Описание Freeduino / Arduino

Введение

Семейство микроконтроллеров Freeduino представлено несколькими вариантами плат, имеющих в общем идентичный функционал, и полностью совместимых с аналогичными вариантами Arduino.

Модули Freeduino основаны на микросхеме ATmega168 (либо ATmega328), и легко программируются через USB порт (либо COM порт в варианте MaxSerial) при помощи бесплатного программного обеспечения.

Модуль имеет 14 контактов цифрового ввода/вывода, шесть из которых могут быть использованы для вывода сигналов ШИМ, и шесть аналоговых входов АЦП с дискретностью в 1024 значений. Модуль также содержит: кварцевый резонатор на 16 МГц, разъем USB (либо COM в варианте MaxSerial), разъем питания типа Mini-Jack, разъем для подключения внешнего программатора ICSP и кнопку сброса.

Приобретенный в готовом виде модуль содержит все необходимое для разработки микроконтроллерных устройств. Нужно просто подсоединить модуль к компьютеру при помощи стандартного USB кабеля.

В качестве периферии могут быть подключены любые устройства - от светодиодов и простейших датчиков до механизмов радиоуправляемых моделей и роботов. Программы пишутся на простом и интуитивно понятном си-подобном языке (с возможностью подключения сторонних библиотек, например, для управления LCD-дисплеями или двигателями), компилируются и загружаются в устройство одной кнопкой, после чего вы тут же получаете работающее автономное устройство.

Память

Микроконтроллер ATmega168 имеет 16 К программной памяти (из которых 2 К используются для хранения загрузчика). Кроме того, микросхема имеет 1 Кбайт ОЗУ (SRAM), 512 байт долговременной памяти данных (EEPROM).

Возможна замена управляющего микроконтроллера на ATmega328, что увеличивает объем доступной памяти в 2 раза: память программ - 32 К (2 К заняты загрузчиком), ОЗУ - 2 Кбайт, EEPROM - 1 Кбайт.

Программное обеспечение

Для работы с модулем используется специальная программа (прилагается на диске, либо скачивается в Internet). При покупке модуля в готовом виде вы получаете плату с микроконтроллером, в программную память которого записана специальная программа – загрузчик. В процессе работы модуля программа загрузчик производит обмен информацией с управляющей программой запущенной на компьютере, что позволяет записывать в программную память микроконтроллера разработанные программы.

Автоматический (программный) сброс

Кроме обычного аппаратного сброса модуль Freeduino имеет возможность производить программный сброс, выполняемый по команде от компьютера. Для этого, один из сигналов (DTR), поступающих с соответствующего выхода микросхемы USB адаптера FT232RL, подключен к входу сброса микроконтроллера через конденсатор 0,1 мкф. В тот момент, когда уровень сигнала на этой линии меняется с высокого на низкий, на входе сброса микроконтроллера появляется короткий отрицательный импульс, длительности которого достаточно для перезагрузки микроконтроллера. Программное обеспечение использует это свойство для автоматического сброса микроконтроллера перед тем, как начать процесс загрузки программы из компьютера в модуль. Это позволяет начать процесс загрузки нажатием всего одной кнопки "Отправить" в окне управляющей программы.

Ввод и вывод данных

Каждый из 14 выводов модуля можно использовать как вход или как выход данных. Эти выводы работают с сигналами уровнем 0...5В. Каждый из выводов рассчитан на входной (или выходной) ток до 20 мА и имеет внутренний программно отключаемый подтягивающий резистор сопротивлением 20-50 кОм, который по умолчанию отключен. Выводы 3, 5, 6, 9, 10 и 11 могут работать в режиме выходов сигналов широтно-импульсного модулирования. Если вы хотите подключить внешние устройства к выводам 1 или 2, то вы должны помнить, что эти выводы совмещены с цепями, работающими в режиме обмена по USB/COM каналу. Поэтому в момент записи программы и при других операциях обмена данными с компьютером внешние цепи нужно будет отключать.

Модуль также имеет 6 аналоговых входов (АЦП), каждый из которых обеспечивает 10 разрядное аналого-цифровое преобразование (т.е. различает 1024 значений уровня сигнала). По умолчанию входы настроены на диапазон входного напряжения от 0 до 5В. Но вы можете снижать верхнюю границу этого диапазона при использовании внешнего источника опорного напряжения и соответствующим образом составленной программы.

Каналы связи

Модуль Freeduino имеет целый ряд возможностей для обмена информацией с компьютером, другими модулями Freeduino/Arduino и любыми микроконтроллерными устройствами. Входящий в модуль микроконтроллер ATmega168 имеет встроенный последовательный интерфейс UART работающий с сигналами TTL уровней (0...+5В), сигналы которого выведены на контакты 1 (RxD) и 2 (TxD) модуля.

USB версии модулей имеют в своем составе USB конвертор FT232RL фирмы FTDI, который обеспечивает работу через создаваемый автоматически виртуальный COM-порт при подключении модуля к компьютеру. Freeduino MaxSerial оснащена преобразователем MAX232 для преобразования TTL уровней к стандарту RS232.

Управляющая программа для компьютера, служащая средой разработки приложений для модулей Freeduino, имеет простые средства, позволяющие получать простые текстовые данные, посылаемые из модуля на компьютер через тот же COM-порт, через который происходит управление и программирование модуля.

Входящая в комплект программного обеспечения библиотека прикладных функций имеет в своем составе несколько функций, позволяющих организовать последовательный канал связи на любом цифровом выходе модуля.

Микроконтроллер ATmega168 также поддерживает I2C (TWI) шину и шину SPI. Библиотека Freeduino содержит функции для работы с I2C. Для реализации работы с SPI можно обратиться к фирменной документации на микроконтроллер.

Питание

Питание USB версий модуля Freeduino можно осуществлять как от шины USB, так и от внешнего источника питания. Питание Freeduino MaxSerial осуществляется от внешнего источника.

Внешнее питание может осуществляться либо от сетевого блока питания, либо от батареи. На центральный контакт разъема питания подается плюс, а на общий провод минус.

Внешнее напряжение питания может быть от 6 до 20 В. Однако, при входных напряжениях менее 7В напряжение на микроконтроллере может упасть ниже 5В что приведет к нестабильной работе модуля. При напряжениях более 12 В встроенный стабилизатор напряжения может перегреваться, что приведет к снижению надежности. При перегреве он даже может выйти из строя. Рекомендуемый диапазон напряжения питания 7...12 В.

Защита от перегрузок шины USB

USB версии Freeduino имеют защиту, которая предохраняет USB порт вашего компьютера от коротких замыканий и перегрузок по току. Если ток по любой из цепей канала USB превысит 500 мА, то срабатывает устройство защиты – встроенный предохранитель разрывает цепь.

