee и подобные им, о которых рассказывается далее в этой главе.

Проект № 47: Пульт дистанционного управления

В этом проекте мы реализуем дистанционное управление двумя цифровыми выходами: вы будете нажимать кнопки, подключенные к одной плате Arduino, и с их помощью управлять соответствующими цифровыми выходами на другой плате Arduino, находящейся на некотором удалении от первой. Этот проект наглядно продемонстрирует, как пользоваться модулями радиосвязи и как определить максимальное расстояние, на котором еще возможно дистанционное управление с применением модулей, прежде чем вы решите использовать их для решения более сложных задач. (На открытом воздухе максимальное расстояние составляет примерно 100 м, но в закрытых помещениях оно будет значительно меньше из-за поглощения радиоволн различными препятствиями, находящимися между модулями.)

Оборудование для передатчика

Для сборки передатчика потребуется следующее оборудование:

- Плата Arduino и кабель USB.
- Одна батарея с напряжением 9 В и кабель с разъемом (использовались в главе 12).
- Один модуль передатчика, работающий в радиодиапазоне 433 МГц (например, модуль с артикулом WRL-10534, производимый компанией SparkFun).

- Два резистора номиналом 10 кОм (R1 и R2).
- Два конденсатора 100 нФ (C1 и C2).
- Две кнопки без фиксации.
- Одна макетная плата.

Схема для передатчика

Схема передатчика включает две кнопки без фиксации с фильтром против дребезга контактов, подключенные к цифровым контактам 2 и 3, и модуль передатчика, подключенный, как описывалось выше (рис. 14.3).

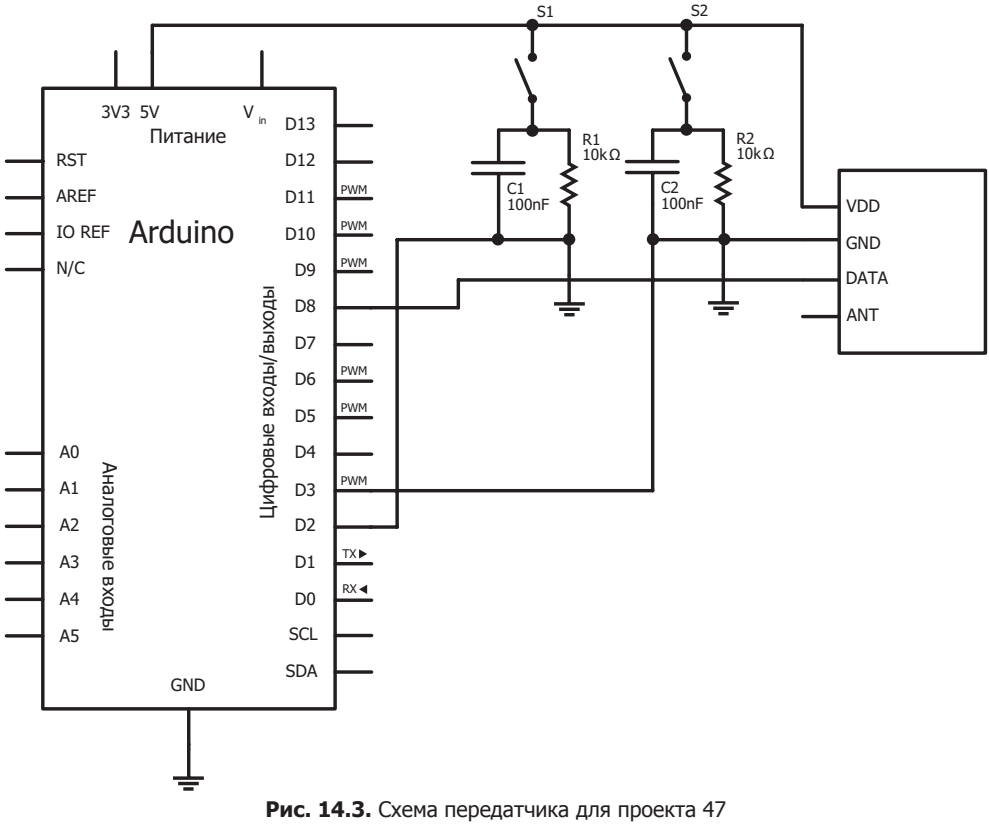


Рис. 14.3. Схема передатчика для проекта 47

Оборудование для приемника

Для сборки приемника потребуется следующее оборудование:

- Плата Arduino и кабель USB.

- Одна батарея с напряжением 9 В и кабель с разъемом (использовались в главе 12).
- Один модуль приемника, работающий в радиодиапазоне 433 МГц (например, модуль с артикулом WRL-10532, производимый компанией SparkFun).
- Одна макетная плата.
- Два светодиода по вашему выбору.
- Два резистора с номиналом 560 Ом (R1 и R2).

Схема приемника

Схема приемника включает два светодиода, подключенных к цифровым контактам 6 и 7. Контакт DATA модуля приемника подключен к цифровому контакту 8, как показано на рис. 14.4.

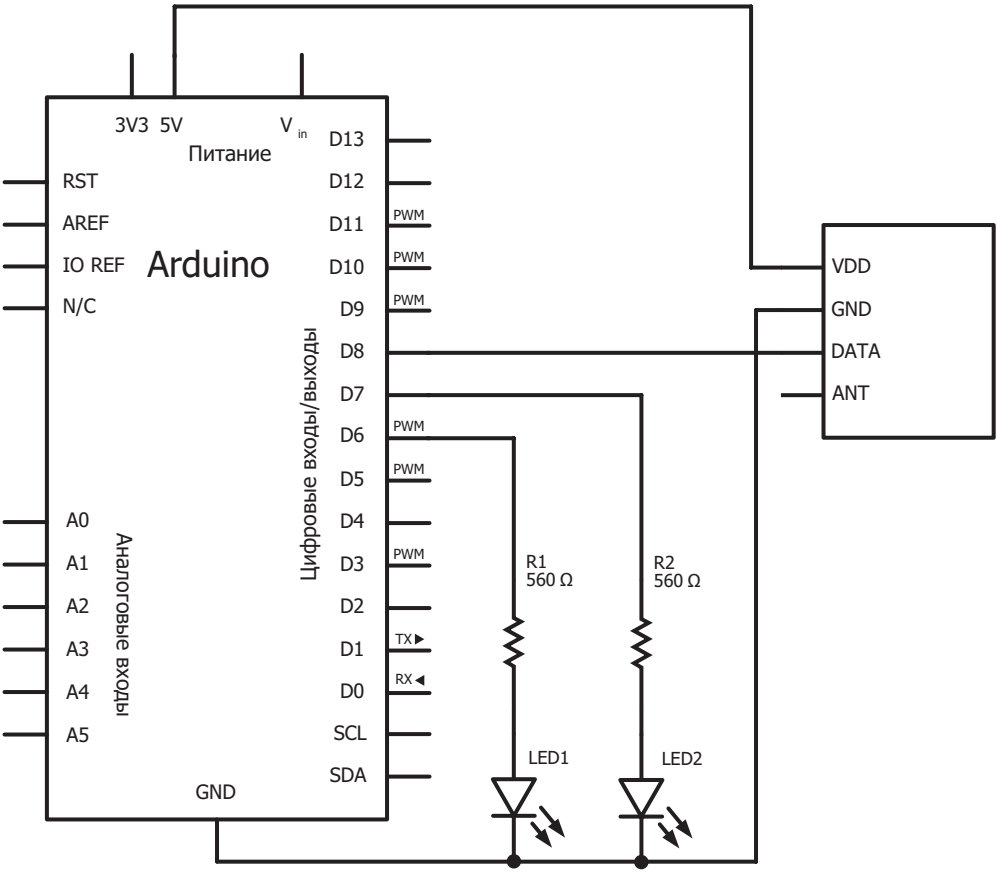


Рис. 14.4. Схема приемника для проекта 47

Вместо макетной платы светодиоды и резисторы можно смонтировать на плате расширения с модулем приемника от компании Freetronics, работающим в радиодиапазоне 433 МГц (рис. 14.5).

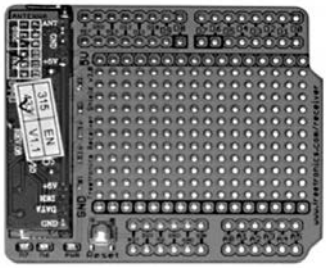


Рис. 14.5. Плата расширения с модулем приемника от компании Freetronics

Скетч для передатчика

Введите и загрузите следующий скетч в плату Arduino с подключенным модулем передатчика:

```
  // Проект 47 - Пульт дистанционного управления.
  //             Скетч передатчика
❶ #include <VirtualWire.h>

  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;

❷ const char *on2 = "a";
  const char *off2 = "b";
  const char *on3 = "c";
  const char *off3 = "d";

  void setup()
  {
❸   vw_set_ptt_inverted(true); // Необходимо для модуля передатчика
    vw_setup(300);             // установить скорость передачи
❹   vw_set_tx_pin(8);
    pinMode(2, INPUT);
    pinMode(3, INPUT);
  }

  void loop()
  {
❺   if (digitalRead(2)==HIGH)
    {
      vw_send((uint8_t *)on2, strlen(on2)); // отправить данные
      vw_wait_tx();                         // выждать немного
      delay(200);
```

```
    }
    if (digitalRead(2)==LOW)
    {
❻     vw_send((uint8_t *)off2, strlen(off2));
      vw_wait_tx();
      delay(200);
    }
    if (digitalRead(3)==HIGH)
    {
      vw_send((uint8_t *)on3, strlen(on3));
      vw_wait_tx();
      delay(200);
    }
    if (digitalRead(3)==LOW)
    {
      vw_send((uint8_t *)off3, strlen(off3));
      vw_wait_tx();
      delay(200);
    }
  }
```

Работа с модулями радиосвязи в скетчах осуществляется посредством функций из библиотеки *VirtualWire* (❶ и ❸). В ❹ выбирается цифровой контакт 8 (он будет использоваться для обмена данными между Arduino и модулем передатчика) и устанавливается скорость передачи. (Вы можете использовать любой другой цифровой контакт, кроме 0 и 1, которые соединены с последовательным портом.)

Скетч передатчика читает состояния двух кнопок, подключенных к цифровым контактам 2 и 3, и посылает в модуль радиосвязи один текстовый символ, соответствующий состояниям кнопок. Например, если кнопка, устанавливающая уровень `HIGH` на цифровом контакте 2 нажата, Arduino посылает символ *a*, а когда кнопка отпущена — символ *b*. Начиная со строки ❷, определяются четыре возможных состояния.

Передача символа выполняется в одной из четырех инструкций `if`, начиная с ❺, — например, в инструкции `if-then` (❻). Переменная, предназначенная для передачи, используется дважды — например, `on2`, как показано ниже:

```
vw_send((uint8_t *)on2, strlen(on2));
```

Функция `vw_send` посылает содержимое переменной `on2`, но ей необходимо знать объем этого содержимого в символах, который определяется вызовом функции `strlen()`.

Скетч для приемника

Теперь добавим скетч для приемника. Введите и загрузите следующий скетч в плату Arduino с подключенным модулем приемника:

```
// Проект 47 - Пульт дистанционного управления.
//             Скетч приемника

#include <VirtualWire.h>

uint8_t buf[VW_MAX_MESSAGE_LEN];
uint8_t buflen = VW_MAX_MESSAGE_LEN;

void setup()
{
❶   vw_set_ptt_inverted(true); // Необходимо для модуля приемника
   vw_setup(300);
❷   vw_set_rx_pin(8);
   vw_rx_start();
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
}

void loop()
{
❸   if (vw_get_message(buf, &buflen))
   {
❹     switch(buf[0])
     {
       case 'a':
         digitalWrite(6, HIGH);
         break;
       case 'b':
         digitalWrite(6, LOW);
         break;
       case 'c':
         digitalWrite(7, HIGH);
         break;
       case 'd':
         digitalWrite(7, LOW);
         break;
     }
   }
}
```

Так же как в скетче передатчика, здесь используются функции из библиотеки *VirtualWire* для настройки модуля приемника (❶), установки скорости приема данных и выбора цифрового контакта (❷) на плате Arduino, через который будут приниматься входящие данные.

Во время работы скетча модуль приемника принимает символы от передатчика и посылает в плату Arduino. Вызовом функции `vw_get_message()` (❸) скетч извлекает символы, полученные платой Arduino, и затем интерпретирует их с помощью инструкции `switch-case` (❹). Например, нажатие кнопки S1 на передатчике вызовет отправку символа *a*. Приемник, получив этот символ, установит уровень `HIGH` на цифровом выходе 6 и включит светодиод.

Эту пару простых устройств можно использовать для реализации более сложного алгоритма дистанционного управления системами на основе Arduino, посылающего коды в виде простых символов, которые будут интерпретироваться на стороне приемника.

Использование модулей XBee для беспроводной передачи данных на большие расстояния с высокой скоростью

Для того чтобы организовать беспроводную передачу данных на большие расстояния и с более высокой скоростью, чем это позволяют простые модули радиосвязи, представленные выше, потребуются модули XBee. Эти модули передают и получают последовательные данные, посылаемые или принимаемые компьютером. Существует несколько моделей модулей XBee, но мы будем использовать передатчики XBee типа Series 1, один из которых изображен на рис. 14.6.

Рис. 14.6. Типичный передатчик XBee

Проще всего подключить передатчик к плате Arduino с помощью платы расширения XBee, изображенной на рис. 14.7.

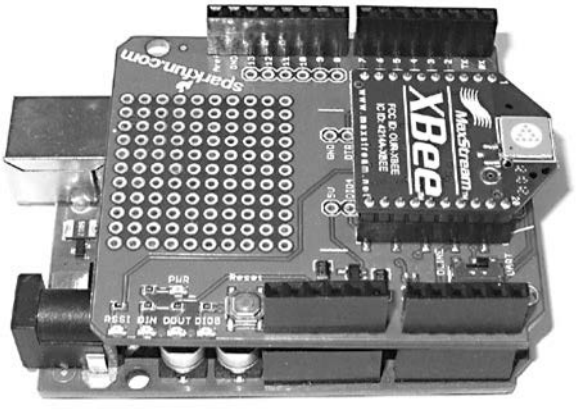


Рис. 14.7. Передатчик XBee, подключенный к Arduino с помощью платы расширения XBee

Подключения к компьютеру осуществляется с помощью платы XBee Explorer, как на рис. 14.8 (производится компанией SparkFun, артикул WRL-08687).

Рис. 14.8. Передатчик XBee, подключенный к порту USB посредством платы Explorer

Модули XBee не требуют применения сторонних библиотек; они действуют подобно простому *мосту данных*, который посылает и принимает данные по последовательной линии Arduino.

Проект № 48: Передача данных с помощью XBee

Этот проект демонстрирует простоту передачи данных из платы Arduino в компьютер, к которому подключена плата Explorer с передатчиком XBee. Проект реализуется в два этапа: сначала мы включим передатчик XBee в плату расширения XBee, а затем — плату расширения в плату Arduino.

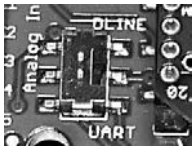


Рис. 14.9. Перемычка на плате расширения XBee

Обратите внимание на маленькую перемычку на плате расширения XBee, она показана на рис. 14.9.

Эта перемычка играет ту же роль, что и перемычка на плате расширения GPS, использовавшейся в главе 13. Эта перемычка управляет направлением записи данных — в порт USB на плате Arduino или в микроконтроллер модуля XBee. Перед загрузкой скетча установите перемычку в позицию DLINE, а после загрузки верните в позицию UART.

Скетч

Введите и загрузите следующий скетч:

```
// Проект 48 - Передача данных с помощью XBee

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

void loop()
```

```
{
  Serial.println("Hello, world");
  delay(1000);
}
```

Этот скетч посылает текст «Hello, world» в последовательный порт и отлично демонстрирует удобство использования модулей XBee для передачи данных: вы просто записываете данные в последовательный порт, а всю остальную работу делает модуль XBee. Отключите кабель USB от платы Arduino и подключите к ней внешний источник питания, например батарею с напряжением 9 В, с помощью кабеля, использовавшегося в главе 12. (Не забудьте вернуть перемычку в позицию UART.)

Подготовка компьютера к приему данных

Теперь подготовим компьютер к приему данных. С помощью платы Explorer (рис. 14.8) подключите другой модуль XBee к своему компьютеру. Загрузите и установите программу Terminal для Windows, доступную по адресу *https://sites.google.com/site/terminalbpp/*. После запуска перед вашим взором предстанет окно, изображенное на рис. 14.10.

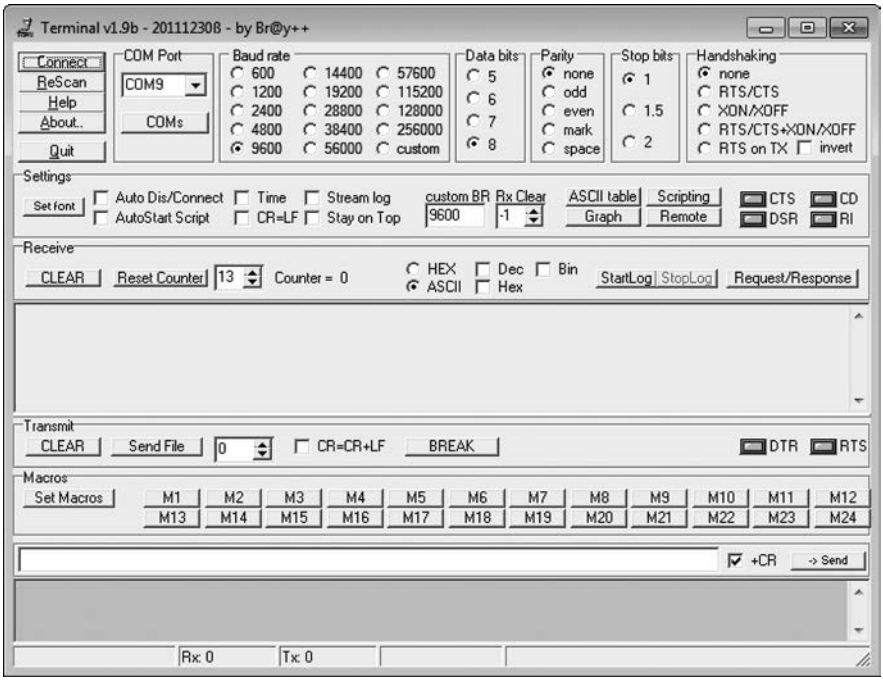


Рис. 14.10. Программа эмулятора терминала

В настройках, расположенных в верхней части окна, в разделе Baud Rate (Скорость) выберите скорость обмена 9600, в разделе Data Bits (Число битов данных) выберите 8, в разделе Parity (Проверка четности) выберите none, в разделе Stop Bits (Стоповых битов) выберите 1 и в разделе Handshaking (Квитирование) выберите none. Затем щелкните на кнопке ReScan (Сканировать) в левом верхнем углу, — это позволит программе выбрать корректный порт USB. Наконец, щелкните на кнопке Connect (Соединиться), и спустя мгновение вы увидите в окне программы данные, посылаемые платой Arduino, как показано на рис. 14.11.

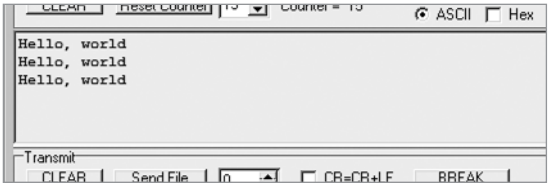


Рис. 14.11. Прием данных от удаленного модуля XBee

С одним компьютером можно связать несколько передатчиков XBee, каждый из которых будет присылать свою информацию. Например, организуйте на рабочем месте прием данных от нескольких датчиков температуры, расположенных в разных местах. В следующем примере мы научимся управлять несколькими платами Arduino с модулями XBee.

Проект № 49: Термометр с дистанционным управлением

В этом проекте мы создадим термометр с дистанционным управлением, возвращающий значение температуры по запросу с компьютера. В процессе реализации вы познакомитесь с основами дистанционного извлечения данных из удаленных датчиков и заложите фундамент для других проектов, так или иначе связанных с телеметрией.

Оборудование

Ниже перечислено необходимое оборудование:

- Плата Arduino и кабель USB.
- Один температурный датчик TMP36.
- Два модуля XBee (Series 1).
- Одна плата расширения XBee для Arduino.
- Одна плата XBee Explorer и кабель USB.

- Несколько отрезков провода разной длины.
- Одна батарея с напряжением 9 В и кабель с разъемом (как в главе 12).
- Одна макетная плата.

Монтаж

Вставьте модуль XBee в плату расширения; затем подключите плату расширения к плате Arduino. На небольшой макетной плате смонтируйте датчик TMP36, как показано на рис. 14.12, и соедините его выводы с контактами 5V, GND и A0 на плате расширения XBee (как в проекте 8).

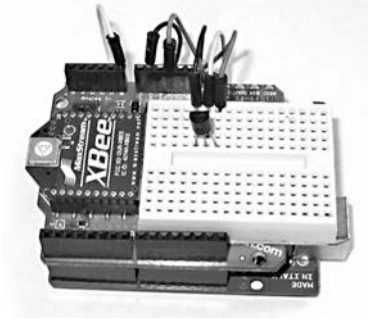


Рис. 14.12. Собранное устройство для дистанционного измерения температуры

Загрузите в плату Arduino скетч, следующий ниже, и затем подключите плату Explorer с другим модулем XBee к порту USB компьютера.

