

Ulli Sommer

Arduino

Mikrocontroller-Programmierung mit Arduino/Freeduino

Mit 125 Abbildungen

Улли Соммер

Программирование микроконтроллерных плат Arduino/Freeduino

Санкт-Петербург
«БХВ-Петербург»
2012

УДК 681.3.068
ББК 32.973.26-018.1
С61

Соммер У.

С61 Программирование микроконтроллерных плат Arduino/Freeduino. — СПб.: БХВ-Петербург, 2012. — 256 с.: ил. — (Электроника)

ISBN 978-5-9775-0727-1

Рассмотрено программирования микроконтроллерных плат Arduino/Freeduino. Описана структура и функционирование микроконтроллеров, среда программирования Arduino, необходимые инструменты и комплектующие для проведения экспериментов. Подробно рассмотрены основы программирования плат Arduino: структура программы, команды, операторы и функции, аналоговый и цифровой ввод/вывод данных. Изложение материала сопровождается более 80 примерами по разработке различных устройств: реле температуры, школьных часов, цифрового вольтметра, сигнализации с датчиком перемещения, выключателя уличного освещения и др. Для каждого проекта приведен перечень необходимых компонентов, монтажная схема и листинги программ. На FTP-сервере издательства выложены исходные коды примеров из книги, технические описания, справочные данные, среда разработки, утилиты и драйверы.

Для радиолюбителей

УДК 681.3.068
ББК 32.973.26-018.1

Die berechtigte Übersetzung von deutschsprachiges Buch Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino, ISBN: 978-3-645-65034-2. Copyright © 2010 Franzis Verlag GmbH, 85586 Poing. Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträger oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt. Die Russische Übersetzung ist von BHV St. Petersburg verbreitet, Copyright © 2011.

Авторизованный перевод немецкой редакции книги Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino, ISBN: 978-3-645-65034-2. Copyright © 2010 Franzis Verlag GmbH, 85586 Poing. Все права защищены, включая любые виды копирования, в том числе фотомеханического, а также хранение и тиражирование на электронных носителях. Изготовление и распространение копий на бумаге, электронных носителях данных и публикация в Интернете, в том числе в формате PDF, возможны только при наличии письменного согласия Издательства Franzis. Нарушение этого условия преследуется в уголовном порядке. Перевод на русский язык "БХВ-Петербург" © 2011.

ArduinoTM является зарегистрированной торговой маркой Arduino LLC и аффилированных компаний.

Группа подготовки издания:

Главный редактор	Екатерина Кондукова
Зам. главного редактора	Игорь Шишигин
Зав. редакцией	Григорий Добин
Перевод с немецкого	Виктория Букирева
Редактор	Леонид Кочин
Компьютерная верстка	Наталья Караваевой
Корректор	Виктория Пиотровская
Оформление обложки	Елены Беляевой
Зав. производством	Николай Тверских

Подписано в печать 30.09.11.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 20.32.

Тираж 2000 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

ISBN 978-3-645-65034-2 (нем.)
ISBN 978-5-9775-0727-1 (рус.)

© 2010 Franzis Verlag GmbH, 85586 Poing
© Перевод на русский язык "БХВ-Петербург", 2011

Оглавление

Предисловие.....	1
Введение	3
Глава 1. Общие сведения о микроконтроллерах.....	5
1.1. Структура и принцип работы контроллера.....	6
1.1.1. Центральный процессор.....	6
1.1.2. Оперативная память и память программ	7
1.2. Внешние устройства	8
1.3. Сравнение технологий RISC и CISC	8
1.3.1. Технология CISC.....	8
1.3.2. Технология RISC.....	10
1.3.3. Выводы	10
Глава 2. Программирование микроконтроллеров.....	11
2.1. Что такое программа?	11
2.2. Программирование на С	11
Глава 3. Краткий обзор семейства микроконтроллеров Arduino	13
3.1. Плата Arduino Mega	14
3.2. Плата Arduino Duemilanove	15
3.3. Плата Arduino Mini	15
3.4. Плата Arduino Nano	16
3.5 Плата Arduino Pro Mini.....	16
3.6. Плата Arduino Pro	17
3.7. Плата LilyPad.....	17
3.8. USB-адаптер	18
Глава 4. Платы расширения Arduino	19
4.1. Плата расширения Arduino ProtoShield	19
4.2. Плата расширения ArduMoto	20
4.3. Плата расширения TellyMate	21
4.4. Плата расширения ArduPilot	22
4.5. Модули XBeeZNet.....	22
4.6. Плата расширения Ethernet	23
Глава 5. Комплектующие изделия	25
5.1. Список основных комплектующих	25
5.2. Список деталей для дополнительных экспериментов	25

5.3. Экспериментальная плата Freeduino	26
5.4. Экспериментальная плата микроконтроллера Freeduino.....	26
5.5. Электропитание.....	27
5.6. Кнопка Reset.....	27
5.7. ISP-подключение	27
5.8. Замечания по технике безопасности	29
Глава 6. Электронные компоненты и их свойства	31
6.1. Светодиоды	31
6.2. Резисторы	32
6.3. Конденсаторы.....	33
6.4. Транзисторы.....	34
6.5. Диод	34
6.6. Акустический пьезопреобразователь ("пищалка")	35
6.7. Монтажный провод.....	35
6.8. Кнопка.....	35
6.9. Потенциометр.....	36
6.10. Фоторезистор	36
6.11. Монтажная панель с контактными гнездами.....	37
Глава 7. Предварительная подготовка	39
7.1. Установка драйвера	39
7.2. Вспомогательная программа MProg для FT232RL	40
7.3. Программирование микросхемы FT232R с помощью MProg	44
7.4. Установка программного обеспечения Arduino	45
Глава 8. Среда разработки Arduino.....	47
8.1. Установки в Arduino-IDE	48
8.2. Наша первая программа "ES_Blinkt"	50
8.3. Что мы сделали?.....	52
Глава 9. Основы программирования Arduino	55
9.1. Биты и байты	55
9.2. Базовая структура программы	56
9.2.1. Последовательное выполнение программы.....	56
9.2.2. Прерывание выполнения программы.....	57
9.3. Структура программы Arduino	57
9.4. Первая программа с Arduino	58
9.5. Команды Arduino и их применение	59
9.5.1. Комментарии в исходном тексте	59
9.5.2. Фигурные скобки {}	60
9.5.3. Точка с запятой ;	60
9.5.4. Типы данных и переменные	60
9.5.5. Имя переменной.....	60
9.5.6. Локальные и глобальные переменные	61
9.5.7. Различные типы данных	61
9.5.8. Операторы	65
9.5.9. Директива #define.....	66

9.5.10. Управляющие конструкции	66
9.5.11. Циклы.....	71
9.5.12. Функции и подпрограммы	75
9.5.13. Функции преобразования типа	78
9.5.14. Математические функции	79
9.5.15. Последовательный ввод/вывод	86
9.5.16. Как функционирует последовательный интерфейс?.....	93
9.5.17. Программная эмуляция UART	96
9.5.18. Конфигурация входа/выхода и установка порта	97
9.5.19. Аналоговый ввод данных и АЦП	103
9.5.20. Аналоговый выход ШИМ	105
9.6. Некоторые специальные функции.....	110
Установка паузы с помощью <i>delay</i>	110
Функции случайных чисел	110
Сколько времени прошло?	113
Глава 10. Дальнейшие эксперименты с Arduino	115
10.1. Регулятор уровня яркости светодиода с транзистором	115
10.2. Плавное мигание	117
10.3. Подавление дребезга контактов кнопок.....	120
10.4. Задержка включения	124
10.5. Задержка выключения	126
10.6. Светодиоды и Arduino	127
10.7. Подключение больших нагрузок	130
10.8. ЦАП на основе ШИМ-порта	132
10.9. С музыкой все веселей.....	136
10.10. Романтический свет свечи с помощью микроконтроллера	139
10.11. Контроль персонала на проходной.....	140
10.12. Часы реального времени	143
10.13. Программа школьных часов	144
10.14. Управление вентилятором	148
10.15. Автомат уличного освещения	151
10.16. Сигнализация.....	153
10.17. Кодовый замок	155
10.18. Измеритель емкости с автоматическим выбором диапазона	159
10.19. Профессиональное считывание сопротивления потенциометра	162
10.20. Сенсорный датчик.....	164
10.21. Конечный автомат	166
10.22. 6-канальный вольтметр на основе Arduino	169
10.23. Программирование самописца напряжения	171
10.24. Осциллограф с памятью на основе Arduino.....	173
10.25. Программа StampPlot — бесплатный профессиональный регистратор данных	175
10.26. Управление через VB.NET	179
10.27. Реле температуры.....	181
Глава 11. Шина I²C.....	185
11.1. Передача бита.....	186
11.2. Состояние "СТАРТ"	186
11.3. Состояние "СТОП"	186

11.4. Передача байта.....	186
11.5. Подтверждение.....	187
11.6. Адресация	187
11.7. 7-битовая адресация.....	187
Глава 12. Arduino и температурный датчик LM75 с I²C-шиной.....	189
Глава 13. Расширителей порта I²C с PCF8574	193
Глава 14. Ультразвуковой датчик для определения дальности.....	197
14.1. Ультразвуковой датчик SRF02	197
14.2. Считывание данных.....	198
Глава 15. Сопряжение платы Arduino с GPS	201
15.1. Сколько требуется спутников?	202
15.2. Как подключить GPS к Arduino?	202
15.3. GPS-протокол.....	203
Глава 16. Сервопривод с платой Servo для Arduino	209
16.1. Как функционирует сервопривод?	209
16.2. Подключение привода к Arduino	210
Глава 17. Жидкокристаллические дисплеи.....	213
17.1. Поляризация дисплеев.....	214
17.2. Статическое управление и мультиплексный режим	214
17.3. Угол обзора	215
17.4. Отражающие, пропускающие и полупрозрачные ЖКИ	215
17.5. Установка контрастности дисплея	216
17.6. Набор отображаемых символов.....	217
17.7. Расположение выводов распространенных ЖКИ	218
17.8. Управление дисплеем от микроконтроллера.....	220
17.9. Инициализация дисплеев	220
17.10. Подключение дисплея к Arduino	222
17.11. Первый эксперимент с ЖКИ.....	223
17.12. Как же все работает?	226
ПРИЛОЖЕНИЯ.....	229
Приложение 1. Соответствие выводов Arduino и ATmega	231
Приложение 2. Escape-последовательности.....	232
Приложение 3. Таблица ASCII	234
Приложение 4. Перечень фирм-поставщиков компонентов	239
Приложение 4. Перечень фирм-поставщиков компонентов	239
Приложение 5. Описание компакт-диска	240
Предметный указатель	241

Предисловие

Многим нелегко сделать первый шаг в области электроники и программирования микроконтроллеров. Для освоения большинства систем микроконтроллеров новичку сначала требуется перевернуть горы литературы, а также прочитать и понять непростые технические паспорта. Среды программирования, как правило, довольно сложны и рассчитаны на профессиональных программистов. Таким образом, доступ в мир микроконтроллеров остается для некоторых навсегда закрытым.

Arduino — это простая для освоения платформа с открытым кодом на основе встроенного микроконтроллера и среды разработки с программным интерфейсом API для микроконтроллеров. Для взаимодействия между человеком и микроконтроллером могут присоединяться различные аналоговые и цифровые датчики, которые регистрируют состояние окружающей среды и передают данные в микроконтроллер. Микроконтроллер обрабатывает входящие данные, а программа выдает новые данные в виде аналоговых или цифровых значений. В результате открываются широкие горизонты для творчества.

В распоряжении разработчика предоставлены готовые программы и библиотеки функций среды программирования Arduino. Комбинируя аппаратные и программные средства, вы сможете с помощью этой книги связать наш реальный мир с миром микроконтроллера, который состоит из битов и байтов.

Желаю приятно провести время при чтении книги и проведении экспериментов!

Улли Соммер (Ulli Sommer)

Введение

Книга представляет собой учебный курс программирования микроконтроллеров. Вы познакомитесь со структурой и принципом действия микроконтроллера, изучите среду программирования Arduino, узнаете о необходимых инструментах и комплектующих для проведения экспериментов. Целая глава книги посвящена основам программирования плат Arduino. Здесь подробно описывается структура программы, команды, операторы и функции, аналоговый и цифровой ввод и вывод данных. Изложение материала сопровождается многочисленными практическими примерами. Вполне вероятно, что экспериментируя, вы сможете изобрести что-то новое в области микроконтроллерной технологии. Не останавливайтесь на достигнутом, старайтесь усовершенствовать конструкцию устройства и код программы.

Книга состоит из 17 глав и 5 приложений. Описано более 80 различных устройств на основе платы Arduino. Для каждого эксперимента приведен перечень необходимых компонентов, монтажная схема макета и листинги программ.

Подготовка к экспериментам. Для предлагаемых экспериментов потребуется всего несколько простых и доступных компонентов из ящика с радиодеталями. Кое-что, возможно, придется приобрести специально. В *приложении 4* указан перечень фирм-поставщиков электронных компонентов.

Для большинства экспериментов не нужны ни батареи, ни внешний источник питания.

Очень полезным в работе будет многофункциональный измерительный прибор (мультиметр) и/или интерфейс к компьютеру. С этими средствами вы сможете провести дополнительные эксперименты и узнать много полезного. Пригодится и стандартный аккумулятор типоразмера AA (Mignon) или AAA (Micro).

Лицензия GPL. Вы можете поделиться своими собственными программами с другими пользователями Интернета. Приведенные примеры программ имеют открытую лицензию GPL (General Public License). Таким образом, вы имеете право на изменение программ, в соответствии с условиями GPL, их публикацию и предоставление другим пользователям, если распространяете свои программы также под лицензией GPL.

Требования к системе. Персональный компьютер не ниже Pentium III, ОС Windows 98SE/ME/XP/Vista/Windows 7, Linux, Macintosh, дисковод компакт-дисков, платформа Java.

Обновления и поддержка. Платформа Arduino постоянно совершенствуется. Обновления можно загрузить бесплатно с Web-сайта www.arduino.cc (вы платите только за доступ в Интернет).

О компакт-диске к книге. Материал прилагаемого к книге компакт-диска можно скачать по ссылке <ftp://85.249.45.166/9785977507271.zip>. Ссылка также доступна на странице книги на сайте www.bhv.ru.

Компакт-диск содержит различные программы, инструменты для программирования, технические паспорта и принципиальные схемы, а также коды примеров из книги. Использование этих материалов облегчает работу с книгой. Описание компакт-диска приведено в *приложении 5*.

Глава 1



Общие сведения о микроконтроллерах

Перед тем как начать работать с аппаратной вычислительной платформой Arduino, важно получить общие сведения о микроконтроллерах. Микроконтроллеры применяются, прежде всего, для автоматизации в метрологии, технике управления и автоматического регулирования. Преимущество микроконтроллеров состоит в том, что можно эффективно и с малыми затратами измерять и интерпретировать физические величины, чтобы потом принимать требуемые решения и выполнять необходимые действия.

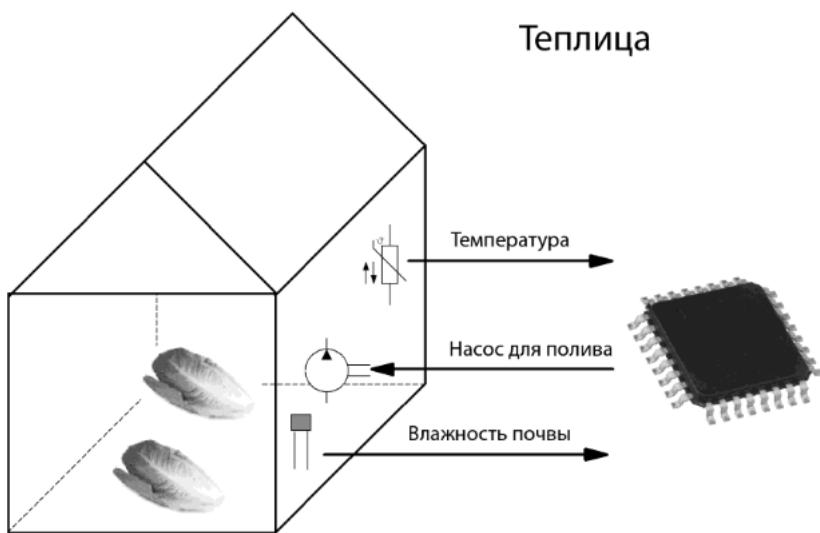


Рис. 1.1. Система управления климатом теплицы

Область возможных приложений микроконтроллеров чрезвычайно обширна: от частного домохозяйства (например, для управления теплицей или освещением) до промышленного производства, где могут обслуживаться и эксплуатироваться комплексные устройства, управляемые системами микроконтроллеров. На рис. 1.1

приведен типичный пример обработки данных для управления оросительной установкой теплицы. Контроллер фиксирует данные о температуре окружающей среды и влажности почвы, полученные от датчиков. Результаты измерения далее подвергаются логической обработке в микроконтроллере. Затем формируются сигналы управления насосом для полива.

1.1. Структура и принцип работы контроллера

Контроллер представляет собой, по сути, микрокомпьютер и содержит все присущие ему основные модули (рис. 1.2). Стандартные блоки каждого микроконтроллера — это центральный процессор (CPU), оперативная память (RAM), а также память программ (Flash-память) и внешние устройства.

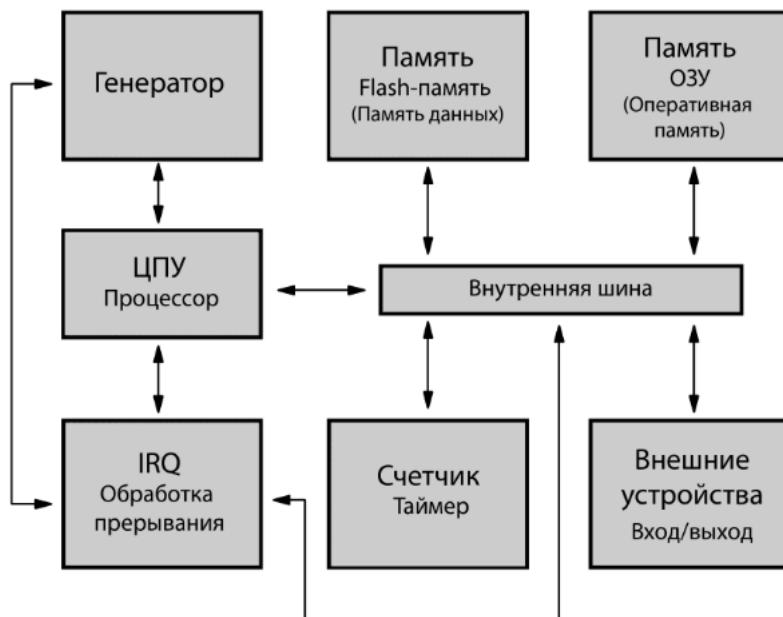


Рис. 1.2. Упрощенная структура микроконтроллера

1.1.1. Центральный процессор

Основное функциональное устройство микроконтроллера — центральный процессор (CPU — Central Processing Unit). Его можно сравнить с "мозгом" микроконтроллера. Сигналы в нем представлены в цифровой форме и над ними выполняются арифметические и логические операции.

1.1.2. Оперативная память и память программ

Оперативная память и память программ традиционно рассматриваются отдельно. Программа пользователя, т. е. наша собственная программа, которую мы сами писали, сохраняется в энергонезависимой Flash-памяти программ (рис. 1.3). В зависимости от типа контроллера память программ может занимать объем от нескольких килобайт до мегабайт. Кроме того, в некоторых вычислительных системах можно увеличить память программ, подключая внешние Flash-накопители.

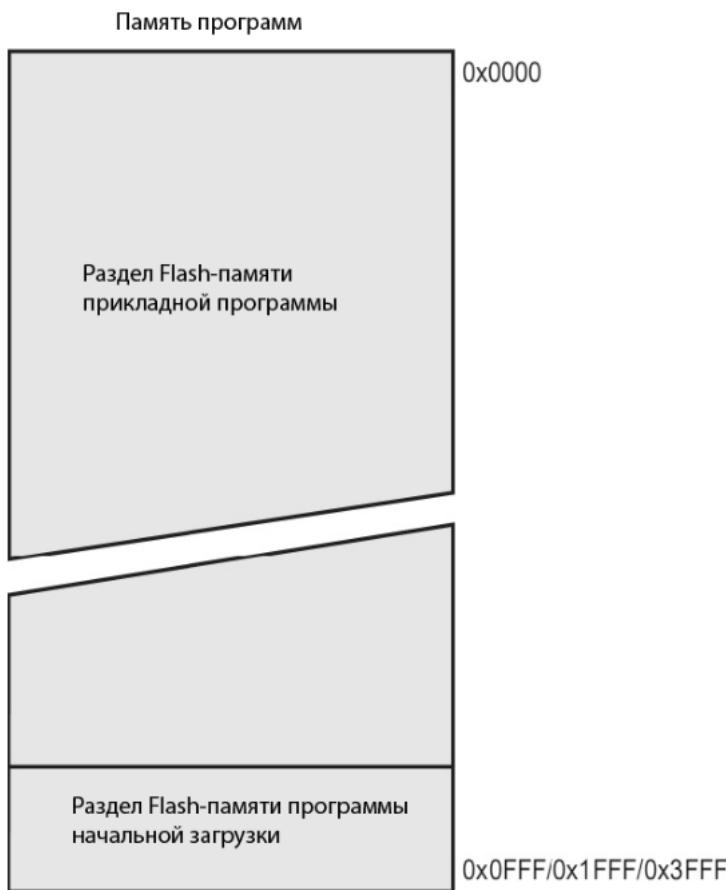


Рис. 1.3. Flash-память микроконтроллера ATmega168PA
(источник: технический паспорт компании ATMEL)

Оперативная память служит для временного хранения различных промежуточных данных. Здесь хранятся и результаты вычислений, полученные во время выполнения программы.

Назначение оперативной памяти ОЗУ (RAM — Random Access Memory) — возможность быстрого обращения к ограниченному количеству данных. Ее объем, как правило, значительно меньше, а быстродействие намного больше, чем Flash-памяти.

Значения записываются и хранятся в ОЗУ во время выполнения и энергозависимы в отличие от Flash-памяти, т. е. после перезагрузки контроллера содержимое ОЗУ полностью стирается. На рис. 1.4 изображена структура ОЗУ микроконтроллера ATmega168PA.

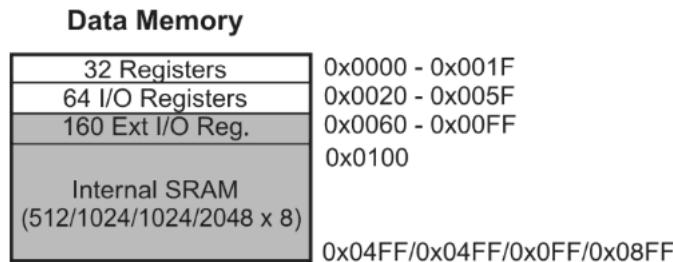


Рис. 1.4. RAM-память микроконтроллера ATmega168PA
(источник: технический паспорт ATMEL)

1.2. Внешние устройства

Внешними (периферийными) устройствами часто называют все компоненты микроконтроллера, кроме центрального процессора и памяти. В частности, к ним относятся внешние интерфейсы, например, цифровые входы и выходы (Input/Output — сокращенно I/O). Большинство микроконтроллеров снабжены также различными цифровыми и аналоговыми входами и выходами.

1.3. Сравнение технологий RISC и CISC

Рассмотрение RISC- и CISC-технологий — это уже более глубокий взгляд на цифровую и микроконтроллерную технику. Сразу заметим, что контроллеры AVR для Arduino базируются на технологии RISC. Кратко опишем технологии RISC и CISC.

1.3.1. Технология CISC

При технологии CISC- в ОЗУ загружается и программа, и данные. Говорят также о том, что код программы и данные делят между собой одну и ту же область памяти. Это имело смысл, в частности, в первых вычислительных системах, т. к. оперативная память была дорога.

Для микроконтроллера гораздо более важный отличительный признак — это структура команд. Компьютер с CISC-технологией располагает большим ассортиментом очень узкоспециализированных команд. В цифровой технике команда — это последовательность определенных байтов. Один байт может принимать 256 (от 0 до 255) различных состояний. Чтобы реализовать более 256 различных команд, нужны дополнительные байты. Таким образом, специальная команда состоит из нескольких, например пяти, байтов. Загрузка этой команды продолжается дольше, чем короткой однобайтовой команды.

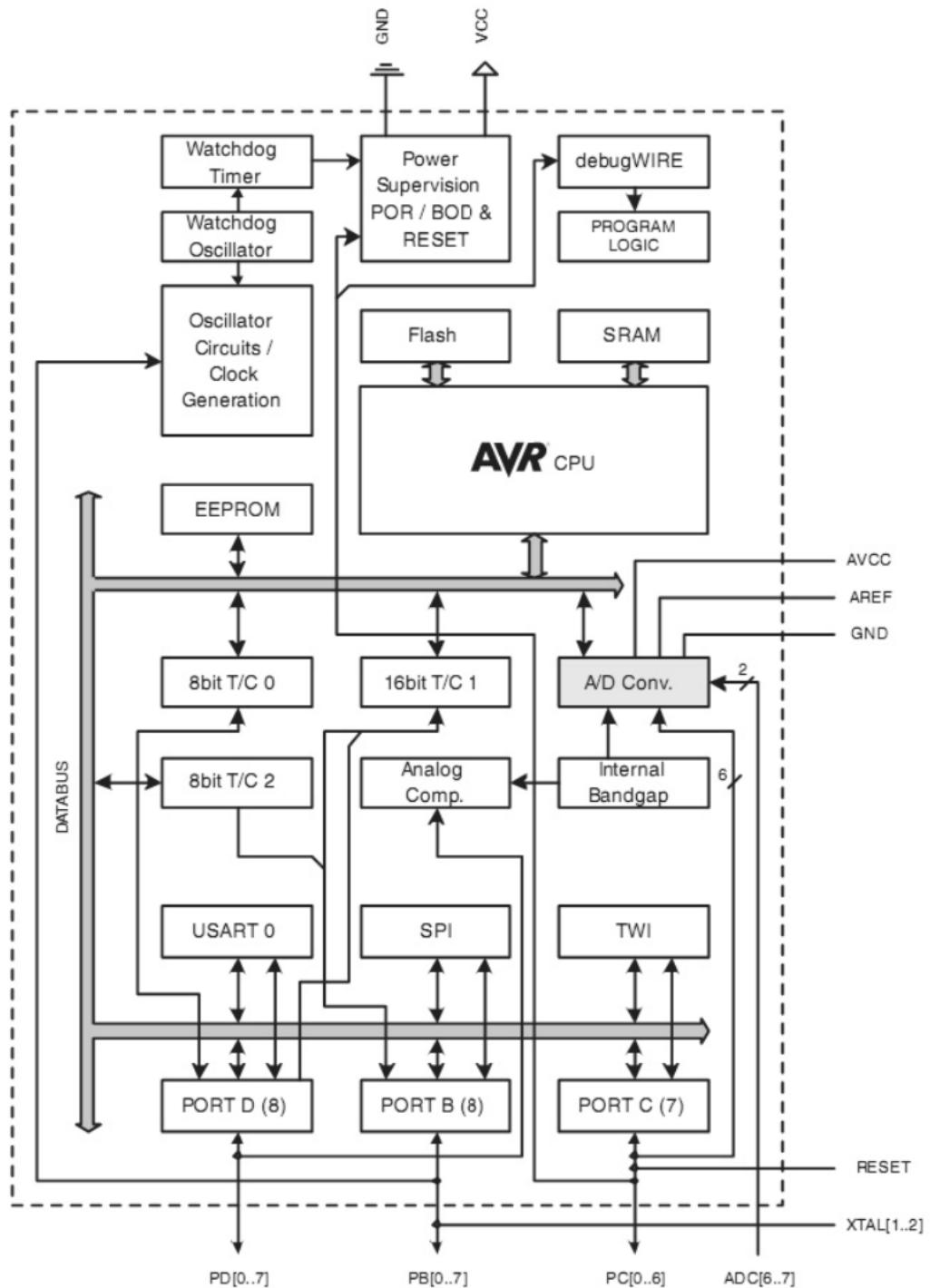


Рис. 1.5. Блок-схема микроконтроллера
(источник: технический паспорт компании ATMEL)

1.3.2. Технология RISC

Было установлено, что в CISC-компьютерах, как правило, около 90% исходного кода программ состоит только из примерно 30 различных команд. Так возникла мысль реализовать в центральном процессоре систему ограниченного числа коротких и быстро выполняющихся команд. Таким образом, в RISC-микроконтроллерах команды, как правило, состоят не более чем из 1–2 байтов. Длинную специальную команду приходится составлять их из нескольких коротких. Чтобы достичь равной производительности с CISC-компьютерами, большинство RISC-компьютеров располагают большим числом регистров. Регистр — это встроенная в центральный процессор сверхскоростная память. Еще один отличительный признак RISC-систем — четкое физическое и логическое разделение между областями памяти программ и данных.

1.3.3. Выводы

CISC-компьютер обладает множеством специальных команд, которые занимают большой объем памяти и требуют, как правило, длительного времени выполнения. Команды RISC-компьютера потребляют меньше памяти и выполняются значительно быстрее. Тем не менее, недостаток RISC-технологии состоит в том, что здесь специальные команды приходится заменять цепочками из нескольких основных команд. Таким образом, и CISC- и RISC-технология имеют свои преимущества и недостатки. Следует отметить, что не существует ни полностью RISC-, ни полностью CISC-систем.

На рис. 1.5 приведена подробная блок-схема микроконтроллера для изучения его внутреннего устройства.



Глава 2

Программирование микроконтроллеров

Степень интеграции микропроцессоров и микроконтроллеров все возрастает и они все шире проникают в прикладные области метрологии, техники управления и автоматического регулирования. Даже в обычной жизни микроконтроллеры становятся все популярнее. Так происходит, с одной стороны, оттого, что сегодня сложные аналоговые схемы заменяются более простыми цифровыми микроконтроллерами. Но решающее преимущество микроконтроллеров — непревзойденное соотношение цена/производительность.

2.1. Что такое программа?

Программа — это описание процесса обработки информации. При выполнении программы рассчитывается совокупность выходных значений исходя из совокупности переменных или постоянных входных значений. Цель выполнения программы — сбор данных либо получение отклика на входные значения. Наряду с собственно вычислениями программа может содержать команды для доступа к аппаратным средствам компьютера и для управления ходом выполнения алгоритма. Программа состоит из нескольких строк так называемого исходного текста. При этом каждая строка содержит один или несколько арифметических или управляющих операторов. Не только сами команды, но и последовательность их выполнения существенно влияет на результат обработки информации. Выполнение соответствующих операций происходит последовательно (по очереди). Упорядоченную определенным образом последовательность инструкций программы называют также алгоритмом.

2.2. Программирование на С

Язык программирования С (ANSI C) прост для изучения. С — это язык программирования высокого уровня, который создал Деннис Ричи (Dennis Ritchie) в начале 1970-х годов в Bell Laboratories для операционной системы UNIX. С тех пор этот язык очень широко распространен. Области применения языка С весьма различны. Он используется, например, в системном и прикладном программировании.

Основные модули всех систем UNIX и ядро многих операционных систем запрограммированы на языке С.

Многочисленные другие языки, например С ++, Objective-C, С #, Java, PHP и Perl ориентируются на синтаксис и свойства языка С. Изучение этого языка программирования очень выгодно, т. к. в дальнейшем легче освоить многие системы микроконтроллеров. Почти для всех микроконтроллеров существует бесплатный компилятор С, предлагаемый производителем микроконтроллера. Компилятор С от Arduino несколько проще, чем профессиональные С-компиляторы, но весьма эффективен. С компилятором Arduino не нужно заботиться о программировании сложных аппаратных средств, поскольку в среде разработки есть соответствующие встроенные команды.



Глава 3

Краткий обзор семейства микроконтроллеров Arduino

Аппаратные средства Arduino включают популярные и доступные комплектующие изделия. Поэтому принцип работы системы понятен, настройка схемы под требования разработчика проста и обеспечена возможность дальнейшей модификации. Основа — контроллер ATmega компании Atmel широко распространенного 8-разрядного семейства AVR. К нему добавляется узел электропитания и последовательный интерфейс. В последних версиях Arduino имеется USB-интерфейс. Через него происходит загрузка программ пользователя и, при необходимости, обмен данными между персональным компьютером и платой Arduino во время выполнения программы.

Плату Arduino часто рассматривают как устройство ввода/вывода. Плата Arduino предоставляет в распоряжение пользователя 14 цифровых входов или выходов, из них шесть можно использовать как аналоговый выход (8-разрядный ШИМ-канал). Следующие шесть входов могут принимать аналоговые сигналы (10-разрядный АЦП). В качестве дополнительных интерфейсов предусмотрены шины SPI и I²C.

Имеется несколько вариантов исполнения плат Arduino (рис. 3.1–3.7). Оригинальные изделия производит итальянская фирма Smart Projects. Существуют также многочисленные клоны и копии других поставщиков, наконец, встречаются свободно распространяемые аппаратные средства. Важный компаньон проекта Arduino — компания Sparkfun (Боулдер, Колорадо). Совместно с американским партнером выпущен ряд оптимизированных плат Arduino, в название которых добавлено обозначение "Pro". Кроме того, с появлением платы LilyPad возникло новое направление, охватывающее область переносных вычислительных средств.

Многие пользователи предпочитают плату Arduino Duemilanove, изготовленную компанией Smart Projects, на которой установлен ATmega-контроллер в DIP-корпусе на панельке. Это изделие несущественно отличается от весьма успешного предшественника — Arduino Diecimilano (рис. 3.2), которая названа в честь первых 10 000 проданных плат. На плате добавлен чип компании FTDI, реализующий интерфейс USB.

Новая плата Arduino Mega (рис. 3.1) имеет более мощный микроконтроллер (ATmega1280) и память большего объема, а также расширенные функции интерфейса ввода/вывода.

Плата Arduino Mini (рис. 3.3) существенно меньше по размеру и имеет формат DIP24. Весь модуль вставляется в 24-контактную панельку. Версия платы Arduino

Pro Mini (рис. 3.5) компании Sparkfun почти идентична, но поставляется в планарном корпусе. Для программирования предусмотрен USB-адаптер, который может вставляться с боковой стороны модуля.

Плата LilyPad Leah Buechley (выщенная в сотрудничестве с компанией Sparkfun) также совместима с Arduino, но обладает своими особенностями. Плата LilyPad и вспомогательные комплектующие специально разработаны для установки в одежду и реализуют симбиоз техники и дизайна. Характерная круглая форма платы LilyPad-Arduino (рис. 3.7) также привлекает внимание необычной окраской и расположением контактов по окружности. Здесь применен микроконтроллер Atmega168 с низким электропотреблением (3,3 В). Многочисленные маленькие платы внешних устройств (датчик, светодиоды, микропереключатель и т. п.) выполнены в едином стиле под девизом "Сочетание электроники со швейным производством".

Информация о новых версиях платы и комплектующих имеется на сайте проекта Arduino и сайте компании SparkFun Electronics.

3.1. Плата Arduino Mega

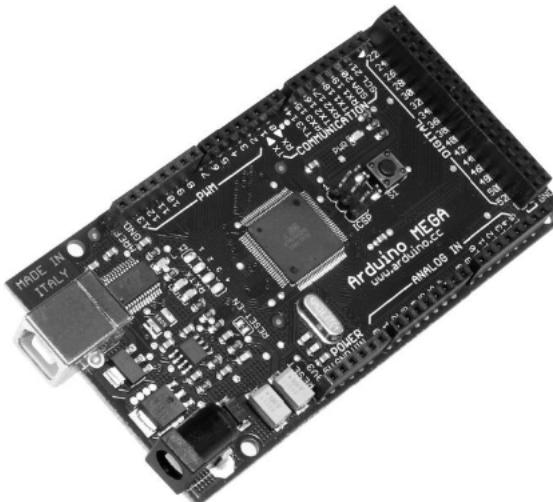


Рис. 3.1. Плата Arduino Mega
(источник: компания Elmicro)

Технические данные:

- Микроконтроллер ATmega1280.
- Flash-память 128 Кбайт.
- RAM-память 8 Кбайт, EEPROM 4 Кбайт.
- Тактовая частота 16 МГц.
- 54 цифровых канала ввода/вывода, из них 14 применимы для ШИМ.
- 4 аппаратных средства UART.
- Интерфейс I²C, SPI.
- 16 аналоговых входов 10-разрядных АЦП.
- Интерфейс USB, источник питания, начальный загрузчик и т. д., как у Arduino Duemilanove.
- Габаритные размеры примерно 101×53×12 мм.

3.2. Плата Arduino DueMilanove

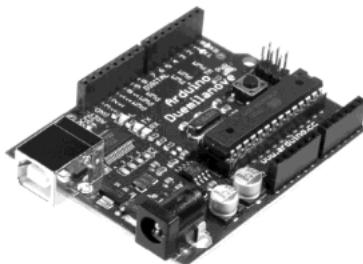


Рис. 3.2. Плата Arduino DueMilanove
(источник: компания Elmicro)

Технические данные:

- Микроконтроллер ATmega328.
- Flash-память 32 Кбайт (из них 2 Кбайт для начального загрузчика системы).
- RAM 2 Кбайт, EEPROM 1 Кбайт.
- Тактовая частота 16 МГц.
- 14 цифровых каналов ввода/вывода, из них 6 применимы для ШИМ.
- 6 аналоговых входов 10-разрядных АЦП.
- Встроенный интерфейс USB FT232RL от компании FTDI.
- Рабочее напряжение 5 В, питание через USB или от стабилизатора напряжения (входное напряжение 7–12 В).
- Габаритные размеры примерно 69×53×12 мм.
- Загрузчик системы уже установлен при поставке, загрузка возможна без программного адаптера.

3.3. Плата Arduino Mini



Рис. 3.3. Плата Arduino Mini
(источник: компания Elmicro)

Технические данные:

- Микроконтроллер ATmega168 с тактовой частотой 16 МГц.
- Программирование через адаптер USB (ARDUINO/USB, адаптер USB с чипом компании FTDI).
- EEPROM 512 бит.
- SRAM 1 Кбайт.
- FLASH 16 Кбайт (2 Кбайт требуется для начального загрузчика системы).
- Рабочее напряжение 5 В.
- 14 цифровых каналов ввода/вывода, 6 из них могут использоваться для формирования ШИМ.
- Восемь аналоговых входов 10-разрядных АЦП.
- Напряжение питания 7–9 В.

3.4. Плата Arduino Nano



Рис. 3.4. Плата Arduino Nano
(источник: компания Elmicro)

Технические данные:

- ATmega328 или более старая версия 168 с тактовой частотой 16 МГц.
- Программирование через встроенный разъем USB.
- Функция автоматического сброса.
- Рабочее напряжение 5 В.
- 14 цифровых каналов ввода/вывода, 6 из них могут использоваться для формирования ШИМ.
- Восемь аналоговых входов 10-разрядных АЦП.
- Flash-память объемом 32 Кбайт или 16 Кбайт.
- SRAM 1 Кбайт.
- EEPROM 512 байт или 1 Кбайт.
- Максимальный выходной ток 40 мА.
- Напряжение питания 6–20 В.
- Габаритные размеры: 18×43 мм.

3.5 Плата Arduino Pro Mini



Рис. 3.5. Плата Arduino Pro Mini
(источник: компания Elmicro)

Технические данные:

- ATmega328 с тактовой частотой 16 МГц (точность 0,5%).
- Программирование через адаптер USB (ARDUINO/USB).
- Функция автоматического сброса.
- Варианты платы под рабочее напряжение 5 и 3,3 В.
- Максимальный выходной ток 150 мА.
- Защита от перегрузки.

- Защита от неправильной полярности.
- Напряжение питания 5–12 В.
- Встроенное питание и светодиоды состояния.
- Габаритные размеры: 18×33 мм.
- Вес менее 2 г.

3.6. Плата Arduino Pro



Рис. 3.6. Плата Arduino Pro
(источник: компания Elmicro)

Технические данные:

- ATmega328 и более старые ATmega168 с тактовой частотой 16 МГц.
- Программирование через адаптер USB (ARDUINO/USB).
- Варианты исполнения под рабочее напряжение 5 и 3,3 В.
- 14 цифровых выводов каналов ввода/вывода, из них 6 применимы для ШИМ.
- Шесть аналоговых входов 10-разрядных АЦП.
- Напряжение питания 3,35–12 В (версия 3,3 В), 5–12 В (версия 5 В).
- Выходной ток цифрового выхода 40 мА.
- FLASH 32 Кбайт или 16 Кбайт (ATmega168).
- SRAM1 Кбайт (ATmega168) или 2 Кбайт (ATmega328).
- 512-(ATmega168) или EEPROM 1 Кбайт.

3.7. Плата LilyPad

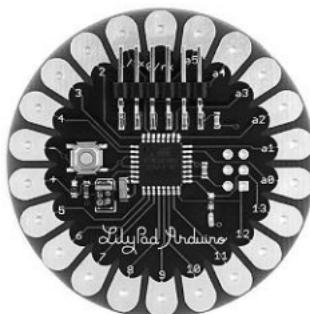


Рис. 3.7. Плата LilyPad Arduino
(источник: компания Elmicro)

Технические данные:

- ATmega328V и более старые ATmega168V с тактовой частотой 16 МГц.
- Программирование через адаптер USB (ARDUINO/USB).

- Напряжение питания 2,7–5,5 В.
- 14 цифровых каналов ввода/вывода, из них 6 применимы для ШИМ.
- Шесть аналоговых входов 10-разрядных АЦП.
- Выходной ток 40 мА.
- FLASH 32 Кбайт или 16 Кбайт (ATmega168).
- SRAM 1 Кбайт (Atmega168) или 2 Кбайт (ATmega328).
- 512-(Atmega168) или EEPROM 1 Кбайт.

3.8. USB-адаптер



Рис. 3.8. USB-адаптер с чипом компании FTDI (источник: компания Elmicro)

Адаптер USB (рис. 3.8) выпускается в исполнении на 3,3 и 5 В.

Адаптер предназначен для программирования Arduino-плат без USB-порта. Расположение выводов соответствует оригинальным спецификациям Arduino. Адаптер может использоваться также для обмена данными через виртуальный последовательный интерфейс. Код программы Arduino можно загружать в плату без аппаратного сброса системы кнопкой Reset.

Глава 4



Платы расширения Arduino

Для плат Arduino выпущено множество различных плат расширения. При просмотре сайтов в Интернете почти ежемесячно можно обнаружить новую плату и полезные расширения. Платы расширения Arduino полностью совместимы с базовыми Arduino-платами, за исключением маленьких модулей и платы LilyPad.

4.1. Плата расширения Arduino ProtoShield

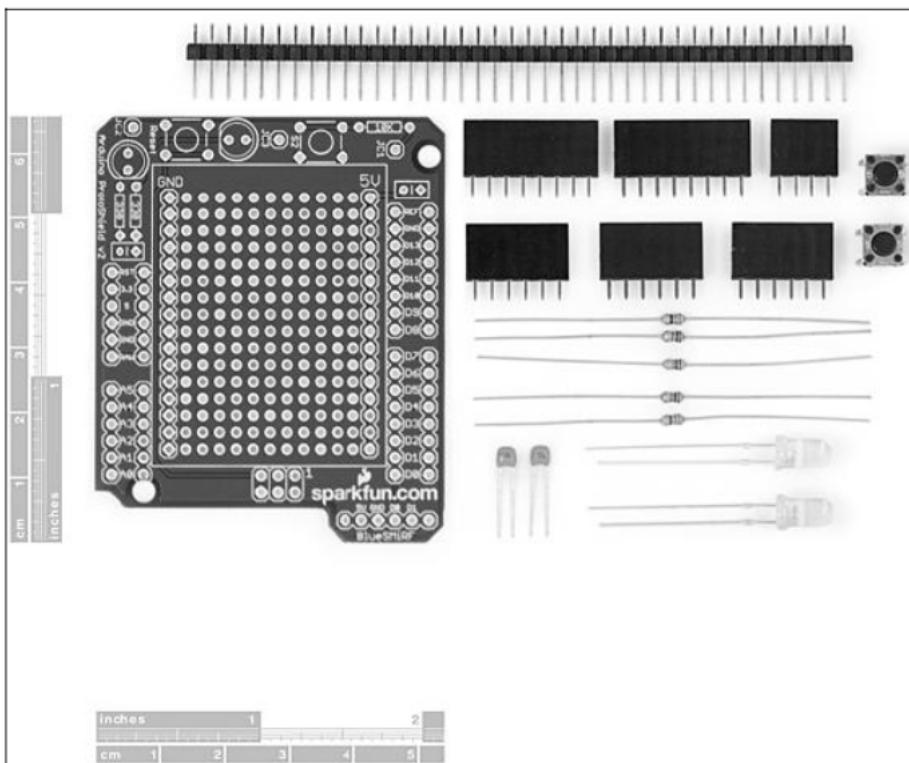


Рис. 4.1. Набор Arduino ProtoShield (источник: компания SparkFun)

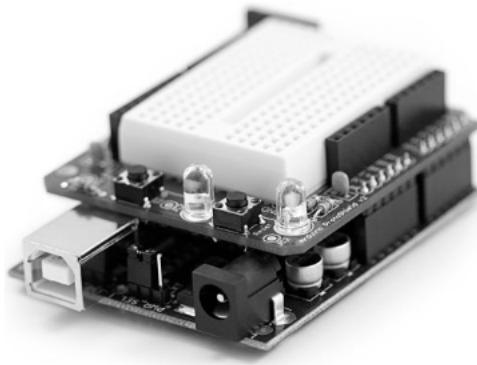


Рис. 4.2. Плата Arduino Duemilanove с платой расширения ProtoShield (источник: компания SparkFun)

Плата расширения ProtoShield (рис. 4.1) предназначена для изготовления самоделок без пайки. Все эксперименты можно проводить на маленькой панели с контактными гнездами. На рис. 4.2 изображен внешний вид платы Arduino Duemilanove с подключенной к ней платой расширения ProtoShield.

4.2. Плата расширения Ardumoto

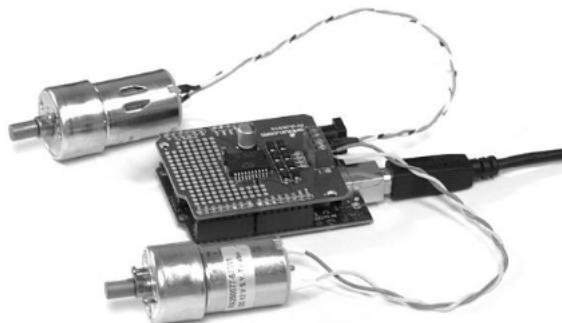


Рис. 4.3. Плата расширения Ardumoto для привода электродвигателя (источник: компания SparkFun)

Плата Ardumoto (рис. 4.3) идеально подходит для управления маленькими электродвигателями. Провода от электродвигателей легко присоединяются к винтовым зажимам макетной платы и с помощью маленькой программы можно задавать желаемую скорость и направление вращения двигателей.

Технические данные соответствуют приводу электродвигателя IC L298. Технический паспорт есть на прилагаемом к книге компакт-диске.

4.3. Плата расширения TellyMate

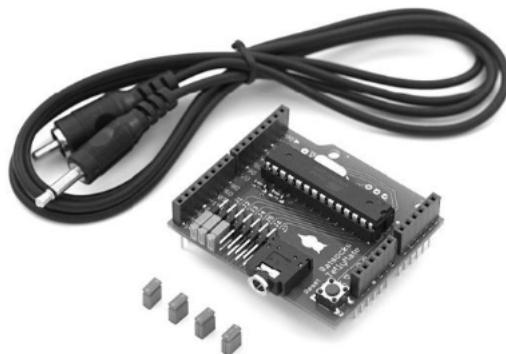


Рис. 4.4. Плата расширения TellyMate
(источник: компания SparkFun)

Плата TellyMate (рис. 4.4) является, пожалуй, наиболее интересной платой расширения, разработанной для платформы Arduino. Ее возможности применения почти не ограничены. С ее помощью можно отображать на телевизионном экране данные, тексты и графические объекты (рис. 4.5). Это позволяет превратить домашний телевизор в Arduino-дисплей. Arduino-микроконтроллер подключается к плате TellyMate через последовательный интерфейс.



Рис. 4.5. Плата расширения TellyMate в действии (источник: компания SparkFun)

Функциональные возможности:

- Телевизионный выход Arduino.
- Композитный видеосигнал PAL или NTSC.
- Вставляемая плата расширения Arduino.
- Работает с последовательным портом и т. д.

- Режим 38×25 символов.
- Представление символов черного и белого цвета.
- Простые графические объекты.
- Несложное программирование.

4.4. Плата расширения ArduPilot

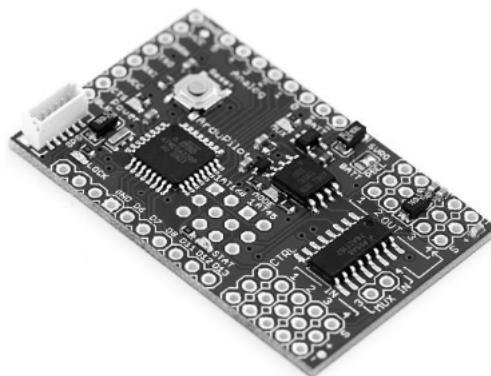


Рис. 4.6. Плата расширения ArduPilot-Arduino с микроконтроллером ATmega328 (источник: компания SparkFun)

Плата расширения ArduPilot (рис. 4.6) для авиамоделей является в высшей степени интересной игрушкой. Она позволяет управлять автономным полетом модели самолета.

Дополнительная информация находится по адресу: <http://diydrones.com>.

4.5. Модули XBeeZNet



Рис. 4.7. Модуль XBee ZNet 2.5 OEM (источник: компания SparkFun)

Для желающих передавать данные по радиоканалу предназначены модули XBee (рис. 4.7). Они реализуют беспроводной последовательный порт UART. Имея две платы Arduino или ПК и плату Arduino, можно общаться по радио.