Габаритные размеры

Размеры самой платы составляют 68,5 X 53 мм. Разъемы USB/COM и разъем питания выступают за эти размеры. Плата имеет три крепежных отверстия, при помощи которых вы можете закрепить модуль в любом необходимом месте.

Дополнительные Преимущества

Вы покупаете законченное устройство, не требующее дополнительного оборудования, такого, как дорогостоящие программаторы и отладочные стенды, устройство не требует платного программного обеспечения.

Программное обеспечение Freeduino работает на Windows, Linux, MacOS X, и других операционных системах, поскольку реализовано на платформе Java.

Программная оболочка является простой в использовании для новичков, но достаточно гибкой для продвинутых пользователей, чтобы быстро достичь нужного результата. Особенно это удобно в образовательной среде, где студенты могут с лёгкостью разобраться с платформой, а преподаватели – разработать учебный курс и задания.

Язык может быть расширен с помощью C++ библиотек, квалифицированные специалисты могут создать свой собственный инструментарий для Freeduino на основе компилятора AVR C. Как само устройство, так и все программное обеспечение является полностью открытым – доступны спецификации, электрические схемы, исходный код.

Проект широко распространен в сети Internet – есть сотни сайтов с готовыми примерами и библиотеками дополнительных функций.

Рекомендуем посетить следующие ресурсы:

на русском языке:

http://www.freeduino.ru

http://habrahabr.ru/blogs/arduino/

http://ru.wikipedia.org/wiki/Arduino

на английском языке:

http://www.arduino.cc/

http://www.freeduino.org/

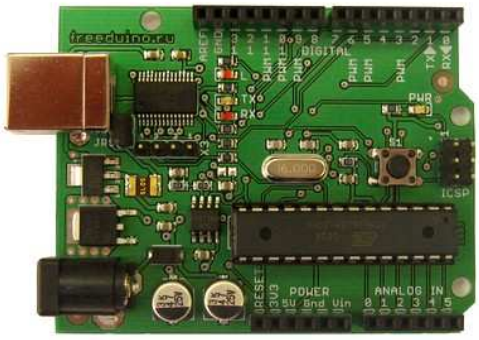
Freeduino 2009 – аналог Arduino Duemilanove

Особенности Freeduino 2009

Это самая новая версия Freeduino на сегодняшний день.

Модуль является полным аналогом Arduino Duemilanove, с той разницей, что неудобная перемычка автоматического сброса, требующая перерезания ножом дорожки на плате для отключения программного сброса и запаивания двух контактных площадок для его включения, заменена на классический «джампер».

Модуль автоматически выбирает питание – от USB, или от внешнего источника.





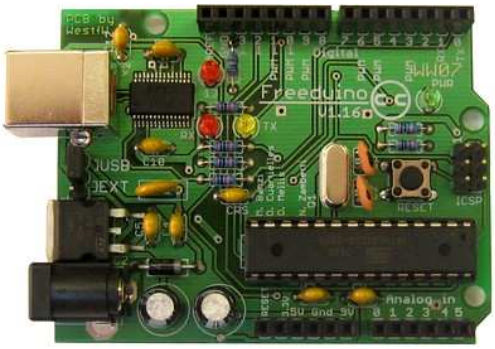
Технические характеристики

- ✓ Микроконтроллер: ATmega168 (ATmega328)
- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 16 К (32 К), из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 1 Кбайт (2 Кбайт)
- ✓ ПЗУ (EEPROM): 512 байт (1024 байт)
- ✓ Тактовая частота: 16 МГц
- ✓ Интерфейс с ПК: USB
- ✓ Питание от USB, либо от внешнего источника, выбор автоматически

Freeduino Through-Hole – аналог Arduino Diecimila

Особенности Freeduino Through-Hole

Модуль является аналогом Arduino Diecimila, но в нем применен штыревой монтаж большинства компонентов, что упрощает ручной монтаж и возможные самостоятельные изменения в модуле.





Технические характеристики

- ✓ Микроконтроллер: ATmega168 (ATmega328)
- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 16 К (32 К), из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 1 Кбайт (2 Кбайт)
- ✓ ПЗУ (EEPROM): 512 байт (1024 байт)
- ✓ Тактовая частота: 16 МГц
- ✓ Интерфейс с ПК: USB
- ✓ Питание от USB, либо от внешнего источника, выбор с помощью перемычки

Freeduino Through-Hole Kit – набор для сборки

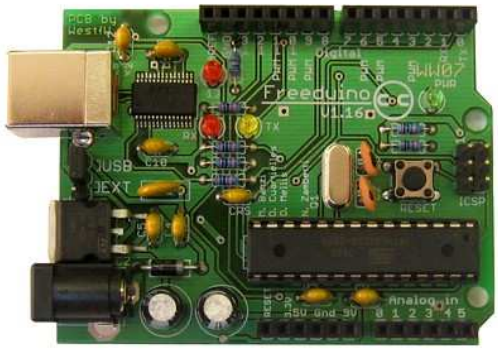
Особенности Freeduino Through-Hole Kit

Модуль поставляется в виде комплекта для самостоятельной сборки, включающего печатную плату, все необходимые элементы, Mini-CD с программным обеспечением и документацией на русском языке.

На печатную плату уже впаяна микросхема FT232RL и стабилизатор напряжения, поскольку самостоятельный монтаж этих компонентов в домашних условиях может вызвать сложности. К комплекту прилагается схема сборки, и монтаж оставшихся штыревых компонентов будет вполне под силу даже начинающим радиолюбителям.

Печатная плата и все элементы комплекта проходят электроконтроль перед отправкой.