Скетч

Введите и загрузите следующий скетч:

```
// Проект 49 - Термометр с дистанционным управлением

char a;
float voltage=0;
float sensor=0;
float celsius=0;
float fahrenheit=0;
float photocell=0;

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

```
  void sendC()
  {
    sensor=analogRead(0);
    voltage=((sensor*5000)/1024);
    voltage=voltage-500;
    celsius=voltage/10;
    fahrenheit=((celsius*1.8)+32);
    Serial.print("Temperature: ");
    Serial.print(celsius,2);
    Serial.println(" degrees C");
  }

  void sendF()
  {
    sensor=analogRead(0);
    voltage=((sensor*5000)/1024);
    voltage=voltage-500;
    celsius=voltage/10;
    fahrenheit=((celsius*1.8)+32);
    Serial.print("Temperature: ");
    Serial.print(fahrenheit,2);
    Serial.println(" degrees F");
  }

  void getCommand()
❶{
    Serial.flush();
    while (Serial.available() == 0)
    {
      // ничего не делать, пока не будут получены данные из XBee
    }
    while (Serial.available() > 0)
    {
      a = Serial.read(); // прочитать число из буфера последовательного порта
    }
  }

  void loop()
  {
    getCommand();// получить команду от компьютера
❷   switch (a)
    {
❸     case 'c':
        // послать температуру в градусах Цельсия
        sendC();
        break;
❹     case 'f':
        // послать температуру в градусах Фаренгейта
        sendF();
        break;
    }
  }
```

Скетч ожидает получения команды, вызывая функцию `getCommand()` (❶). Эта функция выполняет пустой цикл, пока не будет принят символ, после чего сохраняет его в переменной `a`. Затем принятый символ интерпретируется с помощью инструкции `switch-case` (❷). Если получена буква *c*, вызывается функция `sendC()` (❸), которая вычисляет температуру в градусах Цельсия, записывает ее в последовательный порт и посылает компьютеру с помощью модуля XBees. Если получена буква *f*, вызывается функция `sendF()` (❹), которая вычисляет температуру в градусах Фаренгейта и посылает ее компьютеру.

Опробование

Запустите программу Terminal, как это делалось выше в данной главе, а затем пошлите букву *c* или *f*. Примерно через 1 секунду вам вернется значение температуры и появится в области вывода, в разделе Receive (Прием), как показано на рис. 14.13.

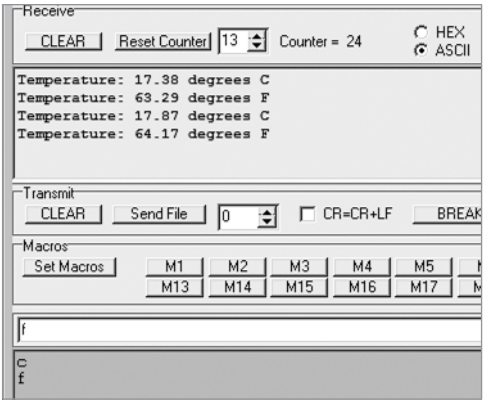


Рис. 14.13. Результаты работы проекта 49

Забегая вперед

Эта глава показала, насколько просто реализовать дистанционное управление системами на основе Arduino. Например, можно управлять цифровыми выходами, посылая символы с персонального компьютера или из другой платы Arduino, или организовать распознавание символов в отдельных платах Arduino. Вооруженные знаниями, полученными к настоящему моменту, вы можете смело продолжать движение по дороге творчества.

Но на этом обсуждение приемов беспроводной передачи данных не заканчивается. В следующей главе вы узнаете, как можно организовать управление платой Arduino с помощью простого пульта управления для телевизора.

15 Инфракрасный пульт дистанционного управления

В этой главе вы:

- ✓ создадите и опробуете простой инфракрасный приемник;
- ✓ реализуете дистанционное управление цифровыми выходами Arduino;
- ✓ добавите дистанционное управление в модель танка, созданную в главе 12.

Вы увидите, как с помощью недорогого модуля приемника плата Arduino принимает сигналы от инфракрасного пульта дистанционного управления и реагирует на них.

Что такое инфракрасный пульт дистанционного управления?

Многие из нас каждый день пользуются инфракрасными пультами дистанционного управления, но далеко не все знают, как они действуют. Инфракрасные (ИК) сигналы фактически являются пучками света инфракрасного диапазона и не воспринимаются невооруженным глазом. Поэтому если посмотреть на маленький светодиод на пульте дистанционного управления и нажать какую-нибудь кнопку, вы не увидите свечения.

Инфракрасные пульты дистанционного управления содержат один или несколько специальных светодиодов, излучающих в инфракрасном диапазоне, которые используются для передачи ИК-сигналов. Например, если нажать кнопку на пульте управления, светодиод многократно включается и выключается, воспроизводя определенные последовательности, уникальные для каждой кнопки. Этот сигнал

принимается специальным ИК-приемником и преобразуется в электрические импульсы, которые в свою очередь преобразуются электроникой приемника в данные. Если вам интересно увидеть световые импульсы, посмотрите на ИК-светодиод, воспользовавшись камерой на сотовом телефоне или цифровой камерой.

Подготовка к приему ИК-сигналов

Прежде чем двинуться дальше, установим библиотеку для Arduino, поддерживающую работу с ИК-приемниками. Для этого откройте в веб-браузере страницу *https://github.com/shirriff/Arduino-IRremote/* и загрузите необходимые файлы, как описывается в разделе «Расширение возможностей скетчей с помощью библиотек» в главе 8.

ИК-приемник

Следующий шаг — установка ИК-приемника и проверка его работоспособности. Вы можете выбрать отдельный ИК-приемник (рис. 15.1) или готовый модуль с разъемом и проводами (рис. 15.2), который проще в использовании.

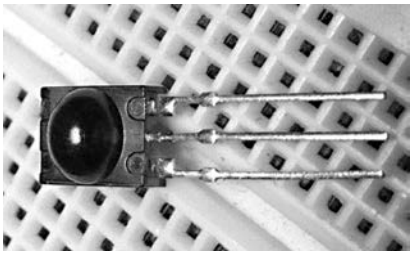


Рис. 15.1. ИК-приемник

Рис. 15.2. Готовый модуль с разъемом и проводами

На рис. 15.1 показан отдельный ИК-приемник Vishay TSOP4138, который можно приобрести у таких продавцов, как Newark (артикул 59K0287) или element14 (артикул 1040743). Нижний вывод (как показано на рис. 15.1) соединяется с цифровым контактом на плате Arduino, центральный — с контактом GND и верхний — с контактом 5V.

На рис. 15.2 показан готовый модуль с разъемом и проводами. Такой модуль можно приобрести у DFRobot или у других продавцов. Преимущество готовых модулей заключается в наличии проводов и маркировки, упрощающей подключение.

Независимо от модели модуля, которую вы выберете для себя, во всех следующих проектах вывод D (или data line — линия данных) приемника будет подключаться к цифровому контакту 11 на плате Arduino, вывод VCC — к контакту 5V и вывод GND — к контакту GND.

Пульт дистанционного управления

Наконец, вам понадобится пульт дистанционного управления. Мы будем использовать пульт от телевизора Sony, изображенный на рис. 15.3. Если у вас нет такого пульта, можете использовать любой недорогой универсальный пульт, достаточно лишь выбрать в нем коды управления Sony; как это сделать, должно быть описано в инструкции к пульту.

Рис. 15.3. Типичный пульт дистанционного управления Sony

Тестовый скетч

Теперь проверим работоспособность всего комплекта оборудования. После подключения ИК-приемника к плате Arduino введите и загрузите скетч из листинга 15.1.

Листинг 15.1. Тест ИК-приемника

```
// Листинг 15.1

int receiverpin = 11; // вывод 1 ИК-приемника - к цифровому
                      // входу 11 на плате Arduino
❶ #include <IRremote.h> // подключить библиотеку
❷ IRrecv irrecv(receiverpin); // создать экземпляр объекта IRrecv
❸ decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // запустить ИК-приемник
}

void loop()
{
❹  if (irrecv.decode(&results)) // ИК-сигнал принят?
  {
❺    Serial.print(results.value, HEX); // вывести ИК-код в монитор порта
    Serial.print(" ");
    irrecv.resume(); // разрешить прием следующего значения
  }
}
```

Этот скетч относительно прост, потому что основная работа выполняется библиотекой *IRremote*. В строке ❹ проверяется, был ли получен сигнал от пульта дистанционного управления. Если «да», то код сигнала выводится в шестнадцатеричном виде в монитор порта (❺). В строках ❶, ❷ и ❸ активируется библиотека *IRremote* и создается экземпляр библиотеки для использования в скетче.

Опробование собранного устройства

Загрузив скетч, откройте монитор порта, наведите пульт управления на приемник и начинайте нажимать кнопки. После нажатия каждой кнопки вы должны увидеть коды в окне монитора порта. Например, на рис. 15.4 показаны результаты однократного нажатия кнопок 1, 2 и 3.

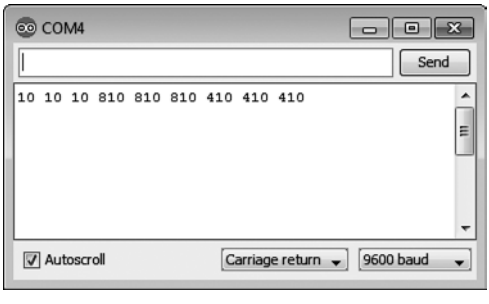


Рис. 15.4. Результаты нажатия кнопок после запуска скетча из листинга 15.1

В табл. 15.1 перечислены коды, генерируемые простым дистанционным пультом управления Sony, которые будут использоваться в последующих скетчах. Тестируя скетч в листинге 15.1, обратите внимание, что каждый код появляется трижды. Это — особенность ИК-систем Sony, которые посылают код три раза для каждого нажатия кнопки. Проявив некоторую изобретательность, повторяющиеся коды можно игнорировать. А сейчас перейдем к реализации следующего проекта с дистанционным управлением.

Таблица 15.1. Примеры ИК-кодов Sony

Кнопка	Код	Кнопка	Код
Power (Вкл./Выкл)	A90	7	610
Mute (Выкл. звук)	290	8	У10
1	10	9	110
2	810	0	910
3	410	Volume up (Увеличить громкость)	490
4	C10	Volume down (Уменьшить громкость)	C90
5	210	Channel up (Следующий канал)	90
6	A10	Channel down (Предыдущий канал)	890

Проект № 50: Дистанционное управление Arduino с помощью ИК-пульта

Этот проект продемонстрирует, как организовать управление цифровыми выходами с использованием ИК-пульта. В этом проекте мы реализуем управление цифровыми контактами со 2-го по 7-й с помощью цифровых кнопок со 2 по 7 на пульте Sony. После нажатия кнопки на пульте управления на соответствующем цифровом выходе устанавливается уровень `HIGH` на 1 секунду, а затем возвращается уровень `LOW`. Вы можете использовать этот проект как основу или руководство для добавления поддержки удаленного управления в своих других проектах.

Оборудование

Здесь используется то же самое оборудование, что было задействовано для проверки работоспособности ИК-приемника в начале главы, с дополнительной самодельной платой расширения LED, созданной в проекте 28. (Вместо этой платы вы можете подключить к цифровым выходам светодиоды, электродвигатели или другие устройства.)

Скетч

Введите и загрузите следующий скетч:

```
// Проект 50 – Дистанционное управление Arduino с помощью ИК-пульта

int receiverpin = 11; // вывод 1 ИК-приемника - к цифровому
                      // входу 11 на плате Arduino
#include <IRremote.h>
IRrecv irrecv(receiverpin); // создать экземпляр объекта IRrecv
decode_results results;

void setup()
{
  irrecv.enableIRIn();          // запустить ИК-приемник
  for (int z = 2 ; z < 8 ; z++) // настроить цифровые выходы
  {
    pinMode(z, OUTPUT);
  }
}

❶ void translateIR()
  // принимает ИК-коды Sony и
  // выполняет соответствующие им операции
  {
    switch(results.value)
    {
❷     case 0x810: pinOn(2); break; // 2
```

```
      case 0x410: pinOn(3); break; // 3
      case 0xC10: pinOn(4); break; // 4
      case 0x210: pinOn(5); break; // 5
      case 0xA10: pinOn(6); break; // 6
      case 0x610: pinOn(7); break; // 7
    }
  }

❸ void pinOn(int pin) // включает контакт pin на 1 секунду
  {
    digitalWrite(pin, HIGH);
    delay(1000);
    digitalWrite(pin, LOW);
  }

  void loop()
  {
❹   if (irrecv.decode(&results))    // ИК-сигнал принят?
    {
      translateIR();
❺     for (int z = 0 ; z < 2 ; z++) // игнорировать 2-е и 3-е повторение
      {
        irrecv.resume();            // разрешить прием следующего значения
      }
    }
  }
```

Этот скетч делится на три основных раздела. Первый ожидает сигнал от пульта дистанционного управления (❹). После приема сигнала полученный код передается в функцию `translateIR()` (❶) для определения нажатой кнопки и выполнения соответствующей операции.

Обратите внимание, как в инструкции `switch-case` ❷ выполняется сравнение шестнадцатеричных кодов, возвращаемых библиотекой *IRremote*. Это те самые коды, которые возвращаются тестовым скетчем из листинга 15.1. Если принят код, соответствующий одной из кнопок со 2 по 7, вызывается функция `pinOn()` (❸), которая устанавливает высокий уровень на соответствующем цифровом выходе на 1 секунду.

Как уже отмечалось, в ответ на однократное нажатие клавиши пульты Sony посылают код трижды, поэтому в скетче добавлен небольшой цикл (❺), позволяющий пропустить второй и третий коды. Наконец, обратите внимание на дополнительную приставку `0x` в начале шестнадцатеричных чисел в инструкциях `case` (❷).

ПРИМЕЧАНИЕ

Шестнадцатеричные числа — это числа в системе счисления с основанием 16, для записи которых используются цифры от 0 до 9 и буквы от A до F. Например, десятичное число 10 в шестнадцатеричном виде записывается как A, десятичное число 15 — как F, десятичное число 16 — как 10, и т. д. Если в скетчах используются шестнадцатеричные числа, они должны предваряться приставкой `0x`.

Расширение возможностей

Добавив в скетч анализ дополнительных кнопок, можно организовать управление движением нашей модели танка. Для этого достаточно с помощью скетча из листинга 15.1 определить коды соответствующих кнопок на пульте и добавить реакцию на новые коды в инструкцию `switch...case`.

Проект № 51: Дистанционное ИК-управление моделью танка

Чтобы показать, как интегрировать дистанционное управление с помощью ИК-пульта в существующий проект, добавим эту возможность в модель танка, создание которой описано в проекте 40. В данном проекте мы посмотрим, как организовать управление движением танка с помощью простого телевизионного пульта Sony.

Оборудование

Здесь используется то же самое оборудование, что было задействовано в проекте создания танка, а также модуль ИК-приемника, описанный в начале главы. Следующий скетч реализует реакцию танка на нажатия кнопок на пульте управления: кнопка 2 — вперед, 8 — назад, 4 — поворот влево и 6 — поворот вправо.

Скетч

После подключения ИК-приемника к модели танка введите и загрузите следующий скетч:

```
// Проект 51 - Дистанционное ИК-управление моделью танка

int receiverpin = 11; // вывод 1 ИК-приемника - к цифровому
                      // входу 11 на плате Arduino
#include <IRremote.h>
IRrecv irrecv(receiverpin); // создать экземпляр объекта IRrecv
decode_results results;

int m1speed = 6;     // цифровые входы для управления скоростью
int m2speed = 5;
int m1direction = 7; // цифровые входы для управления направлением
int m2direction = 4;

void setup()
{
  pinMode(m1direction, OUTPUT);
  pinMode(m2direction, OUTPUT);
  irrecv.enableIRIn(); // запустить ИК-приемник
```

```
}

void goForward(int duration, int pwm)
{
  digitalWrite(m1direction, HIGH); // вперед
  digitalWrite(m2direction, HIGH); // вперед
  analogWrite(m1speed, pwm);       // выбранная скорость
  analogWrite(m2speed, pwm);
  delay(duration);                 // и продолжать
  analogWrite(m1speed, 0);         // затем встать
  analogWrite(m2speed, 0);
}

void goBackward(int duration, int pwm)
{
  digitalWrite(m1direction, LOW); // назад
  digitalWrite(m2direction, LOW); // назад
  analogWrite(m1speed, pwm);      // выбранная скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);        // затем встать
  analogWrite(m2speed, 0);
}

void rotateRight(int duration, int pwm)
{
  digitalWrite(m1direction, HIGH); // вперед
  digitalWrite(m2direction, LOW);  // назад
  analogWrite(m1speed, pwm);       // выбранная скорость
  analogWrite(m2speed, pwm);
  delay(duration);                 // и продолжать
  analogWrite(m1speed, 0);         // затем встать
  analogWrite(m2speed, 0);
}

void rotateLeft(int duration, int pwm)
{
  digitalWrite(m1direction, LOW);  // назад
  digitalWrite(m2direction, HIGH); // вперед
  analogWrite(m1speed, pwm);       // выбранная скорость
  analogWrite(m2speed, pwm);
  delay(duration);                 // и продолжать
  analogWrite(m1speed, 0);         // затем встать
  analogWrite(m2speed, 0);
}

// translateIR выполняет операцию, соответствующую ИК-коду
void translateIR()
{
  switch(results.value)
  {
    case 0x810: goForward(250, 255);   break; // 2
```

```
    case 0xC10: rotateLeft(250, 255);  break; // 4
    case 0xA10: rotateRight(250, 255); break; // 6
    case 0xE10: goBackward(250, 255);  break; // 8
  }
}

void loop()
{
  if (irrecv.decode(&results))    // ИК-сигнал принят?
  {
    translateIR();
    for (int z = 0 ; z < 2 ; z++) // игнорировать повторения
    {
      irrecv.resume();            // разрешить прием следующего значения
    }
  }
}
```

Этот скетч уже знаком вам. Фактически вместо включения цифровых выходов он вызывает функции управления электродвигателями, которые использовались в скетчах управления танком в главе 12.

Забегая вперед

С помощью проектов из этой главы вы научились посылать команды плате Arduino с использованием инфракрасного пульта дистанционного управления. Объединив вновь приобретенные знания с полученными прежде (и с теми, которые вы получите в последующих проектах), вы сможете заменить физические формы ввода, например кнопки, удаленным управлением.

Но наши развлечения на этом не заканчиваются. В следующей главе мы будем использовать Arduino для реализации систем радиочастотной идентификации, которые многими воспринимаются как нечто завораживающее и футуристическое.

16 Чтение радиомаркеров RFID

В этой главе вы:

- ✓ узнаете, как реализовать радиочастотную идентификацию с помощью Arduino;
- ✓ освоите сохранение данных в ЭСППЗУ на плате Arduino;
- ✓ создадите основу системы доступа по радиомаркерам на основе Arduino.

Радиочастотная идентификация (Radio-Frequency Identification, RFID) — это беспроводная система, использующая электромагнитное поле для передачи данных от одного объекта к другому без их прямого контакта друг с другом. Вы можете сконструировать устройство на основе Arduino, которое будет читать радиомаркеры и карты RFID, для создания системы доступа и управления цифровыми выходами. Возможно, вам приходилось использовать карты RFID, например радиоключи, применяющиеся для отпирания дверей, или карты оплаты услуг общественного транспорта, которые вставляются в считыватель при входе, скажем, в автобус. На рис. 16.1 показаны примеры радиоключей и карт RFID.

Рис. 16.1. Примеры устройств RFID

Внутреннее устройство радиомаркеров

Радиомаркер RFID состоит из маленькой микросхемы с памятью, доступной для специализированного устройства чтения. Большинство радиомаркеров не имеют внутреннего источника питания; они питаются энергией электромагнитного поля, генерируемого устройством чтения RFID. Это поле создается миниатюрной катушкой индуктивности, которая одновременно выполняет роль антенны для передачи данных между радиомаркером и устройством чтения. На рис. 16.2 показана катушка-антенна устройства чтения радиомаркеров RFID, которое будет использоваться в этой главе.

Рис. 16.2. Устройство чтения радиомаркеров RFID

Устройство чтения, используемое в этой главе (артикул RFR101A1M), можно приобрести в компании Seeed Studio: *http://www.seeedstudio.com/*. Это недорогое и простое в использовании устройство, действующее на частоте 125 кГц; выбирая радиомаркер RFID, приобретайте модель, действующую на той же частоте, например, с артикулом 113990014 от компании Seeed Studio.

Проверка оборудования

В этом разделе мы подключим устройство чтения радиомаркеров RFID к плате Arduino и проверим его работоспособность с помощью простого скетча, читающего данные с радиомаркера RFID и посылающего их в монитор порта.

Схема

На рис. 16.3 показаны контакты на модуле RFID.

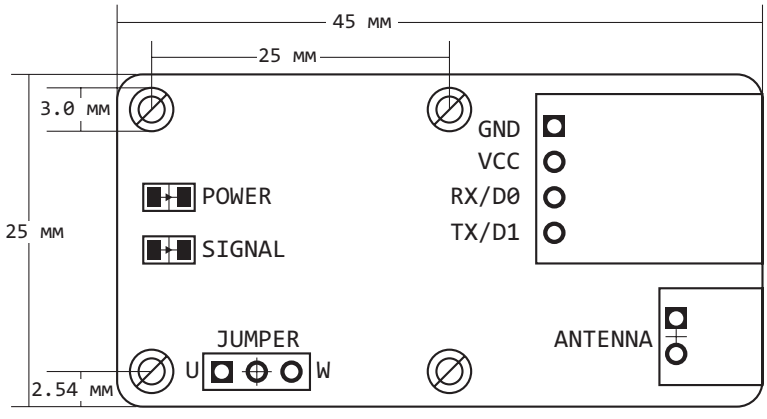


Рис. 16.3. Контакты на модуле RFID

Проверка

После подключения модуля чтения RFID к плате Arduino проверьте его работоспособность, соединив перемычкой левый и средний контакты на колодке JUMPER. Подключите модуль RFID к Arduino проводами, следуя инструкциям ниже:

1. Подключите разъем катушки к контактам ANTENNA.
2. Соедините контакт GND модуля RFID с контактом GND на плате Arduino.
3. Соедините контакт VCC модуля RFID с контактом 5V на плате Arduino.
4. Соедините контакт RX модуля RFID с контактом D0 на плате Arduino.
5. Соедините контакт TX модуля RFID с контактом D1 на плате Arduino.

ПРИМЕЧАНИЕ

Перед загрузкой скетча в Arduino с подключенным модулем RFID отсоедините провод, связывающий контакт RX модуля RFID с контактом D0 на плате Arduino. После загрузки скетча вновь соедините контакты. Это необходимо сделать, так как контакт D0 на плате Arduino используется линией связи, по которой производится загрузка скетча.

Тестовый скетч

Введите и загрузите скетч из листинга 16.1.

Листинг 16.1. Скетч для проверки работоспособности модуля RFID

```
// Листинг 16.1

int data1 = 0;

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

void loop()
{
  if (Serial.available() > 0) {
    data1 = Serial.read();
    // вывести прочитанное число
    Serial.print(" ");
    Serial.print(data1, DEC);
  }
}
```

Исследование вывода в окне монитора порта

Откройте окно монитора порта и поднесите радиомаркер RFID к катушке. В мониторе появятся значения, показанные на рис. 16.4.

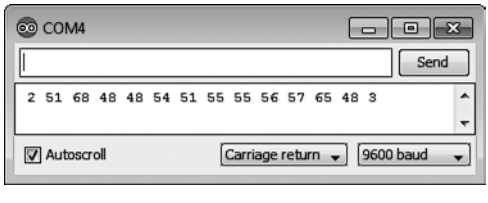


Рис. 16.4. Пример вывода скетча из листинга 16.1

Обратите внимание, что в окне монитора порта отображается 14 чисел. Это уникальный числовой идентификатор радиомаркера RFID, который будет использоваться в будущих скетчах для идентификации радиомаркера. Запишите числа для всех радиомаркеров, имеющихся у вас, это пригодится в следующих проектах.

Проект № 52: Простая RFID-система контроля доступа

Теперь попробуем применить систему RFID на практике. В этом проекте вы узнаете, как генерировать события в плате Arduino в случае чтения допустимого радиомаркера. Этот скетч хранит идентификационные числа двух радиомаркеров RFID; для обоих скетч будет выводить в монитор порта текст «Accepted» (доступ

разрешен). Если поднести к антенне любой другой радиомаркер, скетч выведет в монитор порта текст «Rejected» («Доступ запрещен»). Данный скетч мы используем как основу для добавления функций управления доступом по радиомаркерам RFID в существующие проекты.

Скетч

Введите и загрузите следующий скетч. Но во время ввода замените символы *x* в массивах ❶ и ❷ числами, соответствующими вашим радиомаркерам, которые вы должны были записать, как советовалось выше. (Мы обсуждали массивы в главе 6.)