Технические данные радиомодуля ZigBee:

- Рабочее напряжение — 2,8–3,4 В.
- Частота — стандарт ZigBee, 2,4 ГГц.
- Излучаемая мощность — 1 мВт (0 дБм).
- Чувствительность — -92 дБм.
- Дальность действия — 30 м внутри помещения, 100 м снаружи (в зависимости от условий эксплуатации).
- Ток, потребляемый от источника питания — при передаче — 45 мА, при приеме — 50 мА, в дежурном режиме — 10 мкА.

- Скорость передачи данных (по радио) — 250 000 бит/сек.
- Скорость передачи данных (интерфейс) — 1 200–115 200 бит/сек.
- Уровень сигналов последовательного интерфейса — 3,3 В; для подключения к персональному компьютеру обязательно требуется 3,3-вольтовый преобразователь уровня (MAX3232).
- Стандарт — совместимый с ZigBee/802.15.4.
- Варианты топологии — "точка-точка", "звезда".
- Габаритные размеры — 24,38×27,61×4 мм, шаг 2 мм.

4.6. Плата расширения Ethernet



Рис. 4.8. Плата расширения Ethernet
(источник: компания Solarbotics)

Плата расширения Arduino Ethernet (рис. 4.8) позволяет подключать плату Arduino к локальной сети и Интернету. Она основана на сетевом контроллере Wiznet W5100, который обеспечивает поддержку протоколов TCP и UDP. Допустимо одновременное подключение до четырех сетевых соединений. Программное обеспечение Arduino включает обширную библиотеку и различные программы для сетевых коммуникаций.



Глава 5

Комплектующие изделия

Теперь, когда мы имеем начальные сведения о микроконтроллере, платформе Arduino, ее программировании и применении, пришла пора начать эксперименты с Arduino. Рассмотрим необходимые комплектующие.

5.1. Список основных комплектующих

- Плата микроконтроллера Arduino/Freeduino Duemilanove.
- Панель с контактными гнездами Tiny.
- Две кнопки для печатных плат RM 2,54.
- Датчик освещенности фоторезистор LDR A9060-13 ($R_{100} = 5 \text{ кОм}$).
- $n-p-n$ -транзистор BC548C.
- Кремниевый диод 1N4148.
- Акустический пьезопреобразователь.
- Светодиод красного цвета.
- Светодиод зеленого цвета.
- Два светодиода желтого цвета.
- Три резистора по 1,5 кОм.
- Резистор 4,7 кОм.
- Резистор 47 кОм.
- Резистор 10 кОм.
- Резистор 68 кОм.
- Подстроечный резистор 10 кОм.
- Конденсатор 1 мкФ.
- Моток монтажного провода.

5.2. Список деталей для дополнительных экспериментов

- Резистор 4,7 кОм.
- Датчик температуры LM75 в 8-выводном DIP-корпусе.
- Интерфейс порта I²C PCF8574.

- Ультразвуковой датчик расстояния производства фирмы Devantech SRF02 (подробности приведены по адресу: <http://www.Roboterteile.de>).
- Приемник GPS с TTL-выходом UART.
- Конструкторский набор стандартного сервопривода.
- Жидкокристаллический дисплей 20×4 (5 В).
- 12-контактный разъем RM2,54.

5.3. Экспериментальная плата Freeduino

Малогабаритная экспериментальная плата Freeduino функционально соответствует плате Arduino Duemilanove. Платформа Freeduino представляет собой версию бесплатного программного обеспечения, полностью совместимого с Arduino Duemilanove. Далее мы будем кратко называть их "микроконтроллер" и "плата микроконтроллера".

Подключение между персональным компьютером и микроконтроллером AVR осуществляется через порт USB. В персональный компьютер передаются данные из нашей Arduino-программы. Плата служит как база для экспериментов с уже имеющимся аппаратным оборудованием. Еще для экспериментов потребуется маленькая панель с контактными гнездами на макетной плате, в которую вставляются комплектующие изделия и с помощью которых можно выполнять соединения проводами.

В процессе экспериментов соединяют выбранные комплектующие и через USB-порт загружают подготовленные программы в микроконтроллер. Таким образом можно легко освоить назначение и свойства электронных компонентов и одновременно изучить основы метрологии и техники управления, а также программирование с Visual Basic Dot.Net и обращение с платформой Arduino.

5.4. Экспериментальная плата микроконтроллера Freeduino

Все подключения платы Freeduino (рис. 5.1) можно осуществить через разъемы. Светодиод "PWR" включения питания сигнализирует, что на плату микроконтроллера подается электропитание. Светодиод на плате Freeduino, который постоянно подключен к цифровому выводу 13 платы Arduino, обозначен символом "L". Провода и комплектующие изделия можно непосредственно вставлять в разъемы. Светодиоды с обозначением "TX" и "RX" показывают трафик последовательного интерфейса. В процессе обмена данными через UART-интерфейс светодиоды ритмично мигают.

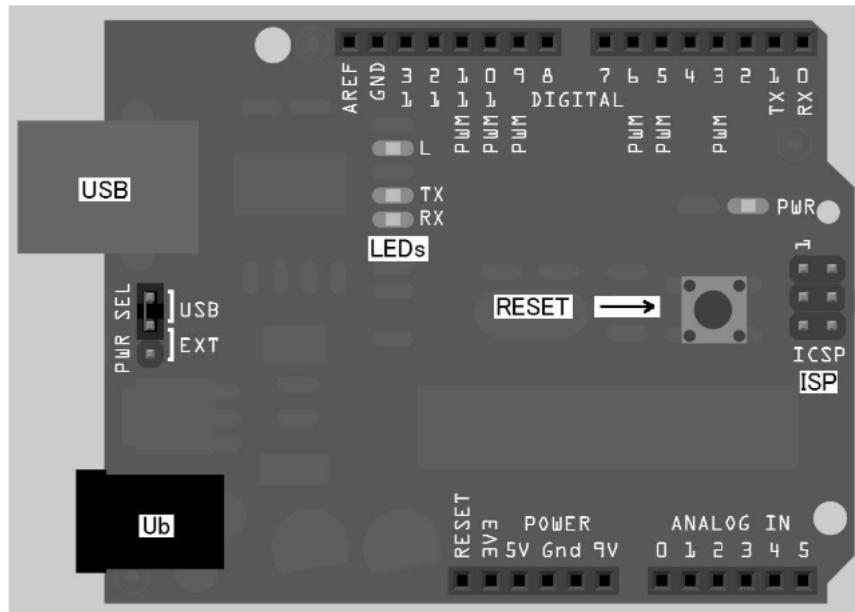


Рис. 5.1. Экспериментальная плата Freeduino

5.5. Электропитание

Электропитание может поступать через USB-разъем или штекер блока питания (расчитанный на ток примерно 500 мА). Выбор источника питания осуществляется установкой перемычки "PWR SEL" в положение "USB" или "EXT". При больших нагрузках платы микроконтроллера советуем не использовать питание от USB, т. к. возможно повреждение порта USB.

5.6. Кнопка Reset

Кнопка Reset (Сброс) выполняет перезагрузку системы. То же самое происходит при выключении и повторном включении питания.

5.7. ISP-подключение

Порт ISP служит для программирования микроконтроллера через ISP-программатор и для начальной загрузки. Для наших экспериментов этот порт не потребуется. Начальный загрузчик (Bootloader) уже установлен на предприятии-изготовителе.

СОВЕТ

При питании платы от USB подключайте Freeduino через USB-разветвитель. Если по ошибке при экспериментировании возникает короткое замыкание, выйдет из строя в большинстве случаев только USB-разветвитель, а не порт USB персонального компьютера.

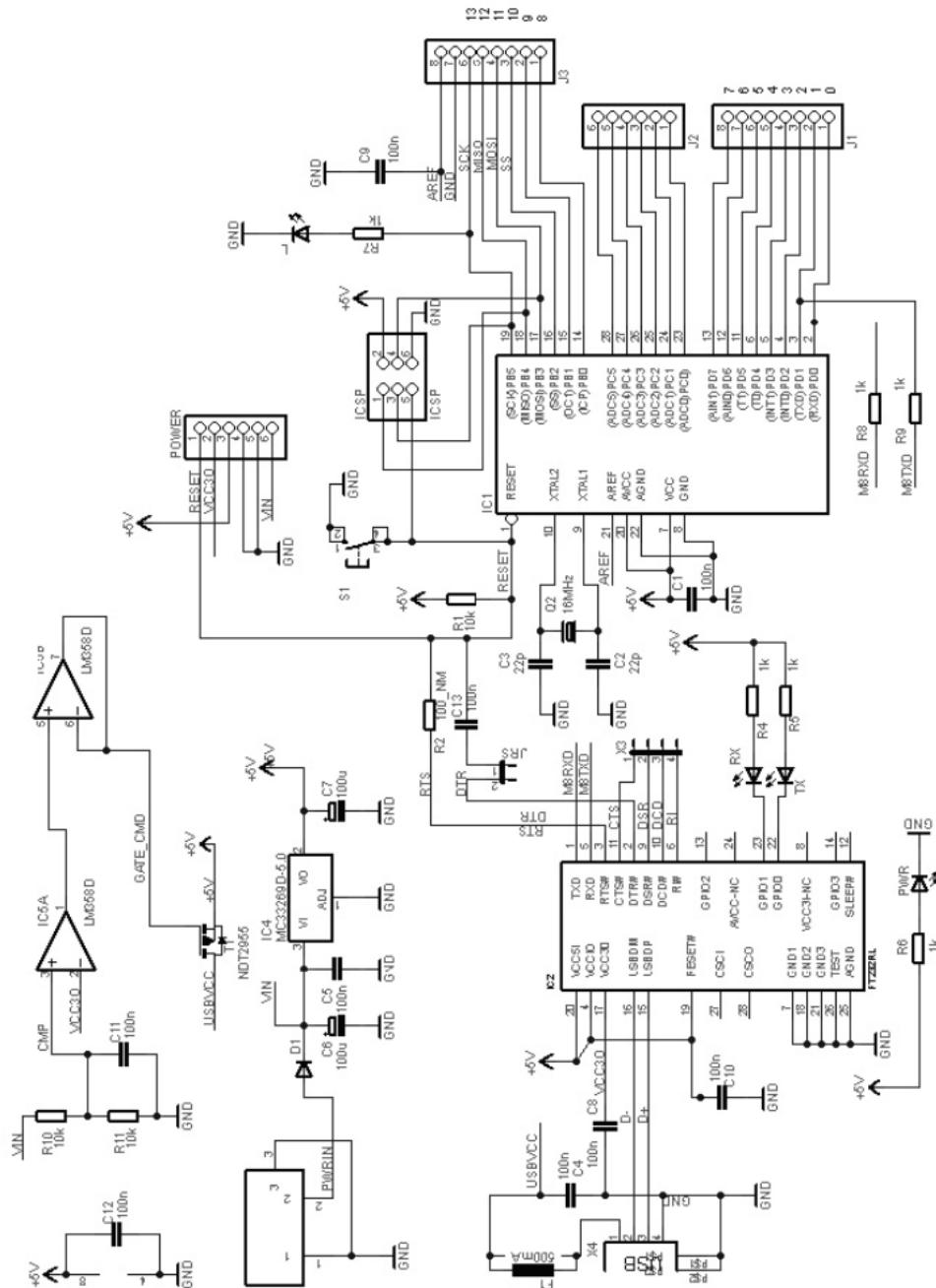


Рис. 5.2. Принципиальная схема экспериментальной платы Freeduino

Подробная принципиальная схема платы Freeduino приведена на рис. 5.2. При проведении экспериментов по ней вы сможете проверить правильность всех подключений.

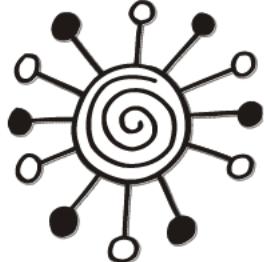
5.8. Замечания по технике безопасности

Плата Freeduino застрахована от ошибок, так что повреждение персонального компьютера едва ли возможно. Однако контакты USB-гнезда на нижней стороне платы не изолированы. Если плата окажется на металлической токопроводящей поверхности, то может произойти замыкание, ток возрастет и произойдет повреждение персонального компьютера и платы.

Согласно спецификации USB-порт нижнего уровня должен быть защищен от перегрузок по току, так что, собственно, ничего особо страшного не произойдет. Разумеется, защитная схема часто состоит лишь из низкоомного сопротивления, которое перегорает подобно плавкому предохранителю.

При экспериментах, пожалуйста, обратите внимание на следующие правила техники безопасности:

- Избегайте металлических предметов под платой или изолируйте всю нижнюю сторону при помощи непроводящей пластины или изоляционной ленты.
- Расположите блоки питания, другие источники напряжения и провода с напряжением более 5 В подальше от макетной платы.
- Присоединяйте плату по возможности не непосредственно к персональному компьютеру, а через разветвитель, который обычно снабжен дополнительной системой защиты.



Глава 6

Электронные компоненты и их свойства

Далее коротко опишем основные электронные компоненты и их назначение. Но получить практические навыки вам помогут только реальные эксперименты с электронными схемами.

6.1. Светодиоды

Выпускают красные, желтые и зеленые светодиоды. При подключении важно обращать внимание на полярность. Минусовой (более короткий) вывод называется катодом. Положительный (более длинный) вывод — это анод (рис. 6.1). Внутри прозрачного корпуса светодиода видна структура, подобная чаше. Это держатель кристалла светодиода, который соединен с катодом. Выход анода соединен с верхним контактом кристалла очень тонкой маленькой проволокой. Всегда обращайте внимание на полярность включения светодиода и никогда не подключайте прибор непосредственно к батарее или к контактам питания USB. Это может вывести светодиод из строя и даже повредить предохранительное сопротивление на экспериментальной плате.

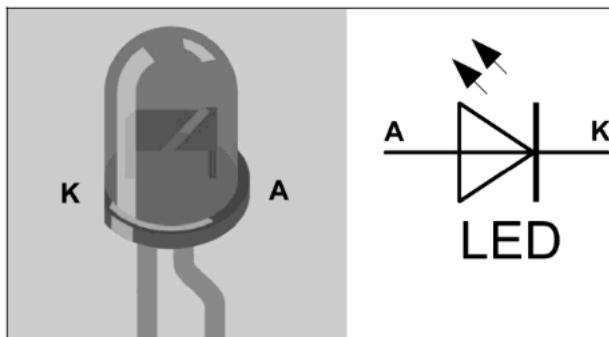


Рис. 6.1. Светодиод и его условное графическое обозначение

6.2. Резисторы

Для экспериментов подойдут пленочные резисторы с допуском $\pm 5\%$. Токопроводящий материал резистора нанесен на керамический сердечник и покрыт защитным лаком. Номинал и допуск резистора обозначают кодом из цветных колец, нанесенных на корпус (рис. 6.2).

Значения сопротивления резисторов с допуском $\pm 5\%$ принадлежат ряду Е24, причем каждая декада содержит 24 равноотстоящих значения.

Вот значения нормального ряда Е24:

1,0 / 1,1 / 1,2 / 1,3 / 1,5 / 1,6 / 1,8 / 2,0 / 2,2 / 2,4 / 2,7 / 3,0 / 3,3 / 3,6 / 3,9 / 4,3 / 4,7 / 5,1 / 5,6 / 6,2 / 6,8 / 7,5 / 8,2 / 9,1

Цветной код читается, начиная с кольца, которое расположено ближе к краю резистора (рис. 6.3). Первые два кольца соответствуют двум цифрам, третье кольцо — это множитель значения сопротивления в омах. Четвертое кольцо указывает допуск. Расшифровка обозначений приведена в табл. 6.1.

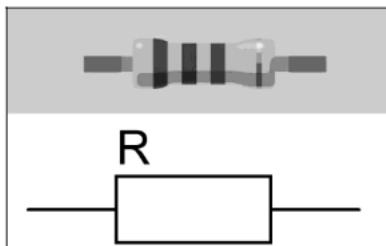


Рис. 6.2. Резистор и его условное графическое обозначение

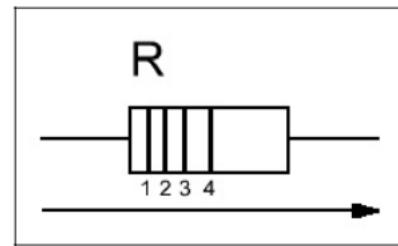


Рис. 6.3. Так определяется значение сопротивления резистора

Таблица 6.1. Цветовая маркировка резисторов

Цвет	Кольцо 1	Кольцо 2	Кольцо 3 (множитель)	Кольцо 4 (допуск)
Серебристый	—	—	$1 \times 10^{-2} = 0,01 \text{ Ом}$	$\pm 10\%$
Золотистый	—	—	$1 \times 10^{-1} = 0,1 \text{ Ом}$	$\pm 5\%$
Черный	0	0	$1 \times 10^0 = 1 \text{ Ом}$	—
Коричневый	1	1	$1 \times 10^1 = 10 \text{ Ом}$	$\pm 1\%$
Красный	2	2	$1 \times 10^2 = 100 \text{ Ом}$	$\pm 2\%$
Оранжевый	3	3	$1 \times 10^3 = 1 \text{ кОм}$	—
Желтый	4	4	$1 \times 10^4 = 10 \text{ кОм}$	—
Зеленый	5	5	$1 \times 10^5 = 100 \text{ Ом}$	$\pm 0,5\%$
Голубой	6	6	$1 \times 10^6 = 1 \text{ МОм}$	$\pm 0,25\%$
Фиолетовый	7	7	$1 \times 10^7 = 10 \text{ МОм}$	$\pm 0,1\%$
Серый	8	8	$1 \times 10^8 = 100 \text{ МОм}$	—
Белый	9	9	$1 \times 10^9 = 1000 \text{ МОм}$	—

Например, резистор с цветными кольцами желтого, фиолетового, коричневого и золотистого цвета имеет значение 470 Ом при допуске $\pm 5\%$. Пользуясь приведенными сведениями, попробуйте самостоятельно расшифровать сопротивление какого-нибудь резистора.

6.3. Конденсаторы

Конденсатор состоит из двух металлических пластин, находящихся друг против друга, и изолирующего слоя между ними. Если подается электрическое напряжение, то между пластинами конденсатора образуется электрическое силовое поле, в котором сохраняется энергия. Чем больше площадь пластин, тем больше энергии может сохранять конденсатор. Емкость конденсатора указывается и измеряется в фарадах (Φ). В нашей книге, да и вообще в электронике, емкость конденсаторов обычно находится в диапазоне от 10 н Φ (0,00000001 Φ) до 1 000 мк Φ (0,001 Φ). Изоляционный материал (диэлектрик) увеличивает емкость по сравнению с воздушной изоляцией. В керамических дисковых конденсаторах применяется специальный материал, который обеспечивает большую емкость при маленьких габаритах.

На рис. 6.4 изображен электролитический конденсатор, который также встречается в нашей книге. При подключении таких конденсаторов нужно обращать внимание на полярность, т. к. ошибка в полярности может привести к взрыву. На схеме отрицательный полюс обозначен зачерненным прямоугольником, положительный — светлым. На корпусе электролитического конденсатора рядом с отрицательным полюсом нанесена белая полоска. Выводы электролитического конденсатора, как и светодиода, имеют разную длину: положительный более длинный, отрицательный более короткий.

Керамические конденсаторы не имеют полярности. На принципиальной схеме эти конденсаторы изображаются двумя параллельными черными полосками без указания полярности (рис. 6.5).

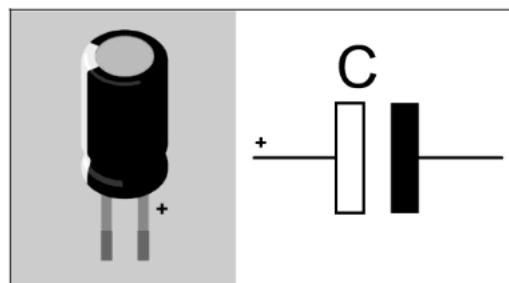


Рис. 6.4. Электролитический конденсатор и его условное графическое обозначение

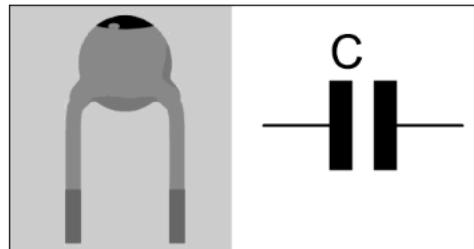


Рис. 6.5. Керамический конденсатор и его условное обозначение

ПРИМЕЧАНИЕ

Для наших экспериментов керамические конденсаторы не потребуются. Информация о них приведена только для сведения.

6.4. Транзисторы

Транзисторы (рис. 6.6) предназначены для усиления токов. Подача небольшого тока в базу приводит к протеканию большого тока коллектора. Эти приборы имеют три вывода: база (Basis, B), коллектор (Collector, C) и эмиттер (Emitter, E).

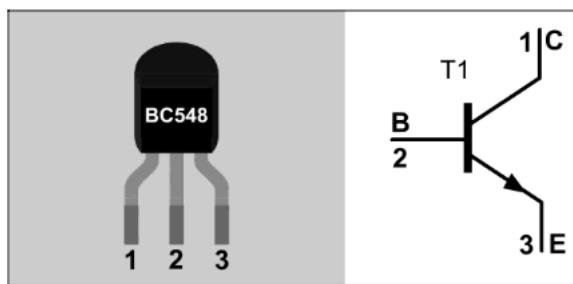


Рис. 6.6. Транзистор структуры n - p - n и его условное графическое обозначение

В наших экспериментах используется транзистор BC548С. Это универсальный маломощный транзистор.

6.5. Диод

Диод — это электронный вентиль, который пропускает ток только в одном направлении. Диоды изготавливают из германия (Ge) или кремния (Si). В наших экспериментах применяются кремниевые диоды типа 1N4148. Это популярные кремниевые диоды, которые выдерживают ток до 100 мА. Включая диод, нужно обращать внимание на полярность. Отрицательный полюс (катод) обозначен маленьким кольцом на краю корпуса (рис. 6.7).

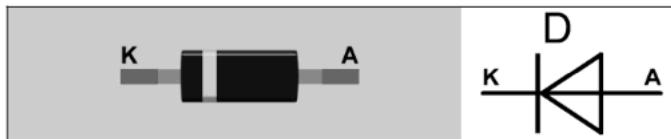


Рис. 6.7. Диод и его условное графическое обозначение

6.6. Акустический пьезопреобразователь ("пищалка")

Акустический пьезопреобразователь (рис. 6.8) может работать как простой маленький динамик, датчик или микрофон. Структура пьезопреобразователя похожа на дисковый керамический конденсатор, но диэлектрик дополнительно электрически заряжен. Вследствие этого возникает взаимосвязь между механическим и электрическим напряжением. Пьезоэлектрическим эффектом обладают кристаллы кварца. Еще пример — электрическая зажигалка, однако в ней генерируемое напряжение значительно выше, чем в нашем пьезоакустическом преобразователе.

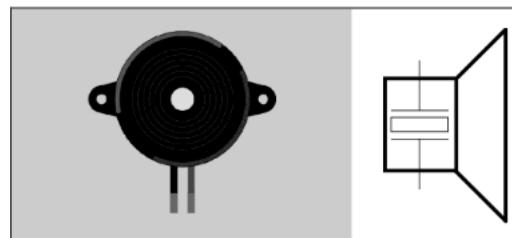


Рис. 6.8. Пьезоакустический преобразователь и его условное обозначение

6.7. Монтажный провод

При помощи провода выполняются все подключения на нашей монтажной плате. Нарежьте маленькими кусачками-бокорезами или ножницами отрезки подходящей длины и снимите изоляцию с обоих концов. Не выбрасывайте использованные провода, они еще пригодятся.

6.8. Кнопка

Кнопка замыкает или размыкает электрическую цепь. В отличие от переключателя после отпускания она не остается в конечном положении, а возвращается в свое первоначальное положение.

Наша кнопка имеет четыре вывода, причем два из них всегда соединены друг с другом (рис. 6.9).

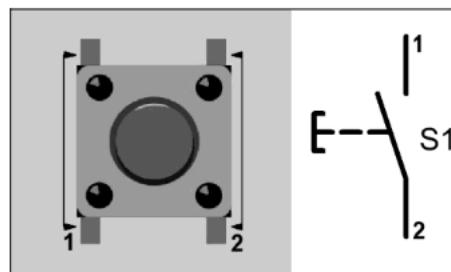


Рис. 6.9. Кнопка

6.9. Потенциометр

Большинство резисторов, позволяющих регулировать сопротивление с помощью ручки, называют потенциометрами. Потенциометр имеет, как правило, три вывода: конечные контакты резистивного слоя и скользящий контакт ползунка (обычно это средний вывод). Миниатюрный потенциометр с регулировкой "под шлиц" (подстроочный резистор) изображен на рис. 6.10.

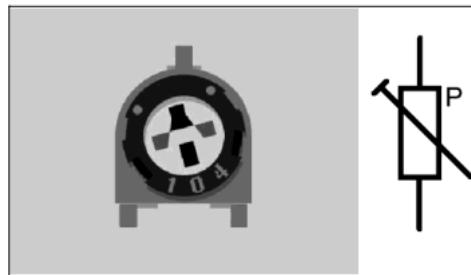


Рис. 6.10. Подстроочный резистор

6.10. Фоторезистор

Резистор, который, реагирует на свет, называется фоторезистором (LDR — Light Dependent Resistor). Фоторезистор состоит из двух медных дорожек, нанесенных на изолированной подложке (белого цвета). Между ними находится полупроводниковый материал в форме извилистой ленты (красного цвета). Если свет (фотоны) падает на светочувствительный полупроводниковый материал, в нем образуются электронно-дырочные пары. В результате проводимость фоторезистора увеличивается, а его сопротивление снижается. Чем больше света попадает на фоторезистор, тем меньше его сопротивление и тем будет больше электрический ток через него. Однако, быстродействие фоторезистора небольшое, переходный процесс продолжается несколько миллисекунд. На рис. 6.11 приведен внешний вид и условное обозначение фоторезистора.

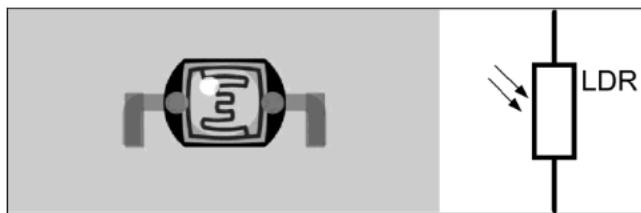


Рис. 6.11. Фоторезистор

6.11. Монтажная панель с контактными гнездами

На монтажной панели с контактными гнездами можно запросто собрать наши схемы без пайки. Контакты обозначены буквами от А до Е и от F до J (рис. 6.12). Наша панель с контактными гнездами состоит из 20 столбцов и 10 рядов (от А до J). Полезно немного обрезать выводы компонента (3 мм) наискось так, чтобы получилось что-то вроде клина. Тогда комплектующие изделия легче вставить в контактные гнезда панели. Если все же возникают сложности со вставкой выводов, то для установки компонента лучше всего воспользоваться маленькими плоскогубцами.

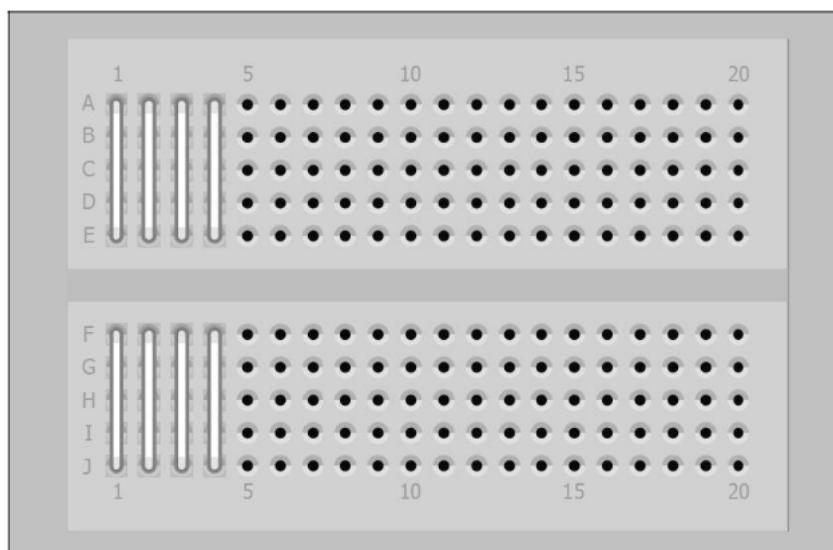
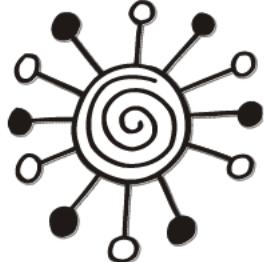


Рис. 6.12. Монтажная панель с контактными гнездами Tiny



Глава 7

Предварительная подготовка

Прежде чем начать экспериментировать и программировать, потребуется некоторая предварительная подготовка. На персональном компьютере нужно установить драйвер для виртуального СОМ-порта (последовательный интерфейс) и среду программирования/разработки для Arduino.

7.1. Установка драйвера

Сначала нужно установить драйвер для микросхемы FT232RL — преобразователя USB/UART фирмы FTDI. Микроконтроллер на экспериментальной плате поставляется с завода с установленным начальным загрузчиком, необходимое сопряжение между интерфейсами USB и UART платы Freeduino выполняет микросхема FT232RL. Поэтому плата Freeduino в менеджере устройств обозначена как виртуальный СОМ-порт (виртуальный последовательный интерфейс).

Возможно, драйвер для FT232RL уже установлен на вашем персональном компьютере. Однако может оказаться, что версия драйвера устарела и не поддерживает все функции чипа. Требуемая версия драйвера устанавливается автоматически.

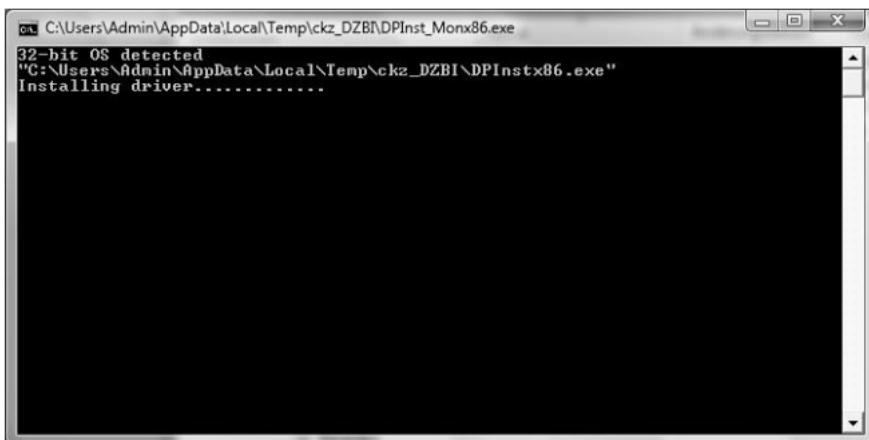


Рис. 7.1. Окно после запуска файла CDMxxx.exe

Выньте все адAPTERы USB/Seriell из разъема USB. Запустите далее автоматическую установку файла CDM20600.exe из папки Software\USB FTDI-Treiber на прилагаемом к книге компакт-диске. Программа удаляет более поздний FTDI-драйвер и инсталлирует новый (рис. 7.1). При необходимости наличие самой последней версии драйвера можно проверить по адресу: www.ftdichip.com. При присоединении USB-платы определяются новые аппаратные средства, и драйвер загружается без дальнейших действий пользователя (рис. 7.2 и 7.3).



Рис. 7.2. Windows обнаружила новое устройство после подключения USB-платы



Рис. 7.3. Сообщение ОС Windows после успешной установки драйверов

Теперь на персональном компьютере существует новый последовательный интерфейс, например, COM2, COM3 и т. д. Узнать номер можно в менеджере аппаратных средств. При повторной установке адAPTERа USB/seriell ОС Windows предоставит максимальный номер COM-порта. В этом случае новое устройство может называться, например, COM35.

7.2. Вспомогательная программа MProg для FT232RL

USB-чип обладает многочисленными настройками, которые пользователь может изменить. Для этого предназначена программа MProg, которая имеется в каталоге SoftwareWProg на прилагаемом к книге компакт-диске. Программа MProg на персональном компьютере устанавливается двойным нажатием на файле MProg3.0_Setup.exe. Появится ряд диалоговых окон, изображенных на рис. 7.4–7.9.



Рис. 7.4. Стартовое окно установки программы MProg.
Для продолжения нажмите кнопку Weiter (Далее)



Рис. 7.5. Теперь можно выбрать язык интерфейса программы

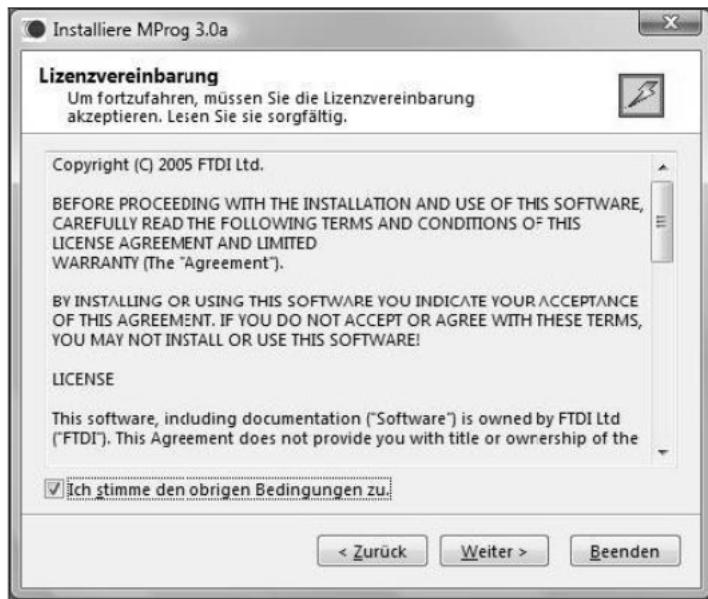


Рис. 7.6. Лицензионное соглашение



Рис. 7.7. В этом диалоговом окне можно свободно выбрать целевую папку

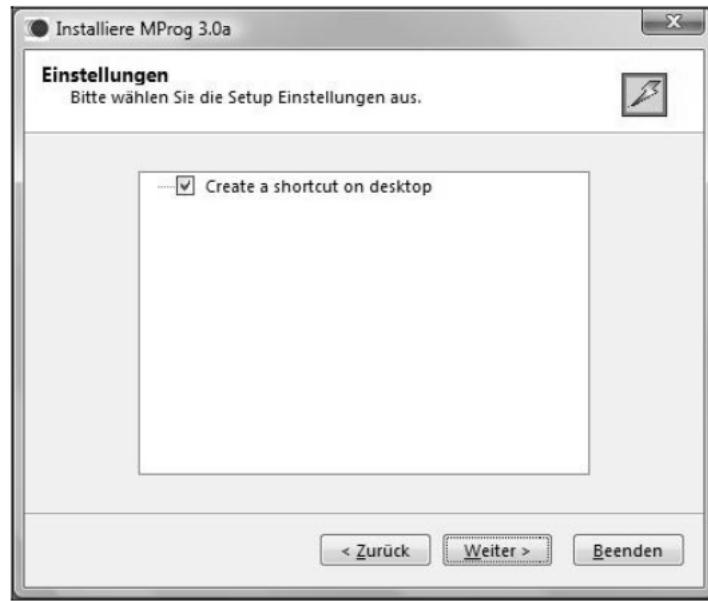


Рис. 7.8. Если нужен ярлык на рабочем столе, установите флажок и нажмите кнопку Weiter (Далее)



Рис. 7.9. Программа успешно установлена на вашем компьютере

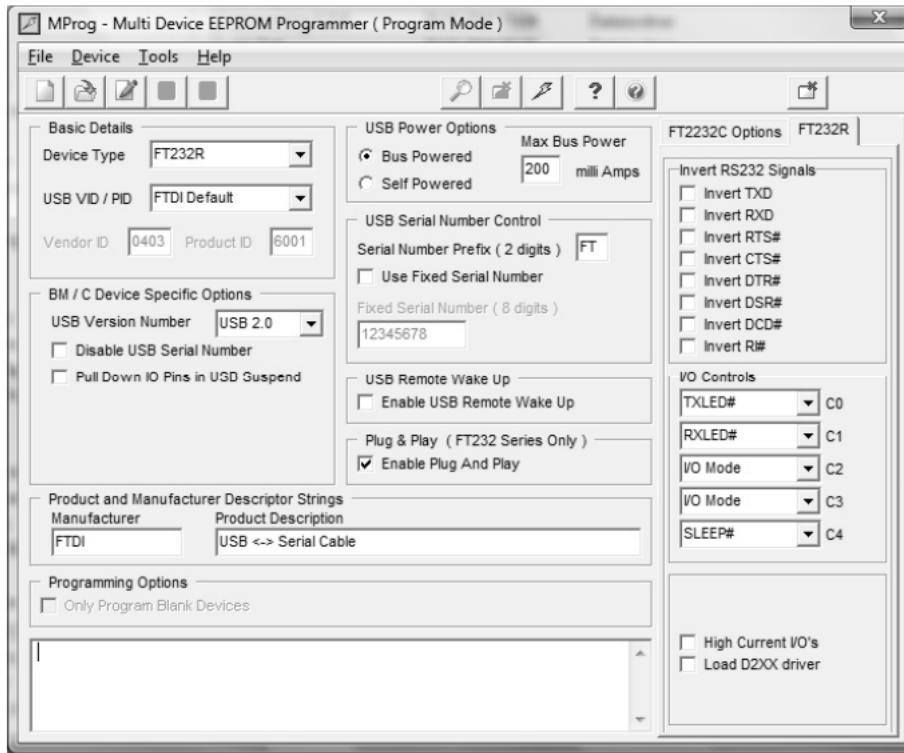


Рис. 7.10. Установки в программе MProg

После завершения инсталляции присоедините плату к USB-порту и запустите программу MProg (рис. 7.10). Нажмите кнопку **File | New** (Файл | Новый) и после этого **Tools | Scan** (Инструменты | Просмотр). Далее выберите опцию **FT232R** в списке **Device Type** (Тип устройства). Откройте файл **Freeduino.ept**, который находится в папке **MProg Files**. Далее выполните команду меню **Device | Program** (Устройство | Программа).

ВНИМАНИЕ!

Убедитесь в том, что никакие посторонние устройства не подключены к USB-порту персонального компьютера.

7.3. Программирование микросхемы FT232R с помощью MProg

После программирования плата появляется снова как новое устройство USB. Выньте и снова вставьте USB-штекер. ОС Windows определяет новое устройство и автоматически устанавливает имеющийся драйвер. Микросхема FT232RL получает в этом случае новый номер COM-порта. В других операционных системах установка происходит аналогично.

Выберите в меню **File** (Файл) пункт **New** (Новый). Вот типичные установки MProg:

- Device Type (Тип устройства) — FT232R.
- USB Power Option (Выбор питания от USB-шины) — Bus Powered (Питание от USB-шины).
- Max Bus Power — 200 мА.
- Invert RS-232 Signals (Сигналы инвертирования RS-232) — флажок отсутствует.
- C0 — TXLED#.
- C1 — TXLED#.
- C2 и C3 — I/O Mode.
- C4 — (SLEEP#).

Сохраните все установки. Теперь можно запустить команду меню **Device | Programm** (Устройство | Программа). Заново отсоедините и присоедините USB-плату. Все готово!

Современные ПК еще зачастую имеют последовательный интерфейс, однако у многих компьютеров он отсутствует. При подключении чипа FT232R персональный компьютер получает дополнительный COM-интерфейс. ОС Windows не делает различий между физическим и виртуальным COM-интерфейсом. Ваш новый интерфейс получает следующее свободное имя, например, COM2. Однако может быть назначен и больший номер COM, если уже раньше было установлено несколько других адаптеров USB/seriell. Благодаря перестановке при помощи программы MProg микросхема получит новый внутренний номер устройства и определится как новое устройство с новым условным номером COM, например, COM3.

Следует отметить, что не каждое программное обеспечение может работать с COM-портами выше COM9. С другой стороны, номера меньше COM10, возможно, раньше были заняты установленными устройствами, которые в настоящее время не используются. Поэтому имеет смысл принудительно определить USB-плату, например, как COM2.

7.4. Установка программного обеспечения Arduino

Теперь установим среду программирования Arduino, которая базируется на программе Processing и имеется на прилагаемом к книге компакт-диске. Processing — это простой подъязык программирования С. Он разработан специально для художника, дизайнера и пользователя, который не так глубоко владеет программированием, но, тем не менее, нуждается в собственной маленькой программе. Дополнительную информацию по этому вопросу можно найти по следующим ссылкам: <http://processing.org/> и www.arduino.cc.

Скопируйте папку Arduino-xxxx, которая находится на прилагаемом компакт-диске в папке Software\Entwicklungsumgebung\Windows на ваш компьютер в желаемый каталог, например, E:\ARDUINO. В этой папке можно создавать другие папки для сохранения в них позже собственных программ.

Скопируете в отдельную папку на жесткий диск примеры к этой книге, чтобы не держать компакт-диск постоянно в дисководе. В данном случае был выбран диск E:, хотя он может быть любым другим. В нашем случае готовый каталог будет выглядеть приблизительно так:

- E:\ARDUINO\Arduino-xxxx\ — здесь находится среда разработки.
- E:\ARDUINO\Meine Programme\ — здесь находятся ваши программы.
- E:\ARDUINO\Franzis\ — здесь находятся примеры из книги.

Для пользователя Macintosh и Linux файлы размещаются в соответствующей папке.

СОВЕТ

Не рекомендуется размещать программы на системном диске C:. При переустановке ОС Windows ваши программы Arduino также могут пропасть.



Глава 8

Среда разработки Arduino

После копирования программы Arduino из папки Entwicklungsumgebung\ с прилагаемого компакт-диска на свой компьютер, можно запустить файл Arduino.exe. Перейдите в каталог Arduino-xxxx. Запустите программу, дважды щелкнув мышью файл с именем Arduino.exe. Для удобства вызова программы можно создать ярлык на рабочем столе.

После запуска программы в течение нескольких секунд появляется стартовое окно, показанное на рис. 8.1.

В Arduino-IDE (Integrated Development Environment) — встроенной среде разработки находятся различные инструменты и настройки, которые облегчают общение с программой Arduino (рис. 8.2).

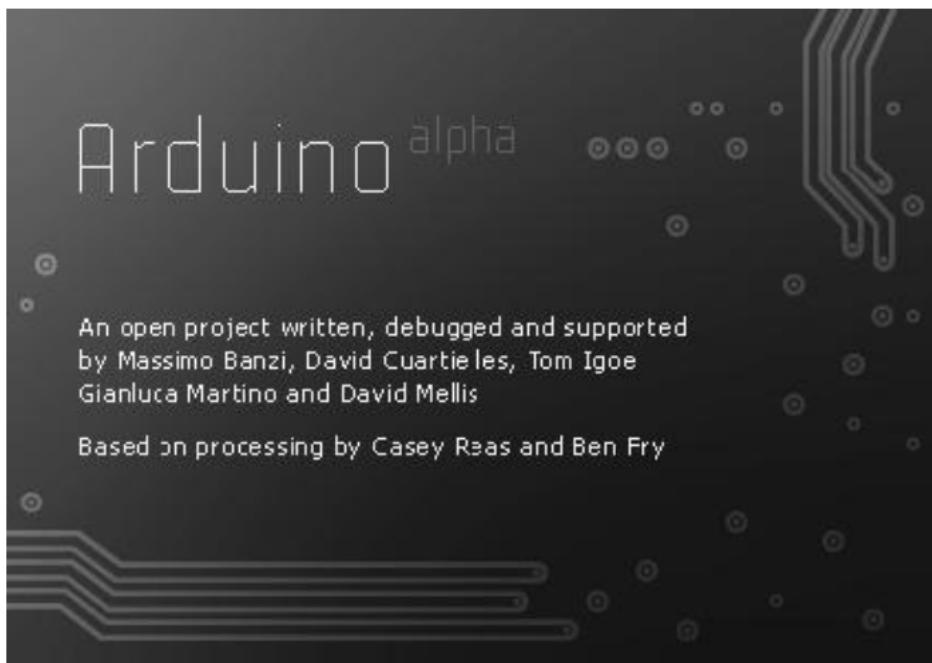


Рис. 8.1. Окно программы Arduino после запуска файла Arduino.exe

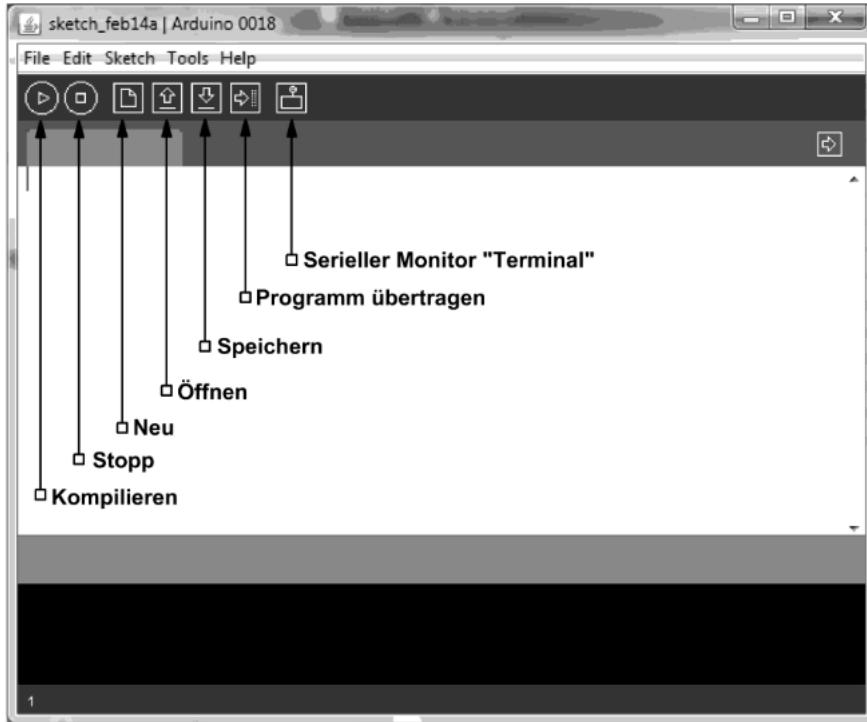


Рис. 8.2. Среда разработки Arduino

Рассмотрим подробнее среду разработки Arduino. Из меню можно вызвать все функции Arduino. Далее мы еще будем знакомиться с отдельными возможностями. Под главным меню находится панель инструментов, где расположены следующие команды:

- Kompilieren (Компиляция) — формирование файла, который будет передан в плату микроконтроллера.
- Stopp (Стоп) — отмена компиляции.
- Neu (Новый) — создание нового файла Arduino.
- Öffnen (Открыть) — открытие текста программы.
- Speichern (Сохранить) — сохранение текста программы.
- Programm übertragen (Транслировать программу) — передача программы в плату микроконтроллера.
- Terminal (Терминал) — открытие встроенного ASCII-терминала.

8.1. Установки в Arduino-IDE

До начала работы необходимо задать исходные установки. Для этого требуется выбрать нужную плату Arduino (Freeduino) и используемый интерфейс (рис. 8.3 и 8.4).

В данном случае указана плата Arduino Diecimila. Если требуется другая, то следует выбрать соответствующую плату из списка.

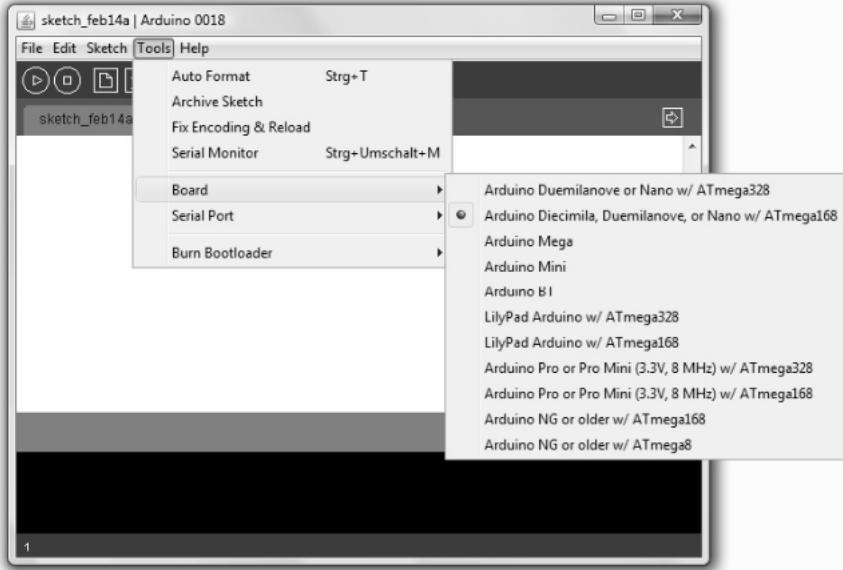


Рис. 8.3. Выбор платы микроконтроллера

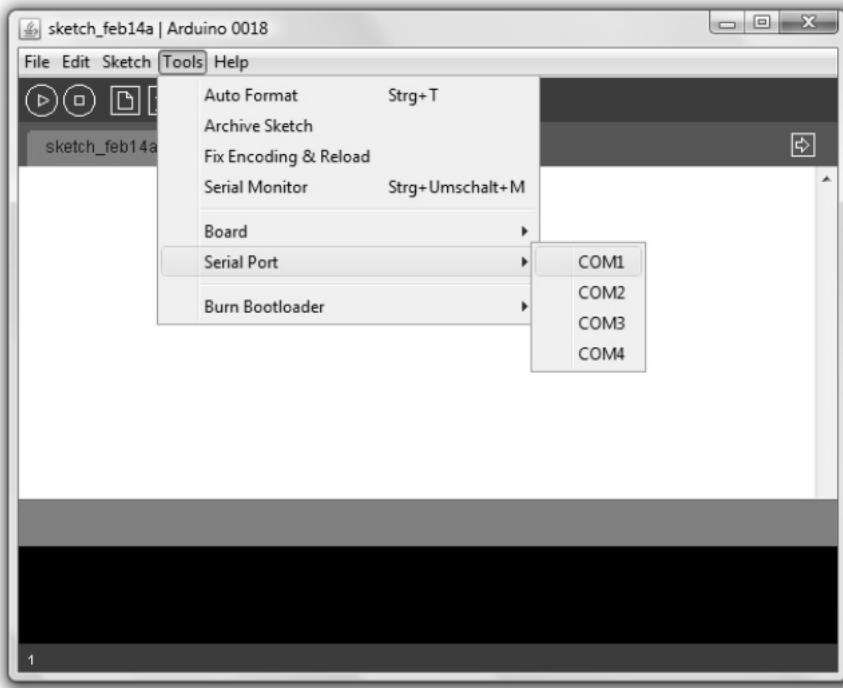


Рис. 8.4. Выбор последовательного интерфейса

Перед тем как открыть среду разработки (IDE) необходимо присоединить плату микроконтроллера к персональному компьютеру и установить драйверы надлежащим образом. Иначе в списке не появится COM-порт. Если имеется несколько альтернативных предложений, удостоверьтесь в менеджере устройств, какой COM-порт назначен плате микроконтроллера (рис. 8.4). В крайнем случае, отсоедините плату и снова присоедините ее. Потом посмотрите, какой COM-порт исчез из менеджера устройств и какой был снова зарегистрирован после присоединения.