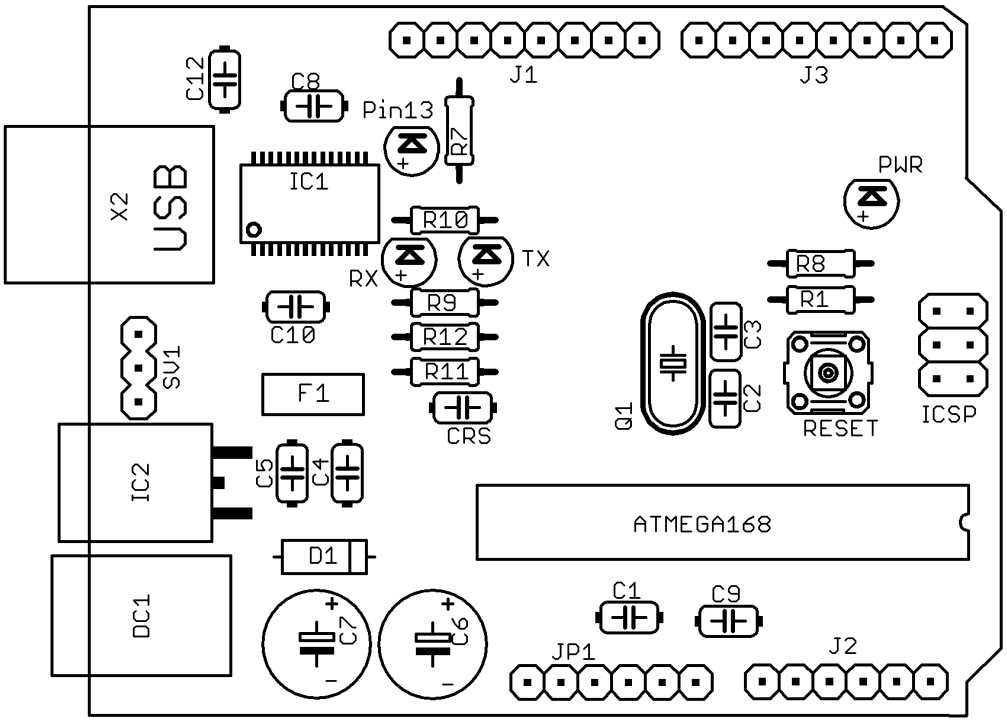




Технические характеристики

- ✓ Микроконтроллер: ATmega168 (ATmega328)
- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 16 К (32 К), из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 1 Кбайт (2 Кбайт)
- ✓ ПЗУ (EEPROM): 512 байт (1024 байт)
- ✓ Тактовая частота: 16 МГц
- ✓ Интерфейс с ПК: USB
- ✓ Питание от USB, либо от внешнего источника, выбор с помощью перемычки

Схема сборки



Перечень элементов:

Наименование	Кол-во	Обозначение на схеме сборки, комментарий
Резистор 10 кОм	1	R1
Резистор 1 кОм	6	R7, R8, R9, R10, R11, R12
Неполярный конденсатор 100 нФ	8	C1, C4, C5, C8, C9, C10, C12, CRS
Неполярный конденсатор 24 пФ	2	C2, C3
Полярный конденсатор 47 мкФ	2	C6, C7
Диод	1	D1
Предохранитель	1	F1
Светодиод красный	2	Pin13, RX
Светодиод желтый	1	TX
Светодиод зеленый	1	PWR
Кварцевый резонатор	1	Q1
Кнопка	1	RESET
Вилка штыревая 3x1	2	ICSP
Вилка штыревая 3x1	1	SV1
Гнездо 6x1	2	JP1, J2
Гнездо 8x1	2	J1, J3
Гнездо USB	1	X2
Гнездо питания	1	DC1
Микроконтроллер ATmega168(328)	1	ATMEGA168
Панель для микроконтроллера	1	ATMEGA168
Микросхема FT232RL	1	IC1 (уже смонтирована)
Стабилизатор 7805D2T	1	IC2 (уже смонтирован)
Перемычка (джампер)	1	Для выбора источника питания на SV1
Разъем питания	1	Для подключения внешнего питания

Замечания по сборке

Обращаем Ваше внимание на следующие замечания:

1. Соблюдайте правильную ориентацию полярных компонентов (светодиоды RX, TX, Pin13, PWR; диод D1; конденсаторы C6, C7) и микроконтроллера с панелью в соответствии со схемой сборки.

- ✓ У светодиодов анод имеет более длинный вывод, а на схеме анод указан знаком "+".
- ✓ У диода D1 катод отмечен белой полосой на корпусе, на схеме также указана полоса.
- ✓ Отрицательные выводы конденсаторов C6 и C7 отмечены знаками "-" на корпусе.
- ✓ Панель под микроконтроллер и сам микроконтроллер имеют вырез с одной из сторон.

2. Резистор R1 имеет номинал 10 кОм, а остальные резисторы - 1 кОм. Внешне резисторы различаются цветовой маркировкой - резистор R1 (10 кОм) отмечен полосками коричневая-черная-черная-красная-коричневая, а резисторы номиналом 1 кОм - коричневая-черная-черная-коричневая-коричневая.

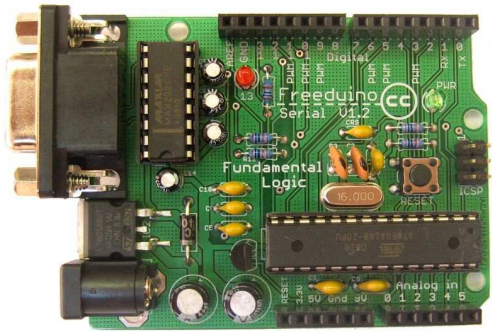
3. Рекомендуем при сборке сверяться с фотографиями собранного модуля, записанными на прилагаемом компакт-диске, а также представленными на сайте.

Freeduino MaxSerial – Arduino на COM-порт

Особенности Freeduino MaxSerial

Freeduino MaxSerial – вариант Freeduino, использующий RS-232 в качестве интерфейса с компьютером (он же COM-порт, он же последовательный порт) вместо USB. Такой вариант может быть интересен при подключении к компьютеру, не имеющего USB порта, либо для упрощения организации связи с другими устройствами, использующими RS-232 интерфейс.

На порту имеются линии RX и TX, поскольку только они реализованы в последовательном интерфейсе микроконтроллера ATmega, а также линия DTR, используемая для программного сброса микроконтроллера при его программировании.





Для организации интерфейса применен специальный преобразователь MAX232, что обеспечивает работу модуля в соответствии со стандартом RS-232.

Для соединения Freeduino MaxSerial с компьютером Вам потребуется модемный кабель 9M-9F и, поскольку COM-порт не имеет линий питания, источник питания 7-20В, в качестве которого можно использовать имеющеюся в ПК линию +12В.

Во всем остальном модуль полностью аналогичен своему USB аналогу Freeduino Through-Hole.

Технические характеристики

- ✓ Микроконтроллер: ATmega168 (ATmega328)
- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 16 К (32 К), из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 1 Кбайт (2 Кбайт)
- ✓ ПЗУ (EEPROM): 512 байт (1024 байт)
- ✓ Тактовая частота: 16 МГц
- ✓ Интерфейс с ПК: RS-232 (COM-порт)
- ✓ Питание от внешнего источника

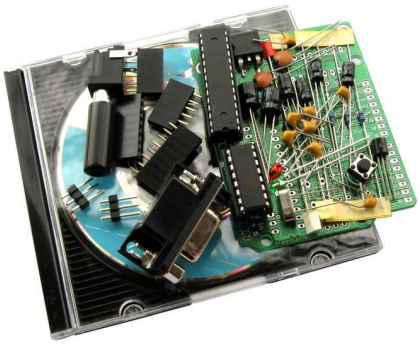
Freeduino MaxSerial Kit – набор для сборки

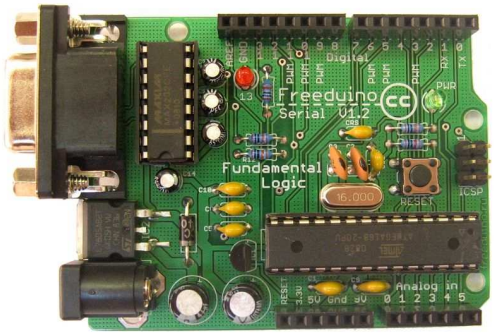
Особенности Freeduino MaxSerial Kit

Модуль поставляется в виде комплекта для самостоятельной сборки, включающего печатную плату, все необходимые элементы, Mini-CD с программным обеспечением и документацией на русском языке.