```
  // Проект 52 – Простая RFID-система контроля доступа

  int data1 = 0;
  int ok = -1;

  // чтобы определить числовые идентификаторы радиомаркеров,
  // воспользуйтесь скетчем из листинга 16.1
❶ int tag1[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};
❷ int tag2[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};

  // следующий массив используется для чтения и сравнения
  int newtag[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0};

  void setup()
  {
    Serial.flush(); // очистить буфер последовательного порта,
                    // иначе первая попытка чтения может дать
                    // ошибочный результат
    Serial.begin(9600);
  }

❸ boolean comparetag(int aa[14], int bb[14])
  {
    boolean ff = false;
    int fg = 0;
    for (int cc = 0 ; cc < 14 ; cc++)
    {
      if (aa[cc] == bb[cc])
      {
        fg++;
      }
    }
    if (fg == 14)
    {
      ff = true;
    }
    return ff;
  }
```

```
  // сравнивает известные числовые идентификаторы
  // с только что прочитанным
❹ void checkmytags()
  {
    ok = 0; // эта переменная помогает принять решение
            // и может иметь три значения:
            // 1 - числовой идентификатор прочитан и совпадает
            // с одним из известных
            // 0 - числовой идентификатор прочитан, но
            // не совпадает ни с одним из известных
            // -1 - числовой идентификатор не был прочитан
    if (comparetag(newtag, tag1) == true)
    {
❺     ok++;
    }
    if (comparetag(newtag, tag2) == true)
    {
❻     ok++;
    }
  }

  void loop()
  {
    ok = -1;
    if (Serial.available() > 0) // если была попытка чтения
    {
      // прочитать число из модуля RFID
      delay(100); // дать время поступить всем данным
                  // в буфер последовательного порта
      for (int z = 0 ; z < 14 ; z++) // прочитать числовой
                                     // идентификатор радиомаркера
❼     {
        data1 = Serial.read();
        newtag[z] = data1;
      }
      Serial.flush(); // аннулировать повторные попытки чтения
      // теперь сравнить числовые идентификаторы
❽     checkmytags();
    }
❾   // выполнить необходимые операции по результатам
    if (ok > 0) // прочитан известный радиомаркер
    {
      Serial.println("Accepted");
      ok = -1;
    }
    else if (ok == 0) // если прочитан неизвестный радиомаркер
    {
      Serial.println("Rejected");
      ok = -1;
    }
  }
```

Принцип действия

Если поднести радиомаркер RFID к антенне, устройство чтения перешлет числовой идентификатор через последовательный порт. Скетч принимает все 14 чисел и помещает их в массив `newtag[]` (❼). Затем прочитанный идентификатор сравнивается с двумя известными идентификаторами, ❶ и ❷, в функции `checkmytags()` (❹ и ❽), при этом фактическое сравнение массивов выполняется в функции `comparetag()` (❸).

Функция `comparetag()` принимает два числовых массива и возвращает логический признак их совпадения (`true`) или несовпадения (`false`). При обнаружении совпадения переменной `ok` присваивается значение 1 (❺ и ❻). Наконец, в ❾ в зависимости от результата совпадения прочитанного идентификатора с одним из известных выполняются те или иные операции.

После загрузки скетча и подключения провода, соединяющего контакт `D0` на плате Arduino с контактом `RX` модуля RFID (см. рис. 16.3), откройте окно монитора последовательного порта и подносите к антенне разные радиомаркеры. Вы должны получить результаты, аналогичные изображенным на рис. 16.5.

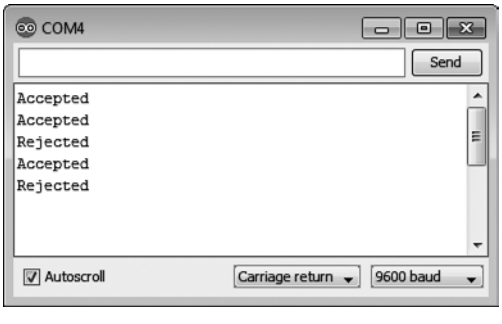


Рис. 16.5. Результаты работы проекта 52

Сохранение данных во встроенном ЭСППЗУ

Когда вы определяете и используете переменные в скетчах для Arduino, они хранят свои данные только до момента сброса платы или выключения питания. Но как быть, если необходимо сохранить значение для использования в будущем?: Например, изменяемый пользователем секретный код для кодового замка из главы 9? Для этой цели используется *ЭСППЗУ* (электрически стираемое программируемое постоянное запоминающее устройство). ЭСППЗУ находится внутри микроконтроллера ATmega328 и не теряет хранящиеся в нем данные при выключении питания.

ЭСППЗУ в Arduino может хранить до 1024 байт данных в ячейках, пронумерованных от 0 до 1023. Напомню, что один байт хранит целое число в диапазоне от 0 до

255. Далее вы узнаете, почему такая организация прекрасно подходит для хранения числовых идентификаторов радиомаркеров RFID. Чтобы получить возможность использовать ЭСППЗУ, в скетче необходимо подключить библиотеку *EEPROM* (входит в состав Arduino IDE), как показано ниже:

```
#include <EEPROM.h>
```

Затем нужно выполнить запись значения в ЭСППЗУ:

```
EEPROM.write(a, b);
```

Здесь параметр `a` определяет номер ячейки (от 0 до 1023) для записи значения, а параметр `b` — байт данных для записи в ячейку с номером `a`.

Для извлечения данных из ЭСППЗУ используется следующая функция:

```
value = EEPROM.read(position);
```

Эта инструкция извлекает байт данных из ячейки ЭСППЗУ с номером `position` и записывает его в переменную `value`.

ПРИМЕЧАНИЕ

ЭСППЗУ имеет ограниченный срок использования, и в конечном итоге хранящиеся в нем данные могут быть потеряны! Согласно утверждениям производителя Atmel, ЭСППЗУ может выдержать до 100 000 циклов стирания/записи в каждую ячейку. Число операций чтения не ограничивается.

Чтение и запись в ЭСППЗУ

Далее приводится пример чтения и записи данных в ЭСППЗУ. Введите и загрузите скетч из листинга 16.2.

Листинг 16.2. Скетч, демонстрирующий использование ЭСППЗУ

```
// Листинг 16.2

#include <EEPROM.h>

int zz;

void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop()
{
  Serial.println("Writing random numbers...");
  for (int i = 0; i < 1024; i++)
```

```
    {
      zz = random(255);
❶     EEPROM.write(i, zz);
    }
    Serial.println();
    for (int a = 0; a < 1024; a++)
    {
❷     zz = EEPROM.read(a);
      Serial.print("EEPROM position: ");
      Serial.print(a);
      Serial.print(" contains ");
❸     Serial.println(zz);
      delay(25);
    }
  }
```

В цикле ❶ в каждую ячейку ЭСППЗУ записывается случайное число от 0 до 255. Во втором цикле ❷ сохраненные значения извлекаются и отображаются в окне монитора порта ❸.

Загрузив скетч, откройте окно монитора порта и посмотрите на картину, аналогичную показанной на рис. 16.6.

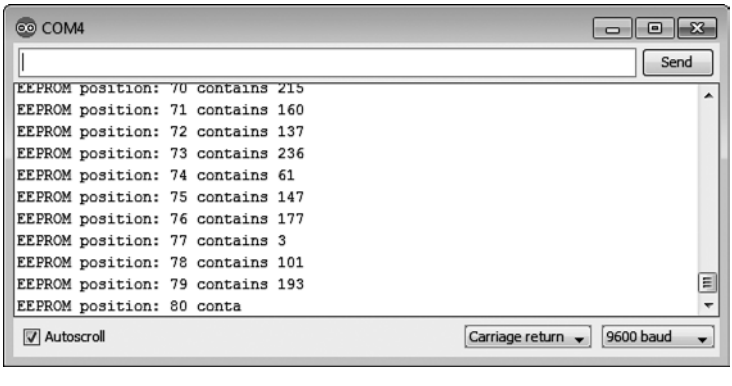


Рис. 16.6. Пример вывода скетча из листинга 16.2

Теперь вы готовы к обсуждению проекта, в котором используется ЭСППЗУ.

Проект № 53: RFID-система управления с запоминанием последнего действия

В проекте 52 было показано, как можно использовать технологию RFID для управления, например, освещением или электрическим замком, но тогда предполагалось,

что система не должна ничего запоминать при выключении питания или сбросе. Например, если после включения освещения устройство на основе Arduino будет выключено, то выключится и освещение. Но зачастую необходимо, чтобы после повторного включения плата Arduino «вспоминала» состояние, имевшее место к моменту отключения питания, и восстанавливала его. Давайте решим эту задачу.

В этом проекте последнее действие будет запоминаться в ЭСППЗУ (например, «заперто» или «отперто»). Когда скетч запустится после восстановления питания или сброса платы Arduino, система восстановит предыдущее состояние, хранящееся в ЭСППЗУ.

Скетч

Введите и загрузите следующий скетч. Как и прежде, замените символы *x* в массивах ❶ и ❷ числами, соответствующими вашим радиомаркерам, как это было сделано в проекте 52.

```
// Проект 53 – RFID-система управления с запоминанием
//             последнего действия

#include <EEPROM.h>

int data1 = 0;
int ok = -1;
int lockStatus = 0;

// чтобы определить числовые идентификаторы радиомаркеров,
// воспользуйтесь скетчем из листинга 16.1
❶ int tag1[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};
❷ int tag2[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};

// следующий массив используется для чтения и сравнения
int newtag[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0};

void setup()
{
  Serial.flush();
  Serial.begin(9600);
  pinMode(13, OUTPUT);
❸  checkLock();
}

// comparetag сравнивает два массива и возвращает true,
// если они идентичны; она отлично подходит для
// сравнения числовых идентификаторов RFID
boolean comparetag(int aa[14], int bb[14])
{
  boolean ff = false;
  int fg = 0;
```

```
    for (int cc = 0; cc < 14; cc++)
    {
      if (aa[cc] == bb[cc])
      {
        fg++;
      }
    }
    if (fg == 14)
    {
      ff = true;
    }
    return ff;
  }

  // сравнивает известные числовые идентификаторы
  // с только что прочитанным
  void checkmytags()
  {
    ok = 0;
    if (comparetag(newtag, tag1) == true)
    {
      ok++;
    }
    if (comparetag(newtag, tag2) == true)
    {
      ok++;
    }
  }

❹ void checkLock()
  {
    Serial.print("System Status after restart ");
    lockStatus = EEPROM.read(0);
    if (lockStatus == 1)
    {
      Serial.println("- locked");
      digitalWrite(13, HIGH);
    }
    if (lockStatus == 0)
    {
      Serial.println("- unlocked");
      digitalWrite(13, LOW);
    }
    if ((lockStatus != 1) && (lockStatus != 0))
    {
      Serial.println("EEPROM fault - Replace Arduino hardware");
    }
  }

  void loop()
  {
    ok = -1;
```

```
    if (Serial.available() > 0) // если была попытка чтения
    {
      // прочитать число из модуля RFID
      delay(100);
      for (int z = 0 ; z < 14 ; z++) // прочитать числовой
                                     // идентификатор радиомаркера
      {
        data1 = Serial.read();
        newtag[z] = data1;
      }
      Serial.flush(); // аннулировать повторные попытки чтения
      // теперь сравнить числовые идентификаторы
      checkmytags();
    }
❺   if (ok > 0) // прочитан известный радиомаркер
    {
      lockStatus = EEPROM.read(0);
      if (lockStatus == 1) // если заперт – отпереть
❻      {
        Serial.println("Status - unlocked");
        digitalWrite(13, LOW);
        EEPROM.write(0, 0);
      }
      if (lockStatus == 0)
❼      {
        Serial.println("Status - locked");
        digitalWrite(13, HIGH);
        EEPROM.write(0, 1);
      }
      if ((lockStatus != 1) && (lockStatus != 0))
❽      {
        Serial.println("EEPROM fault - Replace Arduino hardware");
      }
    }
    else if (ok == 0) // если прочитан неизвестный радиомаркер
    {
      Serial.println("Incorrect tag");
      ok = -1;
    }
    delay(500);
  }
```

Принцип действия

Этот скетч является измененной версией скетча из проекта 52. Здесь с помощью встроенного светодиода имитируется состояние замка, который запирается (включается) или отпирается (выключается) при поднесении известного радиомаркера RFID к антенне модуля чтения. После чтения и сопоставления числового идентификатора RFID (❺) состояние замка изменяется. Состояние сохраняется в первой ячейке ЭСППЗУ и может иметь два значения: `0` — «отперт» и `1` — «заперт».

Каждый раз, когда скетч получает идентификатор известного радиомаркера, он изменяет состояние замка на противоположное («отперт» на «заперт» и обратно на «отперт») в ❻ и ❼.

Мы также добавили обработку ошибки на случай, если значение в ЭСППЗУ стерлось. Если значение, извлеченное из ЭСППЗУ, не равно 0 или 1, в монитор порта выводится предупреждающее сообщение (❽). Кроме того, в момент повторного запуска скетча после сброса вызывается функция `checkLock()` (❶, ❷, ❸ и ❹), которая читает значение из ячейки в ЭСППЗУ, определяет последнее хранящееся состояние и затем восстанавливает состояние замка («заперто» или «отперто»).

Забегая вперед

И снова мы увидели, насколько просто реализуются проекты с применением Arduino, которые в иных условиях оказываются очень сложными. Теперь вы сможете добавлять в свои проекты поддержку технологии RFID и создавать системы контроля доступа профессионального уровня, управляя цифровыми выходами легким взмахом карты RFID. Мы еще раз вернемся к обсуждению этой технологии в главе 18.

17 Шины данных

В этой главе вы:

- ✓ познакомитесь с шиной I²C;
- ✓ узнаете, как использовать ЭСППЗУ (электрически стираемое программируемое постоянное запоминающее устройство) и расширитель порта на шине I²C;
- ✓ познакомитесь с шиной SPI;
- ✓ научитесь использовать цифровой реостат на шине SPI.

Взаимодействие с другими устройствами плата Arduino осуществляет посредством *шины данных* — системы соединений, позволяющей двум и более устройствам обмениваться данными упорядоченным образом. Шина данных используется для подключения к плате Arduino различных датчиков, расширительных устройств ввода/вывода и других компонентов.

Наибольшее значение для Arduino имеют две шины: *шина последовательного периферийного интерфейса* (Serial Peripheral Interface, SPI) и *шина связи интегральных схем* (Inter-Integrated Circuit, I²C). Многие датчики и внешние устройства поддерживают обмен данными по этим шинам.

Шина I²C

Шина I²C, также известная как *двухпроводной интерфейс* (Two Wire Interface, TWI), — простое и удобное устройство, используемое для обмена данными. Передача данных между устройствами и Arduino осуществляется по двум линиям, которые называют *линией данных* (Serial Data Line, SDA) и *тактовой линией* (Serial Clock Line, SCL). В модели Arduino Uno линия SDA выведена на контакт A4, а линия SCL — на контакт A5, как показано на рис. 17.1. Некоторые новейшие платы R3 имеют отдельные контакты, соединенные с шиной I²C и расположенные в верхнем левом углу для удобства доступа к ним.

Рис. 17.1. Контакты на плате Arduino Uno, служащие одновременно выводами шины I²C

Будучи подключенной к шине I²C, плата Arduino считается *ведущим устройством*, а все остальные устройства — *ведомыми*. Каждое ведомое устройство имеет свой адрес — шестнадцатеричное число, — позволяющий плате Arduino обращаться и взаимодействовать с каждым устройством по отдельности. Обычно устройство имеет на выбор диапазон адресов I²C, который указан в документации к нему. Конкретные доступные адреса определяются подключением контактов IC тем или иным образом.

ПРИМЕЧАНИЕ

Поскольку Arduino «питается» напряжением 5 В, подключаемые к ней устройства I²C также должны «питаться» напряжением 5 В или выдерживать его. Обязательно уточните эту информацию у продавца или производителя перед покупкой.

Чтобы использовать шину I²C, скетч сначала активирует библиотеку *Wire* (входит в состав Arduino IDE):

```
#include <Wire.h>
```

Затем, в функции `void setup()`, активируется шина:

```
Wire.begin();
```

Передача данных по шине осуществляется по одному байту. Чтобы послать байт из платы Arduino в устройство на шине, необходимо вызвать три функции:

1. Первая функция инициализирует связь, как показано ниже (где аргумент `address` — это адрес ведомого устройства на шине в шестнадцатеричном виде, например `0x50`):

   ```
   Wire.beginTransmission(address);
   ```

2. Вторая функция посылает 1 байт данных из Arduino в устройство с адресом, указанным в предыдущем вызове функции. Здесь аргумент `data` — это переменная, содержащая 1 байт данных; вы можете послать несколько байтов, но для каждого байта придется вызвать `Wire.write()`:

   ```
   Wire.write(data);
   ```

3. Наконец, по завершении передачи данных определенному устройству следует разорвать связь вызовом:

```
Wire.endTransmission();
```

Чтобы запросить данные из устройства на шине I²C, инициализируйте связь вызовом `Wire.beginTransmission(address)` и отправьте запрос:

```
Wire.requestFrom(address,x);
```

(где `x` — количество запрашиваемых байтов данных). Затем с помощью следующей функции нужно сохранить принятый байт в переменной:

```
incoming = Wire.read(); // incoming - переменная, куда
                        // сохраняется принятый байт данных
```

И по окончании приема следует разорвать связь вызовом `Wire.endTransmission()`. Все эти функции мы используем в следующем проекте.

Проект № 54: Внешнее ЭСППЗУ

Рис. 17.2. Микросхема ЭСППЗУ Microchip Technology 24LC512

В главе 16 мы использовали ЭСППЗУ, встроенное в микроконтроллер на плате Arduino, чтобы обеспечить сохранность данных в случае сброса или отключения питания. Встроенное ЭСППЗУ способно хранить всего 1024 байт данных. Для хранения большего объема необходимо использовать внешние ЭСППЗУ, с которыми вы познакомитесь в этом проекте.

В качестве внешнего ЭСППЗУ мы используем микросхему 24LC512, выпускаемую компанией Microchip Technology, способную хранить 64 кбайт (65 536 байт) данных (рис. 17.2). Ее можно приобрести у таких продавцов, как Digi-Key (артикул 24LC512-I/P-ND) и element14 (артикул 9758020).

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Одна микросхема ЭСППЗУ Microchip Technology 24LC512.
- Одна макетная плата.
- Два резистора с номиналом 4,7 кОм.

- Один керамический конденсатор емкостью 100 нФ.
- Несколько отрезков провода разной длины.
- Плата Arduino и кабель USB.

Схема

Подключите через резисторы 4,7 кОм линии SCL и SDA к контакту 5V, как показано на рис. 17.3.

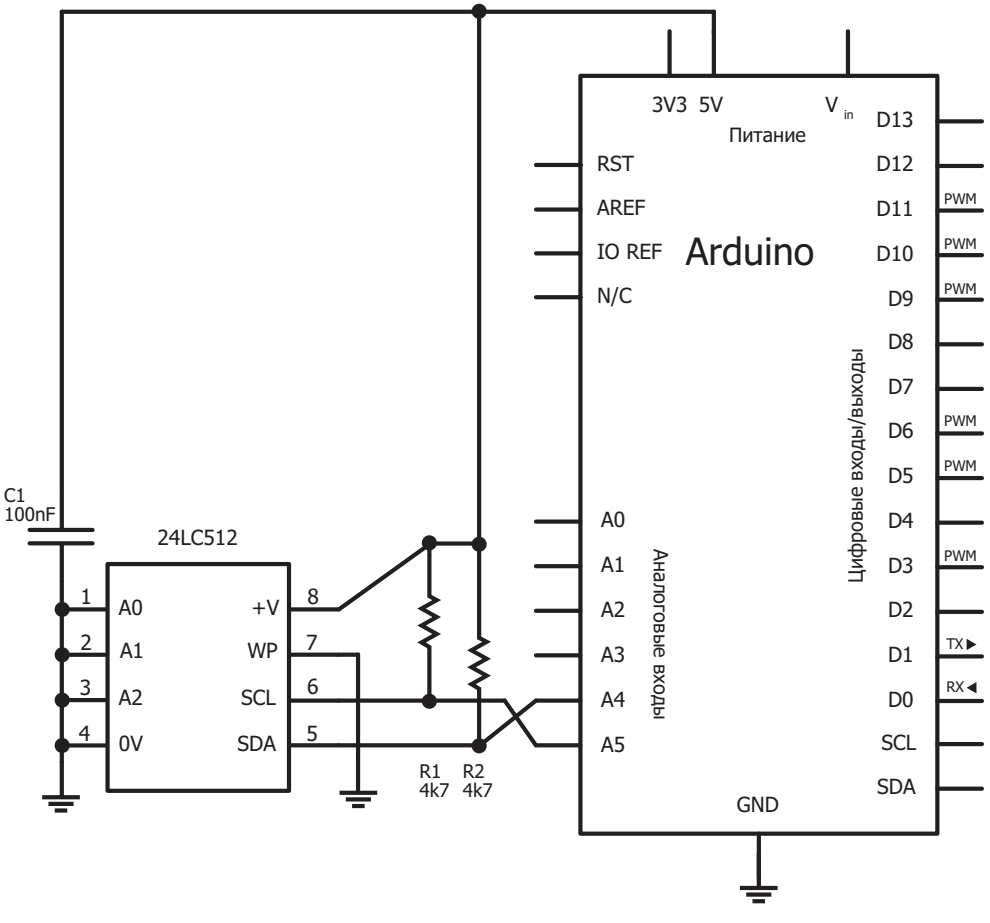


Рис. 17.3. Принципиальная схема проекта 54

Адрес ЭСППЗУ 24LC512 на шине I²C частично определяется схемой подключения. Последние 3 бита адреса на шине определяются состоянием выводов A2, A1

и A0 микросхемы. При подключении их к «земле» они принимают значение 0; при подключении к контакту 5V они принимают значение 1.

Первые четыре бита адреса предустановлены в состояние `1010`. То есть в данной схеме полный адрес ЭСППЗУ на шине I²C имеет вид: `1010000` — в двоичном представлении или `0x50` — в шестнадцатеричном. Это означает, что в скетче мы должны использовать адрес `0x50`.

Скетч

Несмотря на то что внешнее ЭСППЗУ может хранить до 64 Кбайт данных, наш демонстрационный скетч будет сохранять и извлекать байты только из 20 первых ячеек в ЭСППЗУ.

Введите и загрузите следующий скетч:

```
  // Проект 54 - Внешнее ЭСППЗУ
❶ #include <Wire.h>
  #define chip1 0x50

  unsigned int pointer;
  byte d=0;

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

  void writeData(int device, unsigned int address, byte data)
  // записывает байт данных 'data' в ЭСППЗУ с I2C-адресом 'device'
  // в ячейку с номером 'address'
  {
❸   Wire.beginTransmission(device);
    Wire.write((byte)(address >> 8));   // старший байт номера ячейки
    Wire.write((byte)(address & 0xFF)); // и младший байт
    Wire.write(data);
    Wire.endTransmission();
    delay(10);
  }

❹ byte readData(int device, unsigned int address)
  // читает байт данных из ячейки с номером 'address'
  // в ЭСППЗУ с I2C-адресом 'device'
  {
    byte result;                        // возвращаемое значение
    Wire.beginTransmission(device);
    Wire.write((byte)(address >> 8));   // старший байт номера ячейки
    Wire.write((byte)(address & 0xFF)); // и младший байт
    Wire.endTransmission();
```

```
❺   Wire.requestFrom(device,1);
    result = Wire.read();
    return result;                        // вернуть прочитанный байт как
                                          // результат функции readData
}

void loop()
{
  Serial.println("Writing data...");
  for (int a=0; a<20; a++)
  {
    writeData(chip1,a,a);
  }
  Serial.println("Reading data...");
  for (int a=0; a<20; a++)
  {
    Serial.print("EEPROM position ");
    Serial.print(a);
    Serial.print(" holds ");
    d=readData(chip1,a);
    Serial.println(d, DEC);
  }
}
```

Давайте пройдемся по скетчу. В ❶ выполняется подключение библиотеки и определяется адрес ЭСППЗУ на шине I²C в виде константы с именем `chip1`. В ❷ инициализируется монитор порта, а затем — шина I²C. В скетч включены две нестандартные функции — `writeData()` и `readData()`, — чтобы сэкономить время и предоставить в ваше распоряжение некий код, который вы сможете не раз использовать в своих будущих проектах для работы с устройствами ЭСППЗУ, поддерживающими подключение к шине I²C. Они позволяют записывать и извлекать данные в/из ЭСППЗУ.

Функция `writeData()` (❸) инициализирует связь с ЭСППЗУ, посылает номер ячейки, куда предполагается записать байт данных, дважды вызывая функцию `Wire.write()`, посылает байт данных для записи и разрывает связь.

Функция `readData()` (❹) действует похожим образом, но не посылает байт данных в ЭСППЗУ, а вызывает `Wire.requestFrom()`, чтобы прочитать данные (❺). В заключение байт данных, полученный из ЭСППЗУ, присваивается переменной `result` и возвращается как значение функции.

Результат

В функции `void loop()` скетч выполняет 20 итераций и записывает значения (от 0 до 19) в ЭСППЗУ. Затем выполняются еще 20 итераций, в ходе которых извлекаются значения из 20 первых ячеек и выводятся на монитор порта, как показано на рис. 17.4.

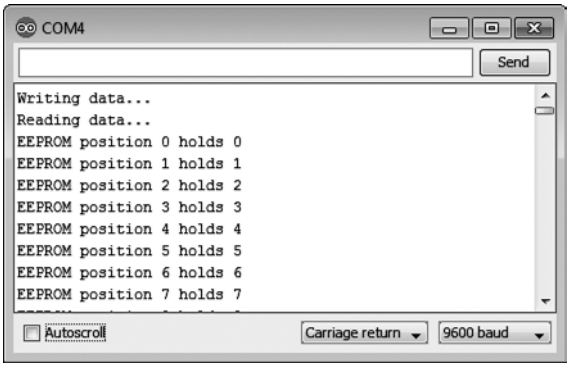


Рис. 17.4. Результаты работы проекта 54

Проект № 55: Расширитель порта

Расширитель порта — еще одно полезное устройство, подключаемое к шине I^2C. Это устройство предназначено для увеличения количества цифровых входов/выходов. В данном проекте демонстрируется 16-битный расширитель порта Microchip Technology MCP23017 (рис. 17.5), имеющий 16 цифровых входов/выходов. Его можно приобрести у таких продавцов, как Newark (артикул 31K2959) и element14 (артикул 1332088).

Рис. 17.5. Расширитель порта Microchip Technology MCP23017

В этом проекте мы подключим MCP23017 к плате Arduino и посмотрим, как управлять входами/выходами 16-битного расширителя порта с помощью Arduino. Все входы/выходы расширителя можно использовать подобно обычным цифровым входам/выходам Arduino.

Оборудование

Ниже перечислено оборудование, необходимое для этого проекта:

- Плата Arduino и кабель USB.
- Одна макетная плата.
- Несколько отрезков провода разной длины.

- Одна микросхема расширителя порта Microchip Technology MCP20317.
- Два резистора с номиналом 4,7 кОм.
- (Необязательно) равное количество резисторов с номиналом 560 Ом и светодиодов.

Схема

На рис. 17.6 изображена базовая схема подключения расширителя порта MCP23017. Так же как при подключении ЭСППЗУ в проекте 54, мы можем установить адрес устройства на шине I^2C, подключив его выводы определенным образом. Подключите выводы MCP23017 с 15-го по 17-й к контакту GND, чтобы установить адрес `0x20`.

При работе с MCP23017 я рекомендую иметь перед глазами схему разводки выводов из документации с описанием, изображенную на рис. 17.7. Обратите внимание, что 16 цифровых входов/выходов делятся на две группы: выводы с GPA7 по GPA0 находятся справа, а выводы с GPB0 по GPB7 — слева. Подключите светодиоды (через резисторы с номиналом 560 Ом) ко всем или к выбранным выходам, чтобы иметь возможность наглядно видеть, когда на них устанавливается высокий уровень.

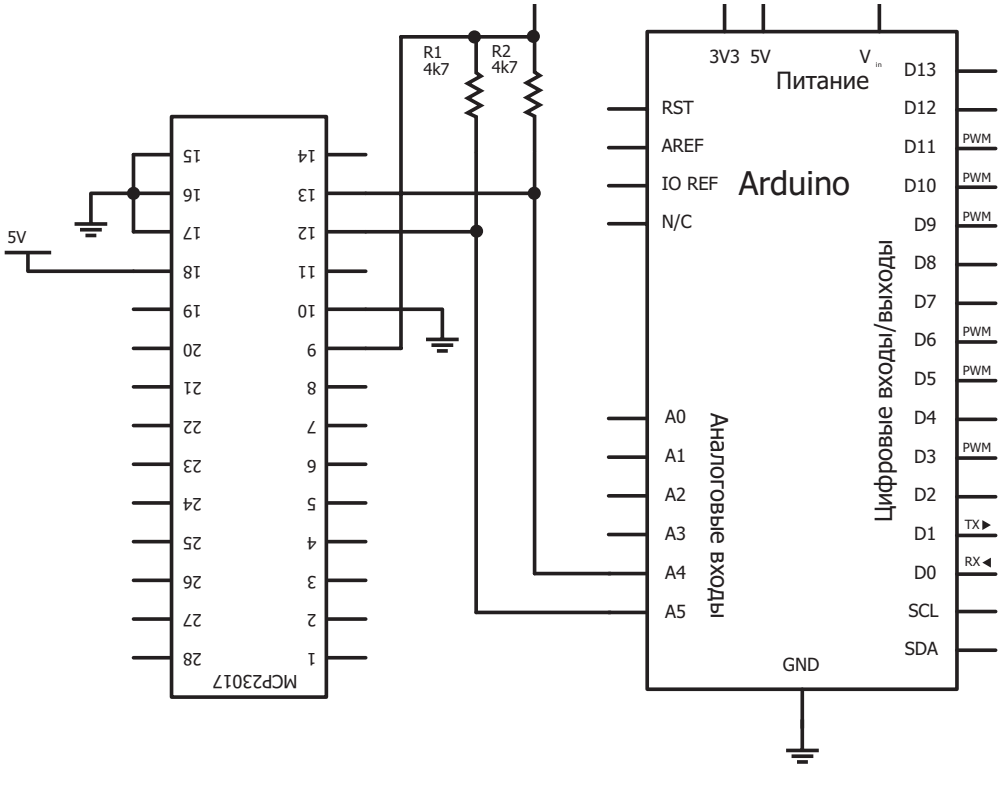


Рис. 17.6. Базовая принципиальная схема проекта 55

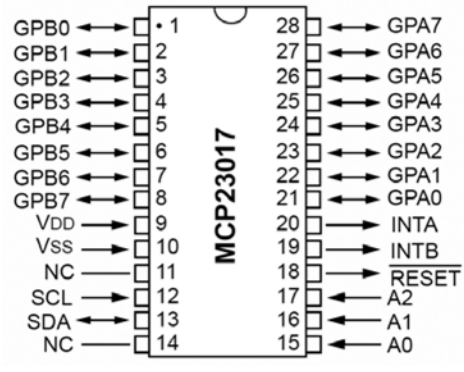


Рис. 17.7. Схема разводки выводов MCP23017

Скетч

Введите и загрузите следующий скетч:

```
// Проект 55 - Расширитель порта

#include "Wire.h"
#define mcp23017 0x20

void setup()
{
  Wire.begin(); // инициализирует шину I2C
❶  // Настроить входы/выходы MCP23017
  // на работу в режиме выходов
  Wire.beginTransmission(mcp23017);
  Wire.write(0x00); // регистр IODIRA
  Wire.write(0x00); // все выводы в группе A - выходы
  Wire.write(0x00); // все выводы в группе B - выходы
❷  Wire.endTransmission();
}

void loop()
{
  Wire.beginTransmission(mcp23017);
  Wire.write(0x12);
❸  Wire.write(255); // группа A
❹  Wire.write(255); // группа B
  Wire.endTransmission();
  delay(1000);
  Wire.beginTransmission(mcp23017);
  Wire.write(0x12);
  Wire.write(0); // группа A
  Wire.write(0); // группа B
  Wire.endTransmission();
  delay(1000);
}
```

Для работы с расширителем MCP23017 необходимы все строки в функции `void setup()` с ❶ по ❷. Чтобы изменить состояние выходов в каждой группе, по очереди посылаются байты, представляющие каждую группу; то есть сначала посылается байт, представляющий группу выходов с GPA0 по GPA7, а затем байт, представляющий группу выходов с GPB0 по GPB7.

Если потребуется изменить состояния отдельных выходов, рассматривайте каждую группу как двоичное число (как описывается в разделе «Краткое введение в двоичную систему счисления» в главе 6). То есть, чтобы установить высокий уровень на выходах с 7-го по 4-й, нужно послать двоичное число 11110000 (или десятичное 240), передав его функции `Wire.write()`, как показано в ❸ для группы GPA0–GPA7 или в ❹ для группы GPB0–GPB7.

Существуют сотни устройств, поддерживающих взаимодействия по шине I²C. Теперь, зная, как работать с этой шиной, вы легко сможете подключать и использовать I²C-устройства с платой Arduino.

Шина SPI

Шина SPI, в отличие от I²C, может применяться для одновременной передачи данных в обоих направлениях и с разными скоростями, в зависимости от типа используемого микроконтроллера. Однако сами взаимодействия все так же осуществляются по схеме *ведущий/ведомый*: плата Arduino играет роль *ведущего* устройства и определяет, с каким устройством (ведомым) она будет взаимодействовать.

Контакты

Каждое устройство, поддерживающее подключение к шине SPI, имеет четыре линии, по которым осуществляется обмен данными: *MOSI* (Master-Out, Slave-In — ведущий посылает, ведомый принимает), *MISO* (Master-In, Slave-Out — ведущий принимает, ведомый посылает), SCK (тактовая линия) и SS или CS (Slave Select или Chip Select — выбор ведомого или выбор устройства). Эти линии шины SPI подключаются к плате Arduino, как показано на рис. 17.8.

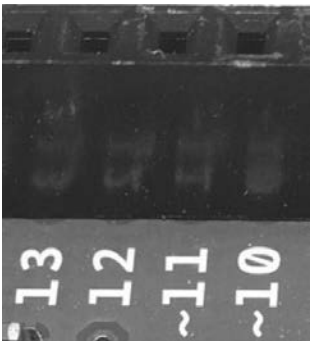


Рис. 17.8. Контакты на плате Arduino Uno, служащие одновременно выводами шины SPI

На рис. 17.9 показана схема типичного подключения устройства к плате Arduino посредством шины SPI. Контакты с D11 по D13 на плате зарезервированы для подключения линий MISO, MOSI и SCK шины SPI, но линию SS можно подключить к любому другому цифровому контакту (часто для этого используется контакт D10, потому что он расположен рядом с контактами, зарезервированными для шины SPI).

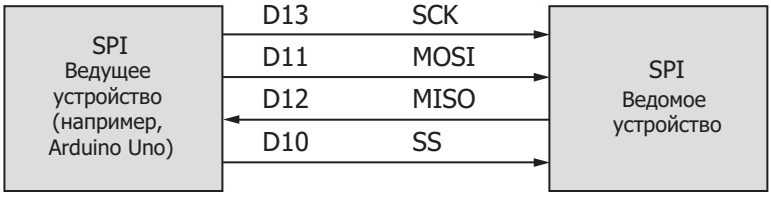


Рис. 17.9. Схема типичного подключения SPI-устройства к плате Arduino

ПРИМЕЧАНИЕ

Подобно I²C-устройствам SPI-устройства должны питаться напряжением 5 В или выдерживать его, поскольку Arduino работает от 5 В. Обязательно уточняйте эту информацию у продавца или производителя перед покупкой.

Реализация обмена данными по шине SPI

Давайте теперь посмотрим, как реализовать в скетче обмен данными по шине SPI. Но прежде познакомимся с некоторыми функциями. Прежде всего, к скетчу нужно подключить библиотеку *SPI* (входит в состав Arduino IDE):

```
#include "SPI.h"
```

Затем, в функции `void setup` настроим контакт, выбранный для подключения линии SS, на работу в режиме цифрового выхода. Так как в примере ниже используется только одно SPI-устройство, мы будем использовать контакт `D10` и сразу же устанавливать на нем уровень `HIGH`, потому что большинство SPI-устройств переходят в активное состояние, когда на линии SS устанавливается низкий уровень:

```
pinMode(10, OUTPUT);
digitalWrite(10, HIGH);
```

Следующая функция инициализирует шину SPI:

```
SPI.begin();
```

Наконец, скетч должен определить, как именно будут посылаться и приниматься данные. Некоторые SPI-устройства требуют, чтобы первым посылался старший бит (Most Significant Bit, MSB), а некоторые, наоборот, требуют, чтобы старший бит посылался последним. (И снова загляните в раздел «Краткое введение в двоичную систему счисления» в главе 6, где рассказывается о том, что такое «старший бит».) Поэтому внутри функции `void setup` сразу вслед за вызовом `SPI.begin` мы вызовем:

```
SPI.setBitOrder(order);
```

где аргумент `order` может иметь значение `MSBFIRST` или `MSBLAST`.

Передача данных SPI-устройству

Чтобы послать данные SPI-устройству, сначала нужно установить на линии SS уровень `LOW` и таким образом сообщить устройству, что оно выбрано ведущим устройством (платой Arduino) для взаимодействий. А затем послать необходимое количество байтов данных соответствующим количеством вызовов следующей функции, — то есть эту функцию придется вызвать для каждого байта:

```
SPI.transfer(byte);
```

Завершив взаимодействие с устройством, следует установить уровень `HIGH` на линии SS, чтобы сообщить устройству, что плата Arduino завершила сеанс связи.

Каждое SPI-устройство должно подключаться отдельной линией SS. Например, если к шине SPI подключено два устройства, линия SS, соединяющая второе устройство, подключается к контакту D9 на плате Arduino, как показано на рис. 17.10.

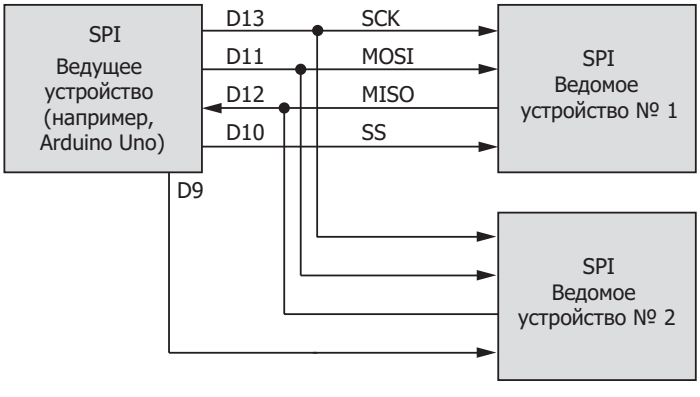


Рис. 17.10. Подключение двух SPI-устройств к плате Arduino

Для взаимодействия со вторым ведомым устройством до и после каждого сеанса связи используйте контакт D9 (вместо D10).

Проект 56 демонстрирует использование шины SPI для взаимодействия с цифровым реостатом.

Проект № 56: Цифровой реостат

Выражаясь простым языком, *реостат* — это устройство, похожее на резистор переменного сопротивления, с которым мы познакомились в главе 4, с той лишь разницей, что реостат имеет два вывода: один является аналогом среднего вывода переменного резистора, а второй — одного из его концов. В данном проекте мы будем менять сопротивление с помощью цифрового реостата вместо поворота ручки

переменного резистора вручную. Реостаты часто применяются для регулировки громкости в аудиооборудовании, где вместо поворотных ручек используются кнопки. Допуск отклонений сопротивления реостата от номинала намного больше, чем обычного постоянного сопротивления, — в некоторых случаях он доходит до 20 %.

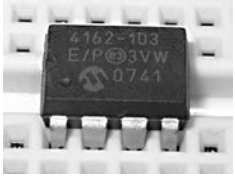


Рис. 17.11. Реостат Microchip Technology MCP4162

В проекте 56 мы используем реостат Microchip Technology MCP4162, изображенный на рис. 17.11. Реостат MCP4162 выпускается с разными номиналами сопротивлений; для нашего проекта выберем модель с номиналом 10 кОм. Ее можно приобрести у таких продавцов, как Newark (артикул 77M2766) и element14 (артикул 1840698). Выходное сопротивление имеет 255 уровней регулировки с шагом около 40 Ом. Для выбора определенного уровня необходимо послать 2 байта: байт команды (который имеет значение 0) и байт значения (определяет уровень от 0 до 255). В реостате MCP4162 используется энергонезависимая память, поэтому после отключения и последующего включения питания устанавливается последний выбранный уровень.

Наш реостат предназначен для управления яркостью свечения светодиода.

Оборудование

Ниже перечислено оборудование, необходимое для этого проекта:

- Плата Arduino и кабель USB.
- Одна макетная плата.
- Несколько отрезков провода разной длины.
- Один реостат Microchip Technology MCP4162.
- Один резистор с номиналом 560 Ом.
- Один светодиод.

Схема

На рис. 17.12 изображена принципиальная схема проекта 56. Нумерация выводов микросхемы MCP4162 начинается слева снизу. Вывод 1 обозначен точкой слева от логотипа Microchip (рис. 17.11).

Скетч

Введите и загрузите следующий скетч:

```
  // Проект 56 - Цифровой реостат
❶ #include "SPI.h" // подключить библиотеку
 int ss=10;       // использовать цифровой контакт 10
```

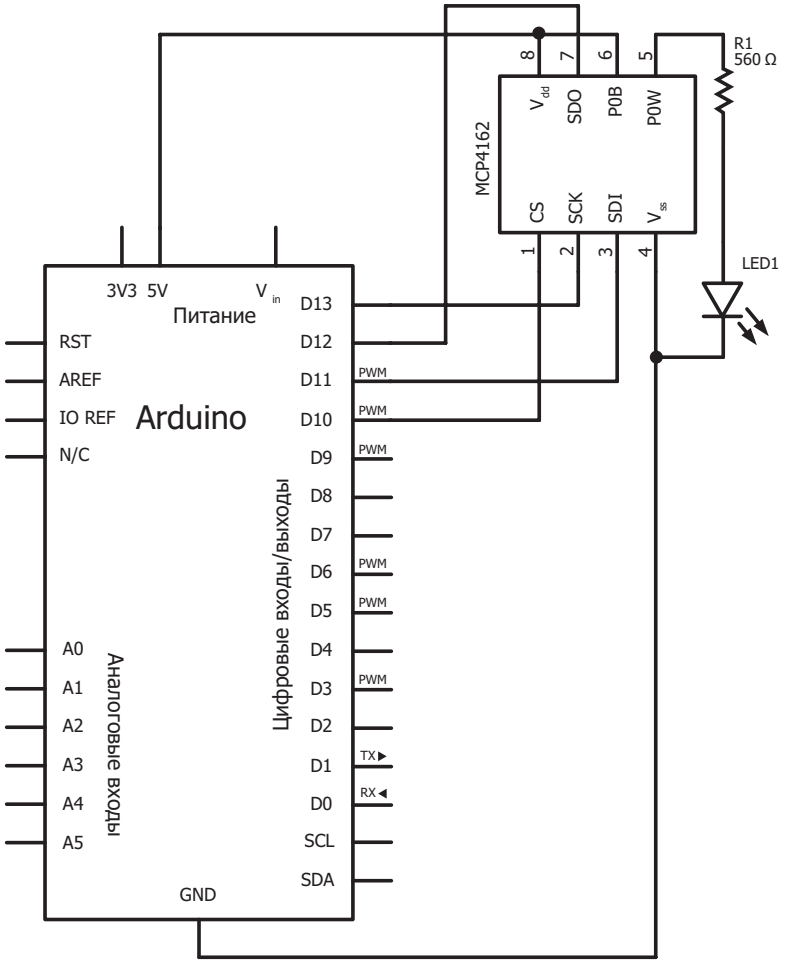


Рис. 17.12. Принципиальная схема проекта 56