В заключение мы устанавливаем еще скорость в бодах в терминале. Для этого нужно нажать пиктограмму **Terminal** (Терминал) и задать скорость 9 600 бод. Эту скорость мы используем в книге для большинства программ.

8.2. Наша первая программа "ES_Blinkt"

Теперь для проверки наших аппаратных средств и функционального контроля настроек Arduino, напишем первую программу при помощи Arduino. Программа будет очень маленькая и простая, чтобы исключить ошибки. Сначала закройте все окна редактирования, которые были открыты при ознакомлении со средой разработки. Далее в меню **File** (Файл) выберите команду **New** (Новый), появится новое окно редактирования. Наберите в нем код листинга 8.1 и сохраните его в папке на жестком диске. Лучше всего создать специальную папку, например, *Meine Programme Arduino* (Мои программы Arduino) и указать ее в настройках как папку для хранения программ. При указании, например, "ES_Blinkt" в качестве имени программы, среда разработки Arduino создает папку в выбранном каталоге с именем ES_Blinkt и сохраняет в ней файл *.pde (код программы) под именем ES_Blinkt.pde.

Листинг 8.1. ES_Blinkt.pde

```
/*
Световая сигнализация

Используется светодиод "L" на плате Arduino / Freeduino

*/
int ledPin = 13;      // Светодиод присоединяется к цифровому выводу 13

// Стандартная программа конфигурирует наш цифровой порт
// Эта программа исполняется только один раз при запуске!
void setup()
{
    // Порт конфигурируется как выход
    pinMode(ledPin, OUTPUT);
```

```
}

// Основная программа – это бесконечный цикл
void loop()
{
    digitalWrite(ledPin, HIGH);      // Включение светодиода
    delay(1000);                  // Ожидание 1 секунда
    digitalWrite(ledPin, LOW);      // Выключение светодиода
    delay(1000);                  // Ожидание 1 секунда
}
```

Теперь можно проверить программу на наличие синтаксических ошибок, нажав в панели инструментов круглую кнопку со стрелкой вправо (см. рис. 8.2) или с помощью комбинации клавиш <Ctrl>+<R>.

При отсутствии сообщения об ошибке (рис. 8.5), можно передать программу в плату микроконтроллера, нажав на панели инструментов (см. рис. 8.2) квадратную кнопку со стрелкой вправо с маленькими точками (::) или с помощью комбинации клавиш <Ctrl>+<U>.

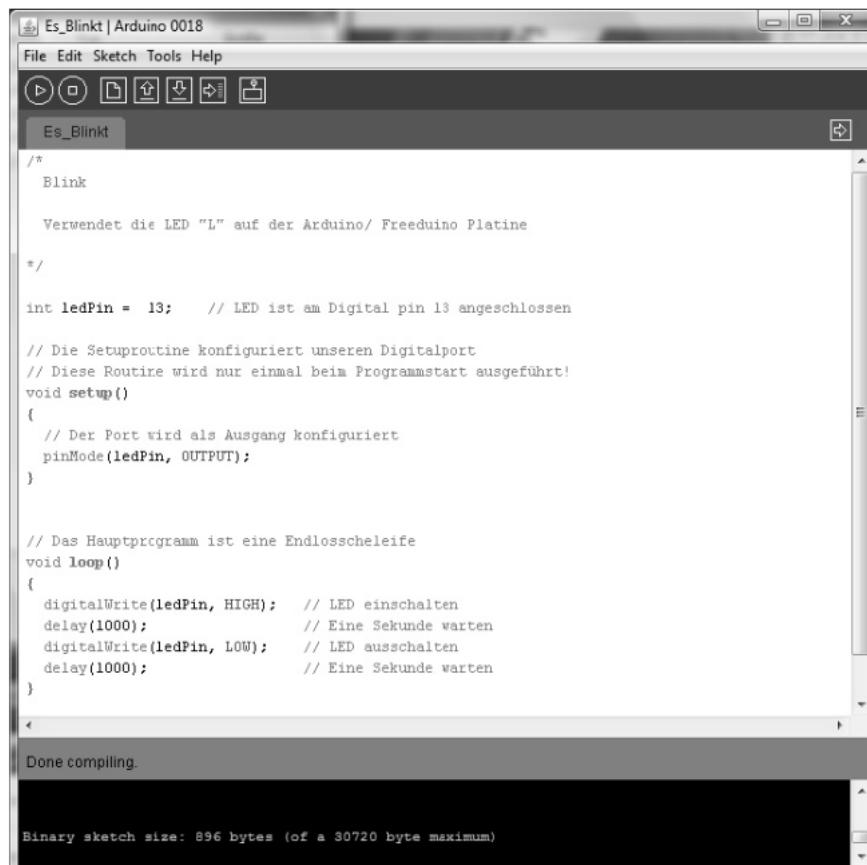


Рис. 8.5. Программа скомпилировалась правильно

Вскоре после загрузки программы должен начать мигать светодиод "L" на плате микроконтроллера.

СОВЕТ

Если начальный загрузчик сообщает об ошибке, то отсоединение и повторное соединение макетной платы с персональным компьютером поможет устранить ошибку.

8.3. Что мы сделали?

Программа Arduino помогла существенно уменьшить объем работ. Рассмотрим подробнее нашу первую программу.

Вначале следуют строки комментариев:

```
// Blink  
// Используется светодиод L на плате Arduino /Freeduino
```

Далее видим, что переменная с именем `ledPin`, определяющая вывод, к которому присоединяется наш светодиод L, содержит число "13":

```
int ledPin = 13;
```

Программа `void setup()` вызывается в начале программы однократно и инициализирует переменные, порты и т. д. при запуске программы:

```
void setup()  
{  
    // Порт конфигурируется как выход  
    pinMode(ledPin, OUTPUT);  
}
```

При помощи команды `pinMode` вывод 13 (установлен в переменной `ledPin`) конфигурируется как выход `OUTPUT`. Команда `pinMode` служит для настройки цифровых портов.

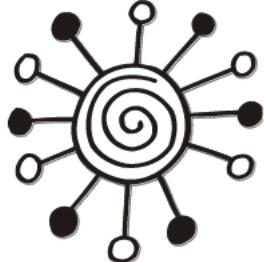
После программы установки `void setup()` следует программа `void loop()`, которая вызывается непосредственно после `void setup()` и всегда запускается как бесконечный цикл. Светодиод включается при помощи команды `digitalWrite`. Переменные в скобках указывают, что речь идет о выводе 13 и состояние вывода должно быть включено на высокий уровень. Можно было написать также число "13" вместо переменной `ledPin`, но тогда ухудшилась бы читабельность программы. Лучше давать уникальное имя portам или переменным. Следующая команда — это пауза `delay`, переменная в скобках указывает время ожидания в миллисекундах. После паузы светодиод снова отключается при помощи функции `digitalWrite` и константы `LOW`. Чтобы продолжительность включения и выключения была одинаковой, снова задана задержка на 1 000 мс. Цикл непрерывно повторяется (листинг 8.2).

Листинг 8.2. Бесконечный цикл

```
void loop()
{
    digitalWrite(ledPin, HIGH); // Включение светодиода
    delay(1000);             // Ожидание 1 секунда
    digitalWrite(ledPin, LOW); // Выключение светодиода
    delay(1000);             // Ожидание 1 секунда
}
```

ПРИМЕЧАНИЕ

Язык программирования Arduino-C поддерживает принцип абстрагирования от аппаратных средств (Hardware-Abstraction-Layer), что сильно облегчает программирование, т. к. не нужно изучать толстые технические описания микроконтроллеров. Непосредственную инициализацию аппаратных средств выполняет компилятор Arduino. Достаточно написать, как в нашем примере, pinMode, чтобы определить вывод как выход или вход без подробного изучения документации микроконтроллера.



Глава 9

Основы программирования Arduino

Для всех, кого пугают сложные конструкции языков программирования, здесь приведено небольшое вступление в основы программирования на языке Arduino C. Вы скоро увидите, как можно просто и быстро написать собственные программы с небольшим числом команд. Все последующие проекты с платформой Arduino основаны на сведениях, приведенных в этой главе.

9.1. Биты и байты

На данный момент самая маленькая единица измерения информации в вычислительной системе — это бит. На английском языке двоичные числа обозначаются как *binary digit* (двоичная цифра). Так возникло искусственно образованное слово "бит" (Bit). Один бит может принимать только два значения: 0 или 1 (наличие или отсутствие электрического тока). Информация может представляться как последовательность из определенного числа битов.

В нашем случае 1 байт — это последовательность из 8 битов. Следовательно, один байт может представляться и интерпретироваться как 256 различных комбинаций. Половину байта, т. е. 4 бита, называют тетрадой или полубайтом:

4 бита = 1 полубайт = 1 тетрада

8 битов = 2 полубайта = 1 байт

Коэффициент пересчета между двоичными единицами измерения количества информации равен 1 024 (табл. 9.1). Исключение — преобразование бита в байт, т. к. при этом коэффициент пересчета равен 8. Эти некруглые числа возникают из-за того, что в информатике вычисления проводятся по основанию "2" (2^n).

Таблица 9.1. Соотношение между единицами измерения количества информации

Наименование	Коэффициент пересчета
1 байт	8 бит
1 килобайт (1 Кбайт)	1 024 байт (2^{10} байт)
1 мегабайт (1 Мбайт)	1 024 Кбайт (2^{10} Кбайт)
1 гигабайт (1 Гбайт)	1 024 Мбайт (2^{10} Мбайт)
1 терабайт (1 Тбайт)	1 024 Гбайт (2^{10} Гбайт)

9.2. Базовая структура программы

Структура большинства программ очень сходна, поскольку широко используется процедурное программирование. Процедурное программирование состоит в разбиении компьютерных программ на небольшие частные задачи, которые обозначаются как процедуры (подпрограммы). Самый маленький и неделимый шаг при этом методе — это команда (оператор, инструкция). Программа выполняется последовательно от команды к команде. Сам термин "процедура" означает "продвигаться" и происходит от латинского слова "procedo". Программист приказывает микроконтроллеру при помощи программы, в какой последовательности нужно действовать. Цель этого принципа программирования — простота разработки и возможность повторного использования фрагментов (модулей) исходного кода.

9.2.1. Последовательное выполнение программы

При последовательном программировании код, состоящий из отдельных процедур, выполняется в цикле. На рис. 9.1 показано выполнение последовательности команд "ввод данных — обработка — вывод данных".

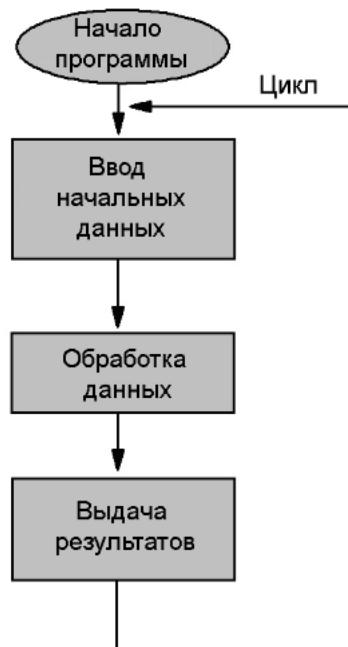


Рис. 9.1. Последовательное выполнение программы

9.2.2. Прерывание выполнения программы

Иногда возникает необходимость прервать последовательное выполнение программы. Это выполняется с помощью прерываний (Interrupt). Основная часть программы выполняется, как и при последовательном программировании — бесконечный цикл, в котором повторяются отдельные процедуры. Как только произойдет внешнее или внутреннее прерывание, например, нажатие кнопки, основной цикл (Main-Loop) прервется и произойдет переход в программу обработки прерываний (Interrupt-Routine). В ней отрабатываются специальные задачи, например, аварийное отключение и т. п. Далее работа продолжается снова в основном цикле, где могут выполняться вспомогательные операции. Выполнение прерываний иллюстрирует рис. 9.2.

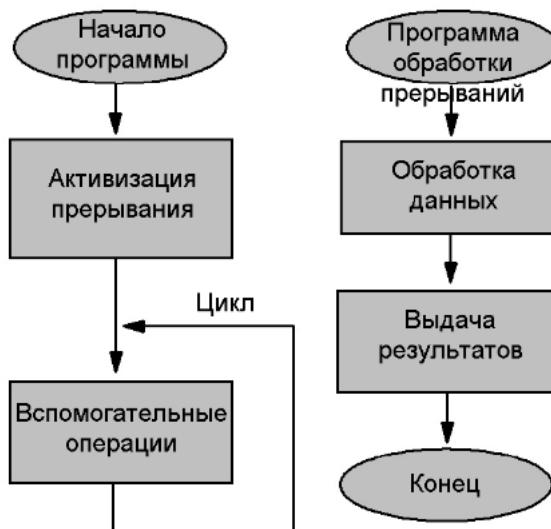


Рис. 9.2. Прерывание выполнения программы

9.3. Структура программы Arduino

Программа Arduino состоит из следующих основных частей:

- Комментарии и описание программы.
- Заголовки файлов и подключенные библиотеки.
- Обявление глобальных переменных.
- Стандартная настройка `void setup()` (порты и конфигурация).
- Основной цикл `void loop()`.
- Собственные процедуры.

Как мы увидим далее, соблюдать немногие основные правила программирования совсем не трудно. Следующий шаг — применение уже изученного материала для написания небольшой программы.

9.4. Первая программа с Arduino

Теперь создадим первую настоящую программу Arduino. В качестве упражнения можно просто перепечатать исходный текст листинга 9.1. Если и это делать не хочется, найдите исходный код на прилагаемом к книге компакт-диске в папке Beispiele\Erstes_Programm. Для выполнения программы потребуется Arduino-терминал.

Запустите терминальную программу после компилирования исходного кода и передачи в контроллер. Наша программа рассчитывает сумму двух чисел и выводит результат через терминал. После вывода результата начинает мигать светодиод L. С нажатием кнопки сброса программа запускается снова.

Листинг 9.1. Erstes Programm.pde

```
// Franzis Arduino
// Первая программа Arduino, которую я сам составил :-)

int ledPin = 13;      // Светодиод присоединяется к цифровому выводу 13

void setup()
{
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT);
    Serial.println("Unser erstes Arduino Programm");
    Serial.println();
}

void loop()
{
    Serial.print("Die Summe aus 5 + 188 lautet ");
    Serial.print(5+188);

    while(true)
    {
        digitalWrite(ledPin, HIGH);      // Включение светодиода
        delay(500);                   // Задержка на 500 мсек
        digitalWrite(ledPin, LOW);      // Выключение светодиода
        delay(500);                   // Задержка на 500 мсек
        continue;
    }
}
```

Программа иллюстрирует самые важные принципы последовательного алгоритма. Сначала контроллеру Arduino сообщается, что переменная ledPin должна содержать число 13. Затем выполняется установка начальных значений. Здесь указывается, к какому выводу присоединяется светодиод и задается скорость последовательного интерфейса, равная 9 600 бод. Установку завершает вывод информации через последовательный интерфейс.

Далее идет переход к основному циклу с именем `loop()`. В нем первая функция `Serial.print` выводит текст, а вторая — сумму $5+188$. Помните, что функция `Serial.println` после вывода осуществляет переход к левому краю с передачей управления на начало следующей строки (CR+LF), а `Serial.print` — нет.

После вывода информации запускается бесконечный цикл `while`, обеспечивающий непрерывное мигание светодиода.

СОВЕТ

При написании текста программы обращайте внимание на правильный стиль оформления: выбор верхнего и нижнего регистра, добавление отступов, расстановку скобок и точек с запятой.

9.5. Команды Arduino и их применение

Тем, кто только начинает программирование с помощью Arduino, следует тщательно изучить этот раздел и опробовать приведенные примеры на практике до тех пор, пока все нюансы не будут поняты. По сути, это маленький учебный курс программирования с помощью Arduino на языке C, на котором базируются все последующие эксперименты.

9.5.1. Комментарии в исходном тексте

Чтобы правильно читать программу по прошествии определенного времени и понимать ее, необходимо тщательно и аккуратно документировать свой исходный код. Можно внедрять документацию прямо в исходный код. Для этого имеются разные символы комментариев, которые предназначены для включения обычных текстов (листинг 9.2). В среде разработки Arduino текст, снабженный комментарием, выделен серым цветом.

Листинг 9.2. Kommentar.pde

```
/*
 Я пишу
 многострочный комментарий,
 который становится все длиннее и длиннее ...
 */
*****  
А так комментарий выглядит еще лучше!
*****
```

ПРИМЕЧАНИЕ

Комментарии в коде программы значительно облегчают ее чтение через некоторое время. Подумайте, сможете ли вы вспомнить через год, что делали сегодня в программе?!

9.5.2. Фигурные скобки { }

Фигурные скобки выделяют фрагмент кода для компилятора.

Блок кода всегда открывается символом { и завершается }. Между скобками располагаются команды.

Пример

```
type Meine_Funktion()
{
// В пределах скобок находятся команды
}
```

9.5.3. Точка с запятой ;

Точка с запятой завершает команду. Если пропустить этот символ, компилятор Arduino выдаст сообщение об ошибке.

Пример

```
Int x=42; // Здесь переменная x объявляется как Integer-переменная и ей задается число 42, точка с запятой завершает определение.
```

9.5.4. Типы данных и переменные

Каждая программа состоит из различных переменных, которые либо зависят от внешнего окружения (например, аналоговый или цифровой вход), либо требуются для расчета в программе и вывода результатов. В распоряжении программиста есть различные типы переменных: `byte` (байт), `Integer` (целое число), `Long` (длинное целое число) и `Float` (число с плавающей запятой). Тип переменной всегда нужно задавать перед ее применением.

9.5.5. Имя переменной

В среде разработки Arduino C имеется различие между верхним и нижним регистром символов имени переменной. В Arduino допускается подчеркивание `_`. Оно часто используется, чтобы сделать длинные имена переменной разборчивее. Ключевые слова (`if`, `while`, `do` и т. д.) не могут быть именами переменной. Имена глобальных переменных и функций не могут совпадать. Кроме того, не допускается одновременное задание функций и локальных переменных с одним и тем же именем.

9.5.6. Локальные и глобальные переменные

Если переменная объявляется в пределах функции, процедуры или как аргумент функции, она является локальной. Это значит, что переменная существует только внутри своей функции. Переменная, объявленная вне функции, является глобальной. Она определена для всех функций в пределах нашей программы.

Пример

```
byte Variable; // Переменная типа Byte, она может принимать значения
                // от 0 до 255
float PI = 3.1415; // Константа PI объявлена как Float
int myArray[9]; // Массив myArray, объявлен состоящим
                // из 10 элементов типа Int.
                // К соответствующему элементу обращаются через
                // указатель: Var(x)
                // Нумерация элементов массивов начинается с 0!
```

9.5.7. Различные типы данных

В следующих разделах показано, какие типы данных существуют и сколько при этом занимается памяти.

Boolean

Переменная типа Boolean может принимать два состояния: `true` (истина) или `false` (ложь). Такая переменная занимает 1 байт в памяти. Значение `true` соответствует логической единице, а `false` — это не логический нуль, как часто предполагают, а состояние, отличное от логической единицы.

```
boolean MeineWahrheit = true; // Переменная истинна
```

Byte

1 байт — это 8 бит, поэтому такая переменная может принимать значения от 0 до 255.

```
byte MeineVariable = 0; // Переменная здесь
                        // инициализируется нулевым значением
```

Char

Символ имеет размер 1 байт. Переменная `char` — это символ в одиночных кавычках, т. е. с апострофом. Если требуется строка из нескольких символов, то их заключают в двойные кавычки (например, "ПРИВЕТ"). Символ получает номер из набора символов ASCII. Например, буква "A" — это число 65. Символьные переменные могут принимать значения от -127 до +127.

```
char MeinCharakter = "A"; // Число 65
```

Unsigned Char

Символы без знака (*Unsigned Char*) ведут себя как символы со знаком, но они могут принимать только положительные значения в диапазоне от 0 до 255.

```
unsigned char = "B"; // Число 66
```

Int(Integer)

Целое число (*Integer*) состоит из двух байтов и может принимать значения от -32 768 до +32 768.

```
int MeineVariable = -32760; // Целая переменная со значением -32 760
```

Unsigned int

Тип *Unsigned Integer* охватывает переменные в диапазоне от 0 до 65 535 ($2^{16} - 1$). В отличие от *int* тип *unsigned int* не имеет знака. Такая переменная занимает в памяти два байта.

```
int MeineVariable = 50000; // Целая переменная без знака со значением 50 000
```

Long

Переменная *Long* состоит из четырех байтов и может принимать значения от -2 147 483 648 до 2 147 483 647 (32 бита *Long*).

```
long MeineVariable = 10000000; // переменная Long со значением 10 000 000
```

Unsigned Long

Переменная *Unsigned Long* состоит из четырех байтов и может принимать значения от 0 до 4 294 967 295 (32 бита *Long*). Переменная не имеет знака и может хранить только положительные значения.

```
unsigned long MeineVariable = 54544454544; // Очень, очень большая переменная
```

Float

Переменные *Float* могут хранить 32 бита значения со знаком. Диапазон приемлемых значений лежит в пределах от -3.4028235E+38 до +3.4028235E+38. Для этой переменной требуется четыре байта в памяти.

```
float MeineVariable = 100.42; // переменная Float со значением 100,42
```

String

Строковая переменная (*string*) — это массив (*Array*) переменных *Char* и нулевого символа.

Для каждого символа требуется, таким образом, один байт и в конце цепочки дополнительно еще один байт для нулевого символа. Таким образом, например, для слова "Hallo" требуется шесть байтов.

```
char MeineZeichenkette[] = "Hallo Welt"; // Требуется 11 байтов
```

Arrays

Массив (Arrays) — это упорядоченный набор переменных (рис. 9.3). В информатике массив обозначает структуру данных. С помощью массива данные, как правило, одинакового типа (Byte, Int и т. д.) заносятся в память компьютера таким образом, что доступ к данным становится возможен через указатель. В языке Arduino массивы должны объявляться как данные типа Integer. Нумерация элементов массива в Arduino начинается с нуля, в некоторых других компиляторах нумерация начинается с единицы. Кроме того, важно знать, что Arduino поддерживает в настоящее время только одномерные массивы. Пример работы с массивами приведен в листинге 9.3.



Рис. 9.3. Структура массива

Листинг 9.3. Arrays.pde

```
// Franzis Arduino
// Arrays

int Array_1[3];
int Array_2[] = {1,2,3};

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Arrays");
    Serial.println();
}

void loop()
{
    byte x;

    Array_1[0] = 1;
```

```
Array_1[1] = 2;
Array_1[2] = 3;
Array_1[3] = 4;

Serial.println("Ausgabe Array 1 ");
Serial.println("-----");

// Выдача данных первого массива
for(x=0;x<3;x++)
{
    Serial.print(Array_1[x]);
    Serial.println();
}

Serial.println("Ausgabe Array 2 ");
Serial.println("-----");

// Выдача данных второго массива
Serial.print(Array_2[0]);
Serial.println();
Serial.print(Array_2[1]);
Serial.println();
Serial.print(Array_2[2]);

while(1);

}
```

В первом массиве мы задаем размер "3". Таким образом, в этом массиве можно сохранять четыре переменные типа `Integer` (16 бит). Размерность второго массива не указана и три его элемента заданы в фигурных скобках. Речь идет о динамическом массиве. В основном цикле мы назначаем элементам первого массива различные значения, в нашем случае 1, 2, 3, 4. Можете указать другие значения, чтобы лучше понять работу с массивами.

Переменная счетчика `x` цикла `for` содержит значение указателя массива для вывода значений в терминале. Значения элементов второго массива и указателя тоже выводятся в терминале.

Цикл `while(1)` представляет собой бесконечный цикл основной программы, поэтому достаточно ее однократного запуска.

9.5.8. Операторы

Каждому типу данных соответствуют свои операторы, которые указывают, какие операции могут применяться. Рассмотрим возможные операторы Arduino и их действие.

Арифметические операторы

- = (присваивание).
- + (сложение).
- (вычитание).
- * (умножение).
- / (деление).
- % (деление по модулю — вычисление остатка от деления).

Операторы сравнения

- == (равно, например `a == b`).
- != (не равно, например `a != b`).
- < (меньше, например `a < b`).
- > (больше, например `a > b`).
- <= (меньше равно, например `a <= b`).
- >= (больше равно, например `a >= b`).

Побитовая арифметика

- & (битовое И).
- | (битовое ИЛИ).
- ~ (битовое НЕ).

Булева арифметика

&& (логическое И, например, если выражение `&& Antwort_B` истинно, тогда делать то-то).
|| (логическое ИЛИ, например, если выражение `|| Antwort_B` истинно, тогда делать то-то).
! (логическое отрицание, например, если выражение `! Antwort_B` истинно, тогда делать то-то).

Увеличение и уменьшение значения

- ++ (инкремент, например `i++`, переменная `i` будет увеличена на единицу).
- (декремент, например `i--`, переменная `i` будет на единицу меньше).
- += (присваивание с инкрементом, например `i += 5`, переменная `i` будет увеличена на 5 и результат присвоен переменной `i`).
- = (присваивание с декрементом, например `i -= 5`, переменная `i` будет уменьшена на 5 и результат присвоен переменной `i`).
- *= (присваивание с умножением, например `i *= 2`, переменная `i` будет умножена на 2 и результат присвоен переменной `i`).
- /= (присваивание с делением, например `i /= 2`, переменная `i` будет разделена на 2 и результат присвоен переменной `i`).

Константы

Вот примеры констант:

HIGH / LOW (HIGH = 1, LOW = 0).

INPUT/OUTPUT (INPUT = 0, OUTPUT = 1).

true/false (true = 1, false != 1).

9.5.9. Директива `#define`

`Define` — это директива препроцессора, которая запускается перед компилированием. Можно сказать, что речь идет о собственном маленьком компиляторе, который заранее преобразует команды `#define` в константы. Директива позволяет задавать постоянное значение переменной:

```
#define Variable1 1 // Без точки с запятой!
```

Здесь переменная `Variable1` получает значение 1. Всякий раз, когда в программе встречается имя `Variable1`, значение переменной заменяется на 1. Для нас эта замена неочевидна, но компилятор действует так, как описано. Тот, кто интересуется подробнее препроцессором и директивой `#define`, может найти более подробную информацию в Интернете.

9.5.10. Управляющие конструкции

Для возможности реагировать на события (условия) каждой программе требуются, так называемые, управляющие конструкции. Они указываются в языке С с помощью управляющих операторов `if... else-if... else` или `switch case`. В приведенных далее листингах вывод осуществляется через встроенную в Arduino терминальную программу. Загрузите примеры в плату микроконтроллера и запустите после этого терминальную программу, чтобы разобраться, что происходит.

Оператор `if`

Листинг 9.4 иллюстрирует синтаксис, а листинг 9.5 — пример использования оператора `if`.

Листинг 9.4. Синтаксис оператора `if`

```
if(Variable A = Variable B)
{
// Здесь находится код, который выполняется,
// когда A равно B
}
```

Листинг 9.5. If.pde

```
// Franzis Arduino
// if...

int x;
```

```
void setup()
{
    Serial.begin(9600);
    Serial.println("If Anweisungen");
    Serial.println();
}

void loop()
{

    if(x==10)
    {
        Serial.println("Die Variable X hat nun den Zaehlerstand 10!");
        while(1);
    }

    x++;
}

}
```

Основной цикл void loop() выполняется до тех пор, пока переменная x типа Integer не будет равна 10. Тело цикла составляют операторы между if и скобками {}. С помощью if можно реализовать простой способ передачи управления (разветвления в программе). Вы можете сами экспериментировать с другими операторами. Здесь допустимы логические операции ! =, <> и т. п.

if ... else

Листинг 9.6 поясняет синтаксис, а листинг 9.7 — пример использования оператора if...else.

Листинг 9.6. Синтаксис оператора if...else

```
if(Variable A > Variable B)
{
    // Код, который должен выполняться
}
else // или когда A не больше B
{
    // Код, который должен выполняться
}
```

Листинг 9.7. Else.pde

```
// Franzis Arduino
// if... else...

int x;

void setup()
{
    Serial.begin(9600);
    Serial.println("If und Else Anweisungen");
    Serial.println();
}

void loop()
{

    if(x==10)
    {
        Serial.println("Die Variable X hat nun den Zaehlerstand 10!");
        while(1);
    }
    else
    {
        Serial.print("X = ");
        Serial.print(x);
        Serial.println();
    }

    x++;
}
}
```

С конструкцией `else` можно реализовать различные альтернативные варианты. Программа выводит значение переменной `x`, пока оно не достигнет 10. Как только переменная `x` будет равна 10, запускается фрагмент программы между скобками `else {}`.

Возможность многократного чередования команд дает оператор `else if`. Здесь могут запрашиваться разные состояния переменных. В зависимости от логического значения (`true` или `false`) в условном операторе `else-if` выполняется соответствующая ветвь программы. Синтаксис поясняет листинг 9.8.

Листинг 9.8. Синтаксис оператора else if

```
if(Variable A != Variable B)
{
// Код, который должен запускаться
}
else if (Variable A == Variable B)
{
// Код, который должен запускаться
}
else (Variable A > Variable B)
{
// Код, который должен запускаться
}
```

В листинге 9.9 в зависимости от значения переменной x выводится разный текст.

Листинг 9.9. ElseIf.pde

```
// Franzis Arduino
// Else If
int x;
void setup()
{
  Serial.begin(9600);
  Serial.println("Else If Anweisungen");
  Serial.println();
}

void loop()
{
  if(x==42)
  {
    Serial.println("Die Variable X hat nun den Zaehlerstand 100
erreicht!");
    while(1);
  }
  else if (x==10)
  {
    Serial.println("Wir haben 10 erreicht");
  }
}
```

```
else
{
    Serial.print("X = ");
    Serial.print(x);
    Serial.println();
}
x++;
}
```

switch case

Конструкция `switch case` ведет себя подобно оператору `else if`. Здесь также в зависимости от логического значения соответствующего выражения будет выполнен соответствующий раздел кода. По умолчанию можно задать альтернативный вариант, при отсутствии соответствия условий внутри оператора `case`. Выход из оператора `case` осуществляется при помощи оператора `break` (листинг 9.10). Пример приведен в листинге 9.11.

Листинг 9.10. Синтаксис оператора `switch case`

```
Switch( Variable )
{
    case 1:
        // Код, который должен запускаться переменной = 1
        break;
    case 2:
        // Код, который должен запускаться переменной = 2
        break;
    default:
        // Альтернативный код, если все другие условия не
        // выполняются
}
```

Листинг 9.11. SwitchCase.pde

```
// Franzis Arduino
// Switch Case

int x;

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

```
Serial.println("Switch Case Anweisungen");
Serial.println();
}

void loop()
{

switch(x)
{
    case 10:
        Serial.println("Wir haben 10 erreicht");
        break;

    case 20:
        Serial.println("Wir haben 20 erreicht");
        break;

    case 30:
        Serial.println("Wir haben 30 erreicht");
        while(1);
        break;

    default:
        Serial.print("X = ");
        Serial.print(x);
        Serial.println();
}

x++;
}
```

В основном цикле значение переменной `x` увеличивается на единицу. При соответствии определенному условию в команде `switch`, в терминал будет выведено сообщение.

9.5.11. Циклы

При программировании часто требуются циклы, например, чтобы реализовать десятичный, двоичный счетчики или заставить программу бесконечное число раз повторять выполнение некоторого фрагмента. Цикл подходит также для организации чтения данных из последовательного интерфейса. Существуют несколько типов циклов. У каждого из них есть свои особенности, с которыми мы сейчас познакомимся.

for

Цикл `for` считает переменную вверх или вниз в пределах одного указанного диапазона значений. При этом может задаваться определенная величина приращения (инкремент). Листинг 9.12 иллюстрирует синтаксис цикла `for`, а листинг 9.13 — пример его использования.

Листинг 9.12. Синтаксис цикла `for`

```
// Структура цикла for
for( Startbedingung ; Stoppbedingung ; Zähler erhöhen )
{
// Блок программы
}
// Этот цикл считает от 0 до 10 с шагом 1
for( x = 0 ; x < 11 ; x++ )
{
// Код, который должен выполняться 10 раз
}
// Теперь значение переменной x увеличивается на 2
for( x = 0 ; x < 11 ; x=x+2 )
{
// Код, который должен выполняться 10 раз
}
// Переменная x уменьшается с 10 до 1 (шаг 1)
for( x = 10 ; x != 0 ; x-- ) {
// Код, который должен выполняться 10 раз
}
```

Листинг 9.13. for.pde

```
// Franzis Arduino
// For

int x;

void setup()
{
  Serial.begin(9600);
  Serial.println("For Anweisungen");
  Serial.println();
}

void loop()
```

```
{  
  
Serial.println("Schrittweite 1");  
for(x=0;x<11;x++)  
{  
    Serial.print("x = ");  
    Serial.print(x);  
    Serial.println();  
}  
  
Serial.println("Schrittweite 2");  
for(x=0;x<11;x=x+2)  
{  
    Serial.print("x = ");  
    Serial.print(x);  
    Serial.println();  
}  
Serial.println("Jetzt zaehlen wir von 10 mit Schrittweite 1 auf 1 herunter");  
for(x=10;x!=0;x--)  
{  
    Serial.print("x = ");  
    Serial.print(x);  
    Serial.println();  
}  
  
// Конец программы!  
while(1);  
  
}
```

Программа содержит три цикла `for` и поясняет принцип работы посредством вывода показаний счетчика. Чтобы считать вплоть до 10, нужно указывать максимальное значение счетчика цикла, равным 11. Таким образом, цикл работает, пока значение счетчика меньше 11. Если задать максимальное значение равным 10, то цикл не отработает последний шаг.

while* и *do while

Еще два варианта цикла — это `while` и `do while`. Операторы в теле цикла `do while` выполняются один раз, затем проверяется условие окончания цикла. Следовательно, этот цикл выполнится, по меньшей мере, один раз. Оператор `while` часто применяется для организации бесконечных циклов. Тем не менее, цикл можно прервать при помощи оператора `break`. Если требуется проверить условие перед вы-

полнением операторов цикла, то нужно использовать оператор `while`. Сначала производится проверка, а затем выполняются операторы тела цикла. Листинг 9.14 иллюстрирует синтаксис цикла `while`, а листинг 9.15 — пример его использования.

Листинг 9.14. Синтаксис цикла `while`

```
// Бесконечные циклы
while(1)
{
    // При любых обстоятельствах здесь мы хотим действовать
}
// Бесконечные циклы при обусловленной отмене
while(1)
{
    Variable++;
    if( Variable > 10 ) break;
}
// While
While( Variable < 10 )
{
Variable++;
}
// Do While
do
{
Variable++;
}while( Variable < 10 );
```

Листинг 9.15. DoWhile.pde

```
// Franzis Arduino
// Do While

int X=0;

void setup()
{
Serial.begin(9600);
Serial.println("Do und Do While Programm");
Serial.println();
}

void loop()
```

```
{  
  
while(1)  
{  
    X++;  
    Serial.print(X);  
    Serial.println();  
    if(X>9) break;  
}  
  
X=0;  
Serial.println();  
  
while(X<10)  
{  
    X++;  
    Serial.print(X);  
    Serial.println();  
}  
X=0;  
Serial.println();  
  
// Do While  
do  
{  
    X++;  
    Serial.print(X);  
    Serial.println();  
}while( X < 10 );  
  
while(1);  
  
}
```

9.5.12. Функции и подпрограммы

Вам очень часто потребуются функции. Они делают программу нагляднее, а также позволяют реализовать собственные команды. Вы можете создать свои функции и процедуры (подпрограммы без возвращаемого значения), которые потребуются снова и снова, и использовать их в будущих проектах (принцип модульного построения). Различие между функцией и процедурой (универсальное название *Sub Routine* (подпрограмма)) состоит в том, что функция в противоположность процедуре возвращает значение. В функции может, например, выполняться математическое выражение, которое возвращает результат передаваемой переменной.

Подпрограмма

Листинг 9.16 иллюстрирует пример подпрограммы.

Листинг 9.16. Sub.pde

```
// Franzis Arduino
// Arduino Sub-Routinen

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Sub-Routinen");
    Serial.println();
}

void loop()
{
    Ausgabe1();
    Ausgabe2();

    while(1);
}

void Ausgabe1()
{
    Serial.println("Ausgabe 1");
}

void Ausgabe2()
{
    Serial.println("Ausgabe 2");
}
```

Функции

Функция — это блок кодов программы, у которых есть имя и ряд команд, которые запускаются при вызове функции. Примеры функций `void setup()` и `void loop()` уже встречались ранее. Существуют также встроенные функции, которые мы рассмотрим далее.

Писать собственные функции имеет смысл, чтобы упростить повторяющиеся задачи и улучшить структуру программы. При объявлении указывают тип функции. Он идентичен типу данных, например, `int` для типов `Integer`. Если функция

не возвращает значения, ее тип будет `void`, как в примере Sub.pde. Кроме типа при объявлении функции задают имя и в скобках перечисляют все параметры, которые должны передаваться. Листинг 9.17 иллюстрирует пример.

Листинг 9.17. Funktionen.pde

```
// Franzis Arduino
// Arduino Funktionen

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Funktionen");
    Serial.println();
}

void loop()
{
    int Erg;
    Erg=Addition(12,55);
    Serial.print("Die Summe aus 12 + 55 = ");
    Serial.println(Erg);

    while(1);
}

int Addition(int x, int y)
{
    int sum;
    sum=x+y;
    return sum;
}
```

continue

По команде `continue` оставшаяся часть текущего цикла (`do`, `for` или `while`) пропускается и запускается код после блока `{}`. Листинг 9.18 иллюстрирует пример.

Листинг 9.18. Continue.pde

```
// Franzis Arduino
// Arduino Continue

int i=0;
```

```
void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Continue");
    Serial.println();
}

void loop()
{

for(i=0;i<10;i++)
{
    if(i%2==0)
    {
        continue;
    }
    Serial.print(i);
    Serial.print(" nicht durch 2 teilbar!");
    Serial.println();
}

while(1);
}
```

Здесь оператор `continue` всегда прерывает цикл `for`, когда переменная `i` не делится на 2.

9.5.13. Функции преобразования типа

Функции `char()`, `byte()`, `int()`, `long()` и `float()` преобразуют тип переменной. Таким образом, мы можем, например, из однобайтовой переменной сделать переменную типа `Long`. Смысл подобных операций — приведение типа данных для последующих вычислений.

Вот перечень функций преобразования типа:

- `Char()` — превращает значение в символ (`Charakter`).
- `Byte()` — превращает значение в `Byte`.
- `Int()` — превращает значение в `Integer`.
- `Long()` — превращает значение в `Long`.
- `Float()` — превращает значение в `Float`.

9.5.14. Математические функции

В следующем разделе будем знакомиться с математическими функциями Arduino. Проверьте как-нибудь результаты при помощи микрокалькулятора.

min(x, y)

Рассчитывает минимум из двух значений и возвращает меньшее значение. Листинг 9.19 иллюстрирует пример.

Листинг 9.19. min.pde

```
// Franzis Arduino
// Arduino min(x,y) Funktion
int x,y,Erg=0;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino min(x,y) Funktion");
    Serial.println();
}

void loop()
{
    Erg=min(10,55);
    Serial.print(Erg);
    Serial.println();

    while(1);
}
```

max(x, y)

Рассчитывает максимум из двух значений и возвращает большее значение. Листинг 9.20 иллюстрирует пример.

Листинг 9.20. max.pde

```
// Franzis Arduino
// Arduino max(x,y) Funktion

int x,y,Erg=0;

void setup()
```

```
{  
    Serial.begin(9600);  
    Serial.println("Arduino max(x,y) Funktion");  
    Serial.println();  
}  
  
void loop()  
{  
    Erg=max(10,55);  
    Serial.print(Erg);  
    Serial.println();  
  
    while(1);  
}
```

abs(x)

Рассчитывает абсолютную величину. Листинг 9.21 иллюстрирует пример.

Листинг 9.21. abs.pde

```
// Franzis Arduino  
// Arduino abs(x,y) Funktion  
  
int Erg;  
  
void setup()  
{  
    Serial.begin(9600);  
    Serial.println("Arduino abs(x,y) Funktion");  
    Serial.println();  
}  
  
void loop()  
{  
    Erg=abs(3.1415);  
    Serial.print(Erg);  
    Serial.println();  
  
    while(1);  
}
```

constrain(x, a, b)

Ограничивает число *x* в определенном диапазоне *a*, *b*. Листинг 9.22 иллюстрирует пример.

Листинг 9.22. constrain.pde

```
// Franzis Arduino
// Arduino constrain(x, a, b) Funktion

int x,Erg;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino constrain(x, a, b) Funktion");
    Serial.println();
}

void loop()
{
    for(x=0;x<60;x++)
    {
        Erg=constrain(x, 10, 50);
        Serial.print(Erg);
        Serial.println();
    }

    while(1);
}
```

map(x, fromLow, fromHigh, toLow, toHigh)

Функция *map* — это полезная функция для пересчета значений из одного диапазона в другой. Она идеально подходит, например, для преобразования большой входной переменной в маленькую выходную переменную. Листинг 9.23 иллюстрирует пример.

Листинг 9.23. map.pde

```
// Franzis Arduino
// Arduino map(x, fromLow, fromHigh, toLow, toHigh) Funktion

int x,Erg;

void setup()
{
```

```
Serial.begin(9600);
Serial.println("Arduino Map Funktion");
Serial.println();
}

void loop()
{
    for(x=0;x<20;x++)
    {
        Erg=map(x,0,20,5,15);
        Serial.print(Erg);
        Serial.println();
    }

    while(1);
}
```

pow(base, exponent)

Функция `pow` выдает результат возведения в степень первого значения аргумента со вторым значением аргумента (первое значение аргумента — число, второе — степень). В листинге 9.24 оба аргумента и результат имеют тип `float`.

Листинг 9.24. pow.pde

```
// Franzis Arduino
// Arduino pow(base,exponent) Funktion

int Erg;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino pow(base,exponent) Funktion");
    Serial.println();
}

void loop()
{
    Erg=pow(10,5);
    Serial.print(Erg);
    Serial.println();

    while(1);
}
```

sq(x)

Функция вычисляет квадрат аргумента. Листинг 9.25 иллюстрирует пример.

Листинг 9.25. sq.pde

```
// Franzis Arduino
// Arduino sq(x)-Funktion

int Erg;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino sq(x) Funktion");
    Serial.println();
}

void loop()
{
    Erg=sq(3);
    Serial.print(Erg);
    Serial.println();
    while(1);
}
```

sqrt(x)

Это функция, обратная `sq(x)`, она вычисляет квадратный корень числа. Листинг 9.26 иллюстрирует пример.

Листинг 9.26. sqrt.pde

```
// Franzis Arduino
// Arduino sqrt(x) Funktion

int Erg;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino sqrt(x) Funktion");
    Serial.println();
}

void loop()
```

```
{  
    Erg=sqrt(9);  
    Serial.print(Erg);  
    Serial.println();  
  
    while(1);  
}
```

sin(rad)

Функция рассчитывает синус. Аргумент указывается в радианах. Возвращаемое значение (от -1 до +1) — это синус аргумента. Листинг 9.27 иллюстрирует пример.

Листинг 9.27. sin.pde

```
// Franzis Arduino  
// Arduino sin(x) Funktion  
  
float Erg;  
  
void setup()  
{  
    Serial.begin(9600);  
    Serial.println("Arduino sin(x) Funktion");  
    Serial.println();  
}  
  
void loop()  
{  
    Erg=sin(1.0);  
    Serial.print(Erg);  
    Serial.println();  
  
    while(1);  
}
```

cos(rad)

Функция рассчитывает косинус. Угол указывается в радианах. Возвращаемое значение находится в пределах от -1 до 1. Листинг 9.28 иллюстрирует пример.

Листинг 9.28. cos.pde

```
// Franzis Arduino  
// Arduino cos(x) Funktion  
  
float Erg;
```

```
void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino cos(x) Funktion");
    Serial.println();
}

void loop()
{
    Erg=cos(1.0);
    Serial.print(Erg);
    Serial.println();

    while(1);
}
```

tan(rad)

Функция вычисляет тангенс. Угол указывается в радианах. Листинг 9.29 иллюстрирует пример.

Листинг 9.29. tan.pde

```
// Arduino tan(x) Funktion

float Erg;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino tan(x) Funktion");
    Serial.println();
}

void loop()
{
    Erg=tan(1.0);
    Serial.print(Erg);
    Serial.println();

    while(1);
}
```

9.5.15. Последовательный ввод/вывод

Связь через интерфейс UART широко применяется. Микроконтроллер может посылать, а также принимать данные от компьютера или других микроконтроллеров. В Arduino для этого существует несколько команд. Некоторые уже встречались в предыдущих примерах, например, `Serial.print()` и `Serial.println()`. Микроконтроллер имеет встроенный аппаратный интерфейс UART. Универсальный асинхронный приемопередатчик (UART — Universal Asynchronous Receiver Transmitter) можно также имитировать с помощью программного обеспечения. Эмуляция будет не настолько скоростной, как аппаратный интерфейс UART, но все-таки позволит устанавливать одновременное подключение к нескольким вызываемым станциям. На рис. 9.4 изображен аппаратный интерфейс UART, реализованный в Arduino.

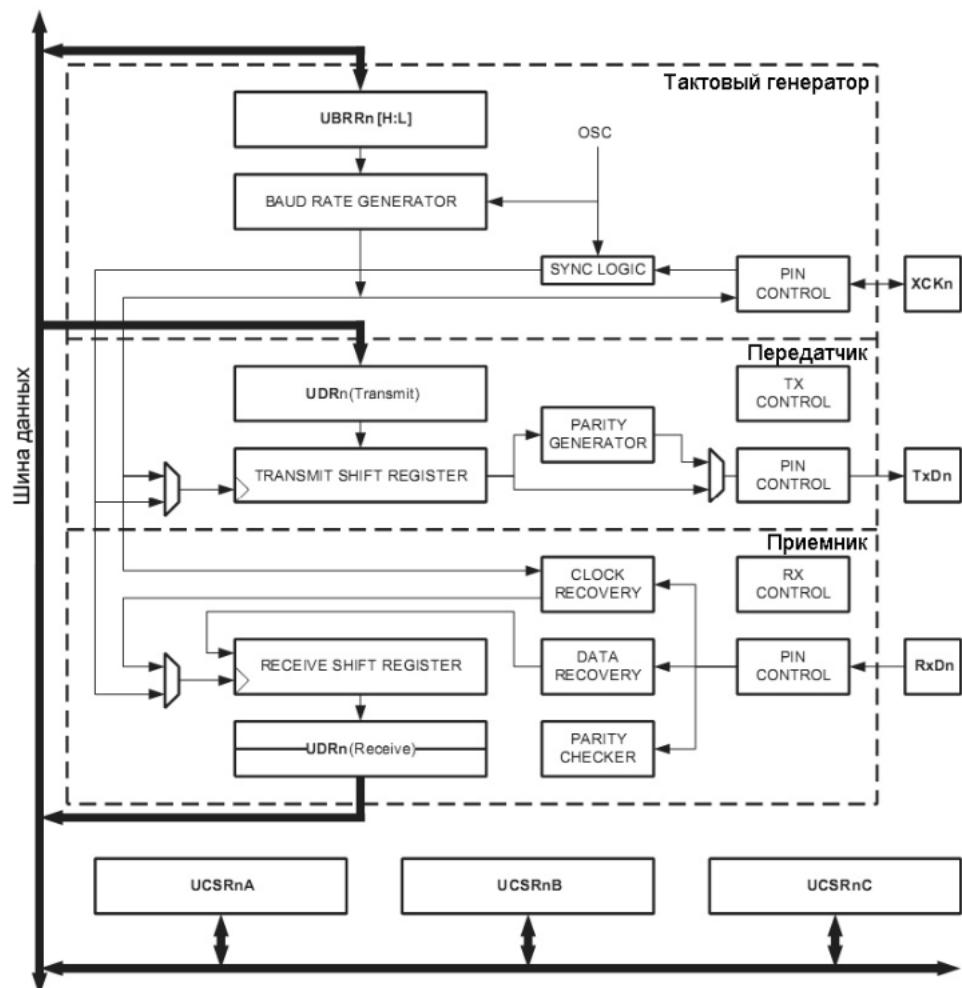


Рис. 9.4. Аппаратный интерфейс UART микроконтроллера
(источник: технический паспорт компании ATME)

Arduino-команда `Serial.println` посыпает последовательность символов (так называемую строку) через интерфейс UART. "Невидимые" символы CR (Carriage Return) и LF (LineFeed) добавляются автоматически и обозначают конец строки. Если переход на другую строку не требуется, то следует указать команду `Serial.print`. Если число передается в `Serialprint`, оно автоматически преобразуется в текст. Передается не число, а ASCII-код для этого числа. Например, число 42 состоит из двух символов 4 и 2. Будут переданы два ASCII-кода для символов "4" и "2" и ASCII-коды обоих служебных символов "CR" и "LF". Важно, чтобы приемники и передатчики были всегда установлены на одинаковую скорость в бодах.

Serial.begin(Baudrate)

Функция `Serial.begin()` задает скорость последовательного интерфейса в бодах. Бод обозначает число бит в секунду. Если мы выбираем 9 600 бод, один бит передается за $1/9\ 600 = 0,000104$ с = 104 мкс. Это действительно быстро.

Функция `Serial.begin(Baudrate)` открывает последовательный порт и устанавливает скорость в бодах (скорость передачи данных) для последовательной передачи. Возможны следующие скорости в бодах:

- 300
- 1 200
- 4 800
- 9 600
- 14 400
- 19 200
- 28 800
- 38 400
- 57 600
- 115 200

При добавлении платы Arduino Pro или платы аналогичного типа с несколькими аппаратными интерфейсами UART, конфигурацию можно задать таким образом:

```
Serial.begin(9600);  
Serial1.begin(38400);  
Serial2.begin(19200);  
Serial3.begin(4800);
```

Все интерфейсы имеют номер. Первый интерфейс — `UART0`. Если требуется вывести символ, то в этом случае применяется функция `Serialx.print()`, где `x` соответствует номеру интерфейса UART.

Примеры

```
Serial.println("Hallo hier ist UART 0");  
Serial1.println("Hallo hier ist UART 1");  
Serial2.println("Hallo hier ist UART 2");  
Serial3.println("Hallo hier ist UART 3");
```

ПРИМЕЧАНИЕ

При последовательной коммуникации цифровые выводы 0 (RX) и 1 (TX) не могут использоваться одновременно.

Serial.end()

Для завершения работы с последовательным интерфейсом и использования выводов по другому назначению, можно применить функцию `Serial.end()`.

Serial.read()

Функция `Serial.read()` считывает один байт из последовательного интерфейса:

```
int x;  
x = Serial.read();
```

Serial.available()

Функция `Serial.available()` указывает, имеется ли символ в последовательном буфере. Эта функция позволяет, например, пропустить блок программы, если в буфере нет никаких данных. Зачем впустую обрабатывать данные и расточать ценнное время для вычислений, если значения в буфере не изменились? Эта функция очень важна для работы с последовательным интерфейсом. В листинге 9.30 показано, как эта функция работает на практике.

Листинг 9.30. `Serial_available.pde`

```
// Franzis Arduino  
// Arduino Serial.available Funktion  
  
byte eingabe, ausgabe;  
  
void setup()  
{  
    Serial.begin(9600);  
    Serial.println("Arduino Serial.available Funktion");  
    Serial.println();  
}  
  
void loop()  
{  
  
    if (Serial.available() > 0)  
    {  
        eingabe=Serial.read();  
        Serial.print("Ich habe folgendes Zeichen empfangen: ");  
    }  
}
```

```
ausgabe=char(eingabe);  
Serial.println(eingabe);  
}  
}
```

Листинг 9.30 иллюстрирует также преобразование типов. Если преобразование не выполнялось, то будет выведен код ASCII. Попробуйте выполнить такой пример:

```
Serial.println(вводимые данные, DEC);
```

Serial.flush()

Эта функция удаляет содержимое последовательного буфера. Применяют ее, чтобы очистить буфер после распределения последовательных данных. Если возникает ошибка коммуникации и данные повреждаются, то они удаляются из буфера.

Serial.print()

Функция `Serial.print()` выводит данные из последовательного буфера передачи. Допустимы типы `Integer`, `Byte`, `Char` и `Float`.

Функция `Serial.print(x)` без указания формата выводит десятичное число из буфера UART:

```
int b = 79;  
Serial.print(b);
```

Выводит число 79 из буфера UART.

Функция `Serial.print(b, DEC)` с указанием формата `DEC` выводит число как ASCII-строку из буфера UART:

```
int b = 79;  
Serial.print(b, DEC);
```

Выводит ASCII-строку "79" из буфера UART.

Функция `Serial.print(b, HEX)` с указанием формата `HEX` выводит из буфера UART число как ASCII-строку в шестнадцатеричном формате:

```
int b = 79;  
Serial.print(b, HEX);
```

Выводит ASCII-строку "4F" из буфера UART.

Функция `Serial.print(b, OCT)` с указанием формата `OCT` выводит из буфера UART число как ASCII-строку в восьмеричном формате:

```
int b = 79;  
Serial.print(b, OCT);
```

Выводит ASCII-строку "117" из буфера UART.

Функция `Serial.print(b, BIN)` с указанием формата `BIN` выводит из буфера UART число как ASCII-строку в двоичном формате:

```
int b = 79;  
Serial.print(b, BIN);
```

Выводит ASCII-строку "1001111" из буфера UART.

Функция `Serial.print(b, BYTE)` с указанием формата `BYTE` выводит из буфера UART число в виде отдельного байта:

```
int b = 79;
```

```
Serial.print(b, BYTE);
```

Выводит ASCII-символ "O" из буфера UART.

Листинг 9.31 иллюстрирует примеры вывода в различных форматах.

Листинг 9.31. Print.pde

```
// Franzis Arduino
// Arduino Serial.print Funktion

int x;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Serial.print Funktion");
    Serial.println();
}

void loop()
{

    Serial.print("NO FORMAT");
    Serial.print("\t");           // Print Tab

    Serial.print("DEC");          // Decimal
    Serial.print("\t");

    Serial.print("HEX");          // Hexadezimal
    Serial.print("\t");
    Serial.print("OCT");          // Octal
    Serial.print("\t");

    Serial.print("BIN");          // Binär
    Serial.print("\t");

    Serial.println("BYTE");        // Byte

    for(x=0; x< 64; x++)
    {
```

```
// Вывод в различных форматах
Serial.print(x);
Serial.print("\t");

Serial.print(x, DEC);
Serial.print("\t");

Serial.print(x, HEX);
Serial.print("\t");

Serial.print(x, OCT);
Serial.print("\t");

Serial.print(x, BIN);
Serial.print("\t");

Serial.println(x, BYTE);
delay(200); // Задержка на 200 мсек
}

Serial.println("");
}
```

Serial.println()

Функция выводит данные из последовательного порта и осуществляет автоматический переход на новую строку.

Serial.write()

Функция `Serial.write()` осуществляет побайтовый вывод данных из последовательного буфера:

```
Serial.write(val);
Serial.write(str);
Serial.write(buf, len);
```

Значение параметров:

- `val` — посыпает отдельный байт;
- `str` — посыпает строку побайтово;
- `buf` — посыпает массив побайтово;
- `len` — длина массива (размер буфера).

Листинг 9.32 иллюстрирует пример.

Листинг 9.32. Serial_write.pde

```
// Franzis Arduino
// Arduino Serial.write Funktion

byte val = 65;
char str[] = "Test";
byte buf[] = {'H','a','l','l','o'};
byte len = 3;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Serial.write Funktion");
    Serial.println();
}

void loop()
{
    Serial.println("ASCII Zeichen");
    Serial.write(val);
    Serial.println();

    Serial.println("String 1");
    Serial.write(str);
    Serial.println();

    Serial.println("String 2");
    Serial.write(buf, len);
    Serial.println();

    while(1);
}
```

В функции `Serial.write(buf, len)` была указана переменная `len`, равная "3". Таким образом, будут выведены только первые три символа.

9.5.16. Как функционирует последовательный интерфейс?

Передача пакета начинается со стартового бита (низкий уровень), далее следуют 8 битов данных. Непосредственно после битов данных передается проверочный бит и, наконец, стоповый бит (рис. 9.5).



St	Стартовый бит, всегда низкий уровень
(n)	Биты данных (от 0 до 8)
P	Проверочный бит
Sp	Стоповый бит, всегда высокий уровень
IDLE	Нет передачи по коммуникационной линии (RxDn или TxDn). При состоянии IDLE линия может иметь высокий уровень

Рис. 9.5. Так выглядит передача через интерфейс UART

Считывание строки символов через последовательный интерфейс

Вам уже известно, как могут передаваться и приниматься отдельные символы и цепочки символов (строки). Для взаимодействия и настоящего ввода данных слов этого еще не достаточно. Нужно написать программу, позволяющую принимать любые строки или блоки данных (листинг 9.33).

Листинг 9.33. Serial_read.pde

```
// Franzis Arduino
// Arduino Serial Read

#define INLENGTH 20
#define INTERMINATOR 13

char inString[INLENGTH+1];
int inCount;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Serial read");
    Serial.println();
}
```

```
Serial.println("Geben Sie einen Text mit max. 20 Zeichen ein: ");
}

void loop()
{

    inCount = 0;

    do
    {
        while (Serial.available()==0);
        inString[inCount] = Serial.read();
        if(inString[inCount]==INTERMINATOR) break;
    }while(++inCount < INLENGTH);

    inString[inCount] = 0;
    Serial.print(inString);

}
```

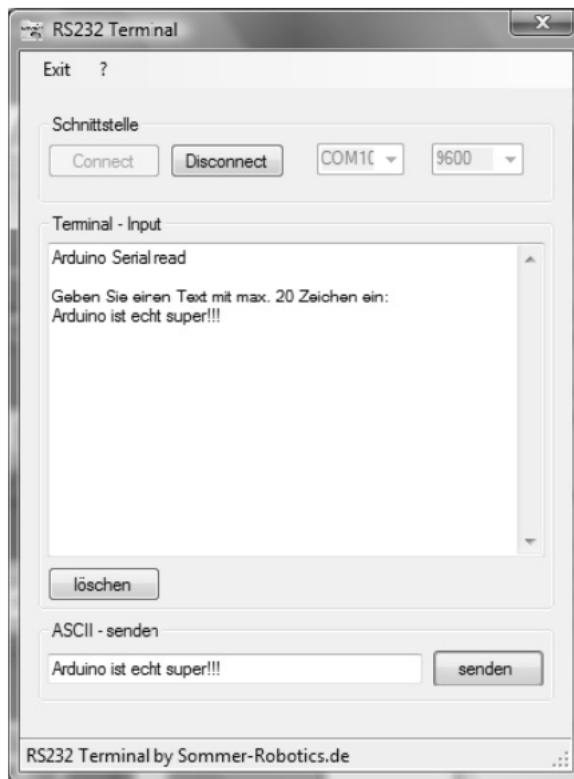


Рис. 9.6. Терминальная программа VB.NET

Программа ждет появления символа в буфере и считывает его в массив `inString`. Если было принято более 20 символов или символ CR, массив выводится в терминал. Терминал Arduino не подходит для этого, т. к. он не воспринимает символ CR в конце передачи. На прилагаемом к книге компакт-диске в папке Software\Terminal есть подходящая программа VB.NET (рис. 9.6).

Последовательный вывод данных с вычислением

Практический пример работы с последовательным интерфейсом иллюстрирует листинг 9.34 — небольшая программа, которая пересчитывает градусы по Цельсию в градусы по Фаренгейту и наоборот.

Листинг 9.34. GradFahrenheit.pde

```
// Franzis Arduino
// Arduino Grad zu Fahrenheit

float Grad = 25.5;
float Fahrenheit = 88.2;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Grad zu Fahrenheit Umrechner");
    Serial.println();
}

void loop()
{
    Serial.print(Grad);
    Serial.print(" Grad sind ");
    Serial.print(Grad_to_Fahrenheit(Grad));
    Serial.println(" Fahrenheit");
    Serial.println();

    Serial.print(Fahrenheit);
    Serial.print(" Fahrenheit sind ");
    Serial.print(Fahrenheit_to_Grad(Fahrenheit));
    Serial.println(" Grad");
    Serial.println();

    while(1);
}

float Grad_to_Fahrenheit(float grad)
```

```
{  
    float erg;  
    erg = grad * 9 ; erg = erg / 5 ; erg = erg + 32;  
    return erg;  
}  
  
float Fahrenheit_to_Grad(float fahrenheit)  
{  
    float erg;  
    erg = fahrenheit - 32 ; erg = erg * 5 ; erg = erg / 9;  
    return erg;  
}
```

9.5.17. Программная эмуляция UART

Когда требуется эксплуатация нескольких последовательных устройств, а в микроконтроллере есть только один аппаратный интерфейс UART, среда разработки Arduino предлагает возможность программной эмуляции интерфейса UART. При программной реализации UART используются выводы 2 и 3. Данныечитываются в стек размером 64 байта. Недостаток программного интерфейса UART состоит в том, что для этого требуются дополнительные системные ресурсы.

Ограничения при применении программного интерфейса UART:

- Максимальная скорость передачи — 9 600 бод.
- Отсутствует функция `Serial.available()`.
- Функция `Serial.read()` ждет до тех пор, пока не появятся данные в буфере.
- Функция `Serial.read()` должна запускаться в цикле, если функция не вызывается; если данные поступают, они пропадают.

В библиотеке UART от Arduino есть следующие функции: `SoftwareSerial()`, `begin()`, `read()`, `print()`, `println()`.

Листинг 9.35 иллюстрирует пример конфигурации последовательного интерфейса UART.

Листинг 9.35. Soft:UART.pde

```
// Franzis Arduino  
// Arduino Software UART  
  
// Здесь подключается библиотека UART  
#include <SoftwareSerial.h>  
  
#define rxPin 2  
#define txPin 3  
#define ledPin 13
```

```
// Die Software UART wird konfiguriert
SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);
byte pinState = 0;

void setup()
{
    // Konfiguration der Pins
    pinMode(rxPin, INPUT);
    pinMode(txPin, OUTPUT);
    pinMode(ledPin, OUTPUT);
    // Установка последовательной скорости в бодах
    mySerial.begin(9600);
}

void loop()
{
    // Теперь мы принимаем данные
    char someChar = mySerial.read();
    // Вывод принятого символа
    mySerial.print(someChar);
    // Светодиод будет переключаться
    toggle(13);
}

void toggle(int pinNum)
{
    digitalWrite(pinNum, pinState);
    pinState = !pinState;
}
```

9.5.18. Конфигурация входа/выхода и установка порта

Чтобы чип Arduino знал, какой вывод мы хотели бы использовать как вход или выход, нужно указать это в подпрограмме `void setup()`. Далее мы рассмотрим, как это сделать. Если такая конфигурация не выполняется, то все выводы микроконтроллера после включения устанавливаются в высокоимпедансное состояние (Z-состояние).

pinMode(pin, mode)

Функция используется в программе `void setup()`, чтобы конфигурировать контакт платы как вход или как выход:

```
pinMode(pin, OUTPUT); // контакт устанавливается как выход
```

В чипе ATmega также имеется встроенный подтягивающий резистор (20 кОм), который управляется посредством программного обеспечения. К встроенному подтягивающему резистору можно обращаться следующим способом:

```
pinMode(pin, INPUT); // Вывод устанавливается как вход
digitalWrite(pin, HIGH); // Подключается подтягивающий резистор
```

Подтягивающий резистор обычно подключают, чтобы присоединять входы как коммутатор. Видно, что вывод конфигурируется как вход, но на нем устанавливается высокий уровень. Это лишь метод активизировать внутренний подтягивающий резистор.

Выводы, конфигурируемые как выход, имеют малое полное сопротивление и могут нагружаться присоединенными элементами и схемами с током максимум 40 мА. Этого хватит для свечения светодиода (с учетом последовательно включенного сопротивления), но недостаточно, чтобы эксплуатировать большинство типов реле, соленоидов или электродвигателей. Короткие замыкания выводов платы Arduino, а также слишком большие токи могут повредить выходной вывод и даже вывести из стоя микроконтроллер. Поэтому лучше подключать внешние компоненты к выходу через резистор 470 Ом или 1 кОм.

digitalRead(pin)

Функция `digitalRead()` считывает значение заданного цифрового вывода, с результатом `HIGH` или `LOW`, что соответствует 1 или 0. Номер вывода может устанавливаться либо как переменная, либо как константа (от 0 до 13).

```
value = digitalRead(Pin); // устанавливает value, равное значению
// на входном выводе
```

digitalWrite(pin,value)

Устанавливает уровень `HIGH` или `LOW` на заданном выводе. Номер вывода может задаваться либо как переменная, либо как константа (от 0 до 13).

```
digitalWrite(pin, HIGH); // устанавливает на выводе высокий уровень (+5 В)
```

Считывание состояния кнопки

В следующем примере (листинг 9.36) считывается состояние кнопки, подключенной к цифровому входу (контакт 12), и результат отображается с помощью светодиода. При нажатии на кнопку светодиод гаснет, а при отпускании — загорается. Здесь дело в том, что мы активизировали внутренний подтягивающий резистор и, таким образом, наш входной вывод 12 имеет высокий уровень. Если мы нажимаем на кнопку, которая подключает вывод к GND (общий провод), светодиод не горит, т. к. на входе теперь низкий уровень. Внешний вид макета приведен на рис. 9.7.

Необходимые комплектующие изделия:

- плата микроконтроллера Arduino/Freeduino;
- панель с контактными гнездами;
- кнопка;
- два гибких монтажных провода длиной примерно 5 см.

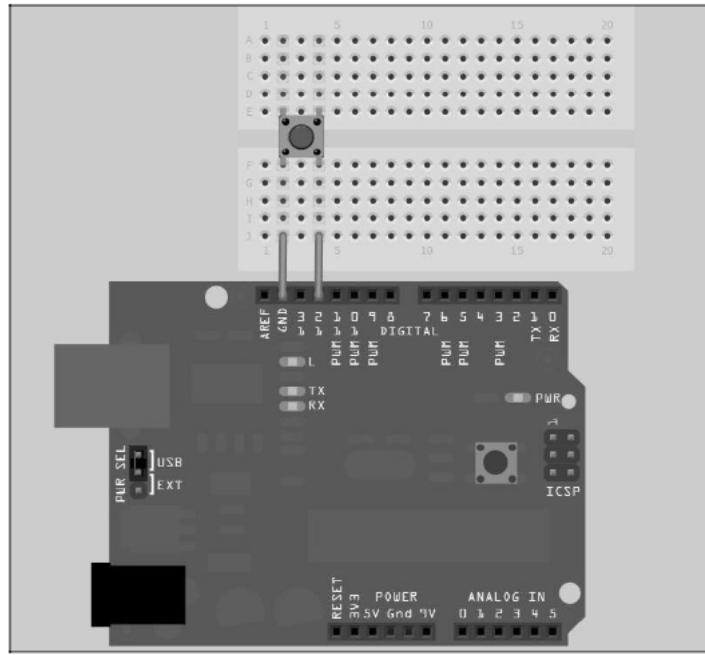


Рис. 9.7. Внешний вид макета

Листинг 9.36. IO.pde

```
// Franzis Arduino
// Taster über IO-Pin einlesen

int led=13;
int pin=12;
int value=0;

void setup()
{
    pinMode(led,OUTPUT);
    pinMode(pin,INPUT);
    digitalWrite(pin, HIGH);
}

void loop()
{
    value=digitalRead(pin);
    digitalWrite(led,value);
}
```

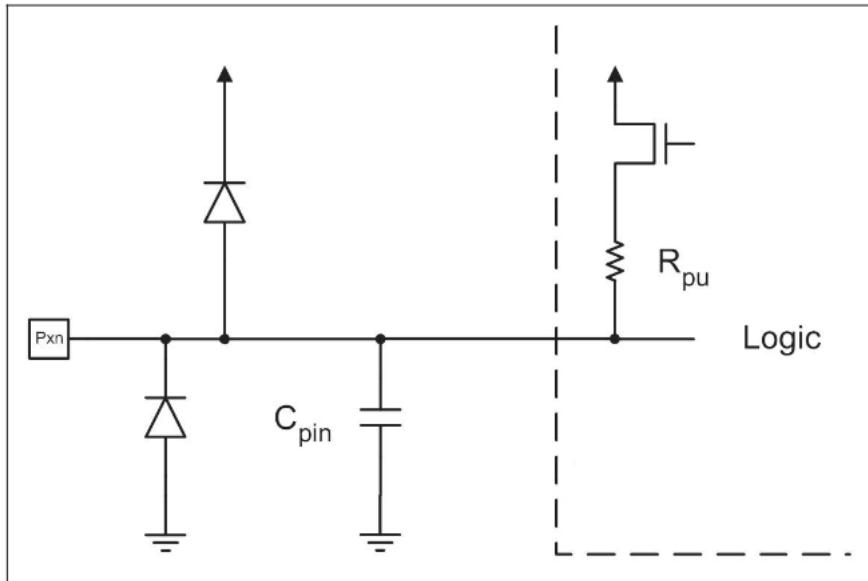


Рис. 9.8. Подключение подтягивающего резистора и защитных диодов к контакту ввода/вывода
(источник: технический паспорт компании ATMEL)

ПРИМЕЧАНИЕ

Вывод контроллера ATmega выдерживает ток до 40 мА. Общая нагрузка микроконтроллера (в зависимости от типа корпуса) не может быть более 200 мА. Выводы могут отличаться друг от друга. Если требуются точные данные, всегда смотрите паспортные данные.

К контакту порта ввода/вывода подключены также защитные диоды и подтягивающий резистор через полевой транзистор (рис. 9.8). Диоды защищают вывод от статических наводок.

Кнопка с согласующим резистором

В предыдущем примере кнопка была подключена к контакту с внутренним подтягивающим резистором. При нажатии кнопки контакт соединялся с GND (общей шиной). В следующем примере (листинг 9.37) показано, как можно опрашивать кнопку через напряжение питания VCC (+5 В). Так как здесь не включен подтягивающий резистор, то нужно добавить внешний согласующий резистор, чтобы вывод не оказался в неопределенном состоянии (рис. 9.9).

Необходимые комплектующие изделия:

- плата микроконтроллера Arduino/Freeduino Duemilanove;
- панель с контактными гнездами;
- кнопка;
- резистор 10 кОм;
- три гибких монтажных провода длиной примерно 10 см.

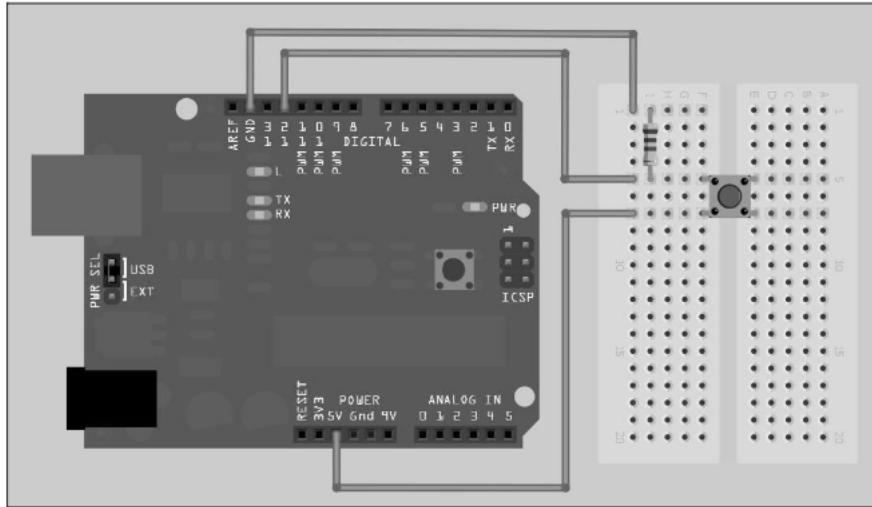


Рис. 9.9. Монтажная схема макета

Листинг 9.37. Pulldown.pde

```
// Franzis Arduino
// Кнопка с согласующим резистором

int led=13;
int pin=12;
int value=0;
void setup()
{
    pinMode(led,OUTPUT);
    pinMode(pin,INPUT);
}

void loop()
{
    value=digitalRead(pin);
    digitalWrite(led,value);
}
```

Кнопка с внешним подтягивающим резистором

Рассматриваемый пример (листинг 9.38) иллюстрирует возможность подключения внешнего подтягивающего резистора к плате Arduino.

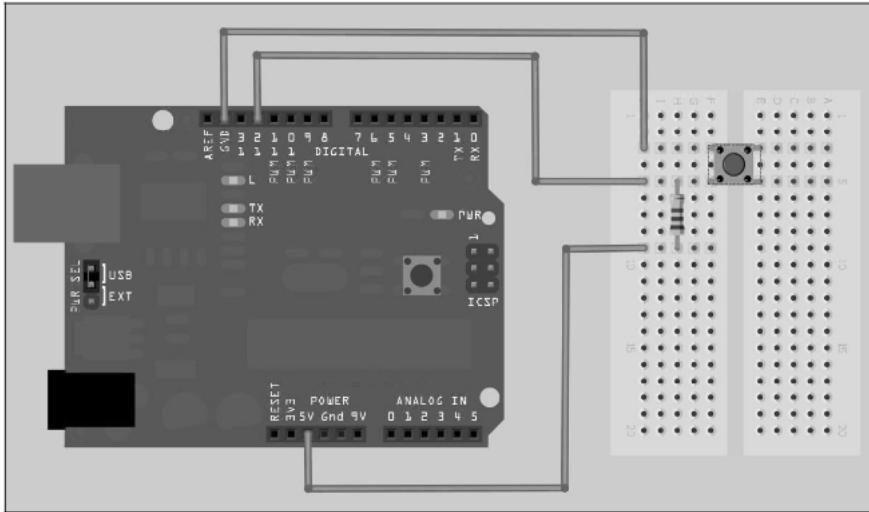


Рис. 9.10. Схема макета

Необходимые комплектующие изделия:

- плата микроконтроллера Arduino/Freeduino;
- панель с контактными гнездами;
- кнопка;
- резистор 10 кОм;
- три гибких монтажных провода длиной примерно 10 см.

В этой схеме (рис. 9.10) резистор подключен не к GND, а к напряжению питания VCC (+5 В). Светодиод светится после включения схемы и гаснет при нажатии кнопки.

Листинг 9.38. Ext_Pullup.pde

```
// Franzis Arduino
// Кнопка с внешним подтягивающим резистором

int led=13;
int pin=12;
int value=0;

void setup()
{
    pinMode(led,OUTPUT);
    pinMode(pin,INPUT);
}

void loop()
```

```
{  
    value=digitalRead(pin);  
    digitalWrite(led,value);  
}
```

9.5.19. Аналоговый ввод данных и АЦП

Для измерения аналогового напряжения платы микроконтроллера Arduino оборудована внутренним аналого-цифровым преобразователем (ADC — Analog Digital Converter). Встроенный АЦП 10-разрядный, т. е. шаг изменения равен значению аналогового напряжения 0,0048 В при опорном напряжении $U_{ref} = 5$ В. Плата микроконтроллера имеет шесть аналоговых входов, но только один внутренний АЦП. В таком случае каналы будут переключаться. Называют их также мультиплексными.

По этой простой формуле можно легко рассчитать разрешение:

$$U_{step} = U_{ref} / \text{Разрядность.}$$

$$U_{step} = 5 \text{ В} / 1\,024 \text{ (от 0 до 1\,023 = 1\,024 шагов)} = 0,0048 \text{ В.}$$

Если требуется узнать цифровое значение, то можно рассчитать его по формуле:
Значение = 1\,024 × (напряжение на входе АЦП) / U_{ref} .

Точность измерения составляет ± 2 шага. При опорном напряжении 5 В точность преобразования составляет $\pm 0,0097$ В. Можно говорить таким образом, что АЦП измеряет приложенное напряжение с точностью до двух цифр после запятой. Естественно опорное напряжение должно быть стабильным.

analogRead(pin)

Функция считывает значение установленного аналогового входа с разрешением 10 бит. Эта функция допустима только для выводов 0–5:

```
value = analogRead(pin); // устанавливает переменную value равной значению напряжения на входе
```

ПРИМЕЧАНИЕ

Аналоговые выводы, в отличие от цифровых, не нужно объявлять как вход или выход в начале программы.

В завершение рассмотрим небольшую программу (листинг 9.39) для измерения напряжения на входе 0 АЦП (ANALOG IN 0). Монтажная схема приведена на рис. 9.11.

Необходимые комплектующие изделия:

- плата микроконтроллера Arduino/Freeduino;
- панель с контактными гнездами;
- построечный резистор 10 кОм;
- три гибких монтажных провода длиной примерно 10 см.

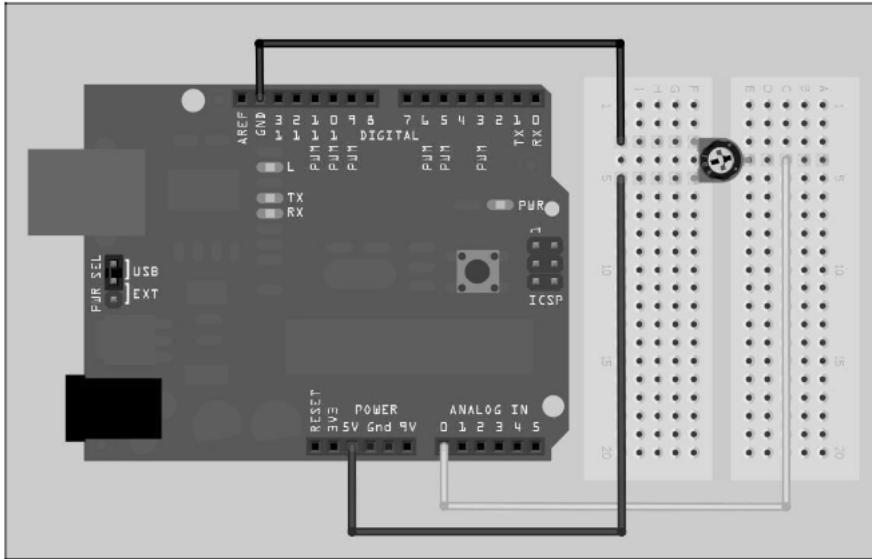


Рис. 9.11. Монтажная схема макета

Листинг 9.39. ADC.pde

```
// Franzis Arduino
// ADC

int ADC0=0;
int value;
int LEDpin=13;

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

void loop()
{
    value=analogRead(ADC0);
    Serial.print("ADC0 = ");
    Serial.println(value);
    delay(1000);
}
```

При вращении ручки потенциометра будет видно, что значение на выходе АЦП изменяется от 0 до 1 023, в зависимости от положения движка. Максимума в 1 023 можно и не достичь, поскольку некоторые потенциометры имеют небольшое остаточное сопротивление. Еще один пример иллюстрирует листинг 9.40.

Листинг 9.40. ADC_Blinker.pde

```
// Franzis Arduino
// Частота мигания светодиода устанавливается через АЦП

int ADC0=0;
int value;
int LEDpin=13;

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

void loop()
{
    value=analogRead(ADC0);
    digitalWrite(LEDpin, HIGH);
    delay(value);
    digitalWrite(LEDpin, LOW);
    delay(value);
}
```

Этот пример можно испытывать на уже имеющейся собранной схеме. Теперь можно устанавливать частоту миганий светодиода. Аргумент `Delay` указывают в миллисекундах.

9.5.20. Аналоговый выход ШИМ

На плате Arduino находятся в распоряжении шесть выходов сигналов с широтно-импульсной модуляцией (ШИМ): контакты 3, 5, 6, 9, 10 и 11 (на более старых контроллерах ATmega8 только контакты 9, 10 и 11). Они могут использоваться для цифроаналогового преобразования, управления сервоэлектродвигателями или для формирования звуковых сигналов. В процессе ШИМ (PWM — Pulse Width Modulation) меняется скважность импульсной последовательности. Скважность указывает соотношение длительности включенного состояния к периоду повторения импульсов. При этом частота и уровень сигнала остаются всегда одинаковыми. Изменяется только длительность перехода от высокого к низкому уровню (рис. 9.12–9.14).

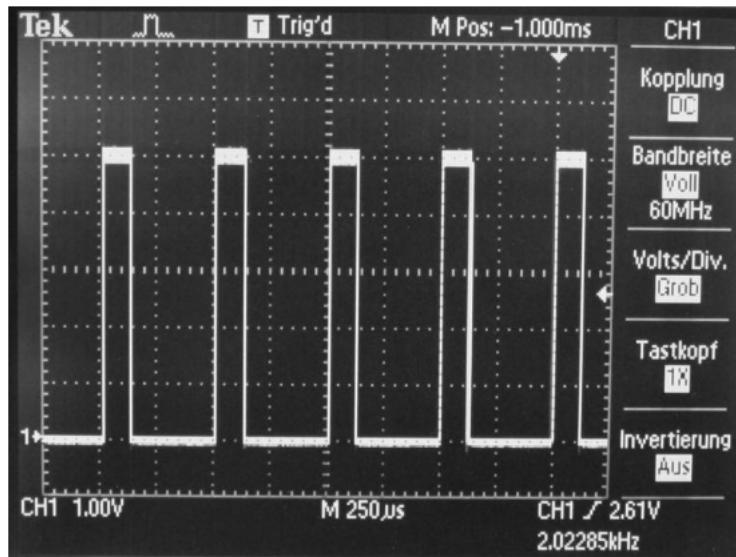


Рис. 9.12. Скважность ШИМ 25%

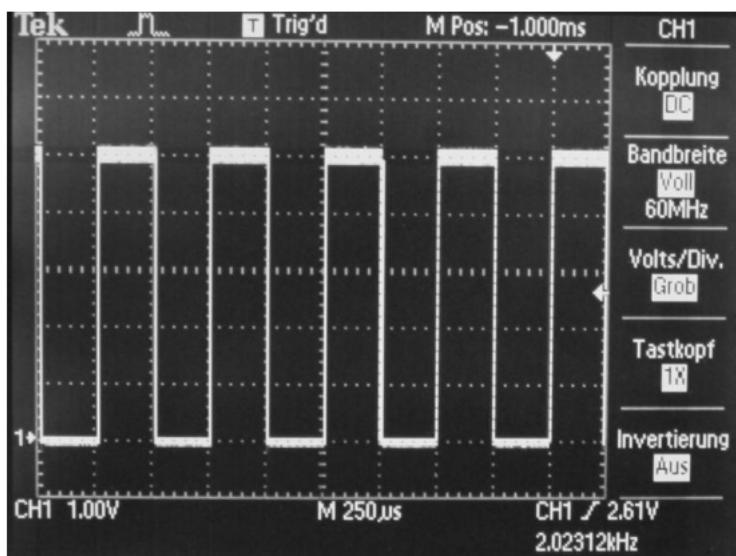


Рис. 9.13. Скважность ШИМ 50%

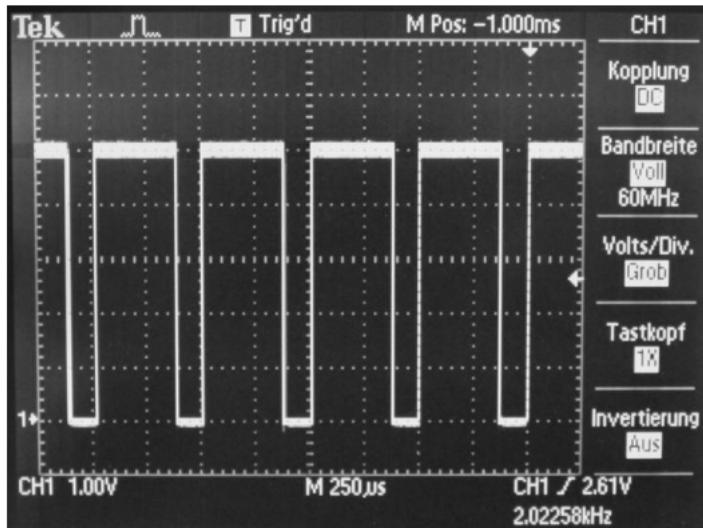


Рис. 9.14. Скважность ШИМ 75%

analogWrite(pin, value)

Эта команда формирует на выходе псевдоаналоговые значения посредством модуляции ширины импульсной последовательности (ШИМ). Значение может устанавливаться в виде переменной или константы в диапазоне 0–255:

```
analogWrite(pin, value); // Запись значения в аналоговый вывод
```

При нулевом значении напряжение на заданном аналоговом выходе тоже будет нулевым. Значение 255 соответствует напряжению 5 В. При значениях между 0 и 255 вывод переключается между 0 и 5 В, чем больше значение, тем дольше вывод имеет высокий уровень (5 В). При значении 64 вывод три четверти периода имеет 0 В и одну четверть — 5 В. Значение 128 приводит к тому, что выходное напряжение одну половину периода времени имеет высокий уровень и вторую — низкий уровень. При значении 192 напряжение на выводе в течение одной четверти периода составляет 0 В и три четверти — полные 5 В. Так как эта функция базируется на аппаратных средствах, то постоянный сигнал запускается независимо от программы вплоть до следующего изменения состояния посредством функции `analogWrite` (или по вызову от `digitalRead` или `digitalWrite` для того же вывода).

ПРИМЕЧАНИЕ

Аналоговые выводы, в отличие от цифровых, не нужно предварительно конфигурировать как вход или выход.

В предлагаемом примере (рис. 9.15, листинг 9.41) светодиод переключается с высокой частотой. В результате значение ШИМ определяет яркость свечения.

Необходимые комплектующие изделия:

- плата микроконтроллера Arduino/Freeduino;
- панель с контактными гнездами;

- гибкий монтажный провод длиной примерно 10 см;
- два гибких монтажных провода длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 1 см;
- резистор 1,5 кОм;
- светодиод красного цвета;
- светодиод зеленого цвета.

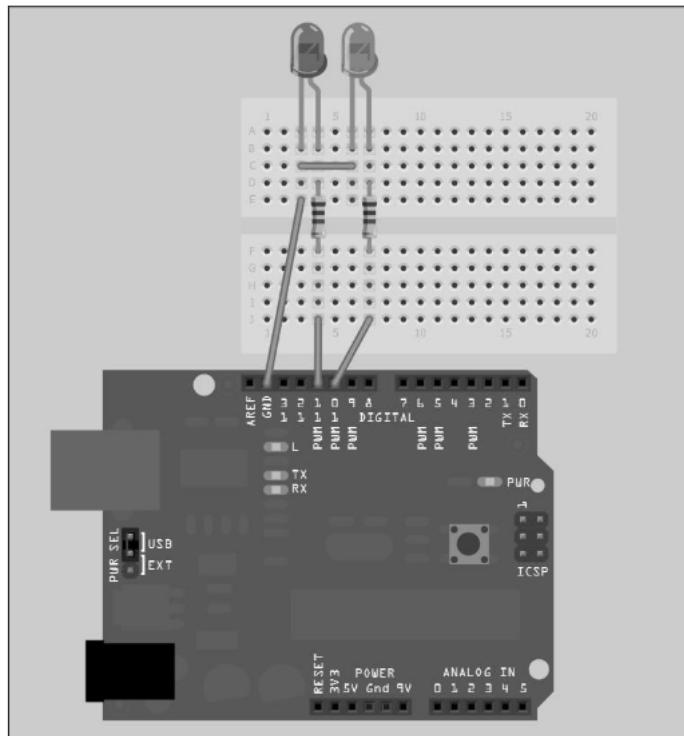


Рис. 9.15. Схема макета

Листинг 9.41. AnalogWrite.pde

```
// Franzis Arduino
// Analog Write

int value;
int LEDgruen=10;
int LEDrot=11;

void setup()
{
  // В этот раз здесь ничего не происходит
}
```

```
}

void loop()
{
    for(value=0;value<255;value++)
    {
        analogWrite(LEDgruen, value);
        analogWrite(LEDrot, 255-value);
        delay(5);
    }

    delay(1000);

    for(value=255;value!=0;value--)
    {
        analogWrite(LEDgruen, value);
        analogWrite(LEDrot, 255-value);
        delay(5);
    }

    delay(1000);
}
```

Если теперь присоединить небольшой пьезоакустический преобразователь к аналоговому выходу (рис. 9.16), то можно "услышать" ШИМ-сигнал. Правда, звук будет негромким.

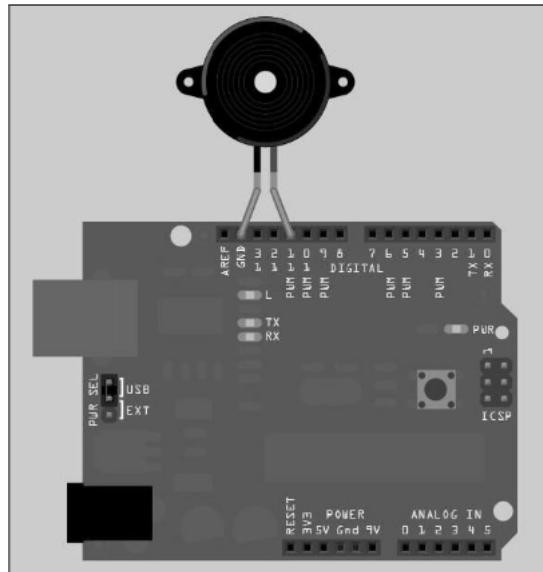


Рис. 9.16. Макет для "прослушивания" ШИМ

9.6. Некоторые специальные функции

Установка паузы с помощью *delay*

В предыдущих примерах мы уже неоднократно задавали паузу при помощи команды `delay()`.

delay(ms)

Останавливает программу на время, указанное в миллисекундах, причем значение 1000 соответствует 1 секунде:

```
delay(1000); // задержка на одну секунду
```

micros()

Останавливает программу на время, указанное в микросекундах, причем значение 1000 соответствует 1 миллисекунде:

```
micros(1000); // задержка на одну миллисекунду
```

Функции случайных чисел

При написании измерительных, управляющих, регулирующих или игровых программ часто требуются случайные числа, например, если в доме в произвольное время должно зажигаться и гаснуть освещение. Для этой цели предусмотрена функция `random`.

randomSeed(seed)

Функция устанавливает значение или начальное число для `random()`:

```
randomSeed(value); // устанавливает значение value как случайное начальное число
```

Применение `randomSeed()` гарантирует лучший результат. Случайные числа можно передать в качестве аргумента, например, функциям `millis()` (см. далее) или `analogRead()`, чтобы сгенерировать электрические помехи на аналоговом выводе.

random(min, max)

Функция `random` осуществляет генерацию псевдослучайных значений в пределах определенного диапазона минимального и максимального значений:

```
value = random(100, 200); // устанавливает value со случайным числом  
// в диапазоне между 100 и 200
```

ПРИМЕЧАНИЕ

Используйте `random(min, max)` после функции `randomSeed()`.

Листинг 9.42 иллюстрирует пример.

Листинг 9.42. Zufallszahlen.pde

```
// Franzis Arduino
// Zufallszahlen

int x,y=0;

void setup()
{
    randomSeed(100);
    Serial.begin(9600);
    Serial.println("Arduino Zufallszahlen");
    Serial.println();
}

void loop()
{

    for(x=0;x<20;x++)
    {
        y=random(0, 10);
        Serial.print(y);
        Serial.print(",");
    }
    Serial.println();

    for(x=0;x<20;x++)
    {
        y=random(10,100);
        Serial.print(y);
        Serial.print(",");
    }
    Serial.println();

    for(x=0;x<20;x++)
    {
        y=random(0,x+1);
        Serial.print(y);
        Serial.print(",");
    }
    Serial.println();

    while(1);
}
```

Вот возможные результаты выполнения листинга 9.42:

0,9,5,5,9,3,2,1,1,9,4,3,9,9,5,6,1,0,4,8
83,24,24,99,92,36,97,35,13,10,43,98,88,52,89,86,29,35,37,58
0,0,1,3,0,1,1,4,6,9,7,3,1,3,5,8,9,9,17,18

Вы получили другие числа? Не удивляйтесь. В конце концов, числа должны быть случайными. Запустите программу несколько раз и удостоверьтесь, что всегда выдаются разные числа. Но результаты всегда будут находиться в заданном диапазоне от минимального до максимального значения.

Еще один пример со случайными числами иллюстрирует листинг 9.43.

Листинг 9.43. Wuerfelspiel.pde

```
// Franzis Arduino
// Würfelspiel

int i,zahl=0;
int Anz=6; // Число бросков

void setup()
{
    Serial.begin(9600);
    Serial.flush();
    randomSeed(6);
}

void loop()
{

    Serial.println("Senden Sie ein Zeichen um zu wuerfeln");

    do
    {
        }while(Serial.available()==0);
    Serial.flush();

    Serial.print("Sie haben folgende Zahlen gewuerfelt: ");

    for(i=0;i<Anz;i++)
    {
        zahl=random(6);
        zahl++;
        Serial.print(zahl);
        Serial.print(" ");
    }

    Serial.println();
}
```

Сколько времени прошло?

Чтобы определить, сколько прошло времени после запуска или экспорта подпрограммы, в Arduino есть специальные функции. Разумеется, можно самому создавать циклы с остановкой всей программы и ожиданием, до тех пор, пока не истечет требуемый интервал времени. Время может определяться в течение миллисекунд или микросекунд.

millis()

Функция возвращает значение времени (в миллисекундах), прошедшее после последнего вызова функции:

```
value = millis(); // возвращает интервал времени в миллисекундах
```

Переменная `unsigned-long` переполняется примерно через 50 дней и обнуляется. Листинг 9.44 иллюстрирует пример.

Листинг 9.44. millis.pde

```
// Franzis Arduino
// Zeitmessung 1

unsigned long value;

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino Zeitmessung 1");
    Serial.println();
}

void loop()
{
    Serial.print("Zeit: ");
    value=millis();
    Serial.println(value);
    delay(1000);
}
```

micros()

Функция возвращает интервал времени (в микросекундах) с момента запуска программы:

```
value = micros(); // возвращает интервал времени в микросекундах
```

Переполнение наступает примерно через 70 минут (при тактовой частоте 16 МГц), и все начинается снова с нуля.

Листинг 9.45 иллюстрирует пример.

Листинг 9.45. micro.pde

```
// Franzis Arduino
// Zeitmessung 1

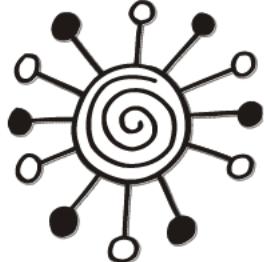
unsigned long value;

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

void loop()
{
    Serial.print("Zeit: ");
    value=millis();
    Serial.println(value);
    delay(1000);
}
```

ПРИМЕЧАНИЕ

1 000 миллисекунд соответствуют 1 000 000 микросекундам.



Глава 10

Дальнейшие эксперименты с Arduino

Вы уже основательно потрудились над фундаментальным курсом и познакомились с программированием Arduino, теперь можно приступать к дальнейшим экспериментам. Далее, опираясь на базовые сведения, мы рассмотрим новые функции и возможности программирования.

10.1. Регулятор уровня яркости светодиода с транзистором

В главе 9 вы уже познакомились с формированием на аналоговом выходе платы Arduino ШИМ-сигнала и теперь можете собрать регулятор яркости свечения светодиода. В предлагаемом эксперименте красный светодиод можно непосредственно подключить к аналоговому выходу (контакт 3). Если взять более яркие светодиоды, например, компании Luxeon, то нужно добавить транзистор в качестве усилителя. С помощью кнопок S1 ("светлее") и S2 ("темнее") можно менять яркость светодиода.

Используемые компоненты:

- светодиод красного цвета;
- две кнопки;
- транзистор BC548C;
- резистор 1,5 кОм;
- резистор 4,7 кОм;
- пять гибких монтажных проводов длиной примерно 10 см;
- два гибких монтажных провода длиной примерно 5 см.

Листинг 10.1 иллюстрирует пример.

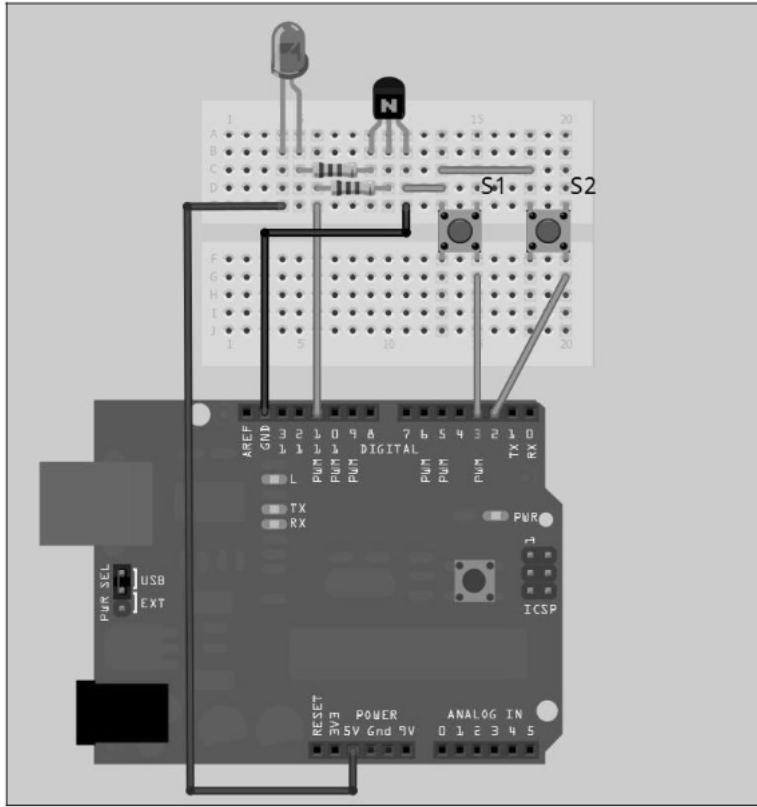


Рис. 10.1. Схема регулятора яркости свечения светодиода с транзистором

Листинг 10.1. LED_Dimmer.pde

```
// Franzis Arduino
// Регулятор уровня яркости светодиода

int helligkeit=0;
int SW1=3;
int SW2=2;
int LED=11;

void setup()
{
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    pinMode(SW2,INPUT);
    digitalWrite(SW2,HIGH);
```

```
}

void loop()
{
    if(!digitalRead(SW1) &&digitalRead(SW2) )
    {
        if(helligkeit<255)helligkeit++;
        analogWrite(LED,helligkeit);
        delay(10);
    }
    else if(digitalRead(SW1) &&!digitalRead(SW2) )
    {
        if(helligkeit!=0)helligkeit--;
        analogWrite(LED,helligkeit);
        delay(10);
    }
}