На печатную плату уже впаян стабилизатор напряжения, поскольку его самостоятельный монтаж в домашних условиях может вызвать сложности. К комплекту прилагается схема сборки, и монтаж оставшихся штыревых компонентов будет вполне под силу даже начинающим радиолюбителям.

Печатная плата и все элементы комплекта проходят электроконтроль перед отправкой.

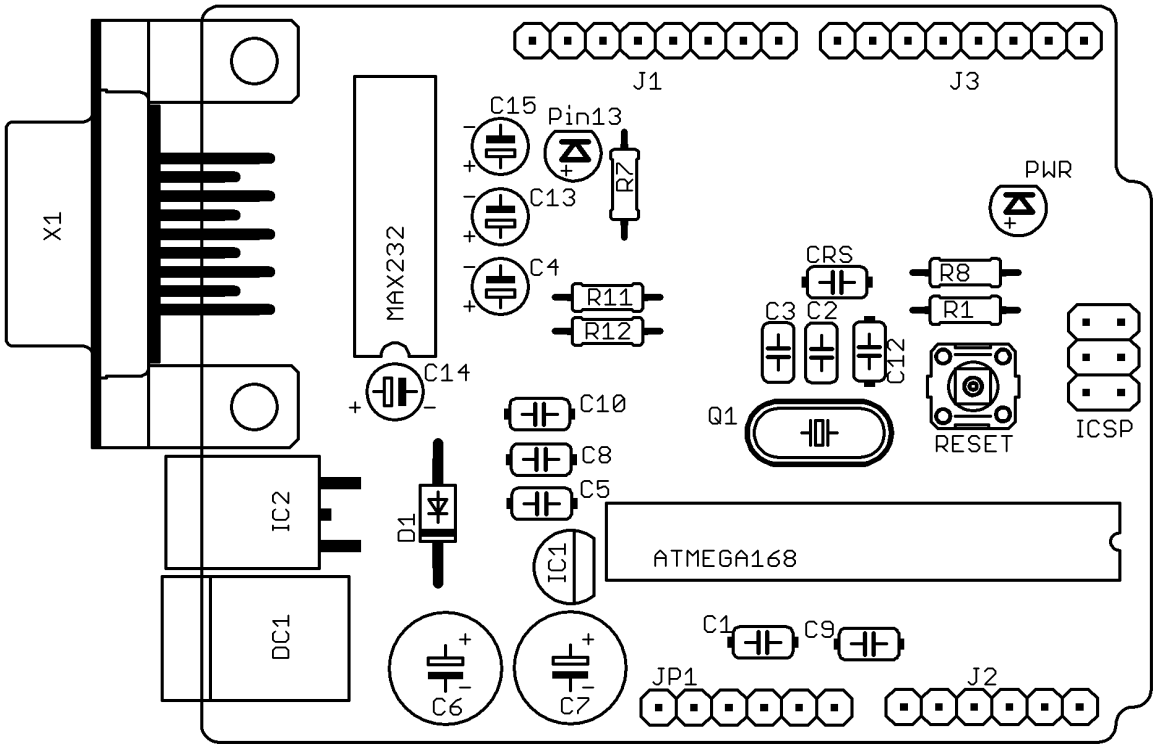




Технические характеристики

- ✓ Микроконтроллер: ATmega168 (ATmega328)
- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 16 К (32 К), из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 1 Кбайт (2 Кбайт)
- ✓ ПЗУ (EEPROM): 512 байт (1024 байт)
- ✓ Тактовая частота: 16 МГц
- ✓ Интерфейс с ПК: RS-232
- ✓ Питание от внешнего источника

Схема сборки



Перечень элементов:

Наименование	**Кол-во**	**Обозначение на схеме сборки, комментарий**
Резистор 10 кОм	1	R1
Резистор 1 кОм	4	R7, R8, R11, R12
Неполярный конденсатор 100 нФ	7	C1, C5, C8, C9, C10, C12, CRS
Неполярный конденсатор 24 пФ	2	C2, C3
Полярный конденсатор 47 мкФ	2	C6, C7
Полярный конденсатор 1 мкФ	4	C4, C13, C14, C15
Диод	1	D1
Светодиод красный	1	Pin13
Светодиод зеленый	1	PWR
Кварцевый резонатор	1	Q1
Кнопка	1	RESET
Вилка штыревая 3x1	2	ICSP
Гнездо 6x1	2	JP1, J2
Гнездо 8x1	2	J1, J3
Гнездо DB9F	1	X1
Гнездо питания	1	DC1
Стабилизатор 78L33	1	IC1
Стабилизатор 7805D2T	1	IC2 (уже смонтирован)
Преобразователь MAX232	1	MAX232
Панель для MAX232	1	MAX232
Микроконтроллер ATmega168(328)	1	ATMEGA168
Панель для микроконтроллера	1	ATMEGA168
Разъем питания	1	Для подключения внешнего питания

Замечания по сборке

Обращаем Ваше внимание на следующие замечания:

1. Соблюдайте правильную ориентацию полярных компонентов (светодиоды Pin13, PWR; диод D1; конденсаторы C6, C7, C4, C13, C14, C15), стабилизатора, микросхем и панелей в соответствии со схемой сборки.

✓ У светодиодов анод имеет более длинный вывод, а на схеме анод указан знаком "+".

- ✓ У диода D1 катод отмечен белой полосой на корпусе, на схеме также указана полоса.
- ✓ Отрицательные выводы полярных конденсаторов отмечены знаками "-" на корпусе.
- ✓ Микросхемы и панели под них имеют вырез с одной из сторон.
- ✓ Стабилизатор 78L33 имеет плоскую грань на корпусе.

2. Резистор R1 имеет номинал 10 кОм, а остальные резисторы - 1 кОм. Внешне резисторы различаются цветовой маркировкой - резистор R1 (10 кОм) отмечен полосками коричневая-черная-черная-красная-коричневая, а резисторы номиналом 1 кОм - коричневая-черная-черная-коричневая-коричневая.