```
                  // для выбора ведомого SPI-устройства
int del=200;      // задержка между изменениями
                  // уровня яркости светодиода

void setup()
{
❷  SPI.begin();
   pinMode(ss, OUTPUT);    // настройка контакта
                           // для подключения линии SS
   digitalWrite(ss, HIGH); // выбор устройства происходит при низком
                           // уровне на линии SS, поэтому первоначально
                           // установить высокий уровень
❸  SPI.setBitOrder(MSBFIRST);
   // реостат MCP4162 требует передавать биты данных в порядке
```

```
    // от старшего к младшему
  }

❹ void setValue(int value)
  {
    digitalWrite(ss, LOW);
    SPI.transfer(0);     // послать байт команды
    SPI.transfer(value); // послать значение (от 0 до 255)
    digitalWrite(ss, HIGH);
  }

  void loop()
  {
❺   for (int a=0; a<256; a++)
    {
      setValue(a);
      delay(del);
    }
❻   for (int a=255; a>=0; a--)
    {
      setValue(a);
      delay(del);
    }
  }
```

Теперь пройдемся по скетчу. В первую очередь, выполняется подключение библиотеки *SPI* и настройка шины (❶ и ❷). В ❸ определяется порядок передачи данных, поддерживаемый реостатом MPC4162. Чтобы упростить регулировку сопротивления, в скетче используется нестандартная функция (❹), которая принимает уровень сопротивления (от 0 до 255) и посылает его в MCP4162. Наконец, скетч выполняет два цикла, перебирающие все уровни сопротивления, от нуля до максимума (❺) и обратно до нуля (❻). Этот последний фрагмент заставит светодиод увеличивать и уменьшать яркость свечения снова и снова, пока выполняется скетч.

Забегая вперед

В этой главе вы познакомились и поэкспериментировали с двумя важными способами взаимодействий, поддерживаемыми платой Arduino. Теперь вы сможете подключить к ней великое множество разнообразных датчиков и других компонентов, имеющихся на рынке. Одним из наиболее популярных компонентов на сегодняшний день являются I²C-часы реального времени, позволяющие проектам определять текущее время и осуществлять синхронную работу, — это станет темой главы 18. Итак, продолжим!

18 Часы реального времени

В этой главе вы:

- ✓ научитесь устанавливать и извлекать время и дату из модуля часов реального времени;
- ✓ познакомитесь с новыми способами подключения устройств к плате Arduino;
- ✓ сконструируете цифровые часы;
- ✓ сконструируете таймер, запускаемый радиомаркером.

I²C-модуль *часов реального времени* (Real-Time Clock, RTC) — это небольшое устройство-хронометр, открывающее новые возможности для проектов на основе Arduino. После установки текущего времени и даты модуль RTC обеспечит высокую точность хода внутренних часов и возврат времени и даты по запросу.

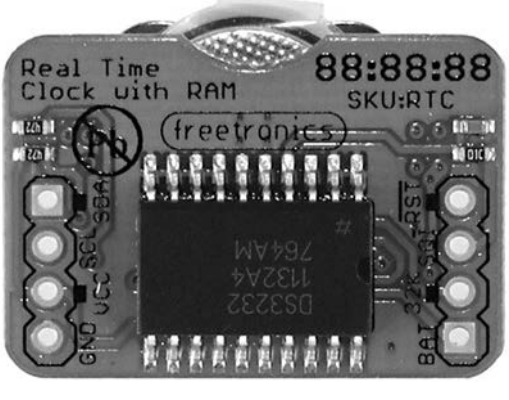


Рис. 18.1. I²C-модуль часов реального времени

На рынке имеется много разных модулей часов реального времени, обеспечивающих разную точность измерения. В этой главе мы воспользуемся модулем Maxim DS3232; он не требует подключения внешних компонентов, кроме батарейки резервного питания, и гарантирует высокую точность и надежность. Модуль DS3232 выпускается разными производителями, он выглядит как небольшая плата: например, модуль от компании Freetronics (*http://www.freetronics.com/rtc/*), приведенный на рис. 18.1.

Подключение модуля RTC

Подключить модуль RTC к плате Arduino просто, для этого используется шина I²C (представленная в главе 17). Берем четыре провода: выводы GND и VCC на плате модуля нужно соединить с контактами GND и 5V на плате Arduino соответственно, а выводы SDA и SCL — с контактами A4 и A5. Из-за особенностей конструкции модуля к шине I²C не требуется подключать дополнительные подтягивающие резисторы, если только кабель, соединяющий модуль и Arduino, не слишком длинный. В противном случае спаяйте перемычки между площадками, подписанными как «Pullups: SDA and SCL» (подтягивающие резисторы на линиях SDA и SCL), чтобы подключить встроенные подтягивающие резисторы.

Для удобства рассмотрим порядок монтирования модуля на пустой макетной плате ProtoShield, это упростит его интеграцию с другими электронными компонентами в будущих проектах. И не забудьте вставить батарейку резервного питания, иначе время будет сбрасываться при каждом выключении питания!

Проект № 57: Установка и отображение даты и времени

В этом проекте вы узнаете, как устанавливать дату и время в модуле RTC и как затем извлекать и отображать их в окне монитора порта. Знание даты и времени может пригодиться в самых разных проектах, например в термометре с памятью или в будильнике.

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Плата Arduino и кабель USB.
- Несколько отрезков провода разной длины.
- Модуль Maxim DS3232 RTC.

Скетч

Подключите модуль к плате Arduino, как описано выше в этой главе, и затем введите, *но не загружайте* следующий скетч:

```
  // Проект 57 - Установка и отображение даты и времени
❶ #include "Wire.h"
  #define DS3232_I2C_ADDRESS 0x68

  // Преобразует обычное десятичное число в двоично-десятичное
```

```
❷ byte decToBcd(byte val)
  {
    return( (val/10*16) + (val%10) );
  }

  // Преобразует двоично-десятичное число в обычное десятичное
  byte bcdToDec(byte val)
  {
    return( (val/16*10) + (val%16) );
  }

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

    // установить начальное время:
    // секунды, минуты, часы, день недели, число, месяц, год
❸   setDS3232time(0, 56, 23, 3, 30, 10, 12);
  }

❹ void setDS3232time(byte second, byte minute, byte hour,
                     byte dayOfWeek, byte dayOfMonth,
                     byte month, byte year)
  {
    // записать дату и время в DS3232
    Wire.beginTransmission(DS3232_I2C_ADDRESS);
    Wire.write(0); // выбрать для записи регистр секунд
    Wire.write(decToBcd(second)); // записать секунды
    Wire.write(decToBcd(minute)); // записать минуты
    Wire.write(decToBcd(hour));   // записать часы
    Wire.write(decToBcd(dayOfWeek));  // записать день недели (1=воскресенье,
7=суббота)
    Wire.write(decToBcd(dayOfMonth)); // записать число месяца (от 1 до 31)
    Wire.write(decToBcd(month));  // записать месяц
    Wire.write(decToBcd(year));   // записать год (от 0 до 99)
    Wire.endTransmission();
  }

❺ void readDS3232time(byte *second, byte *minute, byte *hour,
                      byte *dayOfWeek, byte *dayOfMonth,
                      byte *month, byte *year)
  {
    Wire.beginTransmission(DS3232_I2C_ADDRESS);
    Wire.write(0); // инициализировать регистр указателя в DS3232
    Wire.endTransmission();
    Wire.requestFrom(DS3232_I2C_ADDRESS, 7);

    // прочитать семь байт данных из DS3232, начиная с регистра 00h
    *second     = bcdToDec(Wire.read() & 0x7f);
    *minute     = bcdToDec(Wire.read());
    *hour       = bcdToDec(Wire.read() & 0x3f);
```

```
    *dayOfWeek  = bcdToDec(Wire.read());
    *dayOfMonth = bcdToDec(Wire.read());
    *month      = bcdToDec(Wire.read());
    *year       = bcdToDec(Wire.read());
  }

  void displayTime()
  {
    byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

    // прочитать данные из DS3232
❻   readDS3232time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth,
                   &month, &year);

    // вывести в монитор порта
    Serial.print(hour, DEC);
    // каждый байт данных выводить как десятичное число
    Serial.print(":");
    if (minute<10)
    {
      Serial.print("0");
    }
    Serial.print(minute, DEC);
    Serial.print(":");
    if (second<10)
    {
      Serial.print("0");
    }
    Serial.print(second, DEC);
    Serial.print(" ");
    Serial.print(dayOfMonth, DEC);
    Serial.print("/");
    Serial.print(month, DEC);
    Serial.print("/");
    Serial.print(year, DEC);
    Serial.print(" Day of week: ");
    switch(dayOfWeek){
    case 1:
      Serial.println("Sunday");
      break;
    case 2:
      Serial.println("Monday");
      break;
    case 3:
      Serial.println("Tuesday");
      break;
    case 4:
      Serial.println("Wednesday");
      break;
    case 5:
      Serial.println("Thursday");
      break;
```

```
    case 6:
      Serial.println("Friday");
      break;
    case 7:
      Serial.println("Saturday");
      break;
    }
  }

  void loop()
  {
    displayTime(); // выводить текущее время в монитор порта
    delay(1000);   // каждую секунду
  }
```

Принцип действия

На первый взгляд скетч выглядит сложным, но в действительности в нем нет ничего сложного. В ❶ подключается библиотека поддержки шины I²C и определяется адрес модуля RTC на шине как `0x68`. В ❷ объявляются две нестандартные функции для преобразования десятичных чисел в двоично-десятичные и обратно. Это необходимо, поскольку DS3232 хранит значения в двоично-десятичном формате.

В ❸ вызывается функция `setDS3232time` для записи времени и даты в модуль RTC, которая имеет следующий синтаксис:

```
setDS3232time(second, minute, hour, dayOfWeek, dayOfMonth, month, year)
```

Чтобы воспользоваться этой функцией, просто подставьте необходимые данные взамен параметров. Параметр `dayOfWeek` — это число в диапазоне от 1 до 7, представляющее день недели, от воскресенья до субботы соответственно. Параметр `year` интерпретируется как двузначное число, например 2013 году соответствует число 13. (Номер века 20 подразумевается.) В параметрах можно передавать фиксированные значения (как в скетче) или переменные типа `byte`.

Запись даты и времени в модуль RTC осуществляет функция `setDS3232time` (❹). Теперь загрузите скетч. После этого закомментируйте вызов функции, добавив пару символов `//` в начало строки ❸ перед именем `setDS3232time`, и загрузите скетч повторно, чтобы значение времени не устанавливалось снова и снова после каждого запуска скетча!

Наконец, функция `readDS3232time` (❺) читает дату и время из модуля RTC и сохраняет их в переменных типа `byte`. Она вызывается в ❻, внутри функции `displayTime`, которая извлекает данные и выводит их в монитор порта.

Теперь загрузите скетч и откройте окно монитора порта. Результаты должны выглядеть примерно так, как показано на рис. 18.2.

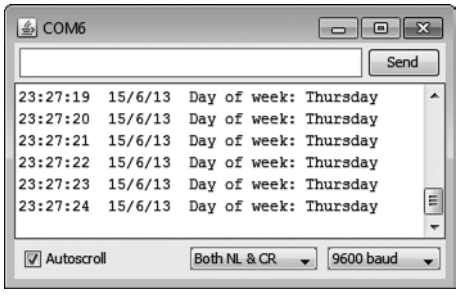


Рис. 18.2. Результаты работы проекта 57

Скетч для проекта 57 можно взять как основу для других проектов, использующих дату и время. Функции `decToBcd`, `bcdToDec`, `readDS3232time` и `setDS3232time` можно свободно копировать и применять в других проектах. Это одно из достоинств платформы Arduino: однажды написанная процедура может использоваться в других проектах с небольшими изменениями или вообще без изменений.

Проект № 58: Простые цифровые часы

В этом проекте мы воспользуемся функцией из проекта 57 для отображения времени и даты на экране стандартного ЖКИ, с подобным экраном мы имели дело в проекте 44 (простой приемник GPS).

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Плата Arduino и кабель USB.
- Несколько отрезков провода разной длины.
- Одна макетная плата.
- Одна макетная плата расширения ProtoScrewShield или подобная ей.
- Модуль ЖКИ или плата расширения Freetronics LCD.
- Модуль часов реального времени (показанный выше в этой главе).

Сначала воссоздадим устройство, использовавшееся в проекте 57. Если в предыдущем проекте вы подключали модуль RTC к плате Arduino проводами, теперь для достижения той же цели воспользуйтесь платой ProtoScrewShield. Затем вставьте плату расширения LCD поверх других плат расширения.

PROTOSCREWSHIELD

Иногда проекты, включающие несколько плат расширения и внешних устройств, превращаются в клубок из перепутанных проводов. Чтобы этого не происходило, используйте платы расширения ProtoScrewShield для Arduino, выпускаемые компанией Wingshield Industries (*http://wingshieldindustries.com/*), изображенные на рис. 18.3.

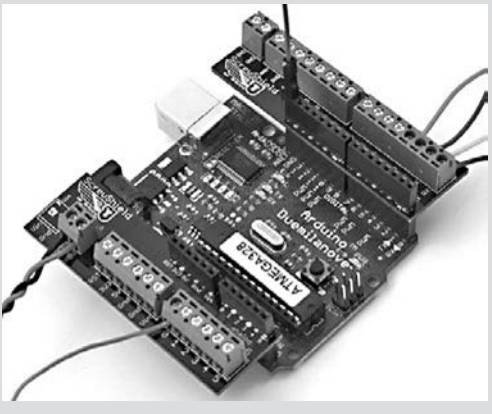


Рис. 18.3. Платы расширения ProtoScrewShield

Один комплект состоит из двух плат — по одной на каждый ряд разъемов на плате Arduino. Вставив такие платы в разъемы, вы не только продолжите пользоваться платой расширения, но и подключите с помощью проводов внешние устройства, такие как датчики или сервоприводы, непосредственно к контактам на плате Arduino, используя для этого удобные винтовые зажимы на платах ProtoScrewShield.

Скетч

Введите и загрузите следующий скетч:

```
// Проект 58 - Простые цифровые часы

#include "Wire.h"
#define DS3232_I2C_ADDRESS 0x68

❶ #include <LiquidCrystal.h>
LiquidCrystal lcd( 8, 9, 4, 5, 6, 7 );

// Преобразует обычное десятичное число в двоично-десятичное
byte decToBcd(byte val)
{
  return( (val/10*16) + (val%10) );
}
```

```
  // Преобразует двоично-десятичное число в обычное десятичное
  byte bcdToDec(byte val)
  {
    return( (val/16*10) + (val%16) );
  }

  void setup()
  {
    Wire.begin();
❷   lcd.begin(16, 2);
    // установить начальное время:
    // секунды, минуты, часы, день недели, число, месяц, год
❸  //setDS3232time(0, 27, 0, 5, 15, 11, 12);
  }

  void setDS3232time(byte second, byte minute, byte hour,
                     byte dayOfWeek, byte dayOfMonth,
                     byte month, byte year)
  {
    // записать дату и время в DS3232
    Wire.beginTransmission(DS3232_I2C_ADDRESS);
    Wire.write(0); // выбрать для записи регистр секунд
    Wire.write(decToBcd(second)); // записать секунды
    Wire.write(decToBcd(minute)); // записать минуты
    Wire.write(decToBcd(hour));   // записать часы
    Wire.write(decToBcd(dayOfWeek));  // записать день недели (1=воскресенье,
7=суббота)
    Wire.write(decToBcd(dayOfMonth)); // записать число месяца (от 1 до 31)
    Wire.write(decToBcd(month));  // записать месяц
    Wire.write(decToBcd(year));   // записать год (от 0 до 99)
    Wire.endTransmission();
  }

  void readDS3232time(byte *second, byte *minute, byte *hour,
                      byte *dayOfWeek, byte *dayOfMonth,
                      byte *month, byte *year)
  {
    Wire.beginTransmission(DS3232_I2C_ADDRESS);
    Wire.write(0); // инициализировать регистр указателя в DS3232
    Wire.endTransmission();
    Wire.requestFrom(DS3232_I2C_ADDRESS, 7);

    // прочитать семь байт данных из DS3232, начиная с регистра 00h
    *second     = bcdToDec(Wire.read() & 0x7f);
    *minute     = bcdToDec(Wire.read());
    *hour       = bcdToDec(Wire.read() & 0x3f);
    *dayOfWeek  = bcdToDec(Wire.read());
    *dayOfMonth = bcdToDec(Wire.read());
    *month      = bcdToDec(Wire.read());
    *year       = bcdToDec(Wire.read());
  }
```

```
void displayTime()
{
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

  // прочитать данные из DS3232
  readDS3232time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth,
                 &month, &year);

  // передать данные в плату расширения LCD
  lcd.clear();
  lcd.setCursor(4,0);
  lcd.print(hour, DEC);
  lcd.print(":");
  if (minute<10)
  {
    lcd.print("0");
  }
  lcd.print(minute, DEC);
  lcd.print(":");
  if (second<10)
  {
    lcd.print("0");
  }
  lcd.print(second, DEC);
  lcd.setCursor(0,1);
  switch(dayOfWeek){
  case 1:
    lcd.print("Sun");
    break;
  case 2:
    lcd.print("Mon");
    break;
  case 3:
    lcd.print("Tue");
    break;
  case 4:
    lcd.print("Wed");
    break;
  case 5:
    lcd.print("Thu");
    break;
  case 6:
    lcd.print("Fri");
    break;
  case 7:
    lcd.print("Sat");
    break;
  }
  lcd.print(" ");
  lcd.print(dayOfMonth, DEC);
  lcd.print("/");
  lcd.print(month, DEC);
```

```
    lcd.print("/");
    lcd.print(year, DEC);
  }

  void loop()
  {
    displayTime(); // отображать время на экране ЖКИ
    delay(1000);   // каждую секунду
  }
```

Принцип действия и результаты

Этот скетч действует почти точно так же, как скетч в проекте 57, с той лишь разницей, что функция `displayTime` выводит время не в монитор порта, а на экран ЖКИ, а сам скетч включает строки, выполняющие настройку платы расширения LCD (❶ и ❷). (Чтобы освежить в памяти приемы работы с модулем ЖКИ, обращайтесь к главе 7.) Не забудьте сначала загрузить скетч, предварительно раскомментировав строку ❸, устанавливающую начальное время, а затем закомментировать ее и загрузить скетч повторно. После загрузки скетча вы увидите на экране ЖКИ результаты, напоминающие изображенные на рис. 18.4.

Рис. 18.4. Отображение даты и времени в проекте 58

Теперь, получив дополнительный опыт при создании проектов 57 и 58, вы должны иметь полное представление о том, как записывать данные в часы реального времени и считывать их оттуда. Теперь, используя новые знания, создадим что-нибудь по-настоящему полезное.

Проект № 59: Система хронометража с радиомаркерами

В этом проекте мы сконструируем устройство для хронометража (напоминающее табельные часы). Вы увидите, как можно использовать сразу несколько плат расширения и как плата расширения ProtoScrewShield помогает подключать дополнительные компоненты, не смонтированные на плате расширения. Это устройство может читать карты RFID и засекать время прихода или ухода владельца карты (например, прихода на рабочее место и ухода домой). Время и числовой идентификатор карты будут записываться на карту microSD для дальнейшего анализа.

Мы уже видели, как записывать данные на карту microSD, в главе 13, как работать с радиомаркерами RFID — в главе 16 и как пользоваться часами реального времени — в этой главе выше. Теперь мы соединим все это воедино.

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Плата Arduino и кабель USB.
- Несколько отрезков провода разной длины.
- Модуль RTC (показан выше в этой главе).
- Модуль LCD или плата расширения Freetronics LCD.
- Плата расширения с картой памяти MicroSD (из главы 13).
- Одна макетная плата расширения ProtoScrewShield или подобная ей.
- Модуль RFID и два радиомаркера (из главы 16).

Начнем сборку устройства: поместим плату Arduino Uno вниз и затем добавим к ней сверху платы расширения ProtoScrewShield, microSD и LCD. Подключите модуль RFID, как описывалось в главе 16, и модуль RTC — как описывалось выше в этой главе. Собранное устройство показано на рис. 18.5.

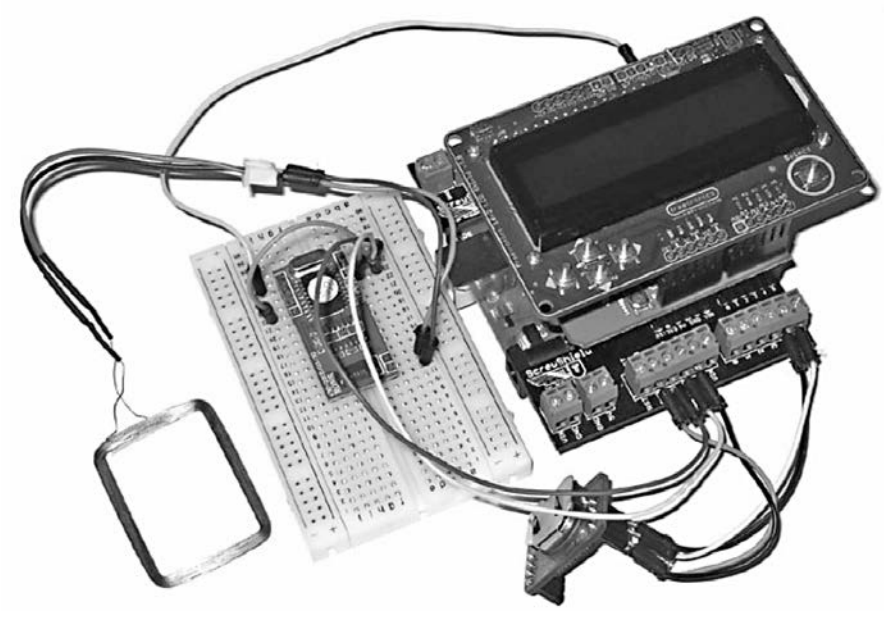


Рис. 18.5. Собранное устройство для хронометража

Скетч

Теперь введите и загрузите следующий скетч. Перед загрузкой скетча не забудьте отсоединить провод, связывающий контакт RX модуля RFID с контактом D0 на плате Arduino, и восстановить соединение после успешной загрузки скетча.