}
```

В примере показано применение функции `not` (!) в операторе `if`. Нажимать можно только на одну из кнопок. При одновременном нажатии на обе кнопки никаких действий не происходит.

10.2. Плавное мигание

При помощи функции `sin()` на аналоговом выходе можно получить сигнал синусоидальной формы. Светодиод при этом плавно включается и выключается, что полезно для некоторых применений. Процесс выглядит так, как будто на экспериментальной плате пульсирует сердце.

Для эксперимента подойдет такая же схема, как и в предыдущем примере (см. рис. 10.1). В листинге 10.2 основная программа содержит цикл, в котором происходит счет от 1 до 255. Значения берутся из таблицы синусов, организованной в виде массива, и передаются в канал ШИМ. Этот метод значительно быстрее и экономичнее, чем расчет значений в микроконтроллере. На компакт-диске, прилагаемом к книге, есть вспомогательная программа Sin Tab (расчет таблицы синусов), окно которой изображено на рис. 10.2.

Листинг 10.2. SinusBlinker.pde

```
// Franzis Arduino
// Плавное мигание с помощью табличных значений

byte i=0;
int LED=11;
```

```

byte Data[] =
{128,131,134,137,140,144,147,150,153,156,159,162,165,168,171,174,177,180,182,185,
188,191,194,196,199,201,204,206,209,211,214,216,218,220,222,224,226,228,230,23
2,234,236,237,239,
240,242,243,244,246,247,248,249,250,251,251,252,253,253,254,254,254,254,255,255,25
5,255,255,255,255,
254,254,253,253,252,252,251,250,249,248,247,246,245,244,242,241,240,238,236,23
5,233,231,229,227,
225,223,221,219,217,215,212,210,208,205,203,200,197,195,192,189,187,184,181,17
8,175,172,169,167,
164,160,157,154,151,148,145,142,139,136,133,130,126,123,120,117,114,111,108,10
5,102,99,96,92,89,
87,84,81,78,75,72,69,67,64,61,59,56,53,51,48,46,44,41,39,37,35,33,31,29,27,25,
23,21,20,18,16,15,
14,12,11,10,9,8,7,6,5,4,4,3,3,2,2,1,1,1,1,1,1,2,2,2,3,3,4,5,5,6,7,8,9,10,12,
13,14,16,17,19,20,
22,24,26,28,30,32,34,36,38,40,42,45,47,50,52,55,57,60,62,65,68,71,74,76,79,82,
85,88,91,94,97,100,
103,106,109,112,116,119,122,125,128};

void setup()
{
    // здесь пусто...
}

void loop()
{
    for(i=0;i<255;i++)
    {
        analogWrite(LED,Data[i]);
        delay(5);
    }
}

```

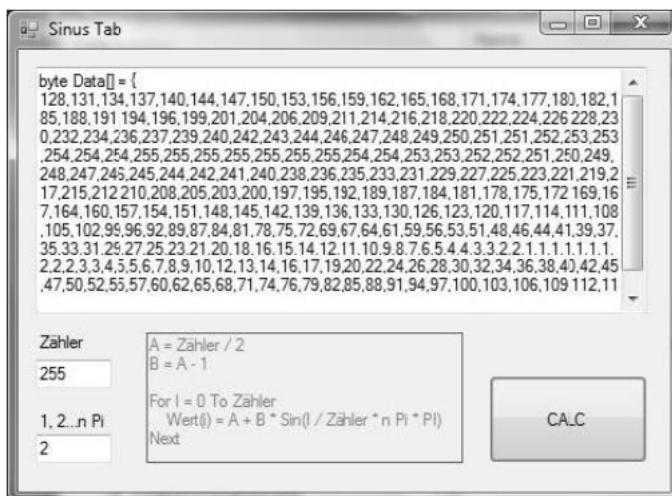


Рис. 10.2. Программа Sinus Tab

Выполнить расчеты, сформировать таблицу значений синуса и даже увидеть форму ШИМ-сигнала можно в программе VB.NET. Рассчитанные значения можно записывать непосредственно в контроллер Arduino.

При наличии осциллографа, можно подключить вместо светодиода RC-цепь в аналоговый выход и наблюдать сигнал синусоидальной формы на экране осциллографа.

Листинг 10.3. SoftBlinkerSinFunc.pde

```
// Franzis Arduino
// Плавное мигание при помощи функции Sinus

int ledPin = 11;
float Val;
int led;

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

void loop()
{
    for (int x=0; x<180; x++)
    {
        Val = (sin(x*(3.1412/180)));
        led = int(Val*255);
        analogWrite(ledPin, led);
        delay(10);
    }
}
```

Во втором примере (листинг 10.3) демонстрируется, как можно реализовать плавное мигание в Arduino через функцию `sinus`. Программа значительно меньше по объему, однако она явно сильнее нагружает микроконтроллер непосредственным вычислением и вследствие этого существенно увеличивается время выполнения.

Для функции `sin()` требуется значение в радианах. Для этого нужно пересчитывать значения в выражении `x * (Pi/180)`. Если теперь умножить результат на 255, то при изменении значения `x`, мы получим синусоиду, значения которой находятся между 0 и 255.

10.3. Подавление дребезга контактов кнопок

Дребезг контактов кнопки возникает из-за особенностей конструкции кнопки. Всякий раз при нажатии и отпускании кнопки сигнал не сразу будет принимать низкий или высокий уровень, а возникнут колебания (дребезг). При подключении лампы накаливания этот эффект незаметен, т. к. длительность колебаний мала, а лампа инерционна. Контроллер работает настолько быстро, что воспринимает эти колебания как несколько включений и отключений (рис. 10.3). Устранить дребезг можно программным путем.

Эту проблему можно решить просто. Нужно дважды с небольшим временным интервалом запросить состояние кнопки. Если сигнал будет одинаковым, то можно считать, что положение кнопки зафиксировано. Интервал между опросами должен быть 20–100 мс. Схема эксперимента приведена на рис. 10.4, а код программы — в листинге 10.4.



Рис. 10.3. Так выглядят дребезги контактов кнопки

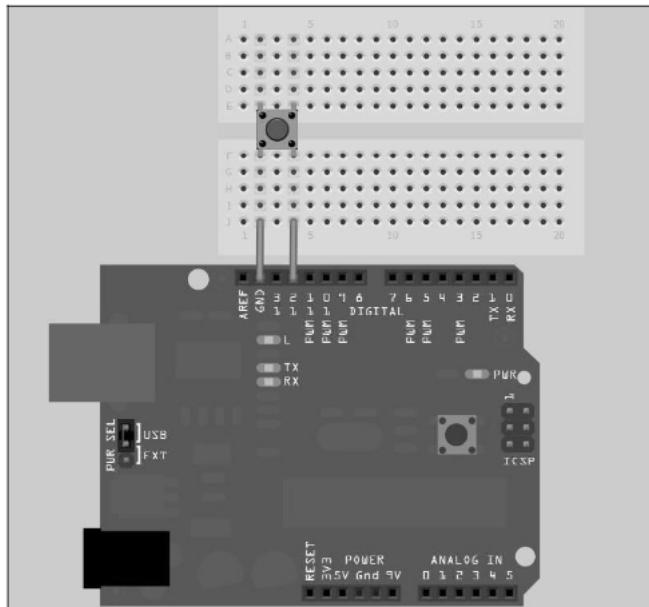


Рис. 10.4. Схема эксперимента

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- кнопка;
- два гибких монтажных провода длиной примерно 5 см.

Листинг 10.4. Taster_Prellen_V1.pde

```
// Franzis Arduino
// Дребезг контактов кнопки, V1

byte i=0;
int SW1=12;

void setup()
{
    Serial.begin(9600);
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    Serial.println("Taster entprellen V1");
}

void loop()
{

    if(!digitalRead(SW1))
    {
        delay(50);
        if(!digitalRead(SW1))
        {
            Serial.println("Taster SW1 wurde gedrueckt");
        }
    }
}
```

Недостаток рассмотренного метода состоит в том, что программа всегда вызывается многократно, пока не будет отпущена кнопка. Другой вариант (листинг 10.5) предусматривает запуск кода программы и после этого ожидание до тех пор, пока кнопка не будет снова отпущена.

Листинг 10.5. Taster_Prellen_V2.pde

```
// Franzis Arduino
// Дребезг контактов кнопки, V2

byte i=0;
int SW1=12;

void setup()
{
    Serial.begin(9600);
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    Serial.println("Taster entprellen V2");
}

void loop()
{

    if(!digitalRead(SW1))
    {
        delay(50);
        if(!digitalRead(SW1))
        {
            i++;
            Serial.print("Taster SW1 wurde ");
            Serial.print(i,DEC);
            Serial.println("x gedrueckt");
            do{
                }while(!digitalRead(SW1));
        }
    }
}
```

Обратный алгоритм реализуется, если цикл `do-while` установить в начале (листинг 10.6). Здесь код запускается только после размыкания контактов кнопки.

Листинг 10.6. Taster_Prellen_V3.pde

```
// Franzis Arduino
// Дребезг контактов кнопки, V3

byte i=0;
```

```
int SW1=12;

void setup()
{
    Serial.begin(9600);
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    Serial.println("Taster entprellen V3");
}

void loop()
{

    if(!digitalRead(SW1))
    {
        delay(50);
        if(!digitalRead(SW1))
        {
            do{
                }while(!digitalRead(SW1));
            i++;
            Serial.print("Taster SW1 wurde ");
            Serial.print(i,DEC);
            Serial.println("x gedrueckt");

        }
    }
}
```

Но самый лучший вариант приведен в листинге 10.7. Здесь объединены последние примеры, выполняется не только двойной опрос, но дополнительно результаты сравниваются друг с другом. Для запуска кода значение `digitalRead` должно дважды совпасть за определенный период. Теперь для этого мы еще включаем или выключаем светодиод L.

Листинг 10.7. Taster_Prellen_V4.pde

```
// Franzis Arduino
// Дребезг контактов кнопки, v4

byte i=0;
int SW1=3;
```

```
int LED=13;
int TOG=0;
byte value_1, value_2=0;

void setup()
{
    Serial.begin(9600);
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    pinMode(LED,OUTPUT);
    Serial.println("Taster entprellen V4");
}

void loop()
{

    value_1=digitalRead(SW1);
    if(!value_1)
    {
        delay(50);
        value_2=digitalRead(SW1);
        if(!value_2)
        {
            i++;
            Serial.print("Taster SW1 wurde ");
            Serial.print(i,DEC);
            Serial.println("x gedrueckt");
            if(TOG!=0)TOG=0;else TOG=1;
            digitalWrite(LED,TOG);
            do{
                }while(!digitalRead(SW1));

        }
    }
}
```

10.4. Задержка включения

Задержка включения, как ясно из названия, задерживает подачу питания на нагрузку (в нашем случае светодиод) после нажатия кнопки (или переключателя). Ожидание реализуется в нашем примере командой `delay()` и циклом (листинг 10.8). При нажатии на кнопку переменная `flag` (флаг состояния) сохраняет

состояние и увеличивает переменную *i*. Значение *i* больше, чем заявленное время (здесь 3 000, что соответствует трем секундам), светодиод L включается и программа остается в цикле `while(1)`. Кнопка снова подключается к цифровому выводу 12 и GND.

Листинг 10.8. Einschaltverzögerung.pde

```
// Franzis Arduino
// Задержка включения

int SW1=12;
int value_1, value_2=0;
int LED=13;
byte Flag=0;
int i=0;

void setup()
{
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    pinMode(LED,OUTPUT);
}

void loop()
{

    value_1=digitalRead(SW1);
    if(!value_1)
    {
        delay(50);
        value_2=digitalRead(SW1);
        if(!value_2)
        {
            Flag=1;
            do{
                }while(!digitalRead(SW1));
        }
    }
    if(Flag==1)i++;
    if(i>3000)
    {
        digitalWrite(LED,HIGH);
    }
}
```

```
    while(1);
}

delay(1);

}
```

10.5. Задержка выключения

Это задача, обратная предыдущей. Здесь нагрузка отключается при нажатии клавиши по истечении установленного времени. Метод идентичен рассмотренному ранее, только здесь переменная *i* не увеличивается, а уменьшается (обратный отсчет). Код программы приведен в листинге 10.9.

Листинг 10.9. Ausschaltverzögerung.pde

```
// Franzis Arduino
// Задержка выключения

int SW1=12;
int value_1, value_2=0;
int LED=13;
byte Flag=0;
int i=3000;

void setup()
{
    pinMode(SW1, INPUT);
    digitalWrite(SW1, HIGH);
    pinMode(LED, OUTPUT);
    digitalWrite(LED, HIGH);
}

void loop()
{

    value_1=digitalRead(SW1);
    if(!value_1)
    {
        delay(50);
        value_2=digitalRead(SW1);
        if(!value_2)
        {
            if(i>0)
                i--;
            else
                digitalWrite(LED, LOW);
        }
    }
}
```

```
Flag=1;
do{
}while(!digitalRead(SW1));
}

if(Flag==1)i--;
if(i==0)
{
    digitalWrite(LED, LOW);
    while(1);
}

delay(1);

}
```

10.6. Светодиоды и Arduino

В большинстве описанных экспериментов для тестирования программного обеспечения применялись один или несколько светодиодов. Человеку, не являющемуся инженером-электронщиком, наверняка интересно знать, как рассчитывается добавочное сопротивление.

Светодиод можно рассматривать как обычный диод, включенный в прямом направлении (анод подключен к плюсу источника питания и катод — к общему проводу). В этом случае на светодиоде падает напряжение, которое зависит от цвета излучения (1,6–3,5 В). Точные данные можно найти в паспорте светодиода. Там будет указано прямое напряжение V_f (Forward Voltage). Для свечения светодиода требуется определенный ток. В паспорте указан ток I_f .

Пример расчета.

Пусть прямой ток $I_f = 2$ мА (светодиод с малым рабочим током). Прямое падение напряжения $V_f = 2,2$ В. Напряжение источника питания $V_{CC} = 5$ В.

Искомая величина — сопротивление добавочного резистора $R = ?$ Ом.

$$R = (V_{CC} - V_f) / I_f.$$

Для нашего примера $R = (5 \text{ В} - 2,2 \text{ В}) / 2 \text{ мА} = 1\,400 \text{ Ом}$.

Целесообразно выбрать сопротивление из ряда Е12, несколько большего номинала. Для надежности ставится резистор 1,5 кОм, чтобы исключить повреждение светодиода. Если внимательно рассмотреть монтажную схему экспериментальной платы, то можно обнаружить перед диодами сопротивление 1,5 кОм.

Рассмотрим практический пример включения светодиодов с попеременным миганием (рис. 10.5). Здесь светодиоды, подключенные к выводам 10 и 11, вспыхивают попеременно, что дает в итоге световой эффект, похожий на мигалку машины скорой помощи, даже если цвет синий, а не красный.

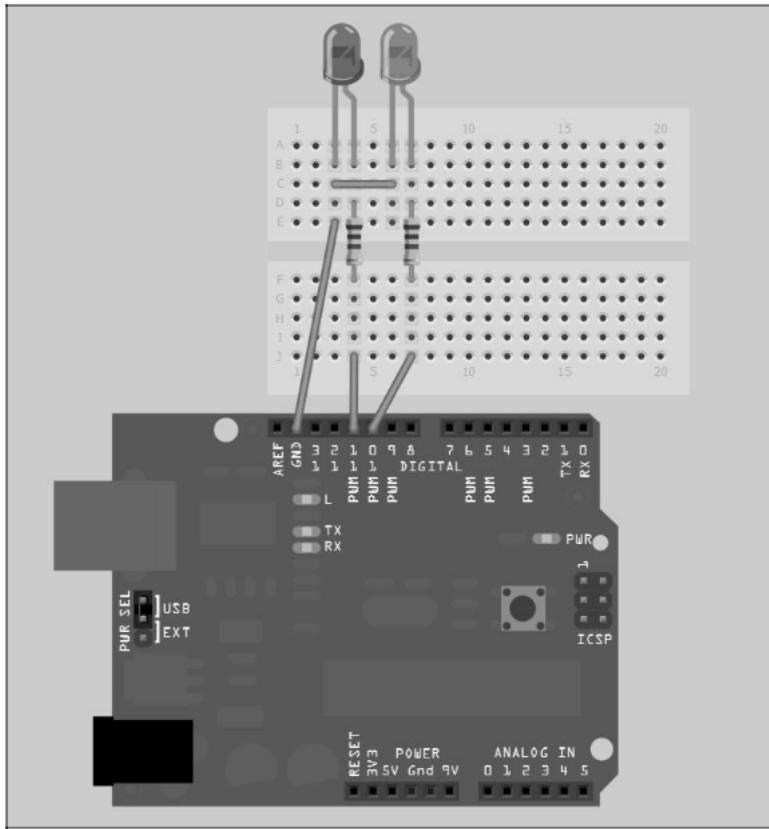


Рис. 10.5. Монтажная схема

Используемые компоненты:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- два светодиода красного цвета;
- два резистора 1,5 кОм;
- три гибких монтажных провода длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

Код программы приведен в листинге 10.10.

Листинг 10.10. Blitz.pde

```
// Franzis Arduino
// Попеременное мигание
```

```
int LED_1=10;
int LED_2=11;
int i=0;
```

```
int TOG=0;

void setup()
{
    pinMode(LED_1,OUTPUT);
    pinMode(LED_2,OUTPUT);
}

void loop()
{

    for(i=0;i<3;i++)
    {
        if(TOG==0) TOG=HIGH;else TOG=LOW;
        digitalWrite(LED_1,TOG);
        delay(40);
    }
    digitalWrite(LED_1,LOW);
    delay(100);

    for(i=0;i<3;i++)
    {
        if(TOG==0) TOG=HIGH;else TOG=LOW;
        digitalWrite(LED_2,TOG);
        delay(40);
    }
    digitalWrite(LED_2,LOW);
    delay(100);

    for(i=0;i<3;i++)
    {
        if(TOG==0) TOG=HIGH;else TOG=LOW;
        digitalWrite(LED_1,TOG);
        delay(40);
    }
    digitalWrite(LED_1,LOW);
    delay(500);

}
```

10.7. Подключение больших нагрузок

Если требуемый ток нагрузки больше, чем 40 мА, нужно предусмотреть дополнительный транзисторный усилитель тока (рис. 10.6). В транзисторе маленький ток, протекая через базу, обеспечивает большой тока коллектора. Коеффициент передачи тока для маломощных транзисторов составляет 100–1 000, в зависимости от примененного типа транзистора. Используемый в нашей схеме транзистор BC548C (рис. 10.7) имеет усиление по току примерно 300. При токе базы 0,1 мА ток в цепи коллектора будет около 30 мА. Максимальный ток в цепи коллектора для нашего транзистора равен 100 мА. Через эмиттер течет и ток базы, и ток коллектора.

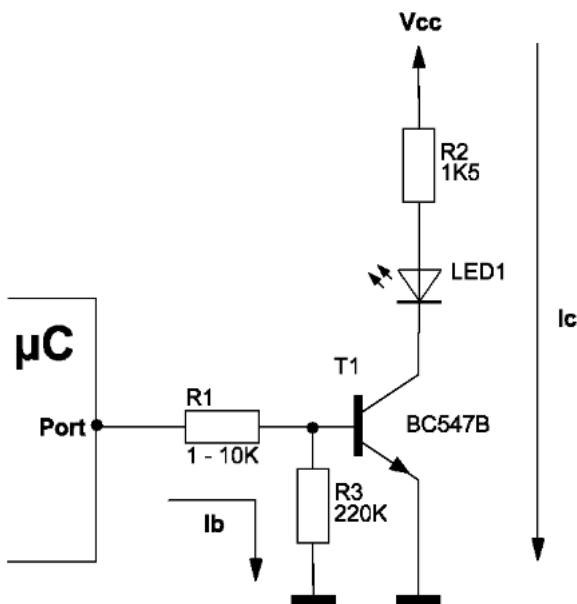


Рис. 10.6. Подключение светодиода через транзистор-усилитель

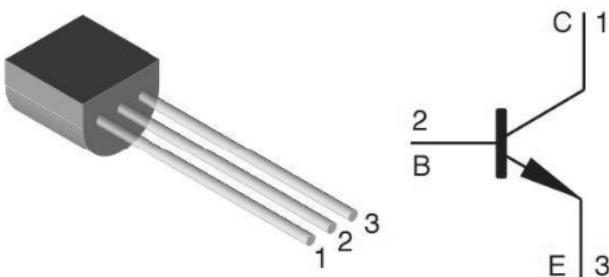


Рис. 10.7. Расположение выводов транзистора BC548
(источник: технический паспорт компании Vishay)

Сопротивление резистора R1 находится в диапазоне от 1 и 10 кОм. При использовании транзистора BC548C для полного свечения светодиода достаточно сопротивления 10 кОм.

Резистор R3 служит для устранения помех базы. Выводы микроконтроллера при включении находятся в высокоимпедансном состоянии. Таким образом, база "висела" бы в воздухе. Чтобы этого не случилось, требуется подключить сопротивление 220–470 кОм между базой и общим проводом. Тогда при протекании через базу большого тока гарантируется срабатывание транзистора. Чем больший ток требуется для нагрузки, тем больше должен быть ток базы.

Ток коллектора равен произведению тока базы на коэффициент передачи по току:

$$I_c = I_b \times \beta.$$

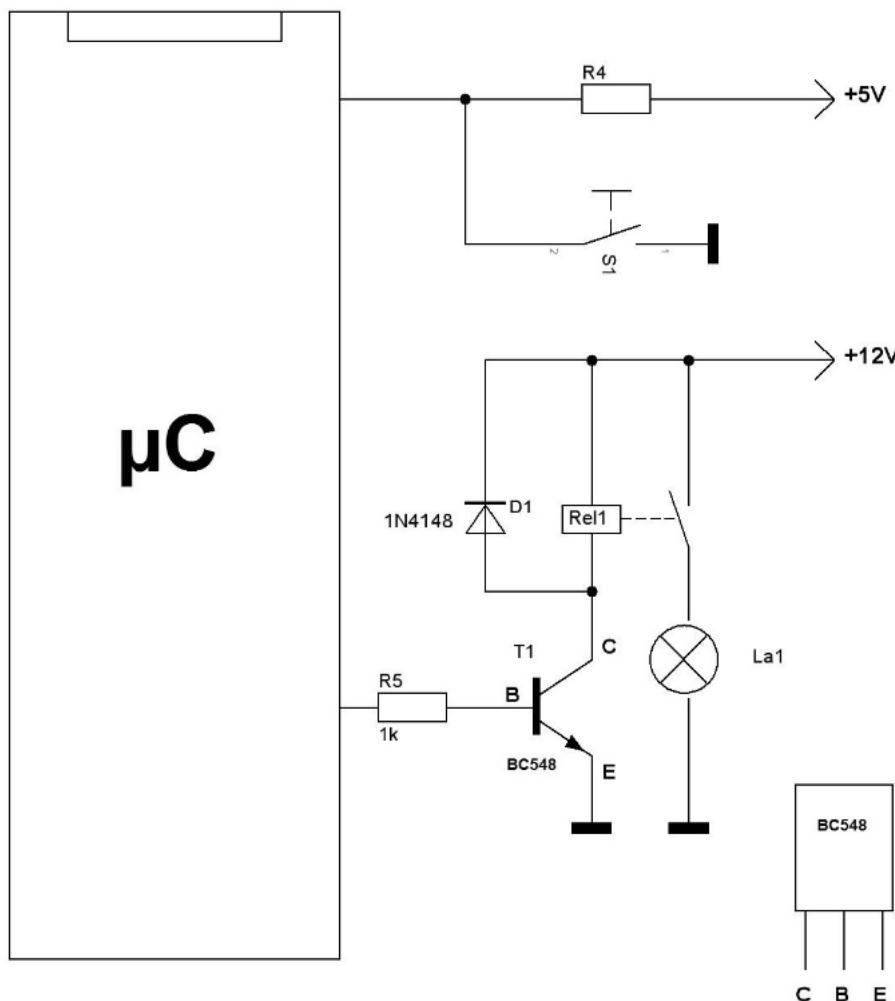


Рис. 10.8. Подключение реле к порту

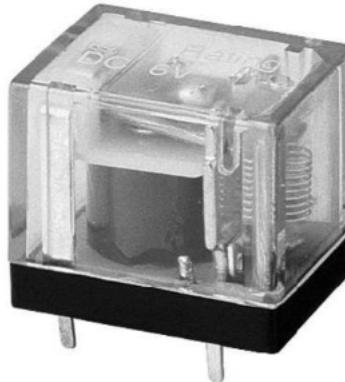


Рис. 10.9. Внешний вид маломощного реле

Через транзистор можно подключать, например, реле (рис. 10.8). Реле выпускают в самых различных исполнениях (рис. 10.9). Коммутационные контакты реле гальванически не соединяются со схемой микроконтроллера. Важно, чтобы параллельно обмотке реле был подключен безынерционный диод для защиты транзистора от импульсов обратной полярности. Диод подключается всегда в обратной полярности по отношению к напряжению питания обмотки, т. к. напряжение самоиндукции направлено противоположно.

10.8. ЦАП на основе ШИМ-порта

Большинство происходящих в природе процессов — аналоговые. Поэтому для управления внешними процессами необходимо преобразовывать цифровые значения в аналоговые величины. Если требуется аналоговое напряжение, то к аналоговому выходу следует присоединить RC-цепь (рис. 10.10), формирующую аналоговое напряжение из сигнала ШИМ. Большинство микроконтроллеров, в том числе и микроконтроллер Arduino, не имеют встроенногоцифроаналогового преобразователя. Разумеется, посредством ШИМ-сигнала можно также осуществитьцифроанalogовое преобразование и получить постоянное напряжение. Если пропустить ШИМ-сигнал через фильтр низких частот, то на выходе будет постоянное напряжение с переменной составляющей, среднее значение которой соответствует среднему значению сигнала ШИМ.

Параметры RC-цепи рассчитывают исходя из следующего соотношения:

$$F = 1/(2\pi RC),$$

где F — частота среза сигнала ШИМ, R — сопротивление в омах, C — емкость в фарадах.

Используемый в нашем примере RC-фильтр низких частот состоит из конденсатора 1 мКФ и резистора 10 кОм и имеет частоту среза 15 Гц.

Конечно, подобную RC-цепь нельзя сильно нагружать, из-за увеличения пульсаций. Лучше всего установить за RC-цепью еще выходной каскад, чтобы увеличить

уровень сигнала до желаемого. В нашем эксперименте напряжение измеряется на аналоговом входе (контакт 0). Монтаж экспериментальной схемы изображен на рис. 10.11. Код программы приведен в листинге 10.11.

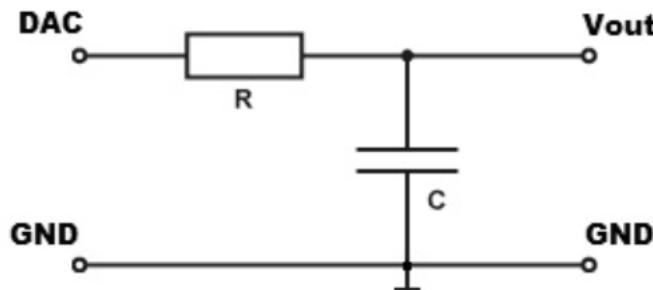


Рис. 10.10. Схема RC-цепи для фильтрации ШИМ-сигнала

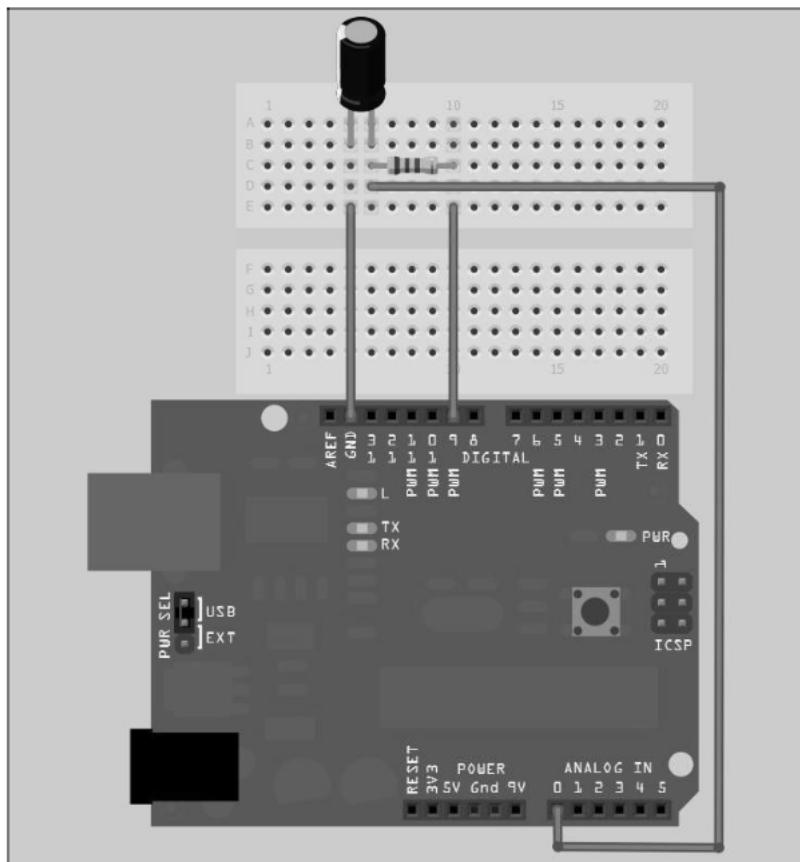


Рис. 10.11. Монтаж схемы ЦАП с выходом ШИМ

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- конденсатор 1 мкФ;
- резистор 10 кОм;
- два гибких монтажных провода длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

Листинг 10.11. DAC.pde

```
// Franzis Arduino
// ЦАП

char buffer[18];
int pinPWM=9;
int raw=0;
float Volt=0;

void setup()
{
    Serial.begin(9600);
    Serial.println("DAC mit PWM-Ausgang");
    Serial.println();
    Serial.println("Geben Sie einen Wert zwischen 0 und 255 ein");
    Serial.flush();
}

void loop()
{
    if (Serial.available() > 0)
    {
        int index=0;
        delay(100); // ожидание символов в буфере
        int numChar = Serial.available();
        if (numChar>15)
        {
            numChar=15;
        }
        while (numChar--)
        {
            buffer[index++] = Serial.read();
        }
    }
}
```

```
    splitString(buffer);
}
}

void splitString(char* data)
{
    Serial.print("Empfangen wurde der Wert: ");
    Serial.println(data);
    char* parameter;
    parameter = strtok (data, " ,");
    while (parameter != NULL)
    {
        setPWM(parameter);
        parameter = strtok (NULL, " ,");
    }
    // Снова очистка буфера
    for (int x=0; x<16; x++)
    {
        buffer[x]='\0';
    }
    Serial.flush();
}

void setPWM(char* data)
{
    int Ans = strtol(data, NULL, 10);
    Ans = constrain(Ans,0,255);
    analogWrite(pinPWM, Ans);
    Serial.print("PWM = ");
    Serial.println(Ans);
    delay(100);
    raw=analogRead(0);
    float ref=5.0/1024.0;
    Volt=raw*ref;
    Serial.print("Die Spannung am ADC0 betraegt: ");
    Serial.print(Volt);
    Serial.println(" Volt");
    Serial.println();
    Serial.println("Geben Sie einen Wert zwischen 0 und 255 ein");
}
```

В рассмотренном примере сигнал ШИМ формируется на аналоговом выходе 9. Сигнал фильтруется RC-цепью и поступает на аналоговый вход 0. Чем меньше введенное значение, тем меньше выходное напряжение на фильтре. Время установки напряжения с 0 до 5 В или с 5 до 0 В составляет примерно 40 мс. Листинг 10.11 также иллюстрирует, как можно оформить ввод данных через последовательный интерфейс.

10.9. С музыкой все веселей

Arduino может быть "музыкален". Чтобы узнать, насколько это так, присоединим пьезоакустический преобразователь к экспериментальной плате, как показано на рис. 10.12.

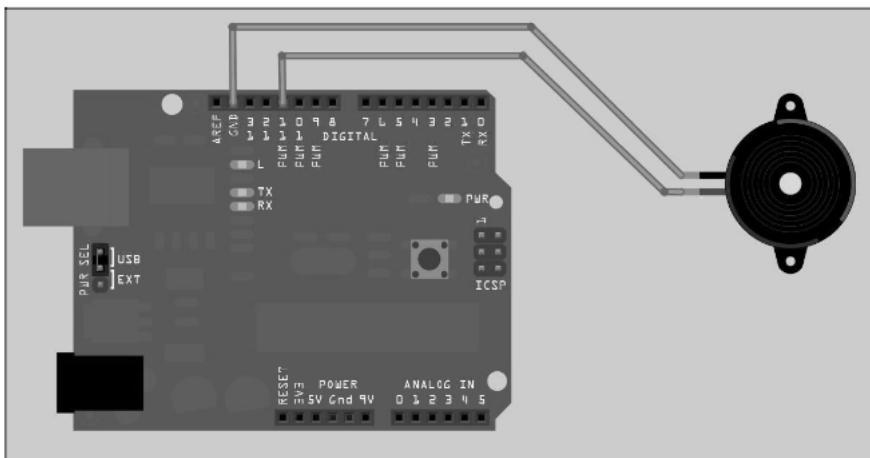


Рис. 10.12. Монтаж схемы с пьезоакустическим преобразователем

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- пьезоакустический преобразователь.

Соединим пьезоакустический преобразователь с цифровым выходом 11 и GND.

Для генерации звуков Arduino предлагает нам команду `tone()`. С ее помощью можно формировать звуковой сигнал на любом выводе. Генерация частоты происходит чисто программным путем (листинг 10.12).

В команде есть еще различные параметры: номер вывода, частота и длительность звукового сигнала:

```
tone(pin, frequency); // Устанавливает непрерывный тон
tone(pin, frequency, duration) // Звук определенной длительности
```

Листинг 10.12. Sound.pde

```
// Franzis Arduino
// Звук

int Speaker=8;

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

void loop()
{
    tone(Speaker,550,450);
    delay(3000);
}
```

Команда `tone()` формирует только простые звуки. Как можно создать собственную программу для генерации мелодий, поясняет листинг 10.13. Здесь звуки генерируются переключением цифрового вывода между высоким (`HIGH`) и низким (`LOW`) уровнем:

```
digitalWrite(Speaker, HIGH);
delayMicroseconds(tone);
digitalWrite(Speaker, LOW);
delayMicroseconds (tone);
```

При запуске кода листинга 10.13 на цифровом выходе формируется мелодия. Период повторения импульсов задается в функции `tone()`.

Листинг 10.13. Melodie.pde

```
// Franzis Arduino
// Мелодия

int Speaker = 8;
int length = 15;
char notes[] = "ccggaagffeeddc ";
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void setup()
{
```

```
pinMode(Speaker, OUTPUT);
}

void loop()
{
    for (int i = 0; i < length; i++)
    {
        if (notes[i] == ' ')
        {
            delay(beats[i] * tempo);
        }
        else
        {
            playNote(notes[i], beats[i] * tempo);
        }

        delay(tempo / 2);
    }
}

void playTone(int tone, int duration)
{
    for (long i = 0; i < duration * 1000L; i += tone * 2)
    {
        digitalWrite(Speaker, HIGH);
        delayMicroseconds(tone);
        digitalWrite(Speaker, LOW);
        delayMicroseconds(tone);
    }
}

void playNote(char note, int duration)
{
    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'c' };
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
    for (int i = 0; i < 8; i++)
    {
        if (names[i] == note)
        {
            playTone(tones[i], duration);
        }
    }
}
```

СОВЕТ

Если приделать бумажный рупор к пьезоакустическому преобразователю, то звук станет значительно громче. Даже если просто при克莱ить к "пищалке" небольшой кусок бумаги, громкость увеличится.

10.10. Романтичный свет свечи с помощью микроконтроллера

Кто не любит уютные вечера при свете свечи, когда огонь мерцает и создает романтическую атмосферу? С помощью Arduino и трех светодиодов (рис. 10.13), а также генератора случайных чисел можно имитировать свет свечи с мерцанием. Программа (листинг 10.14) использует аналоговые выходы микроконтроллера. Однако теперь на выходе не постоянное значение или синусоидальный сигнал, а последовательность импульсов случайной длительности. Светодиоды (красного и желтого цвета) всегда имеют разную яркость. Если установить светодиоды в стеклянный сосуд молочного цвета и оклеить крышку алюминиевой фольгой, свет будет распределяться равномерно. Тогда уже будет сложно определить, светодиоды там или настоящие свечи. Можно поступить еще проще: склеить маленькую бумажную коробочку ($10 \times 10 \times 10$ см) и поместить внутрь нее светодиоды.

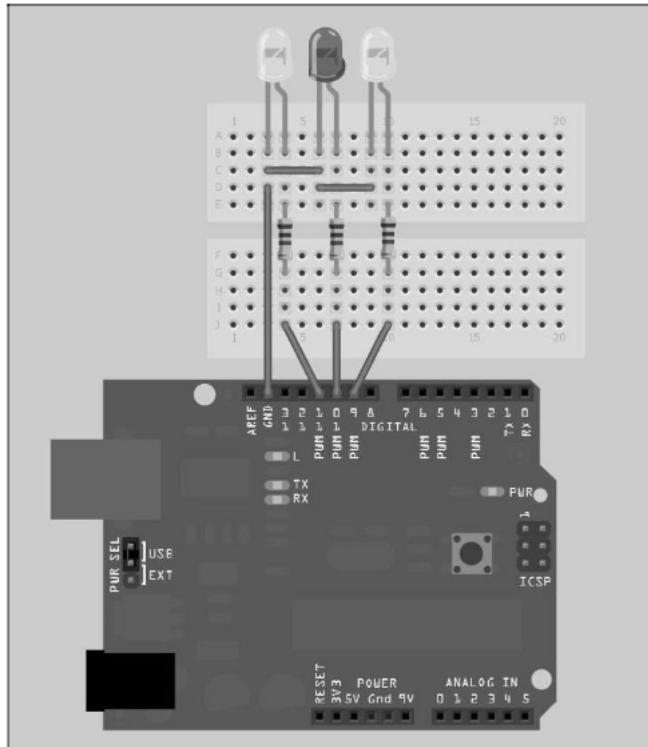


Рис. 10.13. Схема для имитации света свечи

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- светодиод красного цвета;
- два светодиода желтого цвета;
- резистор 1 кОм;
- три гибких монтажных провода длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

ПРИМЕЧАНИЕ

Средний светодиод, подключенный к контакту 10, должен быть красного цвета.

Листинг 10.14. Kerzenlicht.pde

```
// Franzis Arduino
// Свет свечи

int led_gelb1 = 9;
int led_rot = 10;
int led_gelb2 = 11;

void setup()
{
    pinMode(led_gelb1, OUTPUT);
    pinMode(led_rot, OUTPUT);
    pinMode(led_gelb2, OUTPUT);
}

void loop()
{
    analogWrite(led_gelb1, random(120)+135);
    analogWrite(led_rot, random(120)+135);
    analogWrite(led_gelb2, random(120)+135);
    delay(random(100));
}
```

10.11. Контроль персонала на проходной

На большинстве предприятий требуется проверка персонала на проходной. Это можно осуществить с помощью генератора случайных чисел. Установка, которая случайно выбирает отдельных людей для досмотра, позволяет предотвратить совершение краж сотрудниками. Принцип относительно прост: на проходной устанавливается кнопка (звуковой сигнализатор), на которую каждый сотрудник должен

нажать один раз при уходе с территории. При некоторых нажатиях кнопки (за счет генератора случайных чисел) будет включена красная лампа или сирена, сигнализирующие о дополнительном досмотре сотрудника. Затем генератор случайных чисел рассчитывает новое число, и все начинается снова. "Чувствительность" можно устанавливать при помощи потенциометра. Чем больше значение на аналоговом входе (контакт 0), тем меньше выдается тревожных сообщений. Монтаж схемы показан на рис. 10.14. Код программы приведен в листинге 10.15.

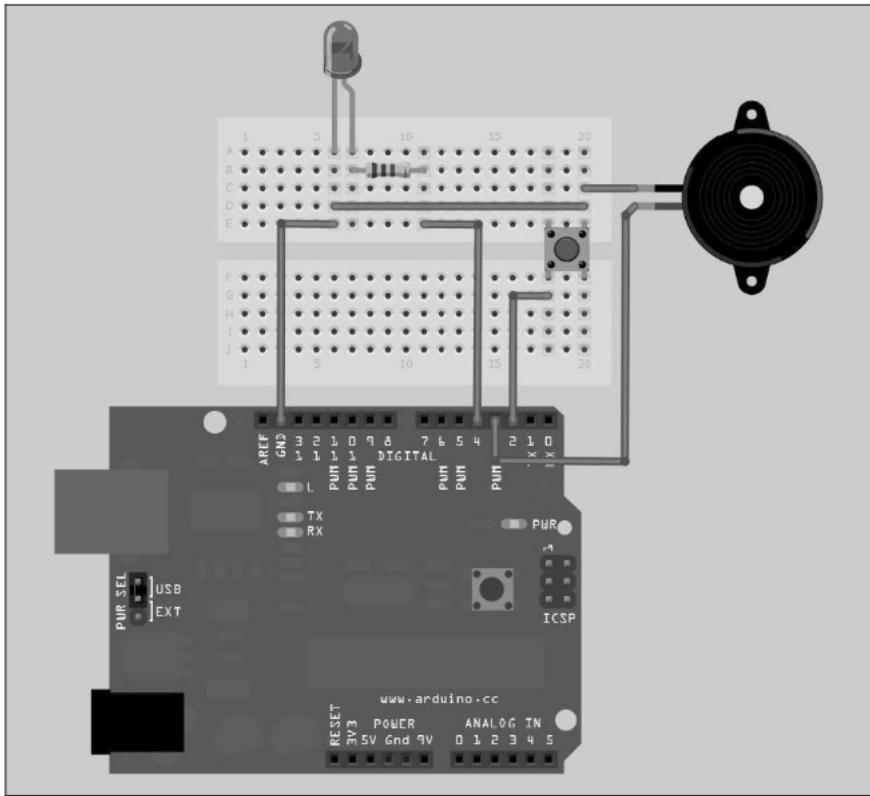


Рис. 10.14. Монтаж схемы соединений

Используемые компоненты:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- светодиод красного цвета;
- пьезопреобразователь;
- резистор 1,5 кОм;
- кнопка;
- три гибких монтажных провода длиной примерно 5 см.

Листинг 10.15. Personalausgang.pde

```
// Franzis Arduino
// Контроль персонала на проходной

int i,x=0;
int LED=3;
int SW1=4;
int Empfindlichkeit=0;
int Speaker=3;
int Person=0;

void setup()
{
    pinMode(LED,OUTPUT);
    pinMode(Speaker,OUTPUT);
    pinMode(SW1,INPUT);
    digitalWrite(SW1,HIGH);
    randomSeed(1000);
}

void loop()
{

Person=(77+analogRead(Empfindlichkeit)/10);
i=random(1,Person);

while(1)
{
    if(!digitalRead(SW1))
    {
        delay(50);
        if(!digitalRead(SW1))
        {
            if(x>Person)x=0;
            if(i==x)
            {
                digitalWrite(LED,HIGH);
                tone(Speaker,500,250);
                delay(3000);
                digitalWrite(LED,LOW);
            }
        }
    }
}
```

```
        break;  
    }  
  
    x++;  
  
}  
}  
}  
}
```

10.12. Часы реального времени

Во многих приложениях требуются часы с программным управлением (RTC — Real Time Clock). Это может быть простое реле времени, срабатывающее в точно установленный срок, или счетчик рабочего времени. Вывод на терминал тоже происходит по секундам. Старое значение секунд всегда сравнивается с новым. Наш светодиод L также мигает ежесекундно. Благодаря ему можно наблюдать за функцией программы, работает ли она без сбоев или имеется ошибка при программировании.

Но следует учитывать, что такие датчики времени не обладают высокой точностью (уход может достигать до 1 минуты за сутки), т. к. тактовая частота и погрешность кварцевого резонатора гораздо больше, чем в прецизионном часовом кварце. К тому же точность отсчета времени существенно зависит от колебаний температуры. Код программы приведен в листинге 10.16.

Листинг 10.16. RTC.pde

```
// Franzis Arduino  
// RTC  
  
int cnt, Sekunde, Minute, Stunde=0;  
int LED=13;  
  
void setup()  
{  
    Serial.begin(9600);  
    pinMode(LED, OUTPUT);  
  
    // Задание времени  
    Sekunde=13;  
    Minute=10;  
    Stunde=0;
```

```
}

void loop()
{
    cnt++;
    if(cnt==50) digitalWrite(LED, LOW);

    if(cnt==100)
    {
        digitalWrite(LED, HIGH);
        Serial.print(Stunde);
        Serial.print(":");
        Serial.print(Minute);
        Serial.print(":");
        Serial.println(Sekunde);
        Sekunde++;
        if(Sekunde==60)
        {
            Sekunde=0;
            Minute++;
            if(Minute==60)
            {
                Minute=0;
                Stunde++;
                if(Stunde==24)
                {
                    Stunde=0;
                }
            }
        }
        cnt=0;
    }
    delay(10);
}
```

10.13. Программа школьных часов

Хорошее применение часов — это, например, подача школьного звонка по расписанию. В определенное время (каждые 45 минут) звенит школьный звонок. Для этого нужно добавить в предыдущую программу часов несколько строк кода и запросить при этом контроль времени через условие *if*. Если условие выполняется, то звучит пьезоакустический преобразователь (листинг 10.17). Макет установки приведен на рис. 10.15.

Вот примерное расписание занятий в школе:

1 час — 7:00–7:45 ч.

2 час — 7:55–8:40 ч., затем 20 мин. перерыва на завтрак.

3 час — 9:00–9:45 ч.

4 час — 9:55–10:40 ч.

5 час — 10:50–11:35 ч., затем 30 мин. обеденного перерыва.

6 час — 12:05–12:50 ч.

7 час — 13:00–13:45 ч.

В начале и конце каждого школьного часа должен звучать звонок. Звонок у нас подает пьезоакустический преобразователь.

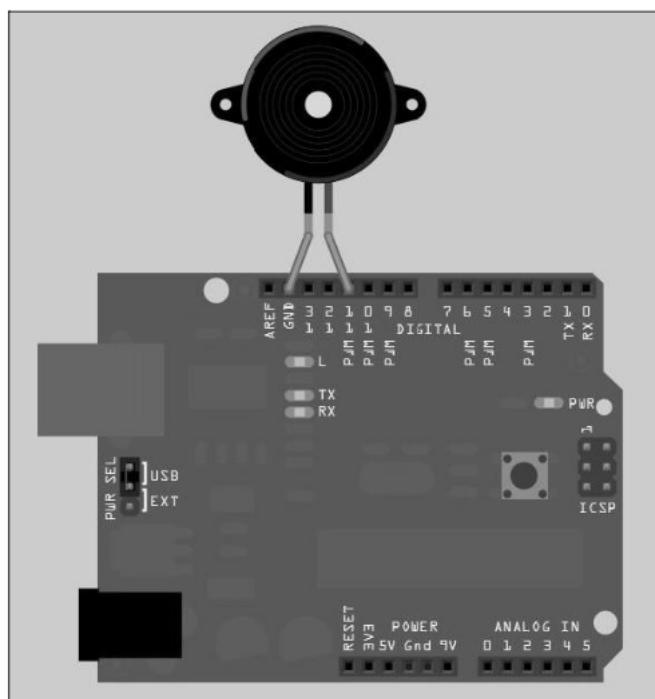


Рис. 10.15. Монтаж схемы для эксперимента со школьными часами

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- пьезопреобразователь.

Листинг 10.17. Schuluhr.pde

```
// Franzis Arduino
// Школьные часы

int cnt, Sekunde, Minute, Stunde=0;
```

```
int LED=13;
int Speaker=11;

void setup()
{
    Serial.begin(9600);
    pinMode(LED,OUTPUT);
    pinMode(Speaker,OUTPUT);

    // Задание времени
    Stunde=6;
    Minute=59;
    Sekunde=58;
}

void loop()
{
    cnt++;
    if(cnt==50)digitalWrite(LED,LOW);

    if(cnt==100)
    {
        digitalWrite(LED,HIGH);
        Serial.print(Stunde);
        Serial.print(":");
        Serial.print(Minute);
        Serial.print(":");
        Serial.println(Sekunde);
        Sekunde++;
        if(Sekunde==60)
        {
            Sekunde=0;
            Minute++;
            if(Minute==60)
            {
                Minute=0;
                Stunde++;
                if(Stunde==24)
                {
                    Stunde=0;
                }
            }
        }
    }
}
```

```
        }
    }
    cnt=0;
}
delay(10);

// Здесь время звонка

// 1. час
if(Stunde==7&&Minute==0)Bell();
if(Stunde==7&&Minute==45)Bell();

// 2. час
if(Stunde==7&&Minute==55)Bell();
if(Stunde==8&&Minute==40)Bell();

// Перерыв на завтрак

// 3. час
if(Stunde==9&&Minute==0)Bell();
if(Stunde==9&&Minute==45)Bell();

// 4. час
if(Stunde==9&&Minute==55)Bell();
if(Stunde==10&&Minute==40)Bell();

// 5. час
if(Stunde==10&&Minute==50)Bell();
if(Stunde==11&&Minute==35)Bell();
// Обеденный перерыв

// 6. час
if(Stunde==12&&Minute==05)Bell();
if(Stunde==12&&Minute==50)Bell();

// 7. час
if(Stunde==13&&Minute==0)Bell();
if(Stunde==13&&Minute==45)Bell();