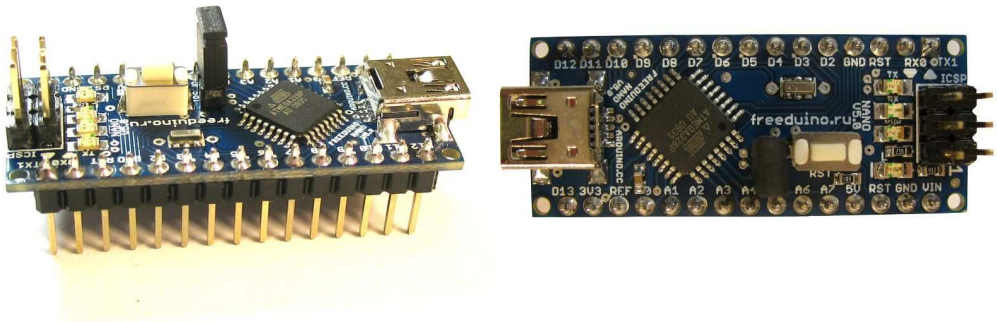
3. Рекомендуем при сборке сверяться с фотографиями собранного модуля, записанными на прилагаемом компакт-диске, а также представленными на сайте.

Freeduino Nano v5

Описание Freeduino Nano v5

Модификации Nano - самые миниатюрные в линейке Arduino-совместимых микроконтроллеров, габариты печатной платы составляют 18x43 мм. Несмотря на размеры, платы является полнофункциональными аналогами соответствующих версий Arduino, и даже превосходят их по количеству аналоговых входов - 8 против 6.

В нашей версии Freeduino Nano v5 предусмотрены даже такие полезные особенности полноразмерных версий, как джампер отключения программного сброса и самовосстанавливающийся предохранитель по питанию USB (у прочих Nano модификаций в случае короткого замыкания перегорает диод Шоттки).



В силу столь радикального уменьшения размеров изменены все внешние разъемы: для связи с ПК используется mini-USB, интерфейсные гнезда заменены на штыревые разъемы со стандартным шагом 2,54 мм, что позволяет установить микроконтроллер в макетную плату или DIP-панель.

Несмотря на то, что непосредственно состыковать Freeduino Nano и дополнительный модуль, вроде Ethernet Shield не получится, совместимость остается – Вам потребуется только правильно соединить выводы Freeduino Nano и дополнительного модуля в соответствии с принципиальными схемами.

Модуль также может поставляться без смонтированных штыревых разъемов - для таких вариантов применения, где удобнее просто подпаять необходимые проводники.

Технические характеристики

- ✓ Миниатюрный размер: 18x43 мм
- ✓ Микроконтроллер: ATmega328
- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 8 портов
- ✓ ППЗУ (Flash Memory): 32 К (из них 2 К используются загрузчиком)
- ✓ ОЗУ (SRAM): 2 Кбайт
- ✓ ПЗУ (EEPROM): 1024 байт
- ✓ Тактовая частота: 16 МГц
- ✓ Интерфейс с ПК: USB (разъем Mini-USB)
- ✓ Питание от USB, либо от внешнего источника, выбор автоматически

Отличия от оригинальной Arduino Nano v3:

- ✓ Устранены ошибки подключения FT232RL
- ✓ Установлен самовосстанавливающийся предохранитель по шине USB
- ✓ Увеличена емкость фильтрующих конденсаторов (C2 с 4,7 до 15 мкФ, C8 с 4,7 до 6.8 мкФ)
- ✓ Добавлен джампер отключения программного сброса
- ✓ Добавлены площадки Bit-Bang программирования
- ✓ Номинал R1 такой же, как и на полноразмерных версиях Arduino/Freeduino

Отличия от Carduino Nano v4 (совместимого клона Arduino Nano):

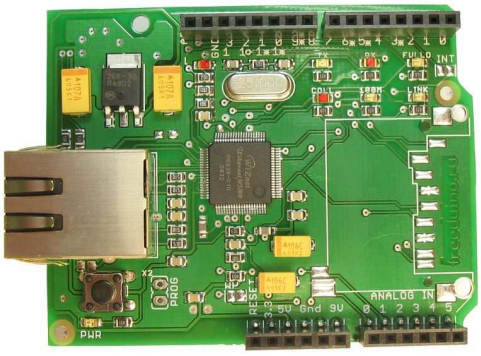
- ✓ Установлен самовосстанавливающийся предохранитель по шине USB
- ✓ Большая емкость фильтрующих конденсаторов - 15мкФ/6,8мкФ (у Carduino 10 мкФ/2,2 мкФ)
- ✓ Существует только версия с более функциональным микроконтроллером ATmega328
- ✓ Джампер отключения программного сброса не требует перерезания ножом/запаивания

- ✓ Стабилизатор питания имеет теплоотводящий контакт, припаянный к плате
- ✓ Порядок и шаг площадок Bit-Bang такой же, как и на полноразмерных версиях Arduino
- ✓ Номинал R1 такой же, как и на полноразмерных версиях Arduino

Ethernet модуль E-Shield v2

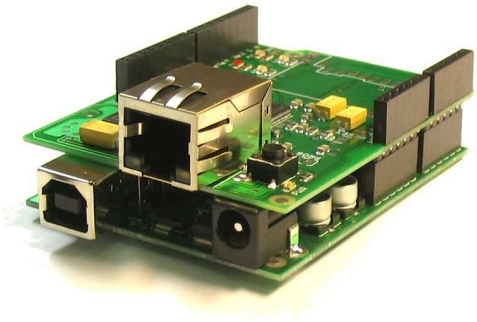
Модуль добавляет Ethernet интерфейс к Вашей Arduino-совместимой плате, и является полным аналогом Arduino Ethernet Shield.

Модуль поддерживает до четырех одновременных соединений по IP протоколам TCP и UDP. Для программирования модуля рекомендуется использовать стандартную Arduino библиотеку Ethernet, входящую в состав программного обеспечения Arduino с версии 0.12. В настоящее время стандартная библиотека позволяет реализовать TCP-сервер и TCP-клиент.





Модуль подключается к Freeduino с помощью установленных на нем штекеров.



Как и оригинальный Arduino Ethernet Shield, модуль не полностью совместим с Arduino Mega, и для работы с данной платой потребуются небольшие изменения конструкции.

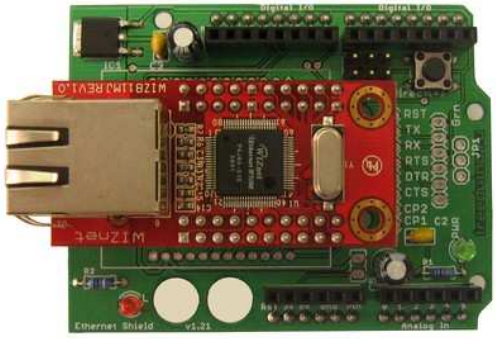
На модуле предусмотрено место для читателя SD-карт, однако такой функционал пока не поддерживается официальным ПО Arduino.