```
  // Проект 59 - Система хронометража с радиомаркерами
❶ #include "Wire.h" // для модуля RTC
  #define DS3232_I2C_ADDRESS 0x68
❷ #include "SD.h" // для платы расширения с картой microSD

  #include <LiquidCrystal.h>
  LiquidCrystal lcd( 8, 9, 4, 5, 6, 7 );
  int data1 = 0;

❸ // Для определения числовых идентификаторов радиомаркеров
  // используйте листинг 16.1
  int Mary[14] = {
    2, 52, 48, 48, 48, 56, 54, 67, 54, 54, 66, 54, 66, 3};
  int John[14] = {
    2, 52, 48, 48, 48, 56, 54, 66, 49, 52, 70, 51, 56, 3};
  int newtag[14] = {
    0,0,0,0,0,0,0,0,0,0,0,0,0,0}; // используется для чтения и сравнения

  // Преобразует обычное десятичное число в двоично-десятичное
  byte decToBcd(byte val)
  {
    return( (val/10*16) + (val%10) );
  }

  // Преобразует двоично-десятичное число в обычное десятичное
  byte bcdToDec(byte val)
  {
    return( (val/16*10) + (val%16) );
  }

  void setDS3232time(byte second, byte minute, byte hour,
                     byte dayOfWeek, byte dayOfMonth,
                     byte month, byte year)
  {
    // записать дату и время в DS3232
    Wire.beginTransmission(DS3232_I2C_ADDRESS);
    Wire.write(0); // выбрать для записи регистр секунд
    Wire.write(decToBcd(second)); // записать секунды
    Wire.write(decToBcd(minute)); // записать минуты
    Wire.write(decToBcd(hour));   // записать часы
    Wire.write(decToBcd(dayOfWeek));  // записать день недели (1=воскресенье,
7=суббота)
    Wire.write(decToBcd(dayOfMonth)); // записать число месяца (от 1 до 31)
    Wire.write(decToBcd(month));  // записать месяц
    Wire.write(decToBcd(year));   // записать год (от 0 до 99)
```

```
  Wire.endTransmission();
}

void readDS3232time(byte *second, byte *minute, byte *hour,
                    byte *dayOfWeek, byte *dayOfMonth,
                    byte *month, byte *year)
{
  Wire.beginTransmission(DS3232_I2C_ADDRESS);
  Wire.write(0); // инициализировать регистр указателя в DS3232
  Wire.endTransmission();
  Wire.requestFrom(DS3232_I2C_ADDRESS, 7);

  // прочитать 7 байт данных из DS3232, начиная с регистра 00h
  *second     = bcdToDec(Wire.read() & 0x7f);
  *minute     = bcdToDec(Wire.read());
  *hour       = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek  = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month      = bcdToDec(Wire.read());
  *year       = bcdToDec(Wire.read());
}

void setup()
{
  Serial.flush(); // очистить буфер последовательного порта
  Serial.begin(9600);
  Wire.begin();
  lcd.begin(16, 2);
  // установить начальное время:
  // секунды, минуты, часы, день недели, число, месяц, год
  //setDS3232time(0, 27, 0, 5, 15, 11, 12);

  // Проверить наличие и готовность карты microSD
❹  if (!SD.begin(8))
  {
    lcd.print("uSD card failure");
    // остановить скетч
    return;
  }
  lcd.print("uSD card OK");
  delay(1000);
  lcd.clear();
}

// Сравнивает два массива и возвращает true, если идентичны
// Удобно использовать для сравнения числовых идентификаторов
// радиомаркеров
boolean comparetag(int aa[14], int bb[14])
{
  boolean ff=false;
  int fg=0;
  for (int cc=0; cc<14; cc++)
```

```
    {
      if (aa[cc]==bb[cc])
      {
        fg++;
      }
    }
    if (fg==14)
    {
      ff=true; // все 14 элементов массивов совпадают
    }
    return ff;
  }

  void wipeNewTag()
  {
    for (int i=0; i<=14; i++)
    {
      newtag[i]=0;
    }
  }

  void loop()
  {
    byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
    if (Serial.available() > 0) // если была попытка чтения
    {
      // прочитать число из модуля RFID
      delay(100); // дать время поступить всем данным
                  // в буфер последовательного порта
      for (int z=0; z<14; z++) // прочитать числовой
                               // идентификатор радиомаркера
      {
        data1=Serial.read();
        newtag[z]=data1;
      }
      Serial.flush(); // аннулировать повторные попытки чтения
      // извлечь данные из модуля DS3232
      readDS3232time(&second, &minute, &hour, &dayOfWeek,
                     &dayOfMonth, &month, &year);
    }

    // выполнить необходимые операции по результатам
❺   if (comparetag(newtag, Mary) == true)
    {
      lcd.print("Hello Mary ");
      File dataFile = SD.open("DATA.TXT", FILE_WRITE);
      if (dataFile)
      {
        dataFile.print("Mary ");
        dataFile.print(hour);
        dataFile.print(":");
        if (minute<10) { dataFile.print("0"); }
```

```
      dataFile.print(minute);
      dataFile.print(":");
      if (second<10) { dataFile.print("0"); }
      dataFile.print(second);
      dataFile.print(" ");
      dataFile.print(dayOfMonth);
      dataFile.print("/");
      dataFile.print(month);
      dataFile.print("/");
      dataFile.print(year);
      dataFile.println();
      dataFile.close();
    }
    delay(1000);
    lcd.clear();
    wipeNewTag();
  }
  if (comparetag(newtag, John)==true)
  {
    lcd.print("Hello John ");
    File dataFile = SD.open("DATA.TXT", FILE_WRITE);
    if (dataFile)
    {
      dataFile.print("John ");
      dataFile.print(hour);
      dataFile.print(":");
      if (minute<10) { dataFile.print("0"); }
      dataFile.print(minute);
      dataFile.print(":");
      if (second<10) { dataFile.print("0"); }
      dataFile.print(second);
      dataFile.print(" ");
      dataFile.print(dayOfMonth);
      dataFile.print("/");
      dataFile.print(month);
      dataFile.print("/");
      dataFile.print(year);
      dataFile.println();
      dataFile.close();
    }
    delay(1000);
    lcd.clear();
    wipeNewTag();
  }
}
```

Принцип действия

В этом скетче система сначала ждет, пока карта RFID окажется в зоне действия антенны. Если карта RFID опознана, в конец текстового файла на карте microSD записывается имя владельца карты, время и дата.

В ❶ подключается библиотека с функциями для работы с шиной I²C и часами реального времени, а в строке ❷ подключается библиотека для работы с картой microSD. В ❹ проверяется и выводится информация о состоянии карты microSD. В ❺ только что прочитанный числовой идентификатор карты RFID сравнивается с известными идентификаторами двух людей — в данном случае Джона (John) и Мэри (Mary). В случае совпадения с одним из идентификаторов выполняется запись данных на карту microSD. Вы можете добавить в систему дополнительные карты, просто вставив числовые идентификаторы ниже уже имеющихся (❸) и добавив инструкцию сравнения по аналогии с инструкцией в строке ❺.

Когда придет время анализировать данные, скопируйте файл *data.txt* с карты microSD и откройте его в текстовом редакторе или импортируйте в электронную таблицу для более детального изучения. Данные отформатированы так, что легко читаются, это показано на рис. 18.6.

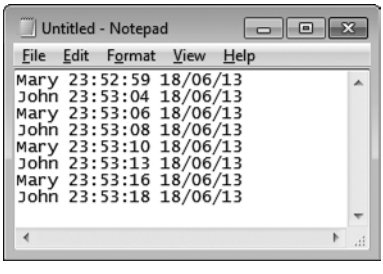


Рис. 18.6. Пример данных, накопленных устройством из проекта 59

Забегая вперед

В этой главе вы узнали, как работать с датой и временем посредством I²C-модуля RTC. Система хронометража с радиомаркерами RFID, описанная в проекте 59, может служить основой для создания ваших собственных систем доступа или даже следить за тем, когда ваши дети приходят домой. В двух последних главах мы создадим проекты на основе Arduino, в которых реализуем обмен данными через Интернет и сети сотовой связи.

19 Интернет

В этой главе вы:

- ✓ создадите веб-сервер для отображения данных на веб-странице;
- ✓ научитесь с помощью Arduino посылать сообщения в Twitter;
- ✓ узнаете, как организовать дистанционное управление цифровыми выходами на плате Arduino из веб-браузера.

Данная глава рассказывает, как связать плату Arduino с внешним миром через Интернет. Благодаря этой связи вы сможете организовать рассылку данных из платы Arduino и дистанционное управление ею из веб-браузера.

Что потребуется

Для создания проектов, так или иначе связанных с Интернетом, вам понадобятся: коммуникационное оборудование, кабель и информация.

Начнем с оборудования. Вам потребуется плата расширения Ethernet с микросхемой контроллера W5100. Предлагаю два варианта: оригинальная плата расширения Ethernet, выпускаемая под торговой маркой Arduino и изображенная на рис. 19.1, или плата, совместимая с Arduino Uno, включающая интегрированное оборудование Ethernet, такая как Freetronics EtherTen, изображенная на рис. 19.2. Последняя отлично подходит для новых проектов, а также в тех случаях, когда имеются ограничения по физическому объему для размещения оборудования или по финансовым соображениям. Как вы видите на рис. 19.2, плата EtherTen имеет разъемы для подключения дополнительных плат расширения, порт USB, разъем Ethernet и гнездо для подключения карты памяти microSD.

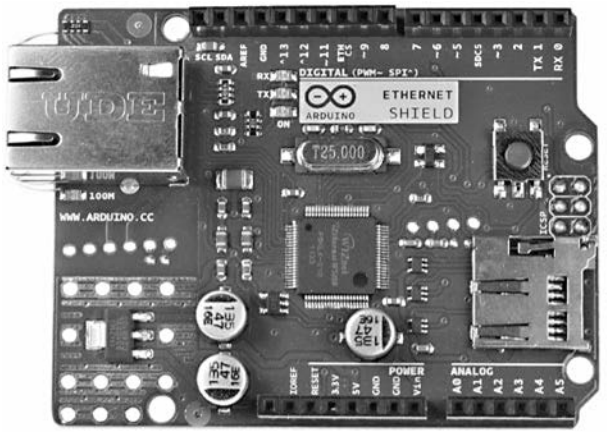


Рис. 19.1. Плата расширения Arduino Ethernet

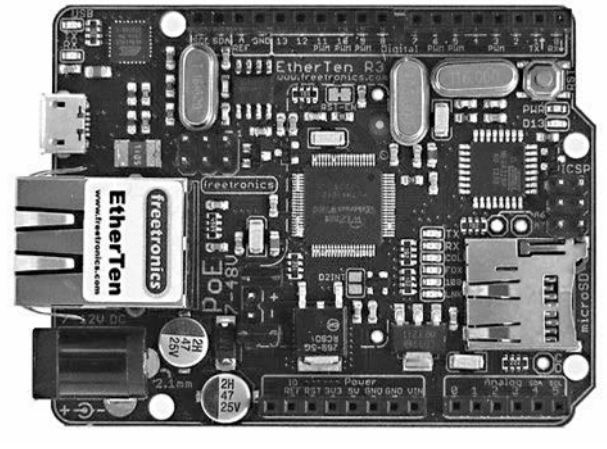


Рис. 19.2. Плата Freetronics EtherTen

Независимо от выбранного оборудования вам также потребуется стандартный сетевой кабель 10/100 CAT5, CAT5E или CAT6 для подключения платы расширения Ethernet к сетевому маршрутизатору или модему.

Кроме того, вам потребуется IP-адрес сетевого маршрутизатора или модема, который должен иметь примерно следующий вид: 192.168.0.1, а также IP-адрес вашего компьютера, имеющий тот же вид, что и IP-адрес маршрутизатора.

Наконец, если необходимо организовать обмен информацией с платой Arduino за пределами домашней или локальной сети, вам понадобится статический, общедоступный IP-адрес. *Статическим* называют фиксированный IP-адрес, присвоенный подключению к Интернету вашим поставщиком услуг Интернета (Internet Service

Provider, ISP). Подключение к Интернету может не иметь статического IP-адреса; свяжитесь с провайдером, чтобы получить такой адрес. Если ваш поставщик услуг не может предложить статический IP-адрес или стоимость его аренды слишком высока, вы сможете получить такой адрес в сторонней компании, такой как no-ip (*http://www.noip.com/*) или Dyn (*http://dyn.com/dns/*). Они настроят вам фиксированный веб-адрес, который будет направлять пользователей по вашему текущему IP-адресу.

А теперь проверим имеющееся оборудование на примере простого проекта.

Проект № 60: Станция дистанционного мониторинга

В проектах, представленных в предыдущих главах, мы собирали данные с датчиков, чтобы оценить температуру. В этом проекте вы узнаете, как отображать подобные значения на простой веб-странице, которую можно просмотреть на практически любом устройстве, поддерживающем выход в Интернет. Этот проект отображает уровни напряжений на аналоговых входах и состояния цифровых входов с 0-го по 9-й на простой веб-странице и служит основой для создания устройства дистанционного мониторинга.

В получившийся каркас можно добавлять датчики с аналоговыми и цифровыми выходами, такие как датчики температуры, освещенности и выключатели, а затем отображать состояния датчиков на веб-странице.

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Один кабель USB.
- Один сетевой кабель.
- Одна плата Arduino Uno с платой расширения Ethernet или одна плата Freetronic EtherTen.

Скетч

Введите следующий скетч, *но пока не загружайте его*:

```
/* Проект 60 – Станция дистанционного мониторинга
создан 18 декабря 2009 года Дэвидом Меллисом (David A. Mellis),
изменен 9 апреля 2012 года Томом Иго (Tom Igoe)
изменен 20 марта 2013 года Джоном Боксоллом (John Boxall)
*/

#include <SPI.h>
```

```
#include <Ethernet.h>

❶ IPAddress ip(xxx,xxx,xxx,xxx); // Замените IP-адресом своего проекта
❷ byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);
void setup()
{
  // Запустить сервер, обслуживающий соединение Ethernet
  Ethernet.begin(mac, ip);
  server.begin();
  for (int z=0; z<10; z++)
  {
    pinMode(z, INPUT); // настроить цифровые входы 0 - 9
                       // на работу в режиме входов
  }
}

void loop()
{
  // ждать запросов от клиентов (на получение веб-страницы)
  EthernetClient client = server.available();
  if (client) {
    // http-запрос должен заканчиваться пустой строкой
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        if (c == '\n' && currentLineIsBlank) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // добавить мета-тег Refresh, чтобы браузер
          // обновлял страницу каждые 5 секунд:
❸          client.println("<meta http-equiv=\"refresh\" content=\"5\">");
          // добавить на страницу значение каждого аналогового входа
          for (int analogChannel = 0; analogChannel < 6; analogChannel++)
          {
            int sensorReading = analogRead(analogChannel);
❹            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          // добавить на страницу значения цифровых входов 0 - 9
          for (int digitalChannel = 0; digitalChannel < 10; digitalChannel++)
          {
            boolean pinStatus = digitalRead(digitalChannel);
            client.print("digital pin ");
            client.print(digitalChannel);
```

```
            client.print(" is ");
            client.print(pinStatus);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // начало новой строки
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // в текущей строке имеются какие-то символы
          currentLineIsBlank = false;
        }
      }
    }
    // дать время браузеру получить данные
    delay(1);
    // закрыть соединение:
    client.stop();
  }
}
```

Мы подробно обсудим этот скетч чуть ниже. А пока, прежде чем загрузить скетч, введите IP-адрес для своей платы расширения Ethernet, чтобы потом ее можно было найти в локальной сети. Первые три компонента адреса можно взять из IP-адреса вашего маршрутизатора. Например, если маршрутизатор имеет адрес 192.168.0.1, подставьте вместо последнего числа другое случайное число в диапазоне от 1 до 254, чтобы получился IP-адрес, не используемый никаким другим устройством в вашей сети. Введите получившийся адрес в строке ❶ в скетче, например:

```
IPAddress ip(192, 168, 0, 69); // IP-адрес платы расширения Ethernet
```

После этого сохраните и загрузите скетч. Затем вставьте плату расширения Ethernet в разъемы на плате Arduino, соедините сетевым кабелем свой маршрутизатор или модем с платой расширения Ethernet и включите питание платы Arduino.

Подождите примерно 20 с и затем в веб-браузере на любом устройстве или компьютере в вашей сети введите IP-адрес, указанный в строке ❶. Если в браузере появится страница, аналогичная показанной на рис. 19.3, каркас станции дистанционного мониторинга работает правильно.

Поиск и устранение неисправностей

Если проект не работает, попробуйте выполнить следующие действия:

- Проверьте правильность IP-адреса в строке ❶.
- Проверьте правильность введенного скетча и его успешную загрузку.

Рис. 19.3. Значения на аналоговых входах и состояние цифровых входов отображаются в виде веб-страницы на любом устройстве с веб-браузером

- Дважды проверьте подключение к локальной сети. Например, проверьте, имеет ли выход в Интернет подключенный компьютер. Затем проверьте, подается ли питание на плату Arduino и ее подключение к маршрутизатору и модему.
- Если вы пытаетесь получить доступ к веб-странице проекта со своего смартфона, убедитесь, что смартфон имеет доступ к вашей локальной сети Wi-Fi, а не пытается обращаться к ней через сеть оператора сотовой связи.
- Если на плате расширения Ethernet не мигает ни один светодиод после включения питания на плате Arduino и подключения кабеля Ethernet к плате расширения и маршрутизатору и модему, попробуйте заменить кабель.

Принцип действия

Если станция мониторинга работает, можно вернуться к наиболее важным участкам в скетче. Код от начала скетча и до строки ❷ является обязательным — он импортирует необходимые библиотеки и активирует оборудование Ethernet, которое затем настраивается и запускается в функции `void setup`. С помощью инструкций `client.print`, предшествующих строке ❸, скетч подготавливает веб-страницу для передачи веб-браузеру. Начиная со строки ❸, функции `client.print` и `client.println` используются для вывода информации на веб-страницу, подобно тому как она выводится в монитор порта. Например, следующий код используется для вывода первых шести строк на веб-странице, изображенной на рис. 19.3:

```
client.print("analog input ");
client.print(analogChannel);
client.print(" is ");
client.print(sensorReading);
```

В строке ❹ можно видеть пример вывода текста и содержимого переменной на веб-странице. Здесь можно использовать разметку HTML для управления внешним видом веб-страницы, если при этом не произойдет переполнения доступной памяти в Arduino. Иными словами, вы можете использовать любой объем разметки HTML, пока размер скетча не превысит максимально допустимое значение, которое ограничивается объемом памяти на плате Arduino. (Объемы памяти каждого типа приводятся в табл. 11.2.)

Еще один важный элемент скетча, который стоит отметить, — это MAC-адрес, он используется для обнаружения отдельных устройств, подключенных к сети. Каждое устройство, подключаемое к сети, имеет уникальный MAC-адрес, который можно изменить подменой одного из шестнадцатеричных чисел в строке ❷. Если в одной сети одновременно должны находиться несколько проектов на основе Arduino, введите разные MAC-адреса для каждого устройства.

Наконец, если у вас появится желание просмотреть веб-страницу на устройстве, не подключенном к локальной сети, например, на планшетном компьютере или телефоне, использующем сети сотовой связи, вам потребуется настроить *переадресацию портов* (port forwarding) на своем маршрутизаторе или модеме с общедоступным IP-адресом, настроенным с помощью таких организаций, как no-ip (*http://www.no-ip.com/*) или Dyn (*http://dyn.com/dns/*). Настройка переадресации портов в разных моделях маршрутизаторов выполняется по-разному, поэтому ищите инструкции по настройке в Интернете или посетите сайт *http://www.wikihow.com/Port-Forward/*, где можно найти дополнительную информацию.

Теперь, когда вы знаете, как распространять текст и значения переменных в виде веб-страницы, давайте научимся посылать сообщения в Twitter с помощью Arduino.

Проект № 61: Arduino Tweeter

В этом проекте вы научитесь посылать сообщения в Twitter с помощью Arduino. Вы можете принимать любую информацию, сгенерированную скетчем, на любом устройстве, с которого можно выйти в Twitter.

Если, к примеру, вы захотите получать ежечасные замеры температуры у себя дома, находясь где-нибудь за границей, или даже извещения о том, когда дети приходят домой, в вашем распоряжении будет недорогое решение.

Вам потребуется создать для платы Arduino отдельную учетную запись в Twitter, поэтому выполните следующие шаги:

1. Посетите сайт *http://twitter.com/* и создайте учетную запись Twitter для платы Arduino. Запишите имя пользователя и пароль.

2. Получите «ключ» от стороннего веб-сайта *http://arduino-tweet.appspot.com/*, который создаст мост между вашей платой Arduino и службой Twitter. На этом сайте вам понадобится выполнить только один шаг.
3. Скопируйте и вставьте ключ (вместе с информацией о новой учетной записи Twitter для Arduino) в текстовый файл на вашем компьютере.
4. Загрузите и установите библиотеку Twitter Arduino, доступную по адресу *http://playground.arduino.cc/Code/TwitterLibrary/*.

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Один кабель USB.
- Один сетевой кабель.
- Одна плата Arduino Uno с платой расширения Ethernet или одна плата Freetronic EtherTen.

Скетч

Введите следующий скетч, *но пока не загружайте его*:

```
// Проект 61 - Arduino Tweeter
#include <SPI.h>
#include <Ethernet.h>
#include <Twitter.h>

❶ byte ip[] = { xxx,xxx,xxx,xxx };
❷ byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
❸ Twitter twitter("token");

❹ char msg[] = "I'm alive!"; // сообщение для вывода в Twitter

void setup()
{
  delay(30000);
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
}

void loop()
{
  Serial.println("connecting ...");
  if (twitter.post(msg)) {
    int status = twitter.wait();
❺    if (status == 200) {
      Serial.println("OK.");
```

```
    } else {
      Serial.print("failed : code ");
      Serial.println(status);
    }
  } else {
    Serial.println("connection failed.");
  }
  do {} while (1);
}
```

Так же как в проекте 60, вставьте свой IP-адрес в строке ❶ и измените MAC-адрес в строке ❷, если необходимо. Затем вставьте ключ доступа к Twitter в строке ❸, заключив его в двойные кавычки. Наконец, в строке ❹ вставьте текст, который требуется послать. Теперь загрузите скетч и подключите собранное устройство к сети. (Не забудьте подписаться на получение сообщений, созданных под учетной записью Arduino!) Примерно через минуту откройте свою страницу в Twitter — на ней должно появиться сообщение, показанное на рис. 19.4.

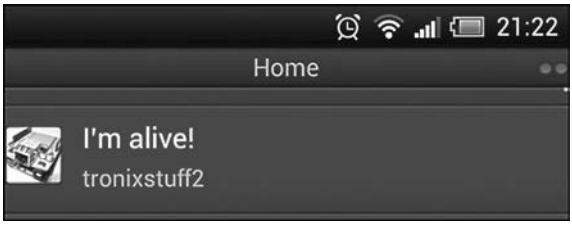


Рис. 19.4. Сообщение, отправленное платой Arduino

Создавая свое устройство, имейте в виду, что посылать сообщения можно не чаще одного раза в минуту и каждое сообщение должно быть уникальным. (Эти ограничения накладываются сайтом Twitter.) В ответ на попытку отправить сообщение Twitter возвращает код результата. Скетч принимает и выводит этот код в монитор порта в строке ❺, как, например, показано на рис. 19.5. Код «403» на рис. 19.5 означает, что был указан неверный ключ или вы пытаетесь посылать сообщения слишком часто. (Полный список кодов ошибок Twitter можно найти по адресу *http://dev.twitter.com/docs/error-codes-responses/*.)

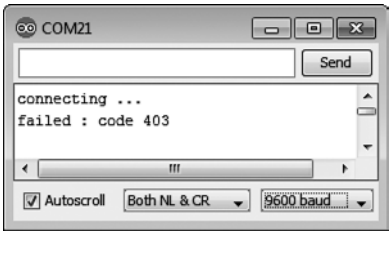


Рис. 19.5. Результат попытки посылать сообщения слишком часто

Управление платой Arduino через Интернет

Существует несколько способов управления платой Arduino с помощью веб-браузера. Проведя некоторые исследования, я обнаружил надежный, защищенный и бесплатный метод: Teleduino.

Teleduino — бесплатная служба, созданная Натаном Кеннеди (Nathan Kennedy) — энтузиастом Arduino из Новой Зеландии. Это простой, но мощный инструмент организации взаимодействий с платой Arduino через Интернет. Он не требует написания сложных или необычных скетчей; вместо этого, чтобы приступить к управлению платой Arduino, достаточно открыть в веб-браузере страницу со специальным адресом URL. С помощью службы Teleduino можно управлять цифровыми выходами и сервоприводами или отправлять I^2C-команды, и круг ее возможностей постоянно расширяется. В проекте 62 вы узнаете, как настроить службу Teleduino и дистанционно управлять цифровыми выходами с любого устройства, имеющего выход в Интернет.

Проект № 62: Настройка дистанционного управления платой Arduino

Прежде чем создать свой первый проект на основе Teleduino, необходимо зарегистрироваться в службе Teleduino и получить уникальный ключ для идентификации вашей платы Arduino. Для этого посетите страницу *https://www.teleduino.org/tools/request-key/* и введите требуемую информацию. В ответ вы получите электронное письмо с уникальным ключом, который будет выглядеть примерно так: `18765432IZ9AEFF952ABCDEF8534B2BBF`.

После этого преобразуйте ключ в массив, посетив страницу *https://www.teleduino.org/tools/arduino-sketch-key/*. Введите свой ключ, и на странице отобразится ключ в виде массива, как показано на рис. 19.6.