}

void Bell(void)
```

```
{  
    if (Sekunde<5)  
    {  
        tone (Speaker,500) ;  
    }  
    else  
    {  
        noTone (Speaker) ;  
    }  
}
```

В начале программы сначала запрашивается время, которое мы задали. В этот момент наши часы запускаются. Затем программа функционирует независимо. Наш светодиод L мигает с тактовой частотой 1 Гц. Если достигнуто одно из заданных значений времени, в течение 5 секунд звучит школьный звонок. При слишком быстрой или слишком медленной работе часов, их можно корректировать, задавая величину `delay()`. Можно использовать для этого функцию `micros()`, что делает настройку еще точнее. В этом случае нужно подключить к выходу светодиода L осциллограф. Можно также сконфигурировать отдельный контакт платы как выход для измерения 10-миллисекундного сигнала при помощи осциллографа или частотомера. В таком случае можно еще точнее настроить тактирование.

10.14. Управление вентилятором

Во многих туалетах имеется система управления вентилятором. Ненадолго после включения света начинает работать вентилятор. Если снова отключить свет, вентилятор продолжает работать еще какое-то время. В большинстве случаев это управление уже встроено в вентилятор. При включении освещения вентилятор запускается не сразу (примерно через 30 секунд), если освещение было включено ошибочно и ненадолго. Как только освещение снова отключается, запускается таймер электронного управления вентилятором, что приводит к его работе в течение 1–5 минут.

ВНИМАНИЕ!

Лучше имитировать работу вентилятора только с экспериментальной платой, чтобы не иметь дела с электросетью 230 В. Тем более для этого потребовался бы специалист-электрик.

Наше устройство интеллектуальнее, чем большинство существующих. Оно управляет при помощи датчика света, который сигнализирует о включении или отключении света. Фоторезистор (LDR — Light Dependent Resistor) — это полупроводниковый прибор, сопротивление которого зависит от интенсивности света. Все полупроводниковые материалы чувствительны к свету и поэтому хорошо бы подошли для фоторезистора. Существуют специальные смеси полупроводников, в которых этот эффект проявляется особенно сильно. Фоторезистор состоит из двух

медных проводников, нанесенных на изолированную подложку (белого цвета). Между ними расположен полупроводниковый материал в форме извилистой ленты (красного цвета). Если свет падает на светочувствительный полупроводниковый материал, то возникают электронно-дырочные пары и сопротивление фоторезистора уменьшается. Чем больше интенсивность света, тем меньше сопротивление и тем больше будет электрический ток. Этот процесс очень инертен — время срабатывания достигает нескольких миллисекунд. Впрочем, этого вполне достаточно для распознавания наличия или отсутствия освещения. Схема макета приведена на рис. 10.16, а код программы — в листинге 10.18.

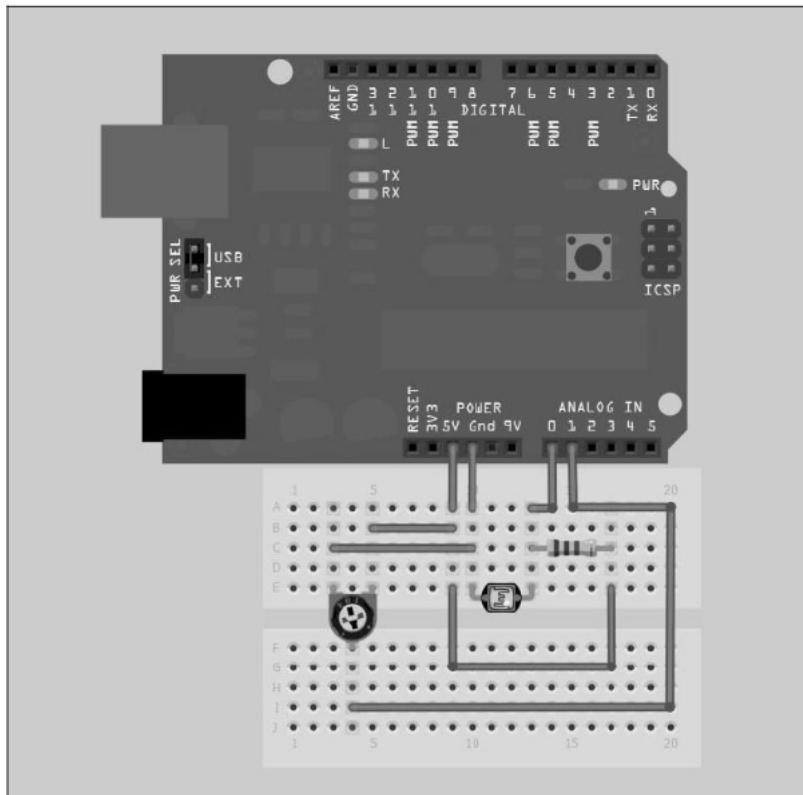


Рис. 10.16. Монтаж схемы управления вентилятором

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- фоторезистор (LDR);
- подстроечный резистор 10 кОм;
- резистор 68 кОм;
- шесть гибких монтажных провода длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

Листинг 10.18. Lüftersteuerung.pde

```
// Franzis Arduino
// Управление вентилятором

int LED=13;
int LDR=0;
int Poti=1;
int cnt=0;
int Flag=0;

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

void loop()
{

    if(analogRead(LDR)<analogRead(Poti))cnt++;
    if(analogRead(LDR)>analogRead(Poti))cnt=0;

    if(cnt>300)
    {
        digitalWrite(LED,HIGH);
        do
        {
            delay(100);
        }while(analogRead(LDR)<analogRead(Poti));
        cnt=0;
        delay(10000);
        digitalWrite(LED,LOW);
    }

    delay(10);

}
```

При помощи потенциометра устанавливается чувствительность устройства. Таким образом задается порог срабатывания схемы. При вращении движка потенциометра вправо, чувствительность уменьшается.

10.15. Автомат уличного освещения

В последнем примере мы познакомились с фоторезистором. Еще одно устройство, которое также часто требуется, — это автомат уличного освещения. Как только становится темно, свет должен автоматически включаться. Мы знаем это на примере уличных фонарей. Если светлеет утром, освещение автоматически отключается.

Программа сравнивает значение на аналоговом входе (контакт 0 платы), к которому присоединяется фоторезистор, с двумя пороговыми значениями: "светлому" и "темному", которые были заданы в программе. В результате получается гистерезис включения/выключения, чтобы небольшие световые колебания не нарушили работу программы. Благодаря гистерезису освещение не отключается (включается) мгновенно при кратковременных перепадах освещения. Дополнительная задержка в несколько секунд надежно подавляет помехи и делает программу устойчивой против внешних мешающих факторов (подъезжающие машины с включенными фарами или случайное затенение датчика).

Рассматриваемая схема (рис. 10.17) похожа на схему управления вентилятором, только что здесь постоянный резистор присоединен к общему проводу, а фоторезистор — к напряжению питания +5 В. Напряжение на аналоговом входе в светлое время суток увеличивается, а в темное — уменьшится. Код программы приведен в листинге 10.19.

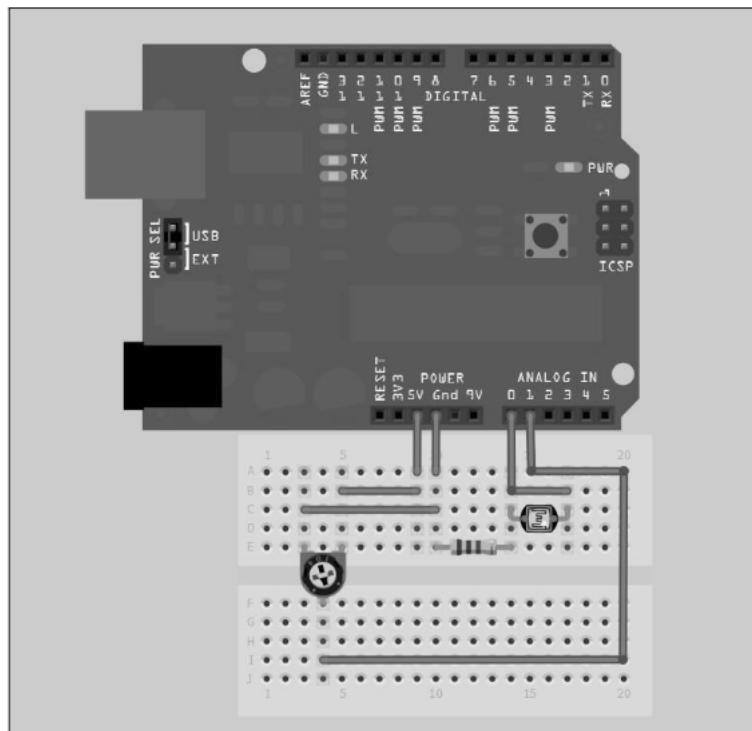


Рис. 10.17. Монтаж схемы выключателя уличного освещения

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- фоторезистор (LDR);
- подстроечный резистор 10 кОм;
- резистор 68 кОм;
- пять гибких монтажных проводов длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

Листинг 10.19. Daemmerungsschalter.pde

```
// Franzis Arduino
// Автомат уличного освещения

int LED=13;
int LDR=0;
int Poti=1;
int cnt=0;

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

void loop()
{

    if(analogRead(LDR)>analogRead(Poti))cnt=0;
    if(analogRead(LDR)<analogRead(Poti))cnt++;

    if(cnt>300)
    {
        digitalWrite(LED,HIGH);
        do
        {
            delay(5000);
        }while(analogRead(LDR)<analogRead(Poti));
        cnt=0;
        digitalWrite(LED,LOW);
    }

    delay(10);

}
```

10.16. Сигнализация

С помощью фотодиода, который мы уже успешно использовали в последних экспериментах, можно сконструировать сигнализацию, которая реагирует на самые небольшие изменения света. В начале программы (листинг 10.20) определяется текущее значение напряжения фотодиода, которое служит в качестве среднего значения. Если напряжение увеличится или уменьшится при изменении освещенности (например, прошел человек) и вследствие этого порог будет занижен или повышен, то сработает сигнализация. Так как освещенность изменяется в течение суток, каждую минуту автоматически определяется новое значение (текущее напряжение), которое служит очередной точкой привязки для измерения. Монтаж схемы показан на рис. 10.18.

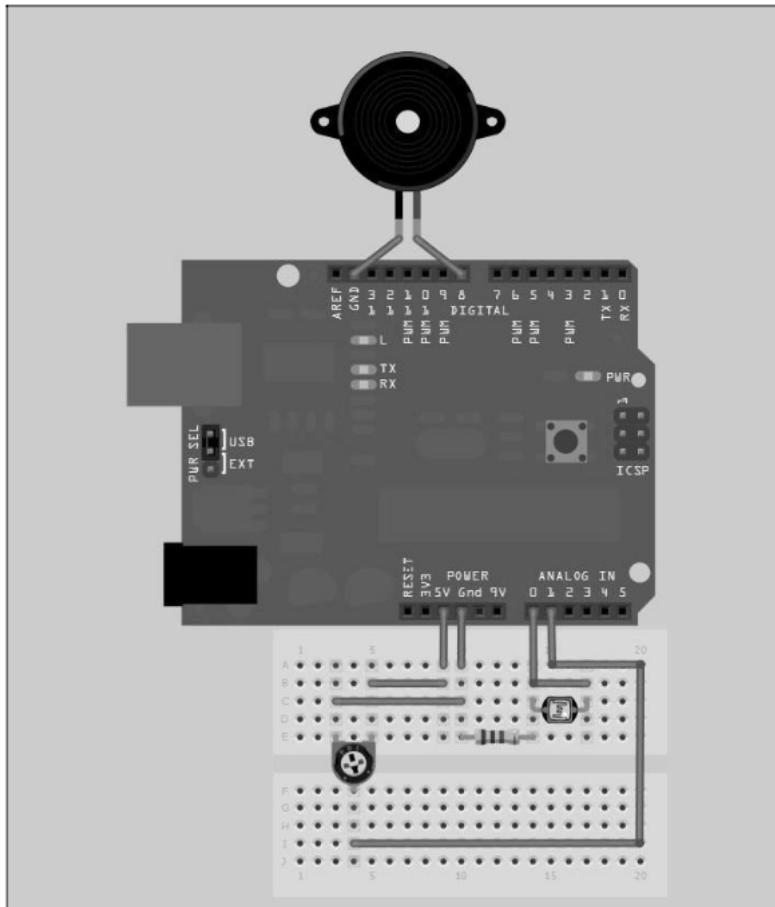


Рис. 10.18. Монтаж схемы сигнализации

Необходимые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- фоторезистор (LDR);
- подстроечный резистор 10 кОм;
- пьезопреобразователь;
- резистор 68 кОм;
- пять гибких монтажных проводов длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

Листинг 10.20. Alarmanlage.pde

```
// Franzis Arduino
// Сигнализация

int LED=13;
int LDR=0;
int Poti=1;
int Speaker=8;
int cnt=0;
int value,Schwelle=0;

void setup()
{
    pinMode(LED,OUTPUT);
    pinMode(Speaker,OUTPUT);
    value=analogRead(LDR);
}

void loop()
{

    cnt++;
    if(cnt>1000)
    {
        cnt=0;
        value=analogRead(LDR);
    }

    Schwelle=(analogRead(Poti)/10);
    if(value>(analogRead(LDR)+Schwelle) || value<(analogRead(LDR)-Schwelle))
```

```
{  
    digitalWrite(LED,HIGH);  
    tone(Speaker,500);  
    delay(2500);  
    noTone(Speaker);  
    digitalWrite(LED,LOW);  
    value=analogRead(LDR);  
}  
  
delay(10);  
  
}
```

10.17. Кодовый замок

Настоящий инженер-электронщик вместо обычного замка непременно установит у себя кодовый замок на основе микроконтроллера. Так как мы уже опытные программисты Arduino, то можем сами сконструировать замок с шифром. Для нашего замка потребуются только две кнопки: кнопка SW1 (вывод 2) и кнопка SW2 (вывод 3) на экспериментальной плате. Для ввода кода нужно нажимать, в зависимости от цифры кода, например, на кнопку SW1 дважды и кнопку SW2 три раза. Нажатие кнопок будет подтверждаться при помощи красного светодиода, подключенного к выводу 4, и звукового пьезопреобразователя. Если код введен правильно, то на 5 секунд включается светодиод красного цвета (вывод 5). Если код набран с ошибкой, то можно очистить ввод данных более длительным нажатием на кнопку SW2. Об удалении сигнализируют мигающий светодиод, подключенный к выводу 7, и "пищалка". Вместо светодиода через транзистор можно присоединить устройство для автоматического открывания двери, и тогда получится настоящий кодовый замок. Монтаж схемы кодового замка показан на рис. 10.19. Код программы приведен в листинге 10.21.

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- две кнопки;
- два светодиода красного и зеленого цвета;
- пьезопреобразователь звука;
- два резистора по 1,5 кОм;
- семь гибких монтажных проводов длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

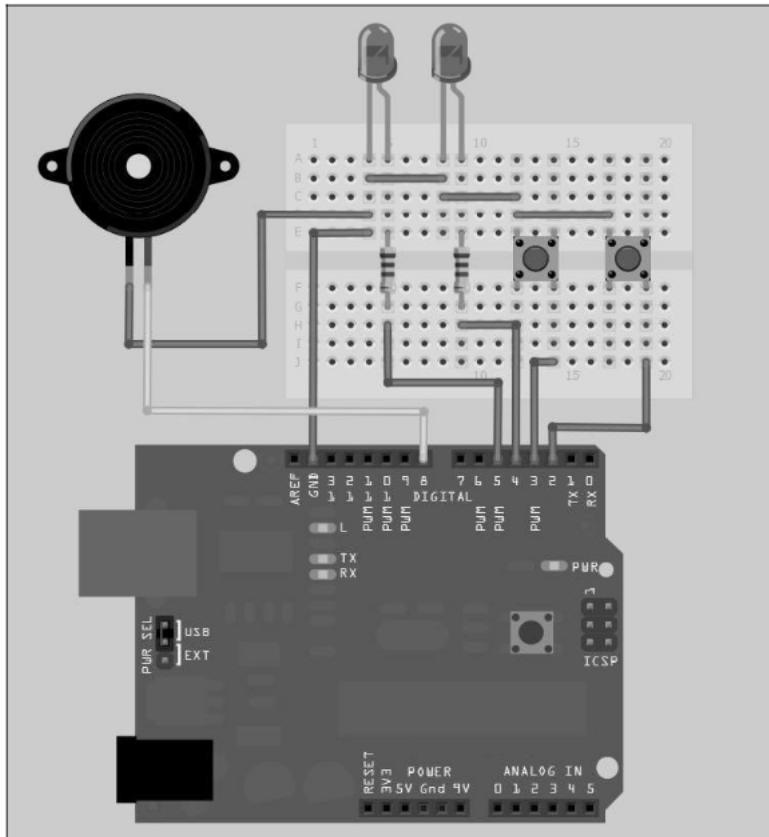


Рис. 10.19. Монтаж схемы кодового замка

Листинг 10.21. Codeschloss.pde

```
// Franzis Arduino
// Кодовый замок

int LED_rot=4;
int LED_gruen=5;
int SW1=2;
int SW2=3;
int Buzzer=8;
int x,y,code1,code2,resetTimer=0;

void setup()
{
    pinMode(LED_rot,OUTPUT);
```

```
pinMode(LED_gruen, OUTPUT);
pinMode(Buzzer, OUTPUT);

pinMode(SW1, INPUT);
digitalWrite(SW1, HIGH);

pinMode(SW2, INPUT);
digitalWrite(SW2, HIGH);
Clr_Code();

}

void loop()
{

// Code 1 = 5
if(!digitalRead(SW1))
{
    delay(50);
    if(!digitalRead(SW1))
    {
        Blink();
        x++;
        if(x==5)
        {
            code1=true;
        }else code1=false;

        do{
        }while(!digitalRead(SW1));
    }
}

// Code 2 = 3
if(!digitalRead(SW2))
{
    delay(50);
    if(!digitalRead(SW2))
    {
        Blink();
        y++;
        if(y==3)
```

```
{  
    code2=true;  
}else code2=false;  
  
do  
{  
    delay(50);  
    resetTimer++;  
  
    if(resetTimer>50)  
    {  
        Toggle_Flash();  
        Clr_Code();  
        break;  
    }  
}  
while(!digitalRead(SW2));  
resetTimer=0;  
}  
}  
  
if(code1==true&&code2==true)  
{  
    digitalWrite(LED_gruen,HIGH);  
    Clr_Code();  
    delay(5000);  
    digitalWrite(LED_gruen,LOW);  
}  
else  
{  
    digitalWrite(LED_gruen,LOW);  
}  
}  
  
void Blink(void)  
{  
    digitalWrite(LED_rot,HIGH);  
    tone(Buzzer,500,150);  
    delay(200);  
    digitalWrite(LED_rot,LOW);  
}  
  
void Toggle_Flash(void)
```

```
{  
    int tog=0;  
    for(x=0;x<6;x++)  
    {  
        if(tog==0) tog=1;else tog=0;  
        digitalWrite(LED_rot,tog);  
        tone(Buzzer,500,250);  
        delay(300);  
    }  
}  
  
void Clr_Code(void)  
{  
    x=0;  
    y=0;  
    code1=0;  
    code2=0;  
    resetTimer=0;  
    delay(1000);  
}
```

10.18. Измеритель емкости с автоматическим выбором диапазона

Снова и снова самостоятельно конструировать измерительные приборы интересно и увлекательно. С нашей экспериментальной платой и языком программирования Arduino C можно создать прибор для измерения емкости конденсаторов в диапазоне от 1 нФ и примерно до 100 мКФ. Потребуется совсем немного компонентов.

Наш измерительный прибор с автоматической установкой диапазона функционирует так. В начале измерения переменной `c_time` задается нулевое значение. Выход 12 конфигурируется как выход и сразу переключается на низкий уровень сигнала, чтобы разрядить присоединенный (тестируемый) конденсатор перед измерением. После разряда в течение 1 секунды вывод 12 конфигурируется как вход и подключается внутренний подтягивающий резистор. Через него тестируемый конденсатор заряжается до тех пор, пока на выводе 12 не будет достигнут высокий уровень. Интервал времени мы измеряем в цикле `do-while c c_time`. Значение переменной `c_time` будет пропорционально емкости конденсатора, т. е. если значение `c_time` большое, то и измеряемая емкость велика. Далее, чтобы получить правильный результат измерения, требуется еще пересчитать величину ($c_{time} \times \text{коэффициент}$). Значение коэффициента должно определяться экспериментально, т. к. распознавание высокого уровня (порог срабатывания) у различных микроконтроллеров оказывается несколько разным. В заключение, результат измерения переводится

в нанофарады (нФ) или микрофарады (мкФ) и выводится в терминал, а затем снова запускается процесс измерения.

Схема измерителя приведена на рис. 10.20, а код программы — в листинге 10.22.

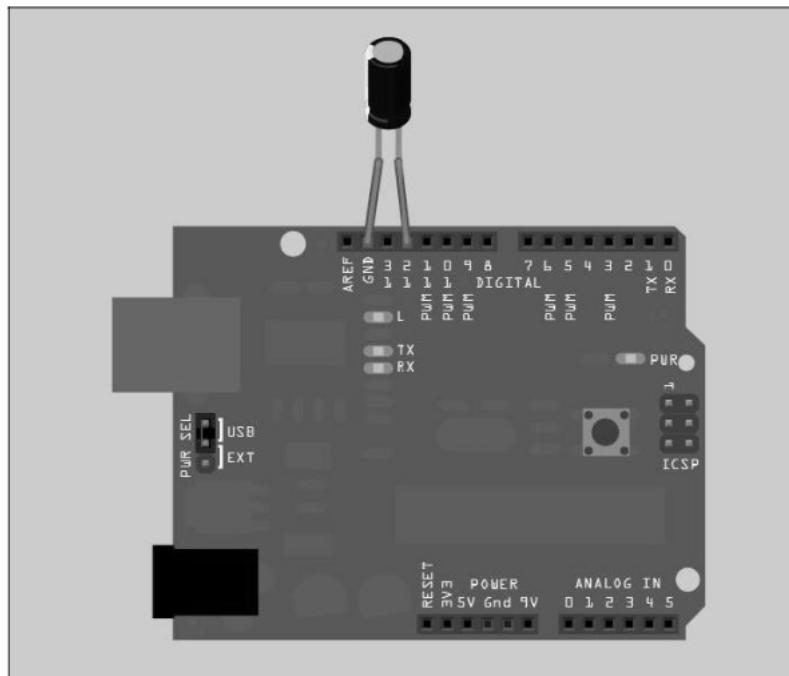


Рис. 10.20. Монтаж схемы измерителя емкости

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- тестируемый конденсатор емкостью от 1 нФ до 100 мкФ с рабочим напряжением не менее 5 В.

В вашем ящике с радиодеталями наверняка найдется несколько "жертв для испытаний"!

ВНИМАНИЕ!

Обращайте внимание на то, чтобы тестируемый конденсатор был разряжен перед измерением. Энергия заряженного конденсатора может повредить микроконтроллер!

Листинг 10.22. Kapazitaetsmessgeraet.pde

```
// Franzis Arduino
// Измеритель емкости 1 нФ...100 мкФ

int messPort=12;
```

```
float c_time=0.0;
float kapazitaet=0.0;

void setup()
{
    Serial.begin(9600);
    Serial.println("Autorange Kapazitaetsmessgeraet 1 nF ... 100uF");
    Serial.println();
}

void loop()
{
    // Разряд
    pinMode(messPort,OUTPUT);
    digitalWrite(messPort,LOW);
    c_time=0.0;
    delay(1000);

    // Заряд
    pinMode(messPort,INPUT);
    digitalWrite(messPort,HIGH);

    // Измерение
    do
    {
        c_time++;
    }while(!digitalRead(messPort));

    // Пересчет
    kapazitaet=(c_time*0.042)*10.0;

    // Выбор диапазона
    if(kapazitaet<999)
    {
        Serial.print(kapazitaet);
        Serial.println("nF");
    }
    else
    {
        kapazitaet=kapazitaet/1000;
        Serial.print(kapazitaet);
    }
}
```

```
    Serial.println("uF");
}

delay(1000);

}
```

10.19. Профессиональное считывание сопротивления потенциометра

Вы уже знаете, как считывать сопротивление потенциометра при помощи команды `analogRead()`. Тем не менее, для некоторых задач этот метод не подходит, т. к. последняя измеряемая позиция (разрядная цифра) постоянно "прыгает" вверх и вниз. Это происходит, с одной стороны, из-за того, что АЦП дает ошибку разряда. С другой стороны, это связано с тем, что потенциометр относительно неточен, и имеет, кроме того, еще так называемый дрейф. Чтобы устранить эти нежелательные явления, можно использовать функцию гистерезиса. Значение будет обновляться только тогда, когда последний измеряемый разряд оказывается больше (или меньше) значения требуемой величины. Кроме того, в нашем примере измеренное значение выводится на терминал только в том случае, если оно отличается от предшествующего результата. Монтаж схемы показан на рис. 10.21, а код программы приведен в листинге 10.23.

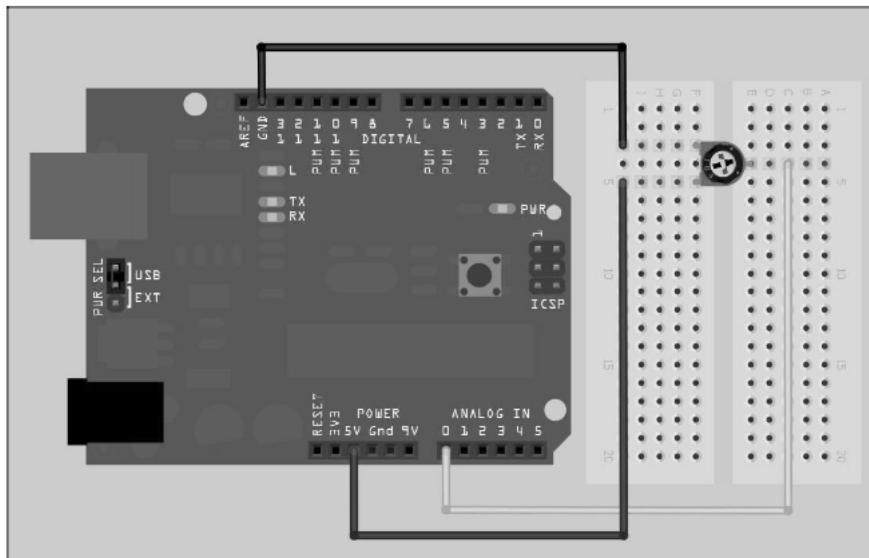


Рис. 10.21. Монтаж схемы

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- подстроечный резистор 10 кОм;
- три гибких монтажных провода длиной примерно 10 см.

Листинг 10.23. Poti.pde

```
// Franzis Arduino
// Считывание потенциометра

int Poti=0;
int raw,raw_last,raw_min,raw_max=0;
int hysterese=10;

void setup()
{
    Serial.begin(9600);
    Serial.println("Potenziometer professionell auslesen");
    Serial.println();
}

void loop()
{

    raw=analogRead(Poti);
    raw_min=raw_last-hysterese;
    raw_max=raw_last+hysterese;

    if((raw!=raw_last))
    {
        if((raw>raw_max) || (raw<raw_min))
        {
            Serial.println(raw);
            raw_last=raw;
        }
    }
}
```

10.20. Сенсорный датчик

Существуют устройства, реагирующие на прикосновение пальца. При касании поверхности без нажатия при этом на механический переключатель или кнопку, устройство включается или выключается, становится громче или тише — в зависимости от функции. Сенсорные датчики широко применяются для включения, переключения или регулирования. Такие сенсорные датчики можно запрограммировать с помощью аналоговых входов. Аналоговые входы являются высокомомными, что и позволяет измерять значение (напряжения) при простом контакте пальцами.

Если в нашем эксперименте слегка коснуться пальцами аналогового входа 0 во время исполнения программы, то загорится светодиод L, а при следующем прикосновении — погаснет. Здесь значение порога включения составляет 50, а отключения — 5. Схема монтажа приведена на рис. 10.22, а код программы — в листинге 10.24.

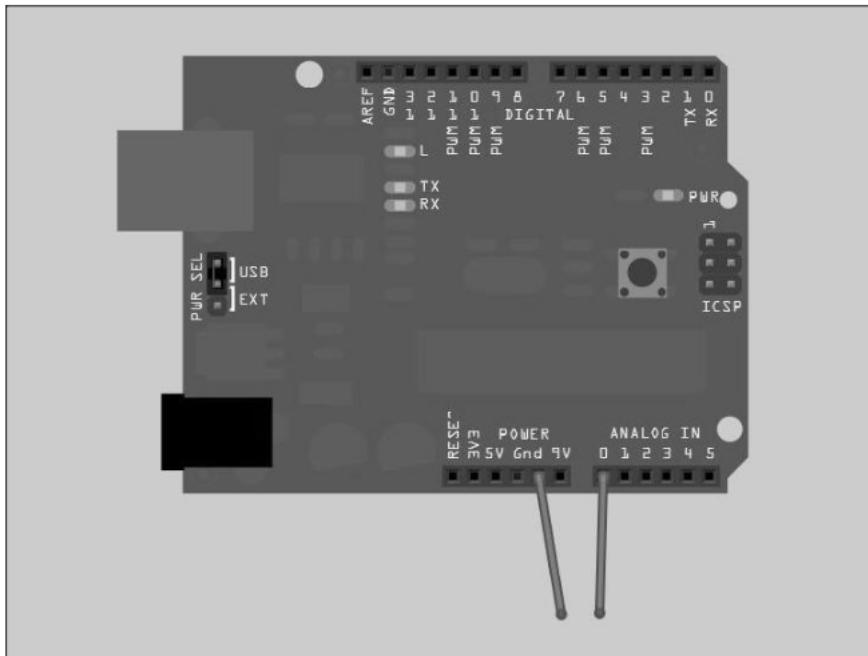


Рис. 10.22. Схема монтажа сенсорного датчика

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- два гибких монтажных провода длиной примерно 10 см.

Листинг 10.24. Sensor_Taster.pde

```
// Franzis Arduino
// Сенсорная кнопка

int LED=13;
int Sensor=0;
int Flag1,Flag2,tog=0;

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

void loop()
{

if((analogRead(Sensor)>50)&&(!Flag2))
{
    delay(50);
    if((analogRead(Sensor)>50)&&(!Flag2))
    {
        if(!tog)tog=1;else tog=0;
        Flag1=tog;
        Flag2=1;
    }
}
else
{
    if((analogRead(Sensor)<5)&&(Flag2))
    {
        delay(50);
        if((analogRead(Sensor)<5)&&(Flag2))
        {
            Flag2=0;
        }
    }
}

if(!Flag1)digitalWrite(LED,LOW);
if(Flag1)digitalWrite(LED,HIGH);

}
```

10.21. Конечный автомат

Конечный автомат (State Machine) — это абстрактный автомат, число возможных состояний которого конечно. Результат работы автомата определяется по его конечному состоянию. Автомат работает во время перехода из одного состояния в другое и запускает при этом действие. Таким образом, последующее состояние получается из предыдущего. Сам автомат приводится в действие через тактирование и не может реагировать на события в произвольный момент времени. На каждом такте в результате анализа текущего состояния и состояния входных каналов решается, в какое состояние нужно перейти и какие действия выполнить.

Абстрактное описание конечного автомата возможно несколькими способами: в форме таблицы, в виде графа или диаграммы состояний.

Проще всего представить себе конечный автомат с помощью светофора.

Вот последовательность световых сигналов светофора:

1. Красный;
2. Красный/желтый;
3. Зеленый;
4. Желтый;
5. Красный.

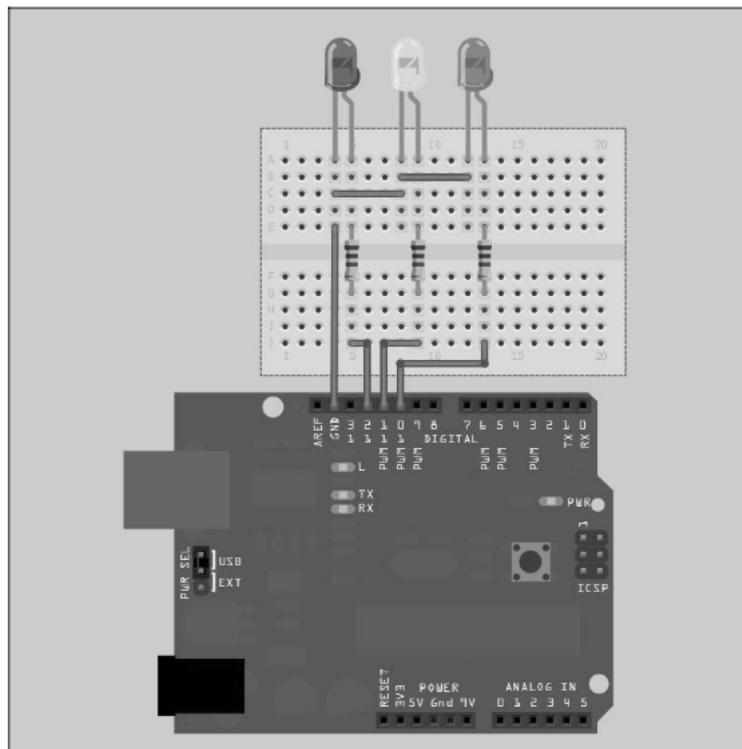


Рис. 10.23. Монтаж схемы светофора

В первом состоянии светофор имеет красный цвет, после заданного времени ожидания светофор переходит на желтый цвет, но красный тоже присутствует. Переход между красным и желтым цветами происходит быстро, и далее следует переключение на зеленый цвет. После разрешающего сигнала светофор переходит снова с зеленого на желтый цвет и после этого снова на красный. В нашем примере (листинг 10.25) предыдущее состояние уступает следующему (изменение состояния). Временная последовательность определяется через `delay()` в `main loop()`. Монтаж схемы светофора показан на рис. 10.23.

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- светодиод красного цвета;
- светодиод желтого цвета;
- светодиод зеленого цвета;
- три резистора 1,5 кОм;
- пять монтажных проводов длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 10 см.

Листинг 10.25. Statemaschine.pde

```
/ Franzis Arduino
// Конечный автомат (светофор)

int LEDrot=12;
int LEDgelb=11;
int LEDgruen=10;
int cnt=0;
int state=1;

void setup()
{
    pinMode(LEDrot,OUTPUT);
    pinMode(LEDgelb,OUTPUT);
    pinMode(LEDgruen,OUTPUT);
}

void loop()
{
    cnt++;
    if(cnt==100)
    {
        cnt=0;
        Statemaschine();
    }
}
```

```
}

delay(10);
}

void Statemaschine(void)
{

switch(state)
{
    case 1:
        digitalWrite(LEDrot,HIGH);
        digitalWrite(LEDgelb,LOW);
        digitalWrite(LEDgruen,LOW);
        state++;
        break;

    case 2:
        digitalWrite(LEDrot,HIGH);
        digitalWrite(LEDgelb,HIGH);
        digitalWrite(LEDgruen,LOW);
        state++;
        break;

    case 3:
        digitalWrite(LEDrot,LOW);
        digitalWrite(LEDgelb,LOW);
        digitalWrite(LEDgruen,HIGH);
        state++;
        break;

    case 4:
        digitalWrite(LEDrot,LOW);
        digitalWrite(LEDgelb,HIGH);
        digitalWrite(LEDgruen,LOW);
        state=1;
        break;

}
}
```

10.22. 6-канальный вольтметр на основе Arduino

Для тех, кто хотел бы увидеть результаты измерения напряжения на персональном компьютере, это можно сделать с помощью описанного здесь макета. Для самостоятельного редактирования исходного кода потребуется Visual Basic Express Version 2008, который можно бесплатно загрузить с сайта компании Microsoft. Программа Arduino считывает аналоговые входы 0–5 и посыпает данные на персональный компьютер. Программа VB.NET на персональном компьютере принимает данные через USB (USB/UART-адаптер), обрабатывает их и отображает на дисплее. Кроме того, результаты измерения АЦП пересчитываются в вольты. В программе (листинг 10.26) показано, как можно оформить передачу данных. Монтажная схема приведена на рис. 10.24.



Рис. 10.24. 6-канальный вольтметр с VB.NET

Программа считывает сначала все каналы АЦП микроконтроллера и сохраняет результаты измерения как переменную величину `Integer` в массиве `Adc_raw[]`. На втором шаге переменные `Word` из массива разбиваются на старший и младший байты. Затем на персональный компьютер сначала посыпается старший байт и потом младший. Выполняется проверка ошибок. Контрольная сумма (CRC) вычисляется посредством XOR-преобразования из фиксированных переменных, старшего и младшего байтов. Процесс передачи данных повторяется до тех пор, пока все шесть каналов не будут переданы в персональный компьютер. Программа VB.NET считывает отдельные байты и записывает их также в массив данных. Старший и младший байты образуют первоначальное значение АЦП. Программа VB.NET рассчитывает таким же способом, как программа Arduino контрольную сумму, только в этот раз с данными, которые принял персональный компьютер. Если принятая контрольная сумма совпадает с рассчитанной, то результат измерения отображается в текстовом окне. В случае ошибки результат измерения не обновляется.

Листинг 10.26. Voltmeter.pde

```
// Franzis Arduino
// 6-канальный вольтметр

int LED=13;
char startbyte=0;
int highbyte=0;
int lowbyte=0;
int adc_raw[6];
int adc_cnt=0;
int cnt=0;
int crc=0;

void setup()
{
    Serial.begin(9600);
    pinMode(LED,OUTPUT);
}

void loop()
{

    startbyte=Serial.read();
    if(startbyte==42)
    {
        digitalWrite(LED,HIGH);
        delay(50);
        digitalWrite(LED,LOW);
        delay(50);
        Serial.flush();

        for(cnt=0;cnt<6;cnt++)
        {
            adc_raw[cnt]=analogRead(adc_cnt);
            adc_cnt++;
        }
        adc_cnt=0;

        for(cnt=0;cnt<6;cnt++)
        {
            highbyte=adc_raw[cnt]/256;
```

```
lowbyte=adc_raw[cnt] %256;
Serial.write(highbyte);
Serial.write(lowbyte);
}

crc=170^highbyte^lowbyte;
Serial.write(crc);
}

}
```

10.23. Программирование самописца напряжения

В этом примере мы используем аналоговый вход как самопищий прибор для записи измеряемого значения. Передача данных формируется так же, как и в предыдущем примере (6-канальный вольтметр). Результат измерения АЦП, который мы записываем в нашей программе как переменную `Integer`, разбивается на старший и младший байты и посыпается через интерфейс UART на персональный компьютер.

Принцип формирования старшего (`Highbyte`) и младшего (`Lowbyte`) байта:

`Highbyte = RAW/256`
`Lowbyte = RAW%256`

Чтобы получить старший байт, результат измерения АЦП делится на 256, что соответствует одному байту. Чтобы получить остаток, применяется операция деления по модулю `%` и в итоге остаются младшие 8 битов (младший байт).

Пример:

Пусть число 766 нужно разложить на старший и младший байты. По упомянутому правилу получаем следующие значения:

`Highbyte = 2`
`Lowbyte = 254`

Теперь чтобы снова получить число 766 из старшего и младшего байтов, нужно умножить старший байт на 256, затем прибавить 254 и результат будет снова 766!

Программа самописца напряжения (листинг 10.27) записывает результаты измерения через интервалы 100, 500 и 1 000 мс. Программа VB.NET посыпает для этого в определенное время стартовый байт (55) в плату микроконтроллера Arduino. После принятия стартового байта начинается измерение и контроллер Arduino посыпает назад результат измерения и контрольную сумму в программу VB.NET. Далее снова происходит пересчет обоих байтов в результат измерения и отображение значения в графическом окне программы (рис. 10.25). Такая регистрация данных подойдет для самых различных приложений. Хорошие примеры — запись температуры, слежение за зарядом и разрядом аккумулятора, контроль напряжения в сети и т. д.

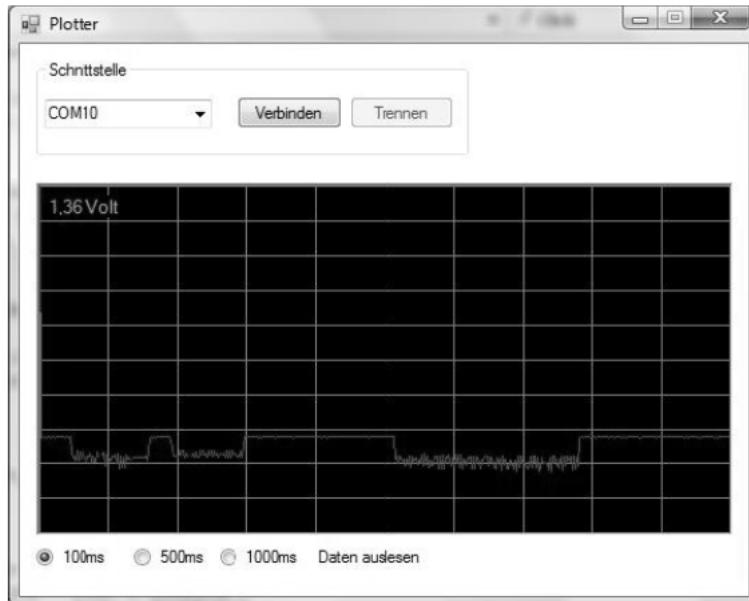


Рис. 10.25. Программа VB.NET для самописца напряжения

СОВЕТ

Можно изготовить простой самописец освещенности с применением светодиодов. Поместите анод в гнездо аналогового входа 0, а катод подключите к GND. Теперь при освещении или затемнении светодиода, напряжение будет меняться. Далее можно попытаться настроить масштабирование амплитуды (уровня напряжения) в коде программы.

Листинг 10.27. Plotter.pde

```
// Franzis Arduino
// Самописец напряжения

char startbyte=0;
int highbyte=0;
int lowbyte=0;
int adc=0;
int crc=0;

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

```
void loop()
{
    startbyte=Serial.read();
    if(startbyte==55)
    {
        Serial.flush();
        adc=analogRead(0);
        highbyte=adc/256;
        lowbyte=adc%256;
        Serial.write(highbyte);
        Serial.write(lowbyte);
        crc=170^highbyte^lowbyte;
        Serial.write(crc);
    }
}
```

10.24. Осциллограф с памятью на основе Arduino

Если требуется исследовать быстропротекающие процессы (примерно до 5 кГц), фиксировать их нужно иначе (листинг 10.28). Для этого сначала следует создать массив данных, в котором можно оставлять 255 результатов измерения. В цикле, который считает от 1 до 256, измеряется при каждом прохождении аналоговый вход и текущее значение пишется в массив. Измерение всегда начинается со стартового байта (55), который должен посыпаться из персонального компьютера. Если измерение завершено, содержание массива передается в программу VB.NET. Разделение старшего и младшего байтов происходит уже известным способом. Программа VB.NET обрабатывает данные и отображает диаграмму напряжения в графическом окне. Таким образом, получается несложный осциллограф с памятью, который позволяет анализировать низкочастотный сигнал. Для измерения высоких напряжений (более 5 В) необходимо перед аналоговым входом дополнительно подключать делитель напряжения. При отсутствии подходящего источника сигналов можно посмотреть наводку сигнала сети 50 Гц. Для этого присоедините кусок провода к аналоговому входу 0 и просто возьмитесь за него пальцами (рис. 10.26).

Можно взять более длинный провод и зафиксировать даже приближение руки. Аналоговый вход высокоомный, поэтому даже малые изменения напряжения становятся заметными.

Поищите в ящике с радиодеталями старый динамический микрофон. Подключив его к нашему осциллографу, можно отчетливо увидеть колебания катушки микрофона.

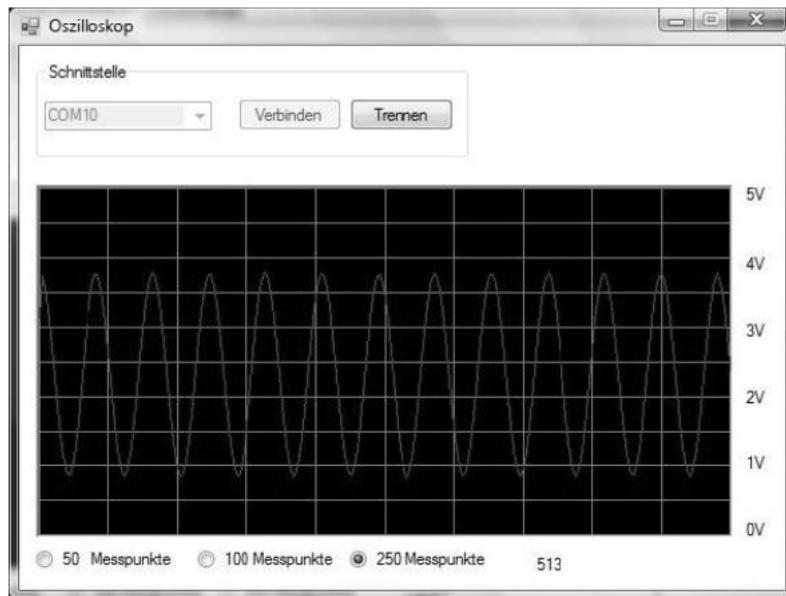


Рис. 10.26. VB.NET-осциллограф с памятью

Листинг 10.28. Oszi.pde

```
// Franzis Arduino
// Осциллограф

char startbyte=0;
int highbyte=0;
int lowbyte=0;
int adc[256];
int cnt=0;
int crc=0;

void setup()
{
    Serial.begin(115200);
}

void loop()

{
    startbyte=Serial.read();
    if(startbyte==55)
```

```
{  
    Serial.flush();  
  
    for(cnt=0;cnt<256;cnt++)  
    {  
        adc[cnt]=analogRead(0);  
    }  
  
    for(cnt=0;cnt<256;cnt++)  
    {  
        highbyte=adc[cnt]/256;  
        lowbyte=adc[cnt] % 256;  
        Serial.write(highbyte);  
        Serial.write(lowbyte);  
    }  
  
    crc=170 ^ highbyte ^ lowbyte;  
    Serial.write(crc);  
}  
}
```

10.25. Программа StampPlot — бесплатный профессиональный регистратор данных

StampPlot — это программа для построения графиков, индикации и контроля последовательных данных от микроконтроллера. В предыдущих разделах мы рассмотрели, как можно реализовать собственные программы для записи данных (осциллограф, самописец напряжения и т. д.) с применением программы VB.NET. Программа StampPlot гораздо профессиональнее и предлагает многочисленные функции для записи данных и интерпретации результата измерения.

В нашем примере (листинг 10.29) мы записываем сначала эпюру напряжений аналогового входа 0, к которому присоединяется, например, потенциометр. Но, конечно же, перед тем как начать эксперименты, нужно установить программу StampPlot. Дистрибутив программы находится на прилагаемом к книге компакт-диске в папке Software (программное обеспечение). После установки запустите программу и выберите один из представленных вариантов отображения (рис. 10.27).

После выбора вида графика нужно сконфигурировать последовательный интерфейс, к которому присоединяется экспериментальная плата. На плате устанавливаем адаптер USB/UART, который предоставляет нам в распоряжение виртуальный COM-порт.

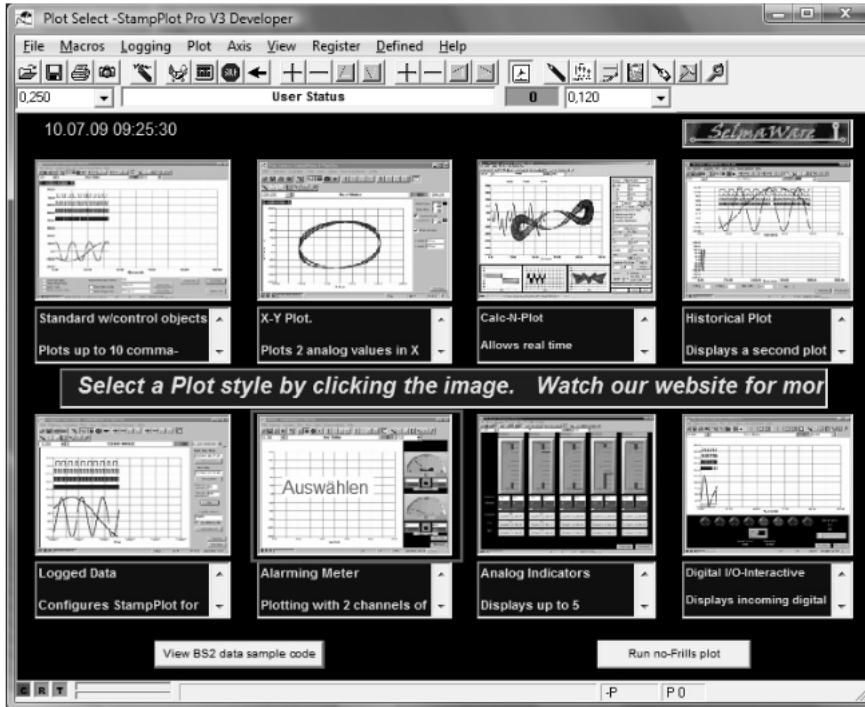


Рис. 10.27. Окно выбора "Plot style" (Вид графика) программы StampPlot

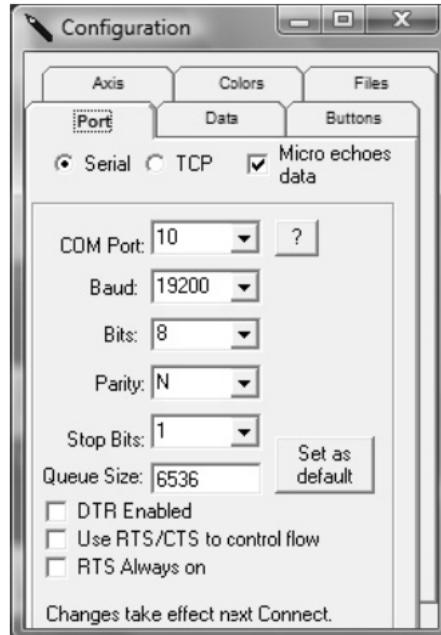


Рис. 10.28. Окно конфигурации программы StampPlot

Нужно задать тот же самый COM-порт, который был назначен в Arduino для программирования и вывода информации на терминал (рис. 10.28).