Поскольку установленный сетевой модуль физически затрудняет доступ к Freeduino, на лицевую сторону выведены все линии ввода/вывода, а также кнопка сброса.

Freeduino использует цифровые порты 10, 11, 12 и 13 для обмена информацией с модулем, и они не могут использоваться для других целей.

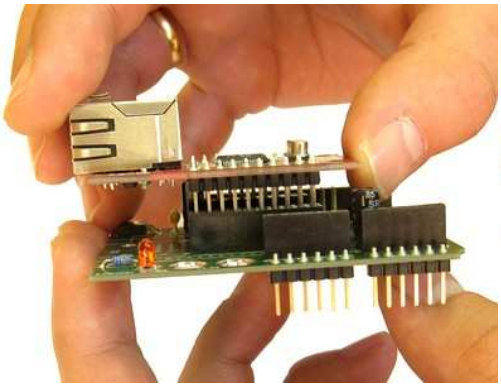
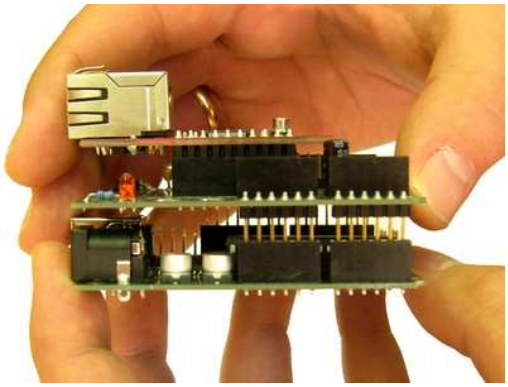
Ethernet модуль E-Shield v1.21

Модуль E-Shield v1.21 предназначен для подключения к Arduino-совместимым платам Ethernet модулей Wiz811mj, X-Port, X-Port Direct. В силу малой распространенности модулей X-Port на территории РФ все дальнейшее описание приведено для варианта работы с модулем Wiz811mj, при подключении которого он полностью совместим со стандартной Arduino библиотекой Ethernet (библиотека входит в состав программного обеспечения Arduino с версии 0.12).




Модуль поддерживает до четырех одновременных соединений по IP протоколам TCP и UDP. В настоящее время стандартная библиотека позволяет реализовать TCP-сервер и TCP-клиент.

Модуль подключается к Freeduino с помощью установленных на нем разъемов как показано на рисунках:

Поскольку установленный сетевой модуль физически затрудняет доступ к Freeduino, на лицевую сторону выведены все порты Freeduino, в том числе ICSP, а также кнопка сброса.

Freeduino использует цифровые порты 10, 11, 12 и 13 для обмена информацией с модулем, и они не могут использоваться для других целей.

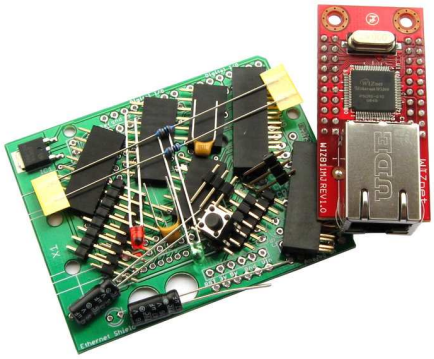
Отдельно следует отметить целесообразность применения модуля E-Shield v1.21 при наличии меньшего по габаритам варианта E-Shield v2, которая заключается в возможности использовать сетевой модуль Wiz811mj самостоятельно. Микросхема W5100, являющаяся основой как E-Shield v2, так и Wiz811mj поддерживает несколько типов интерфейсов с микроконтроллером. Стандартная библиотека Arduino, и соответственно модули E-Shield используют вариант подключения по SPI. При необходимости реализации других интерфейсов одним из простых вариантов может стать применение модуля Wiz811mj.

E-Shield v1.21 Kit – набор для сборки

Особенности Freeduino E-Shield v1.21 Kit

Модуль поставляется в виде комплекта для самостоятельной сборки, включающего печатную плату, все необходимые элементы, а также модуль Wiz811MJ.

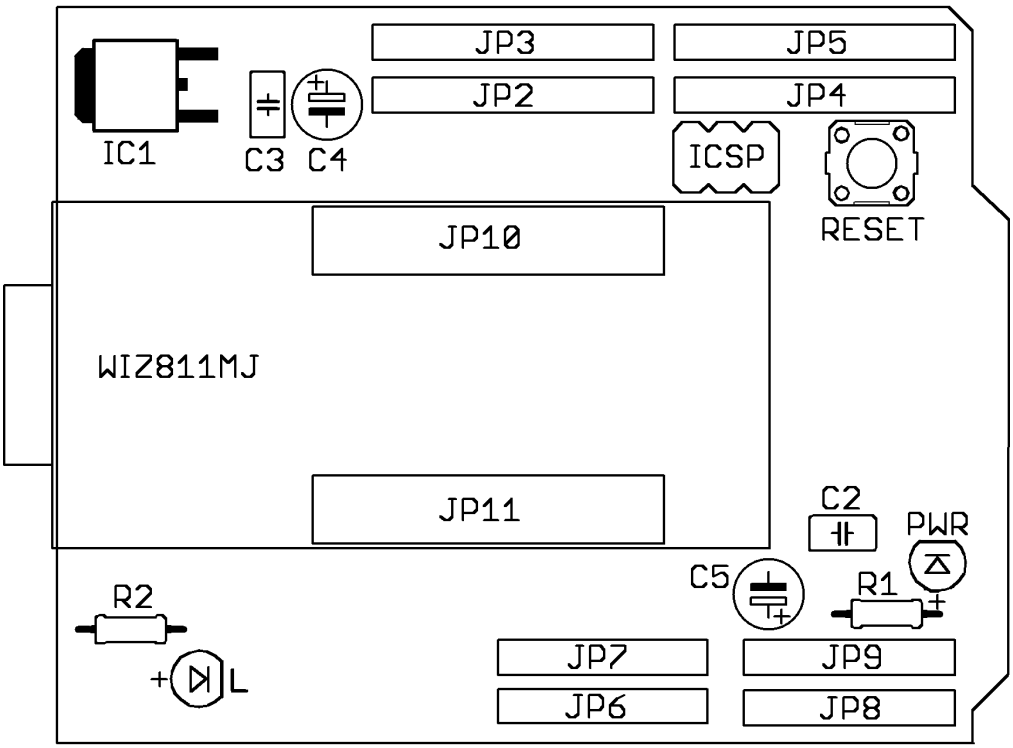
На печатную плату уже впаян стабилизатор напряжения, поскольку его самостоятельный монтаж в домашних условиях может вызвать сложности. К комплекту прилагается схема сборки, и монтаж оставшихся штыревых компонентов будет вполне под силу даже начинающим радиолюбителям.