```
byte key[] = { 0x65, 0x9A, 0xCE, 0xB1,
               0xA7, 0x5E, 0x57, 0x2B,
               0x8F, 0xFE, 0xA4, 0x40,
               0x9E, 0x7B, 0x7A, 0xBC };
```

Рис. 19.6. Ключ Teleduino в виде массива

Каждой плате Arduino соответствует единственный уникальный ключ, но вы можете получить несколько ключей, чтобы работать с несколькими проектами на основе Teleduino.

Оборудование

Ниже перечислено оборудование, необходимое для данного проекта:

- Один кабель USB.
- Один сетевой кабель.
- Одна плата Arduino Uno с платой расширения Ethernet или одна плата Freetronic EtherTen.
- Один резистор с номиналом 560 Ом (R1).
- Одна макетная плата.
- Один светодиод любого цвета.

Соберите устройство, руководствуясь принципиальной схемой с рис. 19.7, и подключите светодиод к цифровому контакту 8.

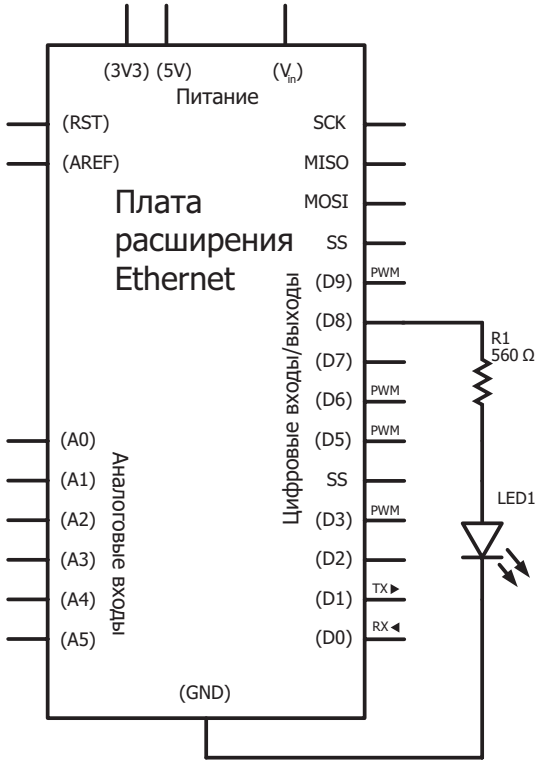


Рис. 19.7. Принципиальная схема проекта 62

Скетч

В проектах на основе Teleduino используется единственный скетч, включенный в состав библиотеки Teleduino. Далее описывается, как получить этот скетч:

1. Загрузите и установите библиотеку Teleduino, доступную по адресу *https://www.teleduino.org/downloads/*.
2. Перезапустите среду разработки Arduino IDE и выберите в меню пункт File ▸ Examples ▸ Teleduino328 ▸ TeleduinoEthernetClientProxy (Файл ▸ Примеры ▸ Teleduino328 ▸ TeleduinoEthernetClientProxy).
3. Теперь вы видите скетч Teleduino. Прежде чем загрузить его в плату Arduino, замените ключ по умолчанию массивом, полученным из своего ключа. Переменная, которую требуется заменить, находится в скетче, в строке 36. Выполнив замену, сохраните скетч и загрузите его в плату Arduino.

Подключите собранное устройство к сети и понаблюдайте за светодиодом. Спустя примерно минуту он должен мигнуть несколько раз и погаснуть. Количество вспышек соответствует состоянию службы Teleduino, это показано в табл. 19.1.

Таблица 19.1. Коды состояния службы Teleduino

Количество вспышек	Значение
1	Инициализация
2	Открывается сетевое соединение
3	Установлено соединение с сервером Teleduino
4	Аутентификация выполнена успешно
5	Сеанс уже существует
6	Неверный или неавторизованный ключ
10	Соединение закрыто

Если вы увидите пять вспышек, значит, ваш ключ уже запрограммирован в другой плате Arduino, которая в данный момент подключена к серверу Teleduino. Увидев 10 вспышек, проверьте свое оборудование и подключение к Интернету. После того как Arduino установит соединение с сервером, плата должна мигать светодиодом один раз в каждые 5 секунд. Так как состоянием светодиода управляет цифровой контакт 8, его нельзя использовать ни для каких других целей, пока плата находится под управлением службы Teleduino.

Дистанционное управление платой Arduino

Для дистанционного управления платой Arduino с помощью службы Teleduino можно использовать любое устройство с веб-браузером. Команда управления посылается вводом созданного вами URL-адреса *http://us01.proxy.teleduino.org/api/1.0/328.php?k=<YOURKEY>&r=setDigitalOutput&pin=<X>&output=<S>*.

Измените три параметра в URL-адресе: во-первых, замените параметр *<YOURKEY>* алфавитно-цифровым ключом, полученным на сайте Teleduino. Во-вторых, замените *<X>* номером цифрового контакта. В-третьих, замените *<S>* значением 0, соответствующим уровню `LOW`, или 1, соответствующим уровню `HIGH`, чтобы изменить состояние цифрового выхода. Например, чтобы перевести контакт 7 в состояние `HIGH`, нужно ввести адрес *http://us01.proxy.teleduino.org/api/1.0/328.php?k=<YOURKEY>&r=setDigitalOutput&pin=7&output=1*.

В случае успешного выполнения команды вы должны увидеть в окне веб-браузера примерно такой текст:

```
{"status":200,"message":"OK","response"
{"result":0,"time":0.22814512252808,"values":[]}}
```

Если команда не будет успешно выполнена, в браузере появится примерно такой текст:

```
{"status":403,"message":"Key is offline or invalid.","response":[]}
```

Вы можете посылать команды для изменения состояния цифровых выходов, просто изменяя URL-адрес. После создания URL-адресов для своего проекта сохраните их в закладках своего браузера или создайте локальную веб-страницу с необходимыми ссылками, оформленными в виде кнопок. Например, можно создать одну закладку с URL-адресом, устанавливающим уровень `HIGH` на цифровом выходе 7, и еще одну закладку — устанавливающую уровень `LOW` на том же выходе.

Иногда состояние выходов на плате Arduino имеет большое значение. Поэтому для безопасности определите состояние по умолчанию цифровых контактов — на случай сброса платы Arduino из-за отключения питания или по другим причинам. Подключив устройство к службе Teleduino, откройте страницу *https://www.teleduino.org/tools/manage-presets/*. После ввода уникального ключа вы увидите массив параметров, в котором можно выбрать режим работы и состояние для каждого цифрового контакта, как показано на рис. 19.8.

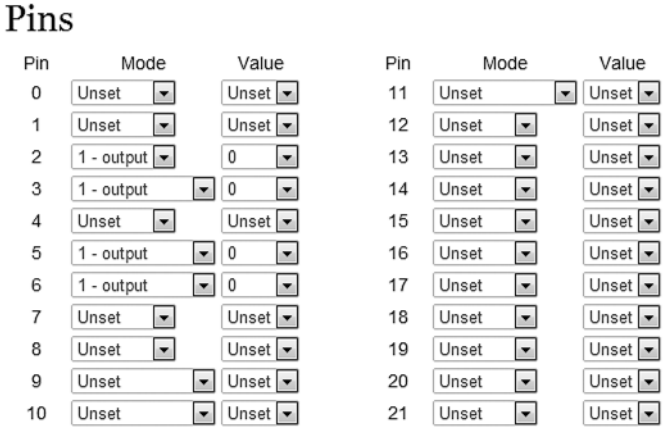


Рис. 19.8. Страница настройки состояния по умолчанию цифровых контактов

Забегая вперед

Устройства на основе Arduino не только легко поддаются мониторингу через Интернет и способны посылать сообщения в Twitter, но ими также можно управлять через Интернет, и для этого не требуется создавать сложные скетчи, обладать навыками программирования сетевых взаимодействий или нести какие-то финансовые расходы. Используя средства дистанционного управления через Интернет, вы сможете контролировать свои устройства на основе Arduino практически из любой точки земного шара и расширить их возможности вывода данных. Три проекта, представленные в этой главе, образуют каркас, опираясь на который вы сможете создавать свои собственные устройства с дистанционным управлением.

Следующая глава, последняя в этой книге, покажет, как реализовать в Arduino отправку и прием команд через сети сотовой связи.

20 Сети сотовой связи

В этой главе вы узнаете, как:

- ✓ с помощью Arduino набрать телефонный номер в случае необходимости;
- ✓ с помощью Arduino послать текстовое сообщение на сотовый телефон;
- ✓ управлять устройствами, подключенными к Arduino, посредством коротких текстовых сообщений.

Проекты на основе Arduino можно подключать к сетям сотовой связи, чтобы обеспечить простейшие взаимодействия между платой Arduino и сотовым или стационарным телефоном. Добавив толику воображения, вы сможете придумать массу применений подобным решениям, в том числе и приводящимся в этой главе.

Обязательно дочитайте эту главу до конца перед приобретением любого оборудования, потому что успех проектов, описываемых здесь, во многом зависит от особенностей вашей сети сотовой связи. Сеть должна:

- ✓ Поддерживать стандарт: GSM 850 МГц, GSM 900 МГц, DCS 1800 МГц или PCS 1900 МГц.
- ✓ Позволять использовать устройства, не поставляемые оператором связи.

Сети сотовой связи, эксплуатируемые в Европейском Союзе, Австралии и Новой Зеландии, обычно отвечают этим требованиям. Если вы находитесь в Соединенных Штатах или в Канаде, свяжитесь со своим оператором сотовой связи, чтобы узнать, соответствует ли ваша сеть этим требованиям, и только потом приобретайте оборудование.

Для описываемых здесь проектов предпочтительнее использовать тарифные планы с предоплатой или планы, допускающие большое количество текстовых сообщений на тот случай, если из-за ошибки в скетче ваш проект пошлет несколько коротких текстовых сообщений SMS (Short Message Service). Кроме того, перед использованием SIM-карты (Subscriber Identification Module — модуль идентификации абонента) отключите требование ввода PIN-кода (Personal Identification Number — личный опознавательный номер). (Для этого вставьте SIM-карту в обычный сотовый телефон и в настройках безопасности отключите параметр, требующий ввода PIN-кода.)

Оборудование

Из-за ограничений, накладываемых оборудованием, проекты в этой главе будут функционировать только с платой Arduino Uno или совместимой с ней (то есть платы серии Arduino Mega не годятся). Во всех проектах используется типовой комплект оборудования, поэтому сначала мы соберем и настроим его.

Для проектов этой главы вам понадобится специализированное оборудование, включая плату расширения SM5100 GSM с антенной, изображенную на рис. 20.1. Эту плату можно приобрести в компании SparkFun и у ее представителей. (Ищите плату с артикулом CEL-09607, комплект наращиваемых контактов (stackable header set) PRT-10007 и антенну с артикулом CEL-00675.)

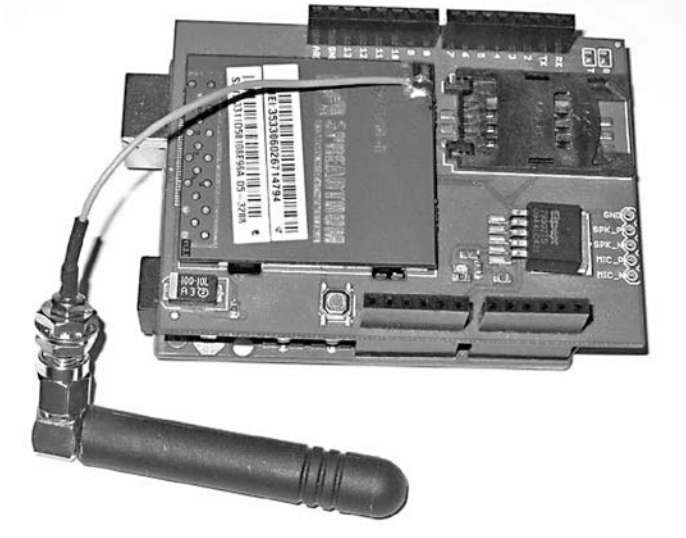


Рис. 20.1. Плата расширения GSM с подключенной антенной

Вам также понадобится плата стабилизатора и внешний источник питания. Плата расширения GSM потребляет ток до 2 А (что превышает возможности Arduino) и может вывести из строя плату Arduino, если не использовать внешний источник питания. Платы стабилизаторов, совместимые с Arduino, выпускает компания DFRobot (*http://www.dfrobot.com/*). Для наших проектов подойдет, например, плата с артикулом DFR0105, изображенная на рис. 20.2.

Наконец, необходим внешний источник питания. Для ваших целей подойдет любой источник постоянного тока (например, зарядное устройство для аккумуляторов с выходным напряжением 7,2 В, солнечные панели, батареи, аккумуляторы с напряжением 12 В или нечто подобное, с единственным условием: выходное напряжение источника постоянного тока не должно превышать 35 В), способный выдавать ток силой до 2 А.

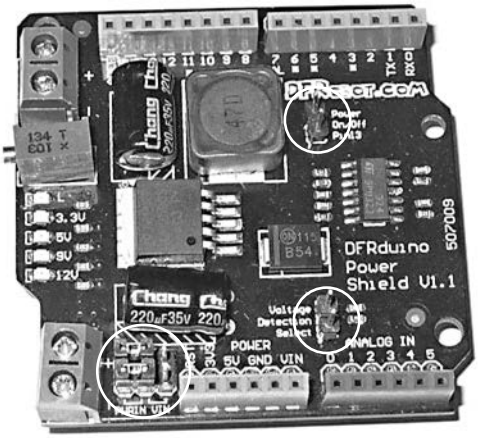


Рис. 20.2. Плата питания от компании DFRobot

Подготовка платы стабилизатора

Подготовьте плату стабилизатора и (что особенно важно) *настройте его выходное напряжение перед подключением к ней любых других компонентов.* Для этого сначала извлеките все перемычки, имеющиеся на колодках контактов справа вверху и справа внизу (обведены окружностями на рис. 20.2 справа). Затем проверьте две перемычки слева внизу — они должны быть установлены горизонтально и замыкать пары контактов в группе PWRIN (обведены окружностью на рис. 20.2 внизу слева).

Теперь подключите внешний источник питания к зажимам PWRIN, расположенным слева внизу на плате стабилизатора. Особое внимание обратите на полярность: положительный вывод источника питания должен подключаться к контакту с меткой +, а отрицательный — к контакту с меткой −. Затем включите источник питания, измерьте выходное напряжение на зажимах PWROUT слева вверху на плате стабилизатора и отрегулируйте его, вращая маленькой отверткой синий потенциометр, изображенный на рис. 20.3.

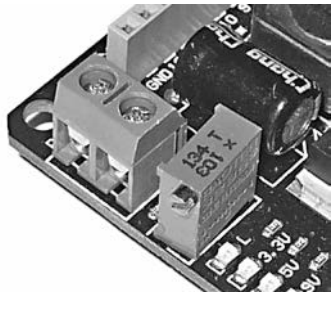


Рис. 20.3. Потенциометр регулировки выходного напряжения на плате стабилизатора

Во время регулировки контролируйте выходное напряжение с помощью мультиметра, оно должно составлять 5 В. Закончив регулировку, выключите источник питания.

Теперь соберем устройство из имеющихся компонентов:

1. Вставьте SIM-карту в плату расширения GSM.
2. Вставьте плату GSM в разъемы на плате Arduino Uno.
3. Вставьте сверху плату стабилизатора напряжения.
4. На плате стабилизатора соедините отрезком провода положительный разъем PWROUT с контактом питания 5V, аналогично соедините отрицательный разъем PWROUT с контактом GND. Эти два провода обеспечат необходимый ток с напряжением 5 В для питания платы Arduino.

ВНИМАНИЕ

Всегда включайте напряжение питания проекта перед подключением кабеля USB, соединяющего плату Arduino с компьютером. И всегда отключайте кабель USB перед выключением внешнего источника питания.

Настройка и проверка оборудования

Теперь настроим и проверим оборудование. В частности, проверим возможность взаимодействий модуля GSM с сетью сотовой связи и с платой Arduino. После сборки устройства и установки SIM-карты введите и загрузите скетч:

Листинг 20.1. Скетч для проверки платы расширения GSM

```
  // Листинг 20.1
  // Написан Райаном Оунсом (Ryan Owens) - SparkFun CC, v3.0 08.03.10
❶ #include <SoftwareSerial.h> // Виртуальный последовательный порт
❷ SoftwareSerial cell(2,3);
  char incoming_char = 0;

  void setup()
  {
    // Инициализировать последовательные порты.
    Serial.begin(9600);
❸   cell.begin(9600);
    Serial.println("Starting SM5100B Communication...");
  }

  void loop()
  {
    // Если получен символ от модуля GSM...
    if( cell.available() > 0 )
    {
      // Прочитать символ из порта, связанного с модулем GSM.
      incoming_char = cell.read();
```