После загрузки кода программы Arduino в микроконтроллер, остается 10 секунд для подключения программы StampPlot с последовательным интерфейсом. Можно нажать также на кнопку Reset.

После подключения программы StampPlot к микроконтроллеру и по истечении 10 секунд, начинается запись. Сначала микроконтроллер посыпает информацию программе StampPlot, которая указывает, как должны отображаться результаты. Здесь сообщаются различные установки, например, сброс графика, заголовок, точки на графике, разрешение и т. д. Затем начинается измерение и передача данных в программу StampPlot. Пример регистрации результатов измерения приведен на рис. 10.29.

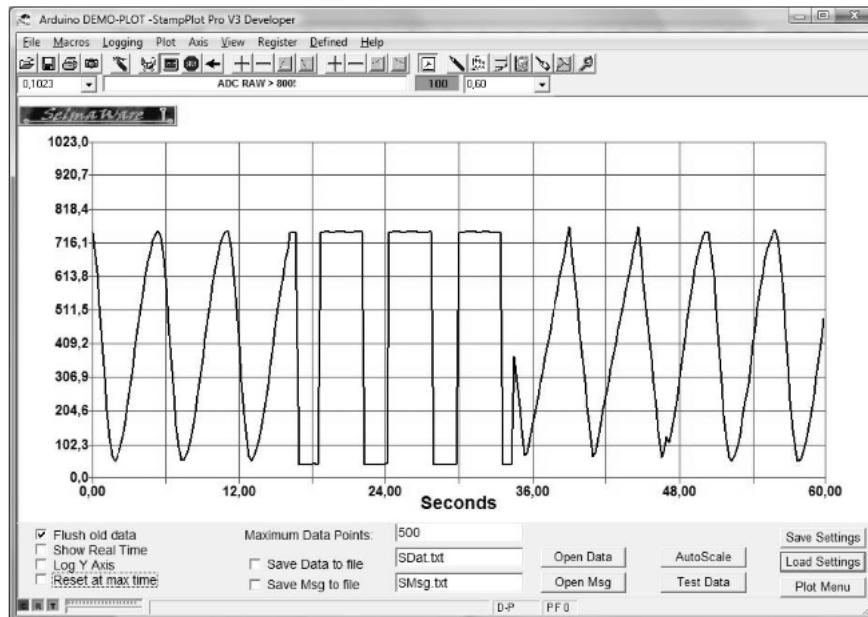


Рис. 10.29. Регистрация результатов измерения в программе StampPlot

Чтобы разобраться подробнее с работой программы StampPlot, нужно посетить сайт разработчиков и изучить руководство. Программа StampPlot очень обширна, и о ней можно написать отдельную книгу. Здесь мы лишь указали дорогу в правильном направлении и немного облегчили первые шаги при работе с программой.

Листинг 10.29. Stampplot.pde

```
// Franzis Arduino
// Устройство регистрации данных Stampplot

int LED=13;
```

```
int adc0=0;

void setup()
{
    Serial.begin(19200);
    pinMode(LED,OUTPUT);

    delay(10000);

    // Передача установок Stampplot
    Serial.println("!RSET"); // Сброс графика для очистки данных
    Serial.println("!TITL Arduino DEMO-PLOT"); // Создание заголовка
    Serial.println("!PNTS 300"); // 1000 образцов базовых
    //точек
    Serial.println("!TMAX 60"); // Макс. 60 секунд
    Serial.println("!SPAN 0,1023"); // Диапазон 0-1023
    Serial.println("!AMUL 1"); // Увеличение данных на 1
    Serial.println("!DELD"); // Удаление файла данных
    Serial.println("!SAVD ON"); // Сохранение данных
    Serial.println("!TSMP ON"); // Включение отметки времени
    Serial.println("!CLMM"); // Очистка Min/Max
    Serial.println("!CLRM"); // Очистка сообщений
    Serial.println("!PLOT ON"); // Запуск построения графика
    Serial.println("!RSET"); // Сброс графика для времени 0
}

void loop()
{
    adc0=analogRead(0);
    Serial.print(adc0);
    Serial.write(13);

    if(adc0>700)
    {
        Serial.println("!USRS ADC RAW > 800!");
        Serial.println("ADC RAW ist größer 800");
        digitalWrite(LED,HIGH);
    }

    if(adc0<150)
    {
```

```
Serial.println("!USRS ADC RAW < 250!");
Serial.println("ADC RAW ist kleiner 150");
digitalWrite(LED, LOW);
}

delay(200);

}
```

10.26. Управление через VB.NET

Теперь рассмотрим, как реализовать передачу данных от персонального компьютера в микроконтроллер и обработку их там. В предлагаемом примере программа VB.NET (рис. 10.30) посыпает единственный байт микроконтроллеру. Но даже при помощи этого единственного байта можно управлять 255 выходами или, например, изменять сигнал на аналоговом выходе. Пример очень прост и не использует контрольную сумму при передаче. Но вы можете предложить свой вариант. Попытайтесь модифицировать программу так, чтобы можно было изменять яркость светодиода через аналоговый выход, и тогда появится возможность управлять остальными портами ввода/вывода.



Рис. 10.30. Программа VB.NET для управления портами ввода/вывода

Естественно, можно добавить контрольную сумму, как мы делали уже в программе для плottера данных или осциллографа. Монтаж схемы приведен на рис. 10.31, а код программы — в листинге 10.30.

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- светодиод красного цвета;
- резистор 1,5 кОм;
- два гибких монтажных провода длиной примерно 5 см.

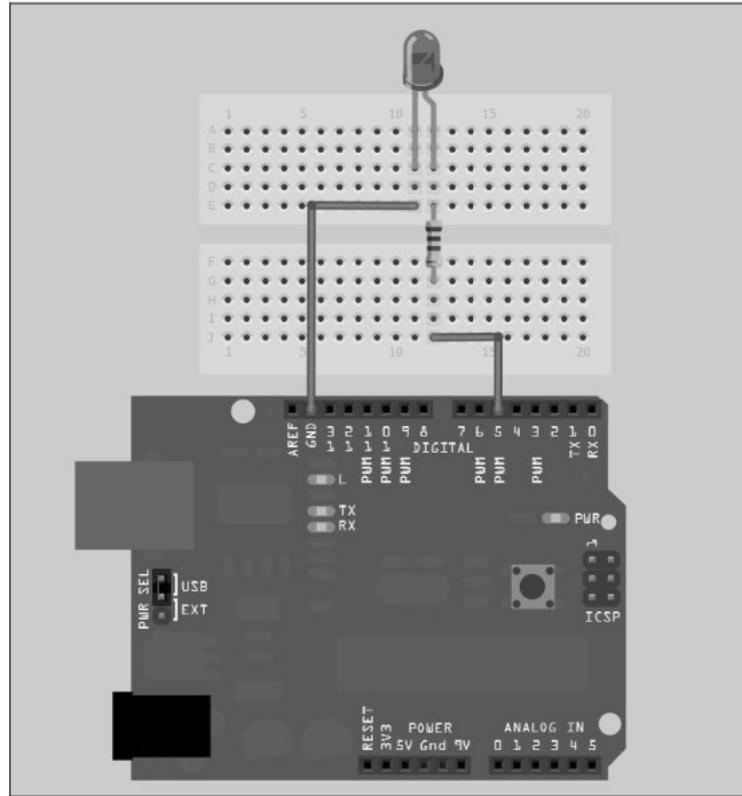


Рис. 10.31. Монтаж схемы

Листинг 10.30. Ports.pde

```
// Franzis Arduino
// Управление портами через DOT.NET

int LED=13;
int IO_5=5;
int input=0;

void setup()
{
    Serial.begin(9600);
    pinMode(LED,OUTPUT);
    pinMode(IO_5,OUTPUT);
}

void loop()
```

```
{  
  
    input=Serial.read();  
  
    switch(input)  
    {  
        case 10:  
            digitalWrite(LED,HIGH);  
            break;  
  
        case 20:  
            digitalWrite(LED,LOW);  
            break;  
  
        case 30:  
            digitalWrite(IO_5,HIGH);  
            break;  
  
        case 40:  
            digitalWrite(IO_5,LOW);  
            break;  
  
        case 100:  
            digitalWrite(LED,LOW);  
            digitalWrite(IO_5,LOW);  
            break;  
  
    }  
  
}
```

10.27. Реле температуры

Этот эксперимент показывает, как при помощи обычного кремниевого диода можно создать реле температуры. Прямое напряжение на диоде (V_f) зависит от температуры окружающей среды (рис. 10.32). Если температура растет, то напряжение V_f уменьшается. Если температура падает, то возрастает напряжение V_f . Можно измерять это напряжение при помощи входа АЦП и реализовать на этом принципе реле температуры.

Изменение напряжения относительно невелико, так что реле температуры можно реализовать только с 10-разрядным АЦП Arduino-микроконтроллера. С помощью измерительного усилителя (операционного усилителя) можно расширить диапазон и сделать термометр.

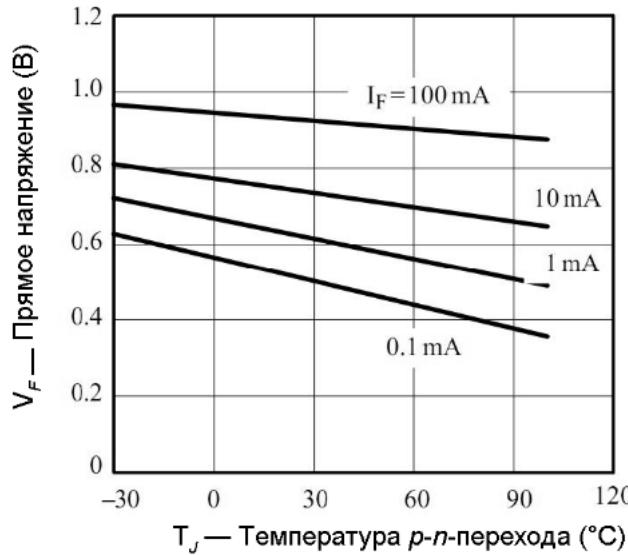


Рис. 10.32. Зависимость между температурой окружающей среды и прямым падением напряжения на диоде V_f при заданном токе (источник: технический паспорт компании Visay)

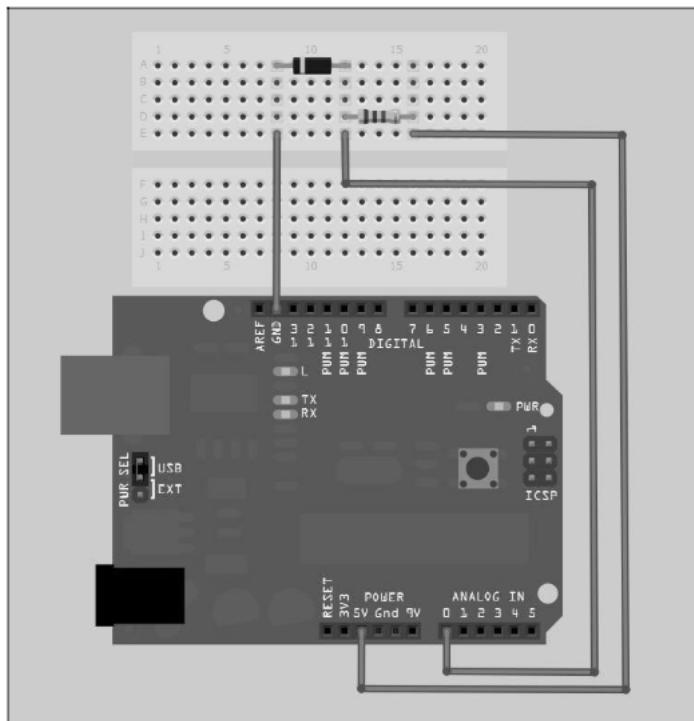


Рис. 10.33. Монтажная схема реле температуры с диодом 1N4148

В нашем примере светодиод L включается, как только температура падает ниже порогового значения. Попробуйте осторожно нагреть диод зажигалкой. Светодиод погаснет. Монтажная схема приведена на рис. 10.33, а код программы — в листинге 10.31.

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- панель с контактными гнездами;
- диод 1N4148;
- резистор 47 кОм;
- гибкий монтажный провод длиной примерно 5 см;
- два гибких монтажных провода длиной примерно 10 см.

Листинг 10.31. Temperaturschalter.pde

```
// Franzis Arduino
// Реле температуры

int LED=13;
int Uf=0;

void setup()
{
    Serial.begin(9600);
    pinMode(LED,OUTPUT);
}

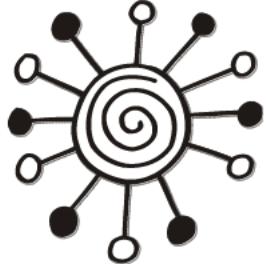
void loop()
{
    Uf=analogRead(0);
    Serial.print("Uf = ");
    Serial.println(Uf);

    if(Uf>40) digitalWrite(LED,HIGH);
    if(Uf<20) digitalWrite(LED,LOW);

    delay(250);
}
```

Значение прямого напряжения на диоде выводится на терминал. Теперь вы узнаете напряжение на диоде при комнатной температуре. Пороговое значение можно настраивать или инвертировать. Это позволяет реализовать, например, управление вентилятором в зависимости от температуры.

Глава 11



Шина I²C

Шина I²C (Inter-Integrated Circuit) — это последовательная синхронная двухпроводная шина, которая была разработана в 1980-е годы компанией Philips для внутрисхемного соединения узлов и микросхем (рис. 11.1). Вопреки ее "преклонному возрасту" она не потеряла актуальность до сегодняшнего дня. Эта шина применяется всюду, где требуется сэкономить на межсоединениях и разводке проводников (из-за недостатка места или для уменьшения затрат). Другое большое преимущество I²C-шины — простота управления. Здесь нет никаких фиксированных тактовых интервалов, соединяемые устройства могут быть как медленными, так и очень скоростными, возможно использование различных микросхем и языков программирования.



Рис. 11.1. Шина I²C (или IIC) — зарегистрированный товарный знак фирмы Philips

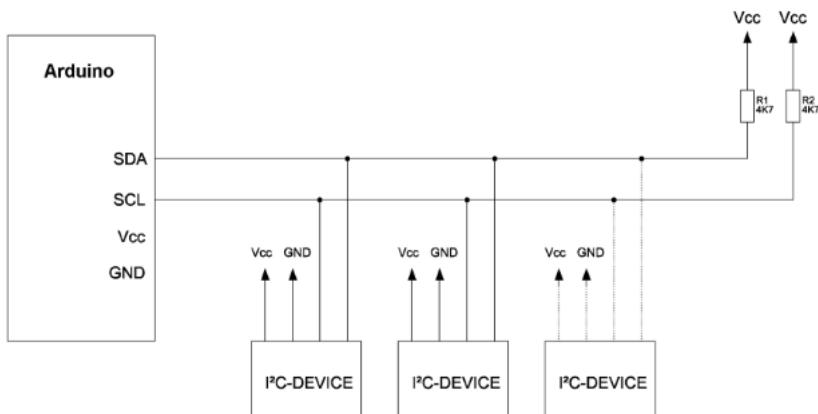


Рис. 11.2. Структура I²C-шины

Шина состоит из двух линий SDA и SCL, на которых изначально присутствует высокий уровень (рис. 11.2). Это обеспечивается резисторами R1 и R2, сопротивлением 1–10 кОм. Каждое из подключаемых устройств идентифицируется собственным адресом. Первые три бита адреса заданы производителем. Последний бит адреса указывает, происходит запись или чтение устройства.

11.1. Передача бита

Чтобы установить текущий бит данных, на линии SCL должен быть высокий уровень. В это время состояние линии SDA не может изменяться (если только речь не идет о состояниях "СТАРТ" или "СТОП" — об этом будет рассказано позже). Чтобы передать, например, "1", на SDA, и SCL должны иметь высокий уровень. Для передачи "0" на SDA должен присутствовать низкий уровень, а на SCL — высокий.

11.2. Состояние "СТАРТ"

Для информирования присоединенных микросхем о начале передачи данных должно быть сформировано состояние "СТАРТ". В этом состоянии линия SCL принимает высокий уровень, а SDA переключается с высокого на низкий уровень.

11.3. Состояние "СТОП"

Состояние "СТОП" устанавливается так: при высоком уровне на SCL линия SDA переключается с низкого на высокий уровень. При этом, как ясно из названия, передача данных завершается. Таким образом, ведущее устройство сигнализирует об окончании приема данных или о переходе к их передаче. На рис. 11.3 показаны состояния СТАРТ и СТОП.

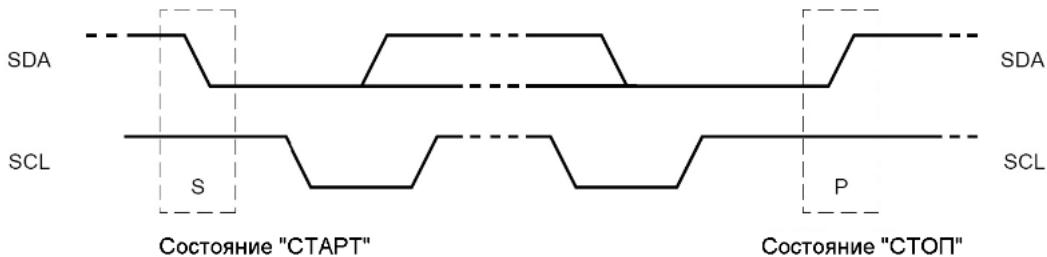


Рис. 11.3. Состояния "СТАРТ" и "СТОП" (источник: технический паспорт компании Philips)

11.4. Передача байта

При передаче байта сначала посыпается самый старший бит. Далее следуют все остальные вплоть до самого младшего бита (рис. 11.4).

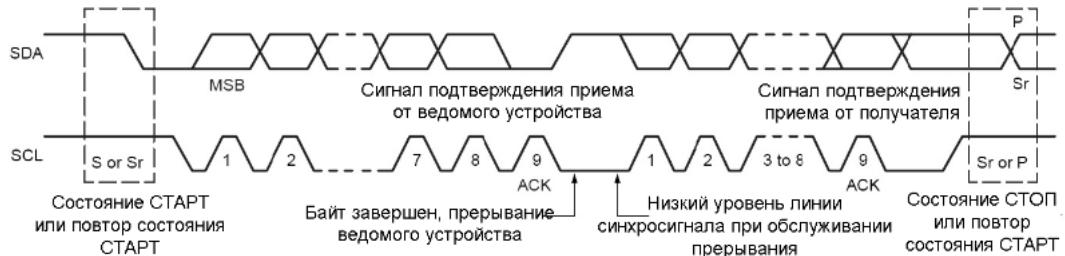


Рис. 11.4. Полный цикл передачи данных (источник: технический паспорт компании Philips)

11.5. Подтверждение

Устройство-приемник подтверждает получение данных путем квитирования (Acknowledgment). Квитирование осуществляется после передачи восьми битов данных (и восьми синхроимпульсов).

11.6. Адресация

Первый передаваемый байт после состояния СТАРТ, который посылает ведущее устройство, — это адрес ведомого, к которому хотел бы обратиться ведущий.

11.7. 7-битовая адресация

7-битовая адресация шины I²C допускает подключение до 128 устройств. Это типичная ситуация.

Дополнительная информация о работе с шиной I²C (I²C) находится на компакт-диске, прилагаемом к книге, в техническом паспорте I²C-шины компании Philips.



Глава 12

Arduino и температурный датчик LM75 с I²C-шиной

Температурные датчики очень важны для измерения температуры окружающей среды и температурного контроля различных узлов, например, двигателей. В первом случае мы хотим знать температуру в помещении или на улице, во втором — защитить, например, коробку передач двигателя от перегрева. Для этих целей хорошо подойдет датчик LM75 компании National Semiconductor (рис. 12.1).

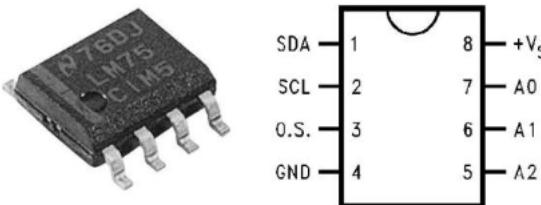


Рис. 12.1. Внешний вид и цоколевка датчика LM75
(источник: технический паспорт Philips)

Этот датчик может измерять температуру от -55 до $+125^{\circ}\text{C}$ с точностью $\pm 3^{\circ}\text{C}$. Конструкция проста и надежна. Для подключения датчика потребуются только два подтягивающих резистора (по $4,7\text{ кОм}$) на I²C-шине.

Вот семь битов адреса:

1	0	0	1	A2	A1	A0
MSB						LSB

Старшие четыре бита заданы производителем, следующие три бита задают адрес. Необходимо учитывать также восьмой бит, установленный в единицу. Полный адрес, при котором выводы от A0 до A1 подключены к общей шине, выглядит тогда следующим образом: 10010001 = Нех 91.

Для нашего эксперимента используем датчик LM75 в 8-выводном DIP-корпусе для монтажа на плате. Монтажная схема приведена на рис. 12.2, код программы — в листинге 12.1.

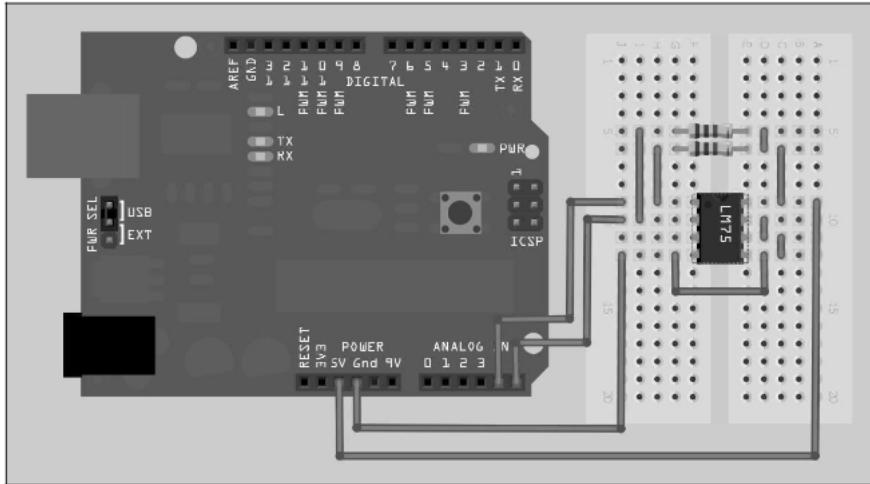


Рис. 12.2. Монтажная схема

Необходимые комплектующие изделия:

- плата Arduino/Freduino;
- панель с контактными гнездами;
- датчик температуры LM75 DIP;
- два резистора по 4,7 кОм;
- семь гибких монтажных проводов длиной примерно 3 см;
- четыре гибких монтажных провода длиной примерно 10 см.

Листинг 12.1. LM75.pde

```
// Franzis Arduino
// Считывание датчика температуры LM75 с I2C-шиной

#include <Wire.h>

#define LM75 (0x90 >> 1) // LM75 7-Bit Adresse

void setup()
{
    Wire.begin();
    Serial.begin(9600);
    Serial.println("LM75 IIC-Bus Temperatursensor");
}

void loop()
{
    byte msb, lsb=0;
```

```
float Grad=0;

Wire.beginTransmission(LM75);
Wire.send(0x00);
Wire.endTransmission();

Wire.requestFrom(LM75, 2);
while(Wire.available() < 2);
msb = Wire.receive();
lsb = Wire.receive();

if(msb<0x80)
{
    Grad=((msb*10)+(((lsb&0x80)>>7)*5));
}
else
{
    Grad=((msb*10)+(((lsb&0x80)>>7)*5));
    Grad-= (2555.0-Grad);
}

Grad=Grad/10;

Serial.print(Grad);
Serial.println(" Grad");

delay(1000);

}
```

В листинге 12.1 показано считывание температурного датчика, преобразование значений старшего и младшего байта в температуру и вывод ее в терминальной программе.

В примере используется библиотека `wire`, которая необходима для коммуникации шины I²C:

```
#include <Wire.h> // Библиотека Wire
```

Здесь устанавливается адрес для датчика. Требуется сдвигать исходный адрес Hех 91 на один бит вправо, т. к. Arduino использует 7-битовую адресацию:

```
#define LM75 (0x90 " 1) // 7-битовый адрес LM75
```

Теперь инициализируем шину I²C. Выводы для SCL и SDA — это аналоговые выводы 5 и 6 платы Arduino:

```
Wire.begin(); // Инициализация аппаратных средств
```

В основном цикле (листинг 12.2) мы считываем температуру через I²C-шину.

Листинг 12.2. Считывание температуры

```
Wire.beginTransmission(LM75); // Запуск I2C с указанием адреса
Wire.send(0x00); // Темп. находится в регистре 0
Wire.endTransmission(); // Конец передачи
Wire.requestFrom( LM75, 2); // Ожидаем 2 байта
while(Wire.available() < 2); // Имеются ли уже оба
// байта?
msb = Wire.receive(); // Получить Byte 1 MSB от датчика
lsb = Wire.receive(); // Получить Byte 2 LSB от датчика
```

Пересчет температуры иллюстрирует листинг 12.3. Старший разряд в байте указывает, положительная температура или отрицательная.

Листинг 12.3. Пересчет температуры

```
if(msb<0x80) {
Grad=( msb*10)+((1 sb&0x80)"7)*5);
}
else {
Grad=( msb*10)+((1 sb&0x80)"7)*5); Grad=(2555.0-Grad);
}
Grad=Grad/10;
```

Глава 13



Расширитель порта I²C с PCF8574

Если требуется большее количество портов ввода/вывода, чем плата Arduino может предоставить в распоряжение, то это можно осуществить при помощи модуля расширителя порта I²C. Популярная микросхема I²C-шины — расширитель порта PCF8574 фирмы Philips. Микросхема имеет восемь двунаправленных линий данных (рис. 13.1). При отсутствии сигнала выводы являются высокоомными, так что могут быть входами. Активное состояние всех линий низкоомное. Тем не менее, микросхема не имеет такого регистра направления передачи данных, как у микроконтроллера Arduino. Базовый адрес микросхемы — 0100 0000 (64) или Hex40. Путем адресации входов A0–A2 к шине I²C можно подключать до 8 одинаковых микросхем. Поэтому микросхема PCF8574 позволяет расширятьпорт до 64 линий. Схема внешних подключений приведена на рис. 13.2. У микросхем других изготовителей могут быть иные базовые адреса. В этом случае потребуется технический паспорт.

Листинг 13.1 демонстрирует ввод и вывод данных по I²C-шине. Здесь исходные последовательности чисел должны посыпаться в порт-расширитель, причем каждый раз состояние порта будет считано снова. При неподключенных выводах порта PCF8574 все посланные данные останутся без изменений. При подключении к общей шине, соответствующий бит данных читается как OFF.

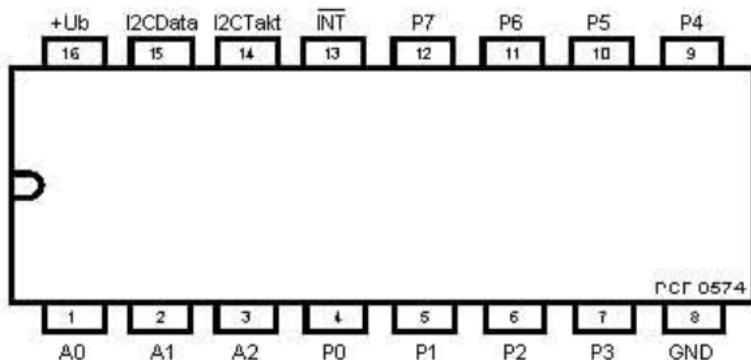


Рис. 13.1. Цоколевка расширителя порта PCF8574

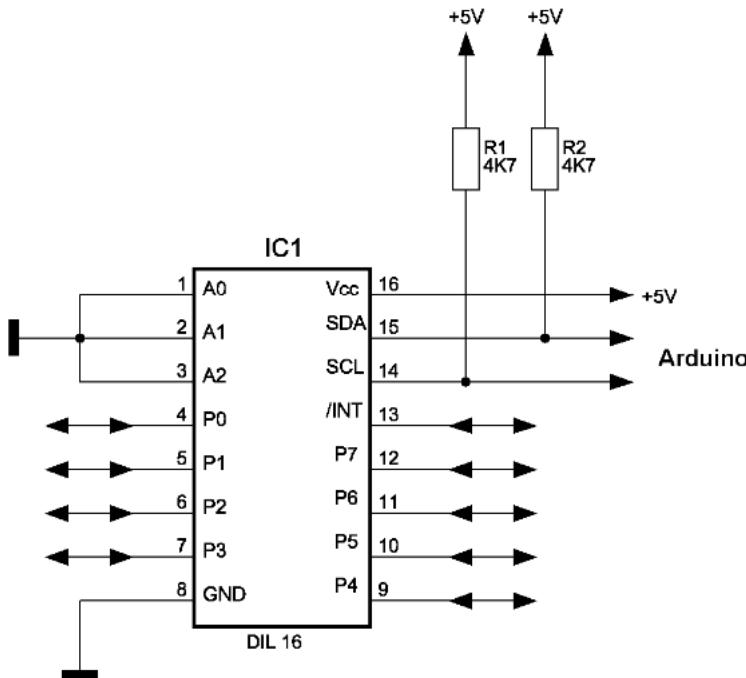


Рис. 13.2. Внешние подключения к PCF8574

При каждом доступе в режиме записи после состояния СТАРТ выдается адрес ведомого устройства. Далее следует передача байта данных, которые выставляются непосредственно на контактах порта. В принципе допустимо любое количество байтов данных, так что можно реализовать быструю смену конфигурации битов.

После каждого цикла записи должно осуществляться чтение. Для этого нужно снова адресовать микросхему. Непосредственный доступ для чтения для одного байта реализует команда `wire.receive()`. В принципе, последовательно может выполняться любое количество запросов на чтение. Но скорость обмена данными по I²C-шине не слишком высока. Основное преимущество шины — возможность управлять большим числом микросхем при помощи всего двух линий.

Листинг 13.1. PCF8574.pde

```
// Franzis Arduino
// Ввод/вывод данных через расширитель порта PCF8574A

#include <Wire.h>
#define addr 0x20

void setup()
```

```
{  
    Wire.begin();  
    Serial.begin(9600);  
}  
  
void loop()  
{  
  
    Serial.println("Alle Ports LOW");  
    PCF8574_Write(0x00);  
  
    Serial.print("Portzustand: ");  
    Serial.println(PCF8574_Read(), BIN);  
    delay(1000);  
  
    Serial.println("Alle Ports HIGH");  
    PCF8574_Write(0xff);  
  
    Serial.print("Portzustand: ");  
    Serial.println(PCF8574_Read(), BIN);  
    delay(1000);  
  
}  
  
void PCF8574_Write(byte data )  
{  
    Wire.beginTransmission(addr);  
    Wire.send(data);  
    Wire.endTransmission();  
}  
byte PCF8574_Read()  
{  
    byte data;  
    Wire.requestFrom(addr, 1);  
    if(Wire.available())  
    {  
        data = Wire.receive();  
    }  
    return data;  
}
```



Глава 14

Ультразвуковой датчик для определения дальности

Теперь подключим к плате микроконтроллера Arduino ультразвуковой датчик дальности. В результате плата Arduino сможет "видеть" как летучая мышь. От ультразвукового датчика посыпаются ультразвуковые импульсы и измеряется время до прихода в приемник первого эхо-сигнала. Зная скорость звука (примерно 300 м/с), можно определить расстояние до объекта. Точность измерения составляет несколько сантиметров. Имеются также системы с другими режимами работы. Вскоре после первого эхо-сигнала приемник снова включается. Таким образом, регистрируются дальнейшие эхо-сигналы. Так можно определить, нет ли первым объектом еще какого-нибудь объекта.

14.1. Ультразвуковой датчик SRF02

Ультразвуковой датчик SRF02 — это первый датчик из серии SRF, который имеет только один преобразователь ультразвука (рис. 14.1). Тем не менее, он заслуживает внимания. Прежде всего, в наличии есть интерфейсы RS-232-и I²C-шины, что порадует многих радиолюбителей. К тому же, это самый популярный и недорогой датчик.

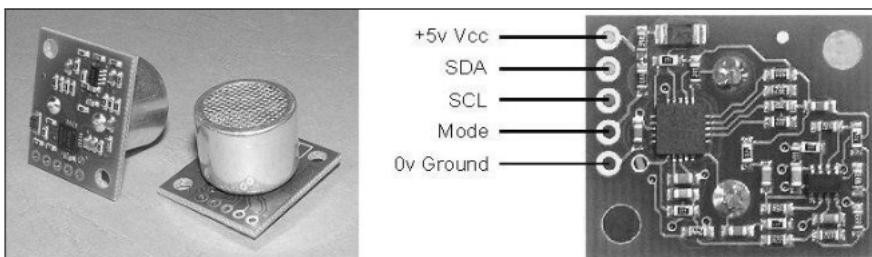


Рис. 14.1. Внешний вид и подключение ультразвукового датчика SRF02

Технические данные датчика SRF02:

- Рабочее напряжение — 5 В (стабилизированное).
- Сила тока — не более 4 мА (среднее значение).

- Частота ультразвука — 40 кГц.
- Дальность действия — от 15 см до 6 м.
- Интерфейсы — RS-232 (TTL) и I²C-шина.
- Единицы измерения — на выбор миллиметры, дюймы или микросекунды.
- Прост в использовании, не нуждается в настройке и калибровке.
- Габариты — 24×20×17 мм.

Датчик можно присоединять к RS-232 или I²C-шине. Соответственно могут объединяться до 16 изделий. В наших экспериментах задействован интерфейс I²C-шины, т. к. он уже есть на плате Arduino.

14.2. Считывание данных

Управление и манипулирование датчиком SRF02 очень просто, как будет видно позже в примерах программы. Каждое устройство, подключенное к шине I²C, в том числе наш SRF02, имеет собственный адрес. Это что-то вроде "номера дома", через который можно обращаться целенаправленно к ведомому устройству (датчику). Адрес ведомого уникальный, иначе возникли бы значительные проблемы. Стандартный адрес датчика SRF02 в шестнадцатеричной системе — Hex 0xE0, в десятичной — 224. Тем не менее, можно программно изменять ID-номер ведомого устройства.

Возможны следующие ID-номера ведомого устройства:

E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC и FE.

Как видим, допустимы 16 различных ID-номеров ведомого устройства. Поэтому можно присоединять одновременно до 16 ультразвуковых датчиков. Исходные адреса, однако, нельзя использовать для Arduino, т. к. адрес должен быть 7-битовым. Придется снова сдвигать адрес, например, Hex E0, на один разряд вправо ($0xE0 >> 1$).

В листинге 14.1 дальность отсчитывается в сантиметрах. Датчик имеет стандартный внутренний адрес Hex E0. Единицу измерения можно изменить на дюймы или микросекунды, поменяв содержимое соответствующих регистров. Предусмотрена и возможность калибровки. Более подробно об этом можно узнать в техническом паспорте SRF02.

Листинг 14.1. SRF02.pde

```
// Franzis Arduino
// Вывод данных ультразвукового датчика SRF02 фирмы Devantech
#include <Wire.h>

#define srfAddress (0xE0 >> 1)
#define cmdByte 0x00
#define rangeByte 0x02

byte MSB = 0x00;
```

```
byte LSB = 0x00;

void setup()
{
    Serial.begin(9600);
    Wire.begin();
    Serial.println("SRF02 Ultraschallsensor");
}

void loop()
{
    int rangeData = getRange();
    Serial.print("Entfernung: ");
    Serial.print(rangeData);
    Serial.println("cm");
    delay(250);
}

int getRange()
{
    int range = 0;

    Wire.beginTransmission(srfAddress);
    Wire.send(cmdByte);
    Wire.send(0x51);
    Wire.endTransmission();

    delay(100);
    Wire.beginTransmission(srfAddress);
    Wire.send(rangeByte);
    Wire.endTransmission();

    Wire.requestFrom(srfAddress, 2);
    while(Wire.available() < 2);
    MSB = Wire.receive();
    LSB = Wire.receive();

    range = (MSB << 8) + LSB;

    return(range);
}
```



Глава 15

Сопряжение платы Arduino с GPS

Глобальная система навигации и определения положения (GPS — Global Positioning System) — это основа всех современных систем навигации и определения местоположения для наземных, воздушных и морских объектов. Эта система мобильной спутниковой связи была разработана в 70-е годы XX в. американским министерством обороны для военных целей и действует до настоящего времени. Система состоит из 24 спутников, которые врачаются круглосуточно вокруг земли по 6 круговым орбитальным траекториям на расстоянии примерно 20 000 км. Такая конфигурация обеспечивает прием сигнала, по меньшей мере, от четырех спутников. Это дает возможность расчета местоположения по принципу триангуляции (минимум три спутника). При этом расстояния до спутников могут определяться по времени распространения сигнала. Так как позиция всех спутников известна, то можно определить и местонахождение. Каждый спутник 50 раз в секунду посылает три различных сигнала: псевдослучайный код (для определения местоположения), альманах-сигнал (местоположение спутника) и сигнал коррекции времени (определение времени). Псевдослучайный код высыпается на двух разных частотах. Один предусмотрен для военного использования, второй — для гражданского. Для гражданского использования до мая 2000 года сигнал имел специально введенную ошибку, которая ограничивала точность определения позиции примерно до 100 м. Второго мая 2000 года в 5 часов 5 минут этот режим "избирательной доступности" (selective availability) был отключен. На рис. 15.1 приведен внешний вид устройства GPS, подключаемого к компьютеру (GPS-мышь).



Рис. 15.1. Стандартное GPS-устройство, подключаемое к компьютеру
(источник: компания Elmicro)

Применяя высококачественный приемник GPS и математические методы коррекции, до 2000 года можно было получить точность до 20–30 м.

После отмены искажения сигнала для гражданского использования точность достигает 5–25 м в зависимости от типа GPS-приемника, что более чем достаточно. Тем не менее, качество приема сигналов GPS зависит от положения спутников на небе, конкретного местоположения объекта, времени суток и метеоусловий.

Для оценки положения спутников указывают два значения: H-DOP — для горизонтальной и V-DOP — для вертикальной позиции (это не является существенным для автомобильной GPS). H-DOP ниже 4 является очень хорошим, а более 8 — плохим. Значения H-DOP становятся хуже, если спутники находятся высоко над горизонтом. Напротив, значение V-DOP хуже, если спутники располагаются очень близко к горизонту. Значения H-DOP и V-DOP выводятся в комплекте NMEA \$GPGSA.

Чтобы без проблем преодолевать большие расстояния, сигналы GPS посылаются в диапазоне сверхвысоких частот (~1,5 ГГц). При этом сигнал существенно ослабляется вне зоны прямой видимости спутника, например, в зданиях. Для приема сигналов GPS в зданиях созданы специальные антенны и приемники. Поэтому, создавая собственное GPS-приложение, нужно обращать внимание, прежде всего, на высокое качество антенны и чувствительность приемника.

В будущем предусмотрена разработка совместного проекта спутниковой системы навигации Европейского союза и Европейского космического агентства под названием Галилео (Galileo).

15.1. Сколько требуется спутников?

Три спутника достаточно для определения точной позиции и высоты. Однако это требует точной синхронизации. На практике большинство приемников GPS не располагает часами, которые были бы достаточно точны, чтобы правильно рассчитывать время задержки. Для более точного определения местоположения в большинстве случаев требуется сигнал четвертого спутника. Тем не менее, теоретически трех спутников всегда достаточно для точного позиционирования на плоскости (без определения высоты).

15.2. Как подключить GPS к Arduino?

Существуют дешевые модули GPS с интерфейсом RS-232 для микроконтроллера. В большинстве случаев они имеют штекер Mini-DIN, но для них отсутствует расположение выводов. В Интернете на сайте производителя можно найти технические паспорта с соответствующими разводками контактов.

Обратите внимание в любом случае, что существуют два типа приемников. Некоторые приемники имеют интерфейс RS-232 с уровнем напряжения ± 12 В и не могут присоединяться непосредственно к нашему микроконтроллеру. Необходимо преобразование к ТТЛ-уровню на напряжение 5 В при помощи микросхемы MAX232, т. к. иначе возможно повреждение контроллера.

Устройства GPS работают в большинстве случаев со скоростью 4 800 бод. Однако с помощью программного обеспечения, поставляемого производителем устройства, например, фирмой SIRF, скорость и желаемый протокол можно перенастраивать.

В нашем примере организован последовательный поток данных через аппаратный интерфейс UART. Вывод 7xD GPS-устройства должен присоединяться к цифровому выводу 0 платы. При программировании Arduino нужно отсоединять этот контакт, т. к. в противном случае коммуникация отсутствует и программирование не произойдет. Также обратите внимание, что необходимо соединить общие шины устройства GPS и экспериментальной платы (рис. 15.2).

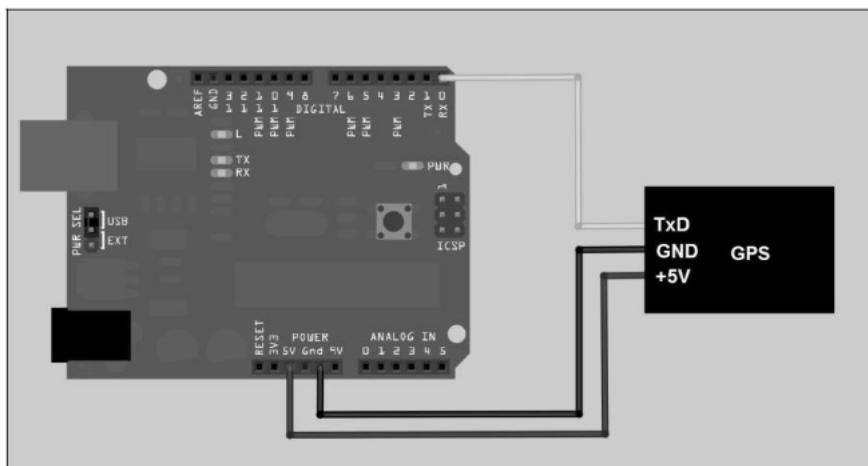


Рис. 15.2. Подключение устройства GPS к Arduino

СОВЕТ

Если у вас возникли проблемы с работой устройства GPS, то можно кратковременно отсоединить небольшой встроенный аккумулятор и потом опять его припаять. В этом случае параметры устройства возвращаются к заводской установке.

15.3. GPS-протокол

В листинге 15.1 приведен протокол GPS.

Листинг 15.1. Протокол GPS

```
SGPRMC,081836,A,3751.65, S, 14507.36,E,000.0,360.0,130998,011.3,E*62
SGPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68
225446      Time of fix 22:54:46 UTC
A            Navigation receiver warning A = Valid position, V = Warning
4916.45,N   Latitude 49 deg. 16.45 min. North
12311.12,W  Longitude 123 deg. 11.12 min. West
```

000.5 Speed over ground, Knots
 054.7 Course Made Good, degrees true
 191194 UTC Date of fix, 19 November 1994
 020.3.E Magnetic Variation, 20.3 deg. East
 *68 mandatory checksum
 SGPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W*70
 1 2 3 4 5 6 7 8 9 10 11 12
 1 220516 Time Stamp
 2 A validity - A-ok, V-invalid
 3 5133.82 current Latitude
 4 N North/South
 5 00042.24 current Longitude
 6 W East/West
 7 173.8 Speed in knots
 8 231.8 True course
 9 130694 Date Stamp
 10 004.2 Variation
 11 W East/West
 12 *70 checksum

В листинге 15.2 приведен обзор RMC-протокола. Более подробная информация по этому вопросу находится в Интернете на странице <http://www.kowoma.de/gps/>. Программа для экспериментов с устройством GPS приведена в листинге 15.3.

Листинг 15.2. RMC-протокол

NMEA 0183 version 3.00 active the Mode indicator field is added
 \$GPRMC,hnnimss.ss,AJlll.lLa,yyyyy.yy,a,x.x,x.x,ddnimyy.x.x,a,m*hh

Field #
 1 = UTC time of fix
 2= Data status (A=Valid position, V=navigation receiver warning)
 3 = Latitude of fix
 4 = N or S of longitude
 5 = Longitude of fix
 6 = E or W of longitude
 7 = Speed over ground in knots
 8 = Track made good in degrees True
 9 = UTC date of fix
 10 = Magnetic Variation degrees (Easterly var. subtracts from true course)
 11 = E or W of magnetic Variation
 12 = Mode indicator, (A=Autonomous, D=Differential. E=Estimated. N=Data not valid)
 13 = Checksum

Листинг 15.3. GPS.pde

```
// Franzis Arduino
// Считывание GPS-мыши 4 800 бод

#include <string.h>
#include <ctype.h>

int ledPin = 13;           // LED
int rxPin = 0;             // RX PIN
int txPin = 1;             // TX TX
int byteGPS=-1;
char linea[300] = "";
char comandoGPR[7] = "$GPRMC";
int cont=0;
int bien=0;
int conta=0;
int indices[13];

void setup()
{
    pinMode(ledPin, OUTPUT);
    pinMode(rxPin, INPUT);
    pinMode(txPin, OUTPUT);
    Serial.begin(4800);
    for (int i=0;i<300;i++)// Инициализация буфера получения данных
    {
        linea[i]=' ';
    }
}

void loop()
{
    digitalWrite(ledPin, HIGH);
    byteGPS=Serial.read();      // Получение байта от интерфейса
    if (byteGPS == -1)
    {
        delay(100);
    }
    else
    {
        linea[conta]=byteGPS;
```

```
conta++;
Serial.print(byteGPS, BYTE);
if (byteGPS==13)
{
    digitalWrite(ledPin, LOW);
    cont=0;
    bien=0;
    for (int i=1;i<7;i++)
    {
        if (linea[i]==comandoGPR[i-1])
        {
            bien++;
        }
    }
    if(bien==6)
    {
        for (int i=0;i<300;i++){
            if (linea[i]==',')
            {
                indices[cont]=i;
                cont++;
            }
            if (linea[i]=='*')
            {
                indices[12]=i;
                cont++;
            }
        }
        Serial.println("");
        Serial.println("");
        Serial.println("-----");
        for (int i=0;i<12;i++){
            switch(i){
                case 0 :Serial.print("Time in UTC (HhMmSs): ");break;
                case 1 :Serial.print("Status (A=OK,V=KO): ");break;
                case 2 :Serial.print("Latitude: ");break;
                case 3 :Serial.print("Direction (N/S): ");break;
                case 4 :Serial.print("Longitude: ");break;
                case 5 :Serial.print("Direction (E/W): ");break;
                case 6 :Serial.print("Velocity in knots: ");break;
                case 7 :Serial.print("Heading in degrees: ");break;
                case 8 :Serial.print("Date UTC (DdMmAa): ");break;
            }
        }
    }
}
```

```
    case 9 :Serial.print("Magnetic degrees: ");break;
    case 10 :Serial.print("(E/W): ");break;
    case 11 :Serial.print("Mode: ");break;
    case 12 :Serial.print("Checksum: ");break;
}
for (int j=indices[i];j<(indices[i+1]-1);j++)
{
    Serial.print(linea[j+1]);
}
Serial.println("");
}
Serial.println("-----");
}
conta=0;
for (int i=0;i<300;i++)
{
    linea[i]=' ';
}
}
}
```



Глава 16

Сервопривод с платой Servo для Arduino

В некоторых конструкциях требуется небольшой привод. Для этих целей созданы недорогие компактные модели сервопривода (рис. 16.1). Сервоприводы выпускают в самых разных исполнениях, но принцип их работы одинаков. Приводом управляют, изменяя длительность подаваемых на него импульсов.



Рис. 16.1. Стандартный сервопривод компании Conrad Electronic SE

16.1. Как функционирует сервопривод?

Сервопривод — это регулируемый редукторный электродвигатель. Он состоит из приводного механизма с электродвигателем постоянного тока, платы управления и потенциометра, который обеспечивает обратную связь.

Сервопривод имеет три вывода: два для питания, которое составляет в большинстве случаев 4,8–6 В, и один для задания значения позиции. Сервоприводу необходимо указывать направление вращения. Это происходит с помощью импульсов, которые, в зависимости от следящего устройства, имеют длительность 1–2 мс

с паузой 20 мс (рис. 16.2). Истинное положение измеряется внутренним потенциометром и возвращается в плату управления, которая подстраивает электродвигатель приводного механизма в зависимости от этого значения.

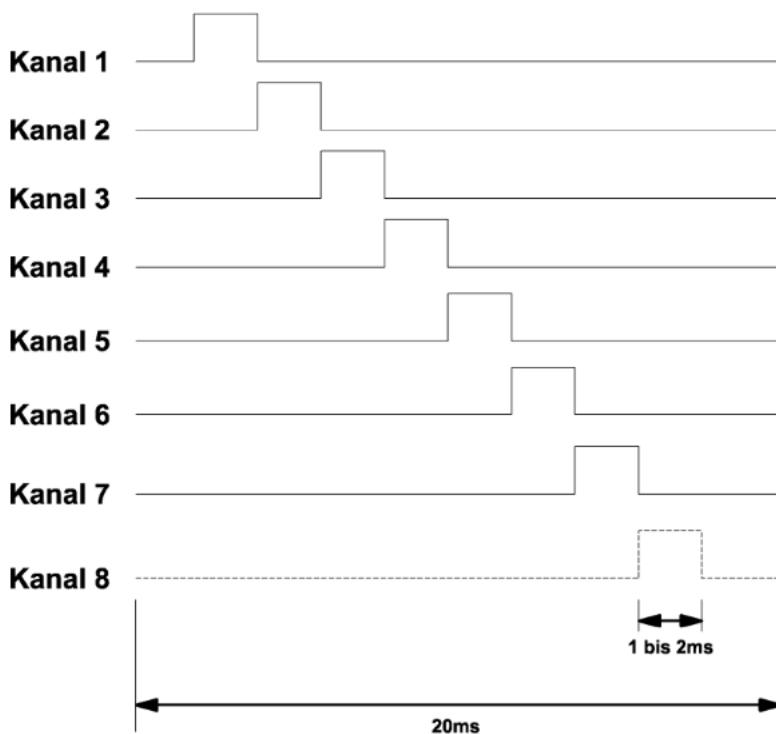


Рис. 16.2. Временная диаграмма управления сервопривода с обратной связью

16.2. Подключение привода к Arduino

При подключении сервопривода непосредственно к плате Arduino, желательно использовать внешний источник питания, т. к. порт USB не может обеспечить требуемый ток. Здесь наилучшим образом подойдут аккумуляторы (например, четыре элемента Mignon-AA). Общую шину (GND) сервопривода следует соединить с контактом GND платы Arduino. Общие провода аккумулятора и платы Arduino также должны быть соединены. Монтажная схема приведена на рис. 16.3, код программы — в листинге 16.1.