Печатная плата и все элементы комплекта проходят электроконтроль перед отправкой.

Схема сборки



Перечень элементов:

Наименование	Кол-во	Обозначение на схеме сборки, комментарий
Резистор 1 кОм	2	R1, R2
Неполярный конденсатор 100 нФ	2	C2, C3
Полярный конденсатор 47 мкФ	2	C4, C5
Светодиод красный	1	L
Светодиод зеленый	1	PWR
Кнопка	1	RESET
Вилка штыревая 3x1	2	ICSP
Вилка штыревая 6x1	2	JP6, JP8
Вилка штыревая 8x1	2	JP3, JP5

Гнездо 6x1	2	JP7, JP9
Гнездо 8x1	2	JP2, JP4
Гнездо 10x2	2	JP10, JP11
Стабилизатор MC33269D-3.3	1	IC1 (уже смонтирован)
Модуль Wiz811mj	1	WIZ811MJ

Замечания по сборке

Обращаем Ваше внимание на следующие замечания:

1. Соблюдайте правильную ориентацию полярных компонентов (светодиоды L, PWR; конденсаторы C4, C5), модуля Wiz811mj в соответствии со схемой сборки.

- ✓ У светодиодов анод имеет более длинный вывод, а на схеме анод указан знаком "+".
- ✓ Отрицательные выводы полярных конденсаторов отмечены знаками "-" на корпусе.

2. Для более качественного и ровного монтажа гнезд JP10, JP11 можно предварительно разместить их на ответных штырях модуля Wiz811mj. Аналогично, при монтаже вилок JP3, JP5, JP6, JP8 можно разместить их в гнездах платы Freeduino.

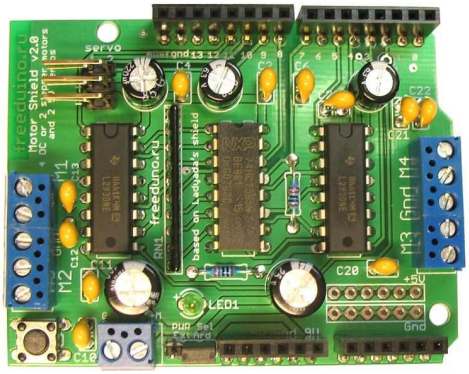
3. Рекомендуем при сборке сверяться с фотографиями собранного модуля, записанными на прилагаемом компакт-диске, а также представленными на сайте.

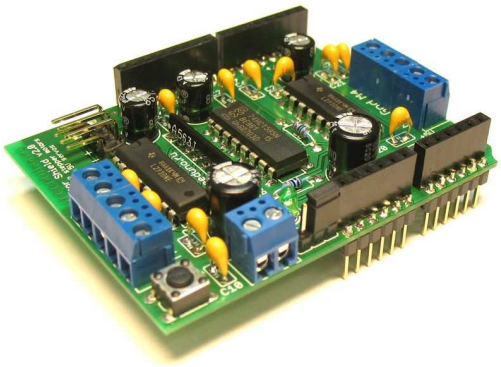
Motor Shield v2

Описание Motor Shield v2

Motor Shield v2 (далее M–Shield) – силовой модуль управления двигателями для микроконтроллеров серии Freeduino/Arduino. Модуль предназначен для упрощения разработки моторизированных и робототехнических устройств и основан на первой версии M–Shield от Adafruit Industries (www.ladyada.net).

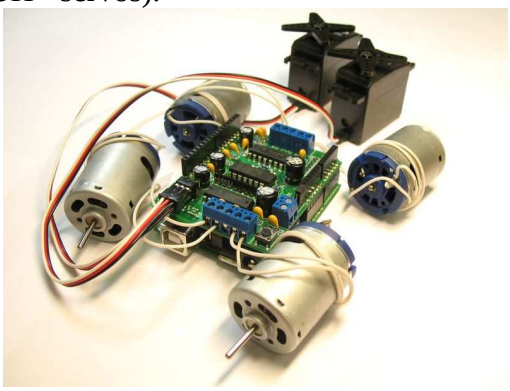
Модуль подключается к Freeduino с помощью установленных на нем разъемов.





Технические возможности M–Shield позволяют реализовать разнообразные проекты, связанные с подключением и управлением слаботочными двигателями следующих типов:

- ✓ четырёх двигателей постоянного тока (ДПТ - DC motors);
- ✓ двух шаговых двигателей униполярных или биполярных с единичной или двойной обмоткой (ШД - stepper motors);
- ✓ двух 5В сервоприводов (СП - servos).



Возможны следующие комбинации подключаемых к M–shield двигателей:

- ✓ 2 СП + 4 ДПТ;
- ✓ 2 СП + 2 ШД;
- ✓ 2 СП + комбинации ШД и ДПТ, например: возможен вариант замены 1 ШД на 2 ДПТ или наоборот.

Технические характеристики

- ✓ напряжение питания: +7...+ 24 В;
- ✓ количество силовых каналов: 4;
- ✓ максимально-продолжительный ток каждого канала: 0,6 А;
- ✓ напряжение питания сервоприводов: 5 В;
- ✓ возможность реверса каждого двигателя;
- ✓ возможность независимого управления каждым каналом;
- ✓ модуль полностью совместим со всеми известными моделями Freeduino/Arduino: Max Serial, Through-Hole, Diecimila, 2009, Duemilanove, а также Arduino Mega.

Драйвер двигателей L293D

Управление двигателями осуществляется двумя микросхемами L293D. Каждая микросхема – это четырёхканальный драйвер со встроенными обратными диодами для защиты микросхемы от перенапряжений при работе на индуктивную нагрузку.

Драйверы управляются парами, что позволяет реализовать на одной микросхеме два двуполярных канала управления с ШИМ управлением.

Основные характеристики микросхемы:

- ✓ напряжение питания: +4,5... +36 В;
- ✓ максимальный продолжительный ток в каждом канале: 0,6 А;
- ✓ максимальный пиковый (<100 мс) неповторяющийся ток в каждом канале: 1,2 А;
- ✓ защита от перегрева.

Более полное описание можно найти в оригинальной документации производителя:
http://focus.ti.com/lit/ds/symlink/l293d.pdf
http://www.freeduino.ru/arduino/files/l293d.pdf

Питание.

В принципиальной электрической схеме M-Shield существуют две отдельные цепи питающего напряжения: слаботочная и силовая.

Питание слаботочной сигнальной цепи, а также подключаемых сервомоторов осуществляется от стабилизированного +5 В источника Freeduino. Недопустимо использовать эту цепь для питания двигателей постоянного тока, т.к. это приведет или к срабатыванию USB предохранителя или перегреву и выходу из строя стабилизатора напряжения +5 В на плате Freeduino.