```
    // Вывести полученный символ в монитор порта.
    Serial.print(incoming_char);
  }
  // Если получен символ из монитора порта...
  if( Serial.available() > 0 )
  {
    incoming_char = Serial.read();// Прочитать символ из монитора порта
    cell.print(incoming_char);    // Послать символ в модуль GSM.
  }
}
```

Этот скетч действует как простое передаточное звено, пересылая всю информацию, поступающую от платы расширения GSM, в монитор порта. Модуль GSM создает последовательное соединение с платой Arduino, используя для этого цифровые контакты 2 и 3, которое не мешает обмену данными по стандартному последовательному порту между Arduino и персональным компьютером, подключенным к цифровым контактам 0 и 1. Для связи с платой GSM мы настроили виртуальный последовательный порт `SoftwareSerial` (❶, ❷ и ❸). (Необходимая библиотека уже входит в состав Arduino IDE.)

Загрузив скетч, откройте окно монитора порта и подождите примерно 30 секунд. В окне должны появиться данные, как показано на рис. 20.4.

Наибольший интерес представляют сообщения, начинающиеся с `+SIND`: они содержат информацию о состоянии модуля GSM и его подключении к сети. Если поток данных заканчивается сообщением `+SIND: 4`, это означает, что плата расширения нашла сеть сотовой связи, подключилась к ней и вы можете переходить к проекту 63. Но если поток данных заканчивается сообщением `+SIND: 8`, измените частоту, используемую модулем, как предлагается в следующем разделе.

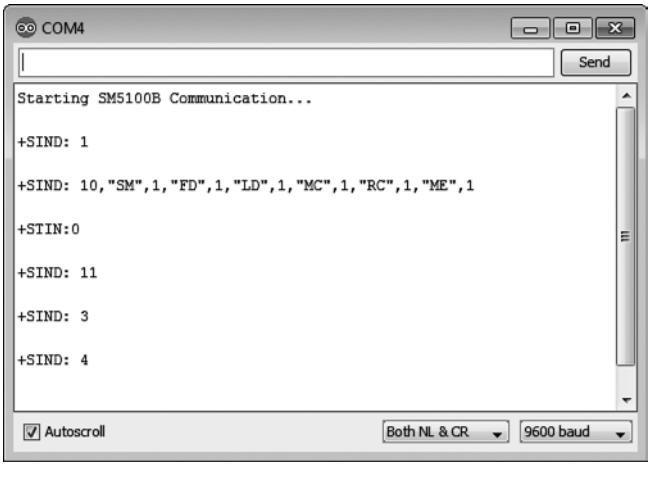


Рис. 20.4. Пример вывода результатов работы скетча из листинга 20.1

Изменение рабочей частоты

Чтобы изменить рабочую частоту модуля GSM, выполните следующие шаги:

1. Закройте среду разработки Arduino IDE и окно монитора порта. Откройте программу терминала, использовавшуюся в проекте 48.
2. Выберите COM-порт, который используется для связи с платой Arduino, и щелкните на кнопке Connect (Соединиться).
3. Найдите в табл. 20.1 рабочую частоту, которую должен использовать ваш GSM-модуль, и запишите номер полосы.

Таблица 20.1. Полосы рабочих частот, используемые модулями GSM

Частота	Номер полосы
GSM 900 МГц	0
DCS 1800 МГц	1
PCS 1900 МГц	2
GSM 850 МГц	3
GSM 900 МГц и DCS 1800 МГц	4
GSM 850 МГц и GSM 900 МГц	5
GSM 850 МГц и DCS 1800 МГц	6
GSM 850 МГц и PCS 1900 МГц	7
GSM 900 МГц и PCS 1900 МГц	8
GSM 850 МГц, GSM 900 МГц и DCS 1800 МГц	9
GSM 850 МГц, GSM 900 МГц и PCS 1900 МГц	10

4. Введите команду AT+SBAND? и нажмите клавишу ENTER. Модуль должен вернуть текущие настройки полосы, как показано на рис. 20.5.

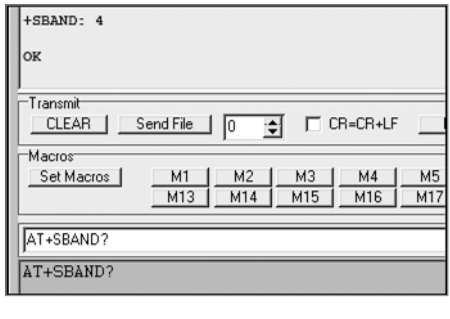


Рис. 20.5. Пример определения текущей полосы рабочих частот

5. Чтобы настроить требуемую полосу рабочих частот в модуле GSM, введите команду `AT+SBAND=x`, где `x` — номер полосы, найденный по табл. 20.1. Плата должна вернуть сообщение `OK`, как показано на рис. 20.6.

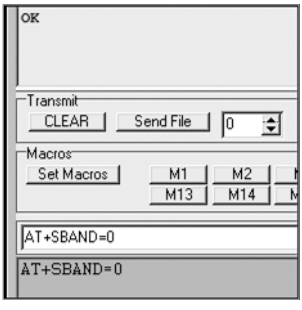


Рис. 20.6. Успешное изменение полосы рабочих частот

6. Наконец, выполните сброс платы Arduino и проверьте, произошло ли подключение к сети, о чем должно свидетельствовать сообщение `+SIND: 4`. Соедините на короткое время контакты GND и RST на плате Arduino отрезком провода, чтобы выполнить сброс, после чего в окне терминала порта должны появиться сообщения, как показано на рис. 20.7.

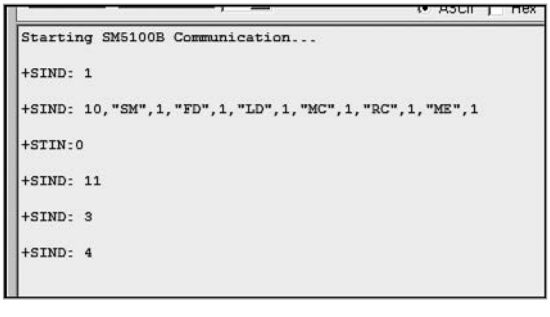


Рис. 20.7. Успешное подключение к сети

ПРИМЕЧАНИЕ

Желательно изменить значение SBAND только один раз, потому что этот параметр хранится в ЭСППЗУ на плате расширения GSM. Но если вы переехали в другую местность, где используется другая сеть сотовой связи, вы можете вновь изменить это значение.

В заключение необходимо выполнить еще одну проверку: позвоните с другого телефона по номеру, соответствующему SIM-карте в модуле GSM. Если вы не подключили службу, препятствующую определению номера вашего телефона, этот номер должен появиться в выводе модуля GSM, как показано на рис. 20.8.

(Обратите внимание, что вместо номера 0452280886, изображенного на рисунке, появится ваш номер.)

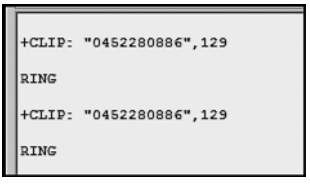


Рис. 20.8. Результат звонка на SIM-карту в модуле GSM

Теперь все работает и можно приступать к разработке новых проектов!

Проект № 63: Автоматический номеронабиратель

К концу этого проекта ваша плата Arduino научится набирать телефонный номер по событиям, определяемым скетчем. Например, если температура в холодильной камере поднялась выше определенного порога или сработала охранная сигнализация, Arduino сможет позвонить по заранее заданному номеру и через 20 секунд повесить трубку. Для идентификации таких звонков занесите номер Arduino в список контактов под неким, понятным вам, именем.

Оборудование

В этом проекте используется оборудование, описанное в начале главы, а также несколько дополнительных компонентов по вашему выбору и в зависимости от прикладного назначения. Для демонстрации мы будем использовать кнопку, нажатие на которую будет вызывать звонок.

В дополнение к оборудованию, описанному выше, для создания этого проекта нам понадобятся:

- Одна кнопка без фиксации.
- Один резистор с номиналом 10 кОм.
- Один конденсатор емкостью 100 нФ.
- Несколько отрезков провода разной длины.
- Одна макетная плата.

Схема

Подключите дополнительные компоненты, как показано на рис. 20.9.

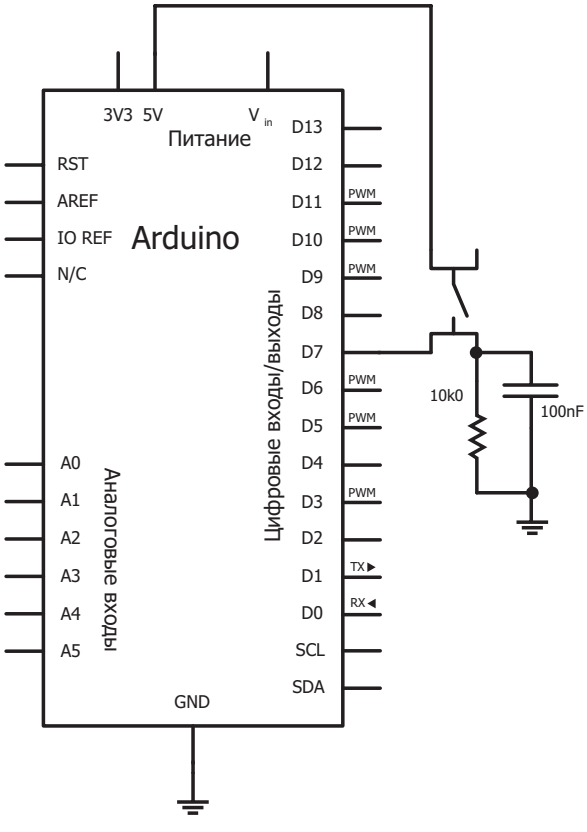


Рис. 20.9. Принципиальная схема проекта 63

Скетч

Введите, *но пока не загружайте* следующий скетч:

```
// Проект 63 - Автоматический номеронабиратель

#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);

void setup()
{
  pinMode(7, INPUT);
  cell.begin(9600);
❶  delay(30000); // дать модулю GSM время для подключения к сети
}

❷ void callSomeone()
{
```

```
❸   cell.println("ATDxxxxxxxxxx"); // набрать номер xxxxxxxxxx
    // замените xxxxxxxxxx номером своего телефона (с кодом города)
    delay(20000);        // ждать 20 секунд.
    cell.println("ATH"); // повесить трубку
    delay(60000);        // ждать 60 секунд, как того требует модуль GSM
  }

  void loop()
  {
    if (digitalRead(7) == HIGH)
    {
❹     callSomeone();
    }
  }
```

Принцип действия

Функция `void setup` инициализирует модуль GSM и дает ему некоторое время (❶) для подключения к сети сотовой связи. В ожидании «события» плата Arduino постоянно проверяет состояние кнопки, подключенной к цифровому контакту 7. В момент нажатия кнопки вызывается функция `callSomeone` (❷), которая посылает модулю GSM команду набора номера (❸).

Замените строку `xxxxxxxxxx` номером телефона, на который будет звонить плата Arduino. Номер должен выглядеть так же, как он выглядит на обычном мобильном телефоне. Например, если вы хотите, чтобы Arduino звонила на номер 212-555-12-12, измените строку ❸ в скетче, как показано ниже:

```
cell.println("ATD2125551212");
```

После ввода номера телефона загрузите скетч, подождите минуту и нажмите кнопку. Функцию автоматического набора номера можно интегрировать в любой имеющийся скетч, потому что ее легко вызвать в нужный момент (❹). Теперь вам остается только найти причины, по которым плата Arduino должна была бы позвонить на ваш номер.

А сейчас перенесем вашу Arduino в XXI век, научив ее посылать текстовые сообщения.

Проект № 64: Отправка текстовых сообщений

В этом проекте плата Arduino будет по событию посылать текстовое сообщение на другой сотовый телефон. Чтобы упростить скетч, мы будем использовать библиотеку *SerialGSM*, доступную по адресу *https://github.com/meirm/SerialGSM/*. После установки библиотеки перезапустите Arduino IDE.

Для этого проекта потребуется то же самое оборудование, что использовалось в проекте 63.

Скетч

Введите следующий скетч в Arduino IDE, но *пока не загружайте его*:

```
// Проект 64 - Отправка текстовых сообщений

#include <SerialGSM.h>
#include <SoftwareSerial.h>
❶ SerialGSM cell(2,3);

void setup()
{
  pinMode(7, INPUT);
  delay(30000); // дать модулю GSM время для подключения к сети
  cell.begin(9600);
}

void textSomeone()
{
  cell.Verbose(true); // для отладки
  cell.Boot();
  cell.FwdSMS2Serial();
❷  cell.Rcpt("+xxxxxxxxxx"); // замените xxxxxxxxxx номером
                             // телефона получателя
❸  cell.Message("This is the contents of a text message");
  cell.SendSMS();
}

void loop()
{
❹  if (digitalRead(7) == HIGH)
  {
    textSomeone();
  }
  if (cell.ReceiveSMS())
  {
    Serial.println(cell.Message());
    cell.DeleteAllSMS();
  }
}
```

Принцип действия

Модуль GSM настраивается как обычно (❶), и функция `void setup()` осталась прежней. В строке ❹ определяется нажатие кнопки и вызывается функция `textSomeone`. Эта простая функция посылает текстовое сообщение на сотовый телефон, номер которого указан в строке ❷.

Прежде чем загрузить скетч, замените строку `xxxxxxxxxx` номером сотового телефона получателя в международном формате: код страны, код области и номер без пробелов и скобок. Например, чтобы отправить текстовое сообщение на телефон с номером 212-555-12-12 в США, введите номер `+12125551212`.

Функция посылает текстовое сообщение, хранящееся в строке ❸. (Обратите внимание, что длина сообщения ограничена 160 символами[1].)

Определив свое текстовое сообщение и номер телефона получателя, загрузите скетч, подождите 30 с и затем нажмите кнопку. Через несколько мгновений сообщение должно быть принято на телефоне получателя, как показано на рис. 20.10.

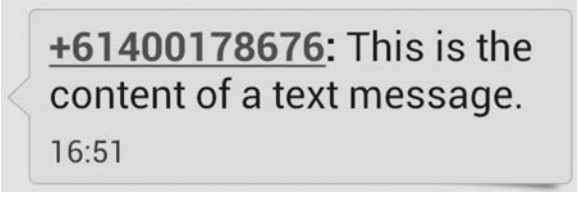


Рис. 20.10. Пример полученного текстового сообщения

Код из проекта 64 легко можно встроить в любые другие скетчи и с его помощью посылать самые разные текстовые сообщения, реализовав выбор с помощью инструкции `switch-case`.

ПРИМЕЧАНИЕ

Не забывайте, что отправка текстовых сообщений производится не бесплатно, поэтому для экспериментов лучше использовать SIM-карту с тарифным планом без ограничений или с большим количеством предоплаченных текстовых сообщений.

Проект № 65: Дистанционное управление устройствами посредством коротких текстовых сообщений

В этом проекте мы будем управлять цифровыми выходами на плате Arduino, посылая текстовые сообщения с сотового телефона. У вас уже достаточно знаний, чтобы организовать управление различными устройствами. В данном проекте будет осуществляться управление четырьмя цифровыми выходами, но вы можете задействовать любое их количество.

Чтобы включить или выключить тот или иной из четырех цифровых выходов (в данном примере — контакты с 10-го по 13-й), нужно послать на номер платы Arduino текстовое сообщение следующего формата: `#axbxcxdx`, заменив `x` цифрой `0` или `1`. Например, чтобы установить высокий уровень на всех четырех выходах, отправьте сообщение `#a1b1c1d1`.

[1] Длина сообщения, написанного кириллицей, не может превышать 70 символов. — *Примеч. пер.*

Оборудование

В этом проекте используется оборудование, описанное в начале главы, плюс дополнительные компоненты по вашему выбору. Для индикации состояния управляемых цифровых выходов будут использоваться четыре светодиода. Соответственно для реализации проекта потребуется следующее дополнительное оборудование:

- Четыре светодиода.
- Четыре резистора с номиналом 560 Ом.
- Несколько отрезков провода разной длины.
- Одна макетная плата.

Схема

Подключите дополнительные компоненты, как показано на рис. 20.11.

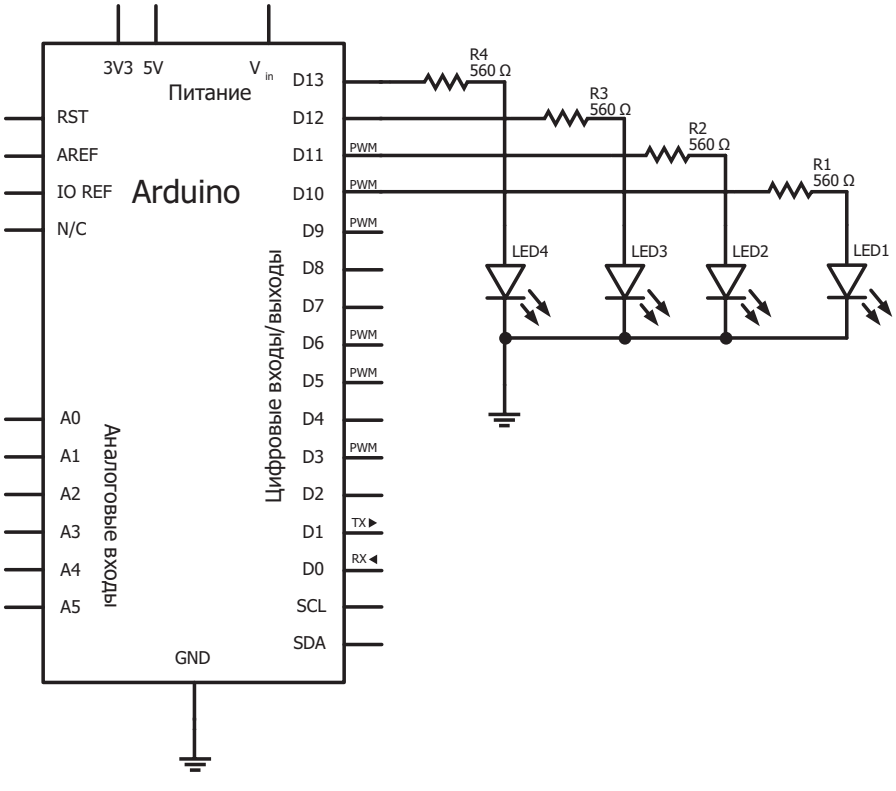


Рис. 20.11. Принципиальная схема проекта 65

Скетч

Введите и загрузите следующий скетч:

```
// Проект 65 - Дистанционное управление устройствами посредством
//             коротких текстовых сообщений

#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);
char inchar;

void setup()
{
  // настроить управляемые цифровые выходы
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

  digitalWrite(10, LOW); // Состояние выходов по умолчанию
  digitalWrite(11, LOW); // после включения питания или сброса
  digitalWrite(12, LOW); // измените в соответствии со своими
  digitalWrite(13, LOW); // потребностями

  // Инициализировать последовательный порт связи с модулем GSM.
  cell.begin(9600);
  delay(30000);
❶  cell.println("AT+CMGF=1");
  delay(200);
❷  cell.println("AT+CNMI=3,3,0,0");
  delay(200);
}

void loop()
{
  // Если получен символ от модуля GSM...
❸  if(cell.available() > 0)
  {
    inchar = cell.read();
❹    if (inchar == '#') // начало команды
    {
      delay(10);
      inchar = cell.read();
❺      if (inchar == 'a')
      {
        delay(10);
        inchar = cell.read();
        if (inchar == '0')
        {
          digitalWrite(10, LOW);
        }
        else if (inchar == '1')
```

```
        {
          digitalWrite(10, HIGH);
        }
        delay(10);
        inchar = cell.read();
        if (inchar == 'b')
        {
          inchar = cell.read();
          if (inchar == '0')
          {
            digitalWrite(11, LOW);
          }
          else if (inchar == '1')
          {
            digitalWrite(11, HIGH);
          }
          delay(10);
          inchar = cell.read();
          if (inchar == 'c')
          {
            inchar = cell.read();
            if (inchar == '0')
            {
              digitalWrite(12, LOW);
            }
            else if (inchar == '1')
            {
              digitalWrite(12, HIGH);
            }
            delay(10);
            inchar = cell.read();
            if (inchar == 'd')
            {
              delay(10);
              inchar = cell.read();
              if (inchar == '0')
              {
                digitalWrite(13, LOW);
              }
              else if (inchar == '1')
              {
                digitalWrite(13, HIGH);
              }
              delay(10);
            }
          }
          cell.println("AT+CMGD=1,4"); // удалить все SMS
        }
      }
    }
  }
}
```

Принцип действия

В этом проекте плата Arduino проверяет каждый символ в текстовом сообщении, полученном от модуля GSM. В строке ❶ плате GSM передается команда преобразовать входящее сообщение SMS в текст и послать его содержимое в виртуальный последовательный порт (❷). Затем Arduino просто ждет получения текстового сообщения от модуля GSM (❸).

Так как команды управления цифровыми выходами, которые посылаются с сотового телефона и передаются модулем GSM в Arduino, начинаются с символа `#`, скетч ждет появления этого символа в текстовом сообщении (❹). В строке ❺ проверяется первый параметр: если за ним следует `0` или `1`, на выходе устанавливается низкий или высокий уровень соответственно. Процесс повторяется для следующих трех выходов, управляемых символами `b`, `c` и `d`.

Представьте, насколько просто с помощью этого проекта можно реализовать дистанционное управление, например, индикаторами, электродвигателями, звонками и многими другими устройствами.

Забегая вперед

Реализовав три проекта из этой главы, вы заложили прочный фундамент, на котором можно строить собственные проекты, поддерживающие взаимодействия с применением сетей сотовой связи. Ваши возможности ограничены только вашим воображением — например, можно организовать посылку текстового сообщения в случае затопления подвала или включать кондиционер со своего сотового телефона. Напомню еще раз: не забудьте положить достаточно средств на счет, прежде чем запустить проект в работу.

В настоящий момент, после знакомства с 65 проектами (и, надеюсь, воплощения), представленными в этой книге, вы уже имеете достаточный объем знаний, чтобы уверенно конструировать собственные устройства на основе Arduino. Вы познакомились с основными строительными блоками, составляющими большинство проектов, и я верю, что с помощью описанных приемов вы сможете решить любые задачи и одновременно получить удовольствие.

Я буду рад обсудить проекты, представленные в этой книге, и получить ваши отзывы и пожелания на моем веб-форуме по адресу *http://www.tronixforum.com/?forum=376318*.

И помните — это только начало. Еще больше информации о самых разных устройствах с идеями и рекомендациями по их использованию вы найдете в обширном сообществе пользователей Arduino в Интернете (например, на сайте Arduino: *http://forum.arduino.cc/*[1]) или в местных клубах.

Поэтому не сидите на месте — действуйте!

---

[1] Существует также русскоязычный сайт пользователей Arduino: *http://arduino.ru/*. — *Примеч. пер.*

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

Все исходные тексты, приведенные в книге, вы можете найти по адресу http://www.piter.com.

На веб-сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

Дж. Бокселл

Изучаем Arduino. 65 проектов своими руками

Перевел с английского А. Киселев

Заведующая редакцией	*Ю. Сергиенко*
Ведущий редактор	*Н. Римицан*
Литературный редактор	*А. Пасечник*
Художник	*С. Заматевская*
Корректоры	*С. Беляева, Н. Викторова*
Верстка	*Л. Егорова*

ООО «Питер Пресс», 192102, Санкт-Петербург, ул. Андреевская (д. Волкова), 3, литер А, пом. 7Н.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 — Книги печатные профессиональные, технические и научные.

Подписано в печать 06.09.16. Формат 70×100/16. Бумага писчая. Усл. п. л. 32,250. Тираж 1200. Заказ 0000.

Отпечатано в ОАО «Первая Образцовая типография». Филиал «Чеховский Печатный Двор».
142300, Московская область, г. Чехов, ул. Полиграфистов, 1.
Сайт: www.chpk.ru. E-mail: marketing@chpk.ru
Факс: 8(496) 726-54-10, телефон: (495) 988-63-87

КНИГА-ПОЧТОЙ

ЗАКАЗАТЬ КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР» МОЖНО ЛЮБЫМ УДОБНЫМ ДЛЯ ВАС СПОСОБОМ:

- на нашем сайте: **www.piter.com**
- по электронной почте: **books@piter.com**
- по телефону: **(812) 703-73-74**

ВЫ МОЖЕТЕ ВЫБРАТЬ ЛЮБОЙ УДОБНЫЙ ДЛЯ ВАС СПОСОБ ОПЛАТЫ:

- Наложенным платежом с оплатой при получении в ближайшем почтовом отделении.
- С помощью банковской карты. Во время заказа вы будете перенаправлены на защищенный сервер нашего оператора, где сможете ввести свои данные для оплаты.
- Электронными деньгами. Мы принимаем к оплате Яндекс.Деньги, Webmoney и Kiwi-кошелек.
- В любом банке, распечатав квитанцию, которая формируется автоматически после совершения вами заказа.

ВЫ МОЖЕТЕ ВЫБРАТЬ ЛЮБОЙ УДОБНЫЙ ДЛЯ ВАС СПОСОБ ДОСТАВКИ:

- Посылки отправляются через «Почту России». Отработанная система позволяет нам организовывать доставку ваших покупок максимально быстро. Дату отправления вашей покупки и дату доставки вам сообщат по e-mail.
- Вы можете оформить курьерскую доставку своего заказа (более подробную информацию можно получить на нашем сайте www.piter.com).
- Можно оформить доставку заказа через почтоматы (адреса почтоматов можно узнать на нашем сайте www.piter.com).

ПРИ ОФОРМЛЕНИИ ЗАКАЗА УКАЖИТЕ:

- фамилию, имя, отчество, телефон, e-mail;
- почтовый индекс, регион, район, населенный пункт, улицу, дом, корпус, квартиру;
- название книги, автора, количество заказываемых экземпляров.

БЕСПЛАТНАЯ ДОСТАВКА:

- курьером по Москве и Санкт-Петербургу при заказе на сумму **от 2000 руб.**
- почтой России при предварительной оплате заказа на сумму **от 2000 руб.**