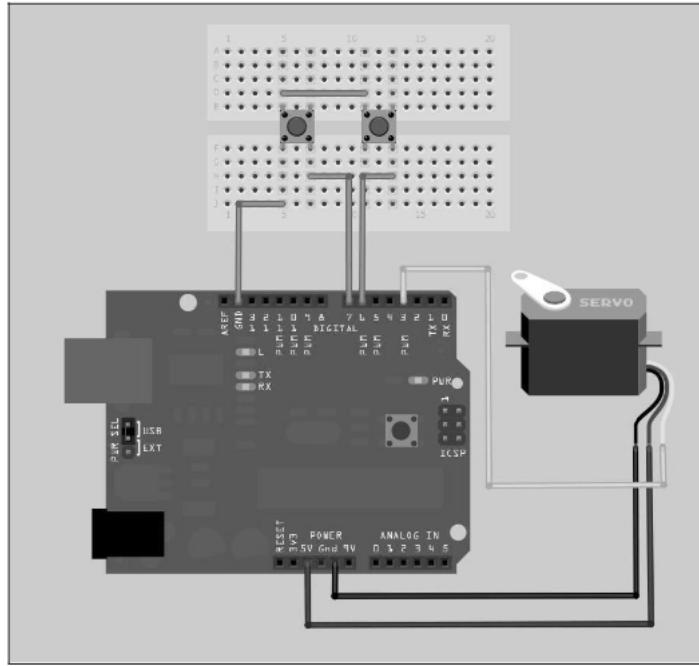


Рис. 16.3. Подключение небольшого сервопривода к плате Arduino

Листинг 16.1. Servo.pde

```
// Franzis Arduino
// Управление сервопривода

int servoPin      = 2;      // Вывод Servo
int minPulse      = 600;    // мин. позиция servo
int maxPulse      = 2400;   // макс. позиция servo
int turnRate       = 1;      // Чем больше скорость, тем быстрее!
int refreshTime   = 20;     // Коррекция частоты 50 Гц = 20 мс

int SW1           = 7;      // Кнопка 1 влево
int SW2           = 6;      // Кнопка 2 вправо

int centerServo;        // Среднее положение
int pulseWidth;         // Ширина импульса
long lastPulse     = 0;    // Сохранение периода последнего импульса

void setup()
{
```

```
pinMode(servoPin, OUTPUT);
pinMode(SW1, INPUT);
digitalWrite(SW1,HIGH);
pinMode(SW2, INPUT);
digitalWrite(SW2,HIGH);
centerServo = maxPulse - ((maxPulse - minPulse)/2);
pulseWidth = centerServo;
}

void loop()
{
    // Wurde ein Taster gedrückt?
    if(!digitalRead(SW1) || !digitalRead(SW2))
    {
        if(!digitalRead(SW1)) {pulseWidth = pulseWidth - turnRate;}
        if(!digitalRead(SW2)) {pulseWidth = pulseWidth + turnRate;}

        // Begrenzung
        if(pulseWidth > maxPulse) {pulseWidth = maxPulse;}
        if(pulseWidth < minPulse) {pulseWidth = minPulse;}
        delay(2);
    }

    // Servo ansteuern
    if (millis() - lastPulse >= refreshTime)
    {
        digitalWrite(servoPin, HIGH);
        delayMicroseconds(pulseWidth);
        digitalWrite(servoPin, LOW);
        lastPulse = millis();
    }
}
```



Глава 17

Жидкокристаллические дисплеи

До сих пор в большинстве случаев для вывода информации использовались светодиоды или терминальная программа. Тем не менее, для многих приложений такое отображение результатов по различным причинам нецелесообразно. Между тем, объединив недорогие жидкокристаллические индикаторы (ЖКИ, LCD — Liquid Crystal Display) с платформой Arduino, можно просто и эффективно осуществить вывод информации. ЖКИ находят применение во многих электронных устройствах, например, в измерительных приборах, мобильных телефонах, электронных часах с цифровой индикацией и микрокалькуляторах. По такой же технологии работают и видеопроекторы. На рис. 17.1 изображен монохромный дисплей "Industriestandard 5×7 Dot Matrix" с 4 строками по 16 символов.



Рис. 17.1. Жидкокристаллический дисплей (источник: компания Conrad Electronic)

Основа ЖКИ состоит из двух стеклянных подложек и слоя жидкокристаллического вещества (ЖК) между ними. Особенность жидких кристаллов состоит в том, что они поворачивают плоскость поляризации света. Этот эффект зависит от действия электрического поля. На обе стеклянные пластины напыляются тонкие металлические слои (электроды). Чтобы получить поляризованный свет, на верхнюю стеклянную пластинку приклеивается поляризационная пленка (поляризатор). На нижнюю стеклянную пластинку такая пленка клеится тоже, но с плоскостью поляризации, повернутой на 90° — анализатор.

В режиме покоя (при отсутствии электрического сигнала) слой ЖК поворачивает плоскость поляризации падающего света на 90° , так чтобы свет мог беспрепятственно пройти анализатор. Структура ЖКИ прозрачна для света. Если приложить определенное напряжение к металлическим электродам, молекулы ЖК начинают переориентироваться. Вследствие этого плоскость поляризации света поворачивается, например, еще на 90° . В результате анализатор преграждает путь свету и ЖКИ становится непрозрачным.

17.1. Поляризация дисплеев

Под поляризацией ЖК-дисплея понимают не полярность источника питания. Речь здесь идет о структуре дисплея. В большинстве ЖКИ используются нематические жидкие кристаллы (TN — Twisted-Nematic), обеспечивающие поворот плоскости поляризации света на 90° . При технологии STN (Super-Twisted-Nematics) плоскость поляризации света поворачивается минимум на 180° . Вследствие этого достигается лучшая контрастность изображения, но проявляется и нежелательное цветное окрашивание (обычно желто-зеленое и голубое). Чтобы устранить этот вредный эффект, в технологии FSTN на внешнюю поверхность экрана наносят дополнительную компенсирующую пленку. Из-за возникающих при этом потерь света данная технология рациональна только для дисплеев с подсветкой. Цветное окрашивание особенно заметно для дисплеев без подсветки или с белой подсветкой. Цветная подсветка (например, подсветка желто-зеленым светодиодом) меняет фоновый цвет изображения. Так, голубой фон жидкокристаллического дисплея с желто-зеленой светодиодной подсветкой будет выглядеть всегда желто-зеленым.

17.2. Статическое управление и мультиплексный режим

Небольшими дисплеями с ограниченным набором отображаемых символов управляют в большинстве случаев статично. У статических дисплеев лучшая контрастность и наибольший угол обзора. Монохромная технология TN здесь полностью оправдывает себя. Однако при увеличении размеров дисплея для управления в статическом режиме необходимо все больше линий (например, матрица $128 \times 64 = 8\,192$ сегмента = 8 192 линии). Для уменьшения числа управляющих каналов применяется мультиплексный режим. Дисплей разделяется на строки и столбцы так, что в каждой точке скрещивания находится сегмент изображения ($128 + 64 = 192$ линии). Стока последовательно переключаются за строкой (т. е. коэффициент сжатия равен 1:64). Поскольку в текущий момент активна только одна строка, то контрастность и угол обзора дисплея уменьшаются.

17.3. Угол обзора

Каждый ЖК-дисплей имеет так называемый оптимальный сектор (или угол) обзора, в пределах которого контрастность максимальна. Большинство дисплеев производятся для обзора в горизонтальной плоскости. Это соответствует, например микрокалькулятору, который лежит на столе.

Дисплеи, предназначенные для обзора по вертикали лучше всего устанавливать на лицевую панель настольного устройства. На любой дисплей лучше смотреть строго перпендикулярно.

17.4. Отражающие, пропускающие и полупрозрачные ЖКИ

Отражающие (неподсвечиваемые) дисплеи имеют на нижней подложке 100%-й рефлектор (отражатель). Задняя подсветка подложки невозможна.

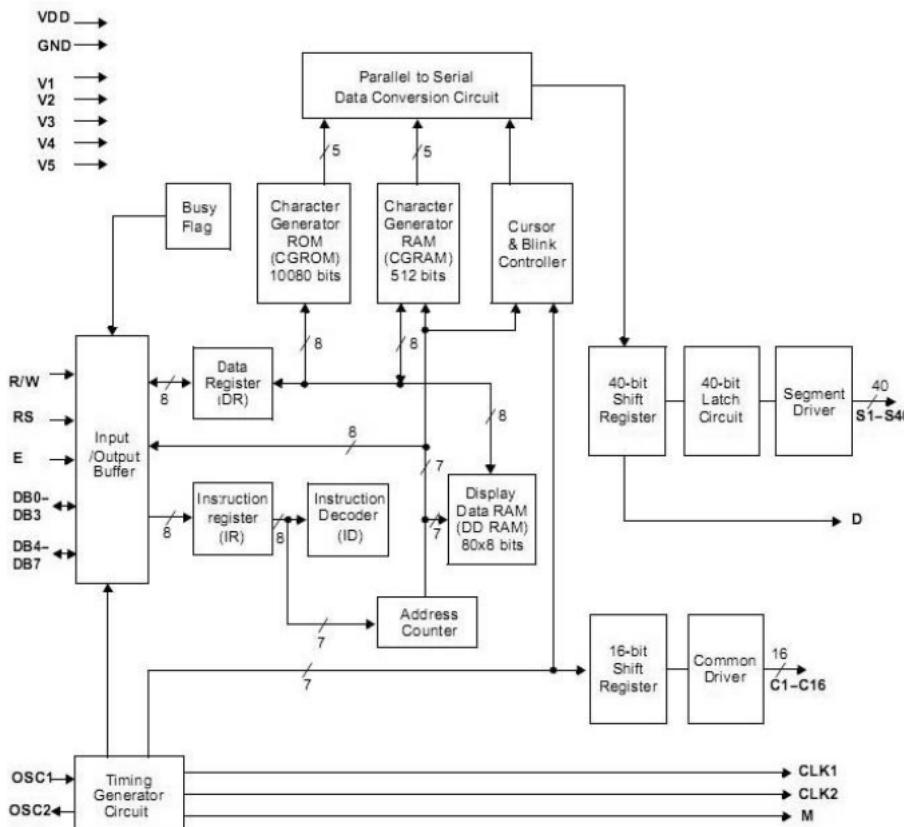


Рис. 17.2. Структурная схема контроллера дисплея KS0066
(источник: технический паспорт компании Samsung)

Отражатель на подложке полупрозрачных дисплеев частично проницаемый. Такие ЖКИ работают как с подсветкой, так и без нее. Хотя без подсветки они выглядят тускло по сравнению с отражающими, все же это, пожалуй, лучший компромисс для подсвечиваемых ЖКИ. В пропускающих дисплеях нет отражателя. Они могут функционировать только с яркой подсветкой снизу.

Дисплеи с точечной матрицей (Dot-Matrix) производят многие фирмы во всем мире (и особенно на Тайване). Наряду с дисплеями известной компании Datavision имеются изделия, производителя которых вовсе определить невозможно. К счастью, назначение и подключение дисплеев всегда одинаковые. В этой книге рассматриваются дисплеи, которые используют контроллер типа HD44780 (или совместимые).

Параметры микросхем контроллеров ЖКИ стандартизированы для всех производителей, поэтому все ЖК-дисплеи совместимы. Далее мы воспользуемся контроллером HD44780 компании Hitachi или совместимым KS0066 (рис. 17.2).

17.5. Установка контрастности дисплея

Контрастность ЖКИ-модулей можно изменять в простейшем случае с помощью потенциометра 10 кОм, который присоединяется к источнику питания Vcc (+5 В) и общейшине GND. Движок потенциометра подключают к выводу Vee в ЖКИ-модуле (рис. 17.3).

Подключение добавочного сопротивления между напряжением Vcc (+5 В) и выводом потенциометра улучшает регулировку контрастности ЖКИ.

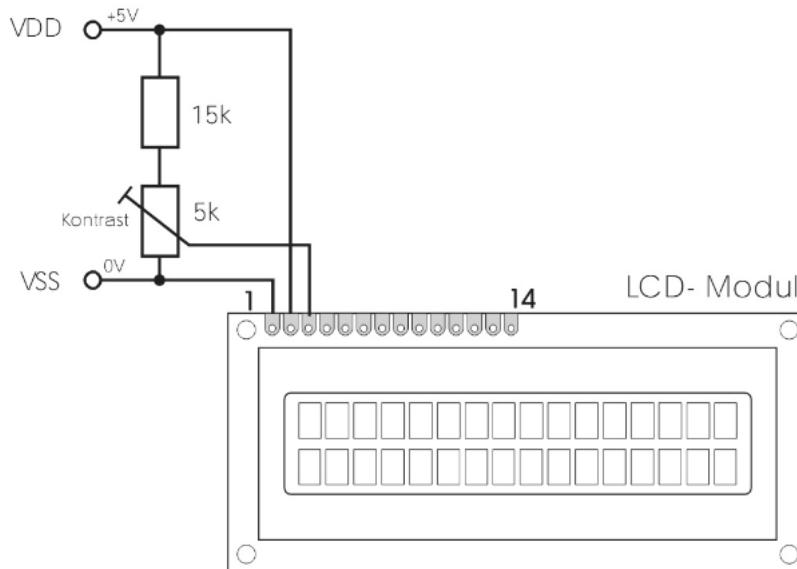


Рис. 17.3. Регулятор контрастности
(источник: технический паспорт компании Electronic Assembly)

Напряжение на выводе V_{ee} должно изменяться от 0 до 1,5 В при температуре окружающей среды от 0 до 40 °С. Если регулируемый диапазон не оптимален (ЖКИ имеют разброс параметров), нужно подкорректировать добавочное сопротивление. На практике его значения находятся в диапазоне 10–22 кОм.

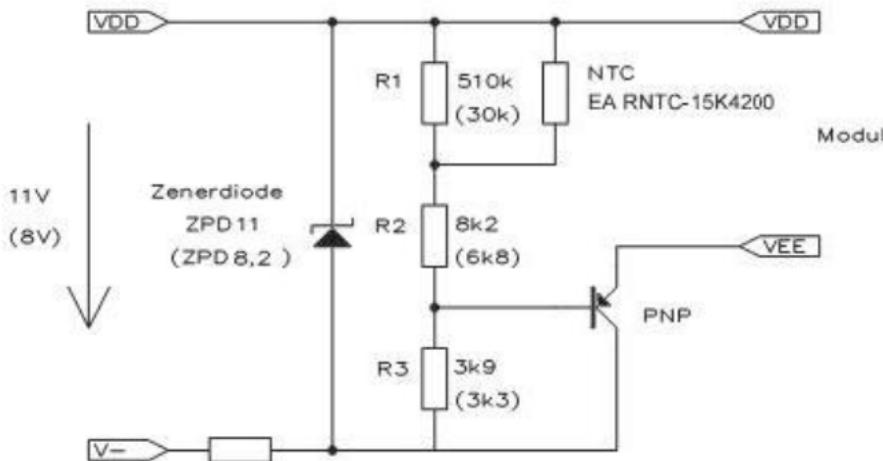


Рис. 17.4. Управление контрастностью с температурной компенсацией
(источник: технический паспорт компании Electronic Assembly)

При эксплуатации дисплея за пределами 0–40 °С целесообразно подключить дополнительную схему (рис. 17.4), адаптирующую контрастность к температуре окружающей среды. Температура измеряется NTC-термистором (Negative Temperature Coefficient Thermistor), который меняет напряжение контрастности через p - n - p -транзистор. Контрастность изображения на ЖКИ сильно зависит от температуры. При слишком низких температурах (менее 0 °С) разборчивость резко ухудшается.

17.6. Набор отображаемых символов

Набор отображаемых символов встроен в контроллер дисплея. При последовательной записи старших и младших битов получится байт данных для соответствующего ASCII-символа. Пример кодировки ASCII-символа "A" — 01000001. Отображаемые символы и их кодировка приведены на рис. 17.5.

Lower 4 Bits	Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	CG RAM (1)	Ø Ø P ^ P											-タミør				
xxxx0000																	
xxxx0001	(2)	! 1 A Q a q											アチムäq				
xxxx0010	(3)	" 2 B R b r											「イツメøø				
xxxx0011	(4)	# 3 C S c s											」ウテモεω				
xxxx0100	(5)	\$ 4 D T d t											、エトムø				
xxxx0101	(6)	% 5 E U e u											・オナムøü				
xxxx0110	(7)	& 6 F V f v											ヲカニエøΣ				
xxxx0111	(8)	' 7 G W g w											アキヌラgπ				
xxxx1000	(1)	(8 H X h x											イクネリjx				
xxxx1001	(2)) 9 I Y i y											ヲルノル"y				
xxxx1010	(3)	* : J Z j z											エコハレj				
xxxx1011	(4)	+ ; K [k {											オサヒロ*ス				
xxxx1100	(5)	, < L ¥ 1											ヲシフワΦ¶				
xxxx1101	(6)	- = M] m }											ユスヘンモ÷				
xxxx1110	(7)	. > N ^ n *											エセホ^n				
xxxx1111	(8)	/ ? O _ o *											レソマ¤ö				

Рис. 17.5. Кодировка отображаемых символов ЖКИ
(источник: технический паспорт компании Samsung)

17.7. Расположение выводов распространенных ЖКИ

В табл. 17.1 приведено описание выводов большинства ЖК-дисплеев без подсветки. Чтобы не повредить ЖКИ, перед подключением к микроконтроллеру рекомендуется ознакомиться с техническим паспортом.

Таблица 17.1. Назначение выводов ЖКИ без подсветки

Номер вывода	Обозначение	Логический уровень	Описание
1	VSS	L	Питание 0 В, GND
2	VDD	H	Питание +5 В
3	VEE	—	Напряжение для установки контрастности дисплея 0–1,5 В
4	RS	H/L	Выбор регистра
5	R/W	H/L	H: Чтение / L: Запись
6	E	H	Разрешение
7	D0	H/L	Линия передачи данных 0 (LSB)
8	D1	H/L	Линия передачи данных 1
9	D2	H/L	Линия передачи данных 2
10	D3	H/L	Линия передачи данных 3
11	D4	H/L	Линия передачи данных 4
12	D5	H/L	Линия передачи данных 5
13	D6	H/L	Линия передачи данных 6
14	D7	H/L	Линия передачи данных 7 (MSB)

Применение ЖКИ-модулей с подсветкой всегда требует осторожности. Некоторые производители подключают выводы светодиодной подсветки не как обычно к контактам 15 и 16, а к контактам 1 и 2. Следует внимательно прочитать паспорт производителя. При отсутствии под рукой паспорта, можно исследовать печатные проводники и найти выводы подсветки. Описание выводов наиболее распространенных ЖКИ с подсветкой приведено в табл. 17.2.

Таблица 17.2. Назначение выводов ЖКИ с подсветкой

Номер вывода	Обозначение	Логический уровень	Описание
1	VSS	L	Питание 0 В, GND
2	VDD	H	Питание +5 В
3	VEE	—	Напряжение дисплея 0–0,5 В
4	RS	H/L	Выбор регистра
5	R/W	H/L	H: Чтение / L: Запись
6	E	H	Разрешение
7	DO	H/L	Линия передачи данных 0 (LSB)
8	D1	H/L	Линия передачи данных 1

Таблица 17.2 (окончание)

Номер вывода	Обозначение	Логический уровень	Описание
9	D2	H/L	Линия передачи данных 2
10	D3	H/L	Линия передачи данных 3
11	D4	H/L	Линия передачи данных 4
12	D5	H/L	Линия передачи данных 5
13	D6	H/L	Линия передачи данных 6
14	D7	H/L	Линия передачи данных 7 (MSB)
15	LED +	-	Плюс питания светодиода подсветки (необходимо добавочное сопротивление)
16	LED-	-	Минус питания светодиода подсветки

17.8. Управление дисплеем от микроконтроллера

Управление ЖКИ-модулем осуществляется через шину данных D0–D7 (D4–D7 при 4-разряднойшине данных), а также по линиям RS, R/W и E. Сигнал RS служит для выбора команды ($RS = 0$) или регистра данных ($RS = 1$). Сигналы R/W указывают, должно ли быть чтение ($R/W = 1$) или запись ($R/W = 0$). Контакт E (Enable, Разрешение) служит для управления передачей данных. В режиме простого вывода Enable имеет уровень логического нуля. Во время доступа для чтения считываемые данныечитываются до тех пор, пока на выводе Enable присутствует логическая единица. При доступе в режиме записи данные дисплеяпринимаются по спаду импульса на контакте Enable.

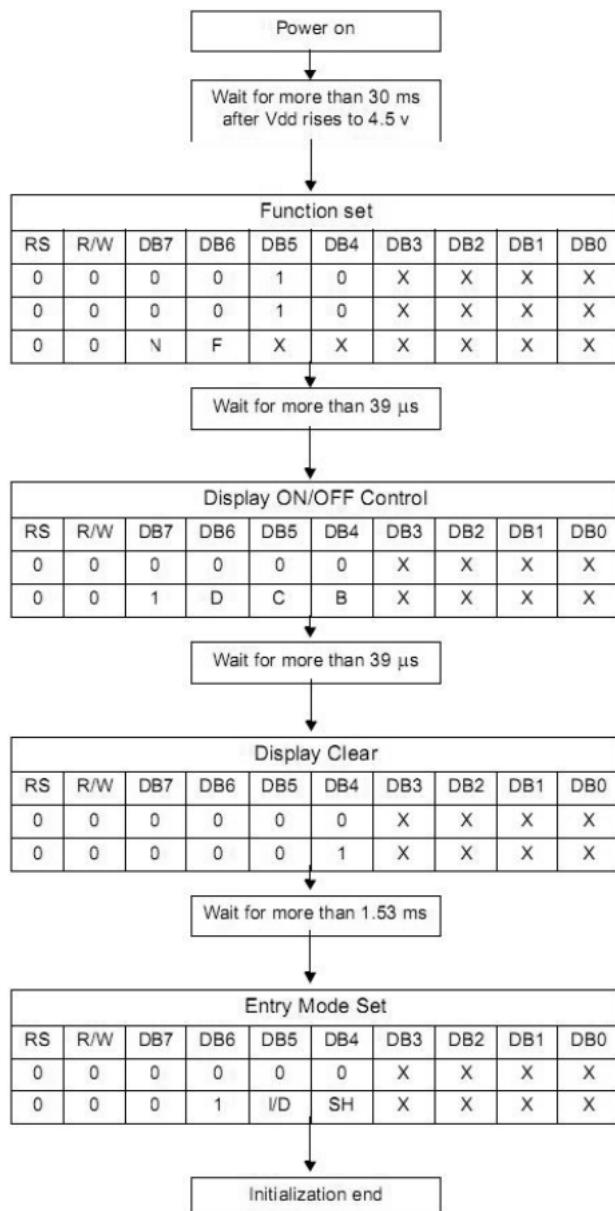
17.9. Инициализация дисплеев

На рис. 17.6 показана инициализация режима 4-разрядной шины данных, который позже потребуется нам для экспериментов с дисплеем. Алгоритм инициализации прост. Всегда, когда посылаются команды в ЖКИ-модуль, на выводе RS должен присутствовать низкий уровень, если посылаются данные, он должен переключаться на высокий уровень. Таким образом, при инициализации вывод RS всегда имеет низкий уровень.

Пример инициализации в 4-разрядном режиме:

1. Ожидание примерно 15 мс.
2. Передача Нех30 (установка интерфейса на 8 бит).
3. Ожидание примерно 5 мс.

4. Передача Hex30 (установка интерфейса на 8 бит).
5. Ожидание примерно 100 мкс.
6. Передача Hex30 (установка интерфейса на 8 бит).
7. Передача Hex20 (установка интерфейса на 8 бит).



N	0	1-line mode
	1	2-line mode

F	0	display off
	1	display on

D	0	display off
	1	display on

C	0	cursor off
	1	cursor on

B	0	blink off
	1	blink on

I/D	0	decrement mode
	1	increment mode

SH	0	entire shift off
	1	entire shift on

Рис. 17.6. Инициализация 4-разрядной шины данных
(источник: технический паспорт компании Samsung)

17.10. Подключение дисплея к Arduino

Теперь присоединим ЖКИ-модуль к плате Arduino. В табл. 17.3 перечислены используемые выводы экспериментальной платы. В нашем эксперименте потенциометр подключен без дополнительного резистора. Монтажная схема изображена на рис. 17.7.

Таблица 17.3. Выходы платы Arduino для подключения ЖКИ

Выход на экспериментальной плате	Подключение у ЖКИ-модуля
GND	RW
Цифровой вывод 12	RS "Выбор регистра"
Цифровой вывод 11	E "Разрешение"
Цифровой вывод 5	Данные D4
Цифровой вывод 4	Данные D5
Цифровой вывод 3	Данные D6
Цифровой вывод 2	Данные D7
+5 В	VDD "Напряжение питания +"
GND	VSS "Напряжение питания GND"
10-килоомный потенциометр соединен с +5 В и GND; движок подключен к ЖКИ	VEE "Контрастность"

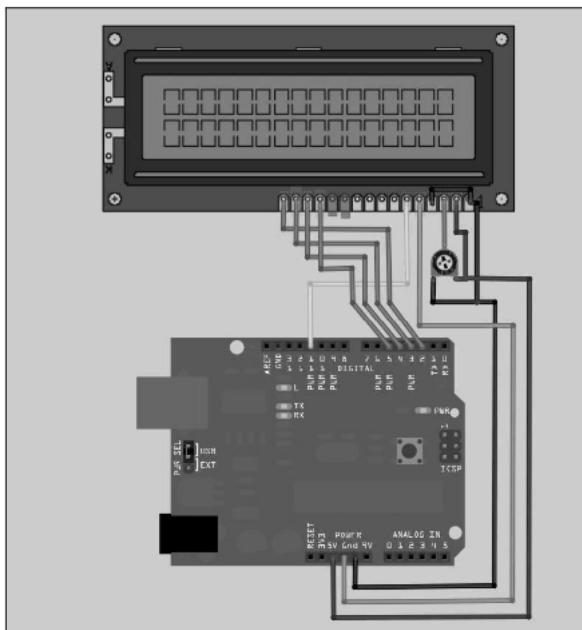


Рис. 17.7. Монтажная схема

Используемые комплектующие изделия:

- плата Arduino/Freeduino;
- семь гибких монтажных проводов длиной примерно 5 см;
- гибкий монтажный провод длиной примерно 3 см;
- восемь штыревых контактов 1×1;
- потенциометр типа PT10 10 кОм;
- ЖКИ-модуль 20×4 символа; контроллер HD44780 (или совместимый KS0066, например, компании Conrad Bestellnummer # 187275).



Рис. 17.8. Штыревой разъем компании Conrad Elektronik

Штырьки разъема (рис. 17.8) необходимо отделить при помощи плоскогубцев. Соединительные провода припаиваются соответственно к выводу ЖКИ и штырьку от разъема, а затем вставляются в плату Arduino. Так будет значительно надежнее по сравнению с вставкой гибких проволочных выводов в разъем платы Arduino.

17.11. Первый эксперимент с ЖКИ

После подключения ЖКИ можно присоединить штекер USB к плате микроконтроллера и передать тестовую программу (листинг 17.1). Если после передачи символов на ЖКИ не видно, то следует отрегулировать контрастность. Для этого вращайте подстроечный резистор до тех пор, пока символы не будут четко видны. Если изображение не появляется, то вероятно, существует ошибка соединения, ЖКИ не соответствует стандартным чипам HD44780/KS0066 или просто неисправен. Однако в большинстве случаев неисправность вызвана плохими контактами или ошибками монтажа.

Листинг 17.1. LCD.pde

```
// Franzis Arduino
// Управление 4-разрядным ЖКИ

/*
---[ Anschluss ]-----

>>> LCD <<<           >>> Arduino <<<

* RW pin          = GND
* LCD RS pin     = digital pin 12
* LCD Enable pin = digital pin 11
* LCD D4 pin     = digital pin 5
* LCD D5 pin     = digital pin 4
* LCD D6 pin     = digital pin 3
* LCD D7 pin     = digital pin 2
* Vee             = Poti Schleifer

*/
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int Poti = 0;
int raw=0;
int x=0;
int i=0;

void setup()
{
    Serial.begin(9600);
    lcd.begin(20, 4);           // ЖКИ с 4 строками и по 20 символов
    lcd.setCursor(0, 0);        // Стока 1 первый символ
    lcd.print("Arduino ist Spitze!");

    lcd.noDisplay();           // Отключение ЖКИ
    delay(1500);
    lcd.display();             // Повторное включение ЖКИ
}

void loop()
```

```
{  
  
lcd.setCursor(0, 1);  
lcd.print(millis()/1000); // Вывод прошедшего времени в секундах  
lcd.print(" Sekunden");  
  
if(x!=(millis()/500))  
{  
    raw=analogRead(Pot1);  
    lcd.setCursor(0,2);  
    lcd.print("ADC0 = ");  
    lcd.setCursor(7,2);  
    lcd.print("      ");  
    lcd.setCursor(7,2);  
    lcd.print(raw);  
    x=millis()/500;  
}  
  
if(Serial.available()>0)  
{  
    if(i==20)  
    {  
        i=0;  
        lcd.setCursor(0,3);  
        lcd.print("          ");  
    }  
  
    lcd.setCursor(i,3);  
    lcd.write(Serial.read());  
    i++;  
}  
}
```

Пример приведен для дисплея с четырьмя строками по 20 символов (20×4).

В первой строке появляется сообщение "Arduino ist Spitze!" (Arduino — это здорово!), во второй строке показываются секунды, прошедшие с начала программы. В третьей строке отображается значение аналогового входа (контакт 0). В четвертой и последней строке можно представлять символ, который посыпается в плату Arduino через последовательный интерфейс.

17.12. Как же все работает?

Команда `#include <LiquidCrystal.h>` подключает библиотеку, которая требуется для управления ЖКИ. В этой библиотеке хранятся программы для управления ЖКИ:

```
#include <LiquidCrystal.h>
```

Команда `LiquidCrystal lcd()` задает выводы, через которые осуществляется соединение ЖКИ с платой Arduino:

```
LiquidCrystal(rs, enable, d4, d5, d6, d7)
```

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // указание цифровых выводов
```

Далее команда `lcd.begin()` инициализирует устройство как ЖКИ с 4 строками по 20 символов:

```
lcd.begin(20, 4); // Инициализация с 4 строками  
// по 20 символов
```

Затем с помощью `lcd.print()` выводится приветственный текст в первой строке. Перед этим курсор позиционируется командой `lcd.setCursor()`:

```
lcd.setCursor(0, 0); // Стока 1 первый символ
```

```
lcd.print("Arduino ist Spitze!");
```

Теперь ЖКИ ненадолго выключается и снова включается. Это выполняют команды `noDisplay()` и `Display()`:

```
lcd.noDisplay(); // Отключение ЖКИ  
lcd.display(); // Повторное включение ЖКИ
```

В основном цикле `loop()` выводится время, прошедшее с начала программы. Также здесь используются уже известные команды `lcd.print()` для вывода данных:

```
lcd.setCursor(0, 1); // Стока 2,  
lcd.print(millis()/1000); // Прошедшее время выводится в секундах  
lcd.print("Sekunden");
```

В третьей строке выводится величина напряжения, измеренная на аналоговом входе 0. Этот процесс реализован довольно хитро (листинг 17.2). Так как основной цикл отрабатывает программу очень быстро и скорость обновления значения на аналоговом входе была бы слишком высокой, устанавливается искусственная задержка, причем не с помощью `delay()`, а через функцию `millis()`. Текущий результат делится на 500 и записывается в переменную `x`. Аналоговое значение с входа 0 обновляется только при изменении `x`. Получается интервал обновления 0,5 с. Большое преимущество описанного способа состоит в том, что задержка не влияет на работу основной программы.

Листинг 17.2. Вывод аналогового значения

```
if(x! = (millis()/500))  
{  
raw=analogRead(Pot1);  
lcd.setCursor(0,2);
```

```
lcd.print('ADC0 = ');
lcd.setCursor(7,2);
lcd.print("    ");
lcd.setCursor(7,2);
lcd.print(raw);
x=millis()/500;
}
```

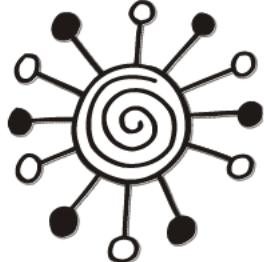
В четвертой и последней строке выводятся символы, которые принимает микроконтроллер через последовательный интерфейс (листинг 17.3). Для эксперимента нужно открыть терминальную программу и послать несколько символов со скоростью 9 600 бод. Символы появятся на ЖКИ.

Листинг 17.3. Отображение заданных символов на ЖКИ

```
if(Serial.available()>0) // Отображаются ли символы?
{
  if(i>20) // Дисплей заполнен?
  {
    // Если да, тогда сброс
    i=0;
    lcd.setCursor(0,3);
    lcd.print("    "); // Удаление строки
  }
  lcd.setCursor(i,3); // Всегда двигаться вперед на одну позицию
  lcd.write(Serial.read()); // Вывод символов
  i++;
}
```

Нужно иметь в виду, что никакая защита от переполнения не предусмотрена. Так, переполнение при выводе значения АЦП произойдет примерно через 16 с при такте 0,5 с.

ПРИЛОЖЕНИЯ



Приложение 1

Соответствие выводов Arduino и ATmega

На рис. П1.1 показано соответствие выводов контроллера ATmega168 и контактов платы Arduino.

Atmega168 Pin Mapping

Назначение вывода Arduino	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	Аналоговый вход 5	
Сброс	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	Аналоговый вход 4	
Цифровой вывод 0 (RX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	Аналоговый вход 3	
Цифровой вывод 1 (TX)	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	Аналоговый вход 2	
Цифровой вывод 2	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	Аналоговый вход 1	
Цифровой вывод 3 (ШИМ)	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	Аналоговый вход 0	
Цифровой вывод 4	VCC	7	22	GND	GND	
VCC	GND	8	21	AREF	Опорное напряжение	
GND	Kварцевый резонатор	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
Кварцевый резонатор	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	Цифровой вывод 13	
Цифровой вывод 5 (ШИМ)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	Цифровой вывод 12	
Цифровой вывод 6 (ШИМ)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	Цифровой вывод 11 (ШИМ)	
Цифровой вывод 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	Цифровой вывод 10 (ШИМ)	
Цифровой вывод 8	(PCINT0/CLK0/CP1) PB0	14	15	PB1 (OC1A/PCINT1)	Цифровой вывод 9 (ШИМ)	

Рис. П1.1. Соответствие выводов ATmega168 и платы Arduino

ПРИМЕЧАНИЕ

Цифровые выводы 11, 12 и 13 используются для сигналов ISP-интерфейса MISO, MOSI, SCK (для ATmega168 — контакты 17, 18 и 19). Избегайте подключения низкоимпедансной нагрузки к этим контактам.



Приложение 2

Escape-последовательности

Перечень Escape-последовательностей приведен в табл. П2.1. Команды всегда начинаются с <ESC>, что соответствует ASCII CHR(27). Для этого терминал должен быть совместимым с терминалом VT100!

Пример П2.1

```
Serial.write(27);  
Serial.println("[01;40H");
```

В результате выполнения примера курсор устанавливается точно в середине верхней строки.

Таблица П2.1. Escape-последовательности

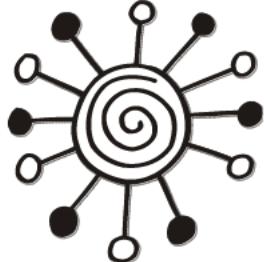
Назначение	Описание	Escape-последовательность
Вывод информации на терминале	Стрелка вверх ↑	<ESC>[A
	Стрелка вниз ↓	<ESC>[B
	Стрелка вправо →	<ESC>[C
	Стрелка влево ←	<ESC>[D
Редактирование	Начало строки (Home)	<ESC> [1~
	Пролистывание вверх (PageUp)	<ESC>[5~
	Пролистывание вниз (PageDown)	<ESC>[6~
Команды терминала	Позиционирование курсора на экране	<ESC> [nn;mmH nn — номер строки 1–24 mm — номер столбца 1–80 или 1–132

Таблица П2.1 (окончание)

Назначение	Описание	Escape-последовательность
Атрибуты выводимого символа	См. пример П2.2	<ESC>[param Возможные значения param: 0 — обычный режим; 1 — полужирный; 4 — подчеркивание; 5 — мигание; 7 — реверсирование (с темного на светлый или наоборот)
Команды удаления	Удаление строки справа от курсора	<ESC>[K
	Удаление строки слева от курсора	<ESC>[1K
	Удаление всего экрана ниже курсора	<ESC>[2J
	Удаление всего экрана выше курсора	<ESC>[J

Пример П2.2

```
// При указании нескольких параметров их разделяют точкой с запятой
<ESC>[param1;param2;param3m
// Устанавливается атрибут "revers"
Serial.write(27); Serial.println("[7m")
// Текст
Serial.println("VORNAME ?>");
// Снова атрибут "normal"
Serial.write(27); Serial.println("[0m");
```



Приложение 3

Таблица ASCII

CHAR	DEC	HEX	BIN	Описание
NUL	000	000	00000000	Пустой символ
SOH	001	001	00000001	Начало заголовка
STX	002	002	00000010	Начало текста
ETX	003	003	00000011	Конец текста
EOT	004	004	00000100	Конец передачи
ENQ	005	005	00000101	Запрос подтверждения
ACK	006	006	00000110	Подтверждаю
BEL	007	007	00000111	Звонок, звуковой сигнал
BS	008	008	00001000	Возврат на один символ
HAT	009	009	00001001	Горизонтальная табуляция
LF	010	00A	00001010	Перевод строки
VT	011	00B	00001011	Вертикальная табуляция
FF	012	00C	00001100	Новая страница
CR	013	00D	00001101	Возврат каретки
SO	014	00E	00001110	Переход на верхний регистр
S1	015	00F	00001111	Переход на нижний регистр
DLE	016	010	00010000	Следующие символы имеют специальный смысл
DC1	017	011	00010001	1-й символ управления устройством — включить устройство чтения перфоленты
DC2	018	012	00010010	2-й символ управления устройством — включить перфоратор
DC3	019	013	00010011	3-й символ управления устройством — выключить устройство чтения перфоленты

CHAR	DEC	HEX	BIN	Описание
DC4	020	014	00010100	4-й символ управления устройством — выключить перфоратор
NAK	021	015	00010101	Отсутствие подтверждения
SYN	022	016	00010110	Синхронизация IDLE
ETB	023	017	00010111	Конец текстового блока
CAN	024	018	00011000	Отмена
EM	025	019	00011001	Кончилась перфолента
SUB	026	01A	00011010	Подставить
ESC	027	01B	00011011	Начало Escape-последовательности
FS	028	01C	00011100	Разделитель файлов
GS	029	01D	00011101	Разделитель групп
RS	030	01E	00011110	Разделитель записей
US	031	01F	00011111	Разделитель устройств
SP	032	020	00100000	Пробел
!	033	021	00100001	Восклицательный знак
"	034	022	00100010	Двойные кавычки
#	035	023	00100011	Знак числа
\$	036	024	00100100	Знак доллара
%	037	025	00100101	Процент
&	038	026	00100110	Амперсанд
'	039	027	00100111	Одинарные кавычки
(040	028	00101000	Открывающая скобка
)	041	029	00101001	Закрывающая скобка
*	042	02A	00101010	Звездочка
+	043	02B	00101011	Плюс
,	044	02C	00101100	Запятая
-	045	02D	00101101	Минус или тире
.	046	02E	00101110	Точка
/	047	02F	00101111	Косая черта, прямой слэш
0	048	030	00110000	-
1	049	031	00110001	-
2	050	032	00110010	-
3	051	033	00110011	-

CHAR	DEC	HEX	BIN	Описание
4	052	034	00110100	-
5	053	035	00110101	-
6	054	036	00110110	-
7	055	037	00110111	-
8	056	038	00111000	-
9	057	039	00111001	-
:	058	03A	00111010	Двоеточие
;	059	03B	00111011	Точка с запятой
<	060	03C	00111100	Меньше чем
=	061	03D	00111101	Равно
>	062	03E	00111110	Больше чем
?	063	03F	00111111	Знак вопроса
@	064	040	01000000	Символ AT
A	065	041	01000001	-
B	066	042	01000010	-
C	067	043	01000011	-
D	068	044	01000100	-
E	069	045	01000101	-
F	070	046	01000110	-
G	071	047	01000111	-
H	072	048	01001000	-
L	073	049	01001001	-
J	074	04A	01001010	-
K	075	04B	01001011	-
L	076	04C	01001100	-
M	077	04D	01001101	-
N	078	04E	01001110	-
O	079	04F	01001111	-
P	080	050	01010000	-
Q	081	051	01010001	-
R	082	052	01010010	-
S	083	053	01010011	-
T	084	054	01010100	-

CHAR	DEC	HEX	BIN	Описание
U	085	055	01010101	-
V	086	056	01010110	-
W	087	057	01010111	-
X	088	058	01011000	-
Y	089	059	01011001	-
Z	090	05A	01011010	-
[091	05B	01011011	Открывающая квадратная скобка
\	092	05C	01011100	Обратная косая черта
]	093	05D	01011101	Закрывающая квадратная скобка
^	094	05E	01011110	Знак вставки
_	095	05F	01011111	Символ подчеркивания
`	096	060	01100000	-
a	097	061	01100001	-
b	098	062	01100010	-
c	099	063	01100011	-
d	100	064	01100100	-
e	101	065	01100101	-
f	102	066	01100110	-
g	103	067	01100111	-
h	104	068	01101000	-
i	105	069	01101001	-
j	106	06A	01101010	-
k	107	06B	01101011	-
l	108	06C	01101100	-
m	109	06D	01101101	-
n	110	06E	01101110	-
o	111	06F	01101111	-
p	112	070	01110000	-
q	113	071	01110001	-
r	114	072	01110010	-
s	115	073	01110011	-
t	116	074	01110100	-
u	117	075	01110101	-

CHAR	DEC	HEX	BIN	Описание
v	118	076	01110110	-
w	119	077	01110111	-
x	120	078	01111000	-
y	121	079	01111001	-
z	122	07A	01111010	-
{	123	07B	01111011	Открывающая фигурная скобка
	124	07C	01111100	Вертикальная линейка (чертка)
}	125	07D	01111101	Закрывающая фигурная скобка
~	126	07E	01111110	Тильда
DEL	127	07F	01111111	Удалить последний символ



Приложение 4

Перечень фирм-поставщиков компонентов

Elektronikladen Zentrale
Hohe Straße 9-13
04107 Leipzig

Elektronikladen Vertrieb
Bielefelder Straße 561
32758 Detmold
www.elmicro.com

Conrad Electronic SE
Klaus-Conrad-Straße 1
92240 Hirschau
www.conrad.de

Reichelt Elektronik GmbH & Co. KG
Elektronikring 1
26452 Sande
www.reichelt.de

Roboter-Teile
EDV-Beratung & Robotertechnik Jörg Pohl
Baluschekstraße 9
01159 Dresden
www.roboter-teile.de

Electronic Assembly GmbH
Zeppelinstraße 19
82205 Gilching bei München
http://www.Icd-module.de

Sommer-Robotics
Ulli Sommer Bahnhofstraße 8
92726 Waidhaus
www.sommer-robotics.de



Приложение 5

Описание компакт-диска

Архив компакт-диска к книге можно скачать по ссылке <ftp://85.249.45.166/9785977507271.zip>. Ссылка также доступна на странице книги на сайте www.bhv.ru.

Компакт-диск содержит программное обеспечение, инструменты для программирования, технические паспорта, коды примеров из книги, а также принципиальные схемы (табл. П5.1).

Таблица П5.1. Описание компакт-диска

Папка	Описание	Главы
\Adobe	Дистрибутив Adobe Reader 9.1	1–17
\Beispiele	Файлы *.pde — коды примеров к главам книги	8–10, 12–17
\Daten	Технические описания и принципиальные схемы	3–17
\Freeduino_Board	Техническое описание платы Freeduino	5–17
\Software	Различное программное обеспечение	7–17

Предметный указатель

A

abs(x) 80
Analog Digital Converter, ADC 103
analogRead() 103, 110, 162
analogWrite() 107
ANSI-C 11
Ardumoto 20
ArduPilot 22

B, C

binary digit 55
Central Processing Unit, CPU 6
CISC 8
СОМ-интерфейс 45
СОМ-порт 50
constrain(x, a, b) 81
cos(rad) 84
Скорость в бодах 87

D

delay() 110, 226
digitalRead() 98, 107, 123
digitalWrite() 98, 107
Display() 226

E

else-if 69
Escape-последовательности 232

F

Flash-память 7
for 72, 73
FT232RL 39, 44
FTD-драйвер 40

G

Global Positioning System, GPS 201
GPS-мышь 201
GPS-протокол 204
GPS-сигнал 202

H

Hardware-Abstraction-Layer 53
HD44780 216
H-DOP 202

I

if 68
if-else 67
Integrated Development Environment, IDE 47
Inter-Integrated Circuit, I²C 185
ISP 27

K, L

KS0066 216
lcd.print() 226
lcd.setCursor() 226
Light Dependent Resistor, LDR 36, 148
Liquid Crystal Display, LCD 213
LM75 189
loop() 226

M

map(x, fromLow, fromHigh, toLow, toHigh) 81
max(x, y) 79
MAX232 202
micros() 110, 113, 148
millis() 113, 226
min(x, y) 79
MProg 40, 44

N, P

noDisplay() 226
 PCF8574 193
 pinMode 52
 pinMode() 97
 pow(base, exponent) 82
 ProtoShield 20
 Pulse Width Modulation, PWM 105

R

random(min, max) 110
 randomSeed(seed) 110
 RC-цепь 132
 Real Time Clock, RTC 143
 RISC 8

S

SCL 186
 SDA 186
 Serial.available() 88
 Serial.begin() 87
 Serial.end() 88
 Serial.flush() 89
 Serial.print() 86, 89
 Serial.println() 86, 91
 Serial.read() 88
 Serial.write() 91
 sin(rad) 84
 sq(x) 83
 Sqrt(x) 83
 SRF02 197
 StampPlot 175
 State Machine 166
 switch case 70

T

tan(rad) 85
 TellyMate 21
 tone() 136
 Twisted-Nematic 214

U

Universal Asynchronous Receiver
 Transmitter, UART 86, 87, 96

V, W

V-DOP 202
 while 73

A

Авиамодель 22
 Автомат:
 конечный 166
 уличного освещения 151
 Адаптер USB 18
 Анализатор 213
 Анод 31
 Аппаратные средства 13
 Аналогово-цифровой преобразователь,
 АЦП 103, 162, 169, 171, 181

В, Д

Внешние устройства 8
 Дальномер 197
 Датчик:
 времени 143
 расстояния 26
 света 148
 сенсорный 164
 температуры 189
 ультразвуковой 197
 Декремент 65
 Динамический массив 64
 Диод 34
 Дребезг 120

Е, Ж

Емкость 33
 Жидкокристаллический индикатор,
 ЖКИ 217, 223
 инициализация 220
 контрастность 216
 назначение выводов 218
 управление 220

З

Задержка:
 включения 124
 выключения 126
 Защитный диод 100

И

Измеритель емкости 159
 Инкремент 65
 Интерфейс UART 39

K

Катод 31
Кнопка 35
Кодовый замок 155
Команды удаления 233
Конденсатор 33

Контроллер:
ATmega 13
Atmega168 14
ATmega168 231
ATmega8 105
KS0066 216

M

Монтажная панель 37
Монтажный провод 35
Мультиплексный режим 214

O

Обработка данных 6
Оперативное запоминающее устройство,
ОЗУ 7
Оперативная память 6, 7
Операторы 65
Осциллограф с памятью 173

P

Память программ 6, 7
Панель инструментов 48
Плата расширения Ethernet 23
Платы расширения 19
Подпрограмма 75
Полубайт 55
Поляризатор 213
Потенциометр 36
Преобразование типа 78
Преобразователь
FT232RL 39
MAX232 202
аналого-цифровой См. АЦП
пьезоакустический 35, 109, 136, 144
ультразвука 197
уровня 23
цифроанalogовый См. ЦАП
Прерывания 57
Программный интерфейс UART 96

Процедурное
программирование 56
Процесс обработки информации 11

P

Расширителей порта 193
Редукторный электродвигатель 209
Резистор:
внешний подтягивающий 101
внутренний подтягивающий 159
встроенный подтягивающий 98
добавочный 127
подстроечный 36
цветовая маркировка 32
Реле 132
температуры 181
Ряд Е24 32

C

Самописец напряжения 171
Светодиод 31
Светофор 166
Сервопривод 209
Синтаксическая ошибка 51
Скважность импульсов 105
Среда программирования 45

T

Терминал VB.NET 95
Термистор 217
Тетрада 55
Техника безопасности 29
Технология:
CISC 8
RISC 10
STN 214
Типы данных 61
Транзистор 34

У

Угол обзора 215

Ф

Фарада 33
Фильтр нижних частот, ФНЧ 132
Фоторезистор 36, 148

Функции 77

Функция:

not 117

sin() 117

Ц

Цифроаналоговый преобразователь,

ЦАП 132

Центральный процессор 6

Циклы 71

Ч

Часы реального времени 143

Ш

ШИМ 105, 117

ШИМ-сигнал 132, 136

Шина I²C 185, 189

Штыревой разъем 223

Э

Экспериментальная плата 26

Электропитание 27

Я

Язык программирования С 11, 59