Выбор источника питания силовой части модуля осуществляется при помощи джампера питания на M-shield: либо от источника внешнего питания +7...+12 В управляющей платы Freeduino, либо от дополнительного источника постоянного напряжения +6...+ 24 В, подключаемого к двум клеммам разъема питания на M-Shield.

В настоящей версии M–Shield v2 верхнее значение напряжения питания ограничено используемыми электролитическими конденсаторами на максимальное напряжение 25 В и может быть увеличено до максимального для микросхемы L293D значения 36 В заменой конденсаторов C7 и C8.

Управление двигателями постоянного тока

Определение параметров двигателя

Убедитесь, что параметры двигателя – номинальное напряжение и ток соответствуют используемому источнику питания и параметрам M-Shield.

Если потребляемый двигателем ток превышает номинальное для драйвера L293D значение в 0,6 А, можно увеличить значение продолжительного тока до 1,2 А путем параллельного подключения двигателя одновременно к двум портам, например к M1 и M2 и составлением соответствующего алгоритма управления.

Кроме того, с целью увеличения допустимой токовой нагрузки в два раза, можно напаять на микросхему L293D «нога к ноге» ещё одну L293D, это позволит управлять более мощными двигателями с максимально-продолжительным током в каждом канале управления до 1,2 А.

Вы также можете использовать две платы M-Shield, установив одну в разъемы другой, и подключив каждый двигатель к одноименным каналам обеих плат.

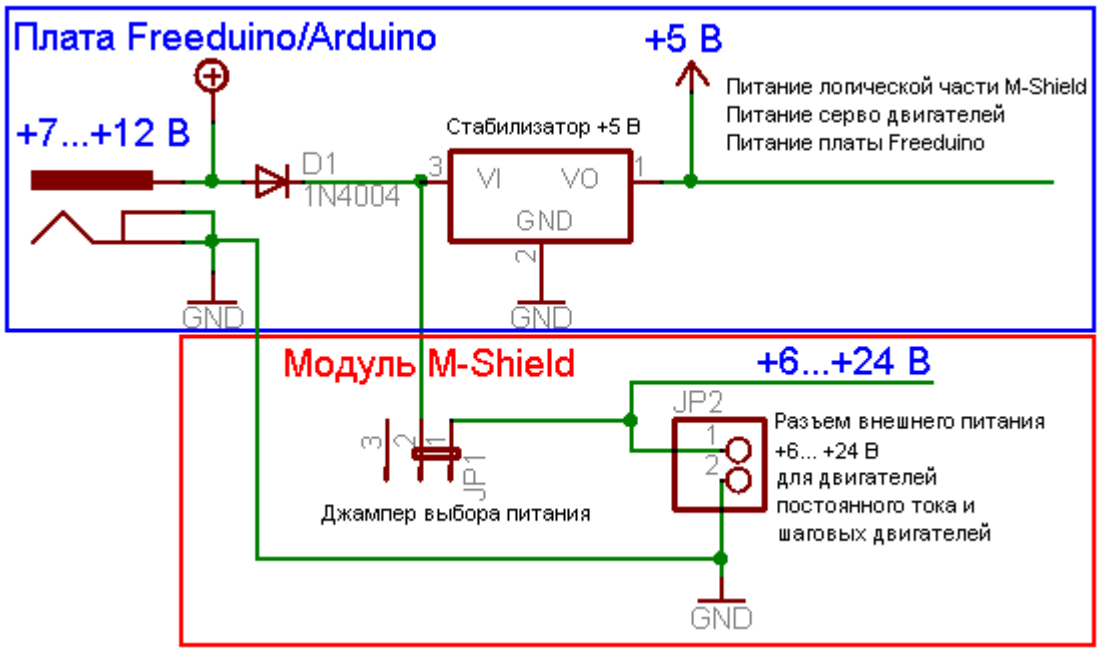
Установка джампера выбора источника питания

Крайнее левое положение джампера выбора источника питания соответствует подключению внешнего источника питания +6... +24 В силовой части модуля, при этом цепь питания Freeduino/Arduino отключена.

Крайнее правое положение джампера соответствует питанию +7...+12 В от основной платы Freeduino/Arduino.

Перед подключением внешнего источника питания к клеммам M–shield обязательно убедитесь в правильности установки джампера питания, поскольку неверная его установка может привести к замыканию двух источников.

На рисунке приведена схема, поясняющая электрические соединения шин питания основной платы Freeduino/Arduino совместно и модуля M-Shield. 3-й штекер джампера питания не связан ни с одной из электрических цепей и предназначен для предотвращения утери перемычки при её переключении из одного положения в другое.



Подключение двигателей постоянного тока

Расположенные на плате M-Shield 5-контактные клеммники предназначены для подключения двигателей. Центральный контакт каждого клеммника – «земля», двигатели постоянного тока подключаются к крайним парам контактов, обозначенных по номерам каналов M1, M2, M3, M4.

Управление двигателями с помощью библиотеки AFMotor

Существует удобная библиотека, упрощающая работу с модулем M-Shield, скачать которую можно с сайта разработчика, или у нас:

http://www.ladyada.net/make/mshield/download.html

http://www.freeduino.ru/arduino/files/AFMotor-08_12_2009.zip

Как и в большинстве случаев, установка библиотеки сводится к распаковке архива в подпапку \hardware\libraries\ папки с ПО Arduino.

Для управления двигателями постоянного тока используется класс AF_DCMotor. Рассмотрим его основные методы.

AF_DCMotor – конструктор объекта

Вызов:

AF_DCMotor имя_объекта(номер_канала);

Создает экземпляр класса AF_DCMotor, принимает номер канала После создания объекта можно вызывать его методы.

Метод AF_DCMotor::setSpeed

Вызов:

имя_объекта.setSpeed(скважность)

Задает скважность ШИМ на канале в диапазоне от 0 до 255. Значение 0 соответствует напряжению 0 В на двигателе, значение 255 – полному напряжению питания.

Метод AF_DCMotor:: run

Вызов:

имя_объекта.run(направление)

Задает направление движения двигателя (полярность прикладываемого напряжения). Параметр «направление» может принимать одно из следующих значений:

FORWARD – прямое направление вращения

BACKWARD – обратное направление вращения

RELEASE – остановка двигателя

Пример управления двигателями постоянного тока

Ниже приведен пример простой программы, осуществляющей ступенчатое изменение скорости вращения двигателя с интервалами в 3 секунды в следующей последовательности: +50, +75%, +50%, STOP, -50%, -75%, -50%, STOP. После завершения цикл будет повторяться.

```
#include <AFMotor.h>  //Подключаем заголовочный файл библиотеки
```

```
//Создаем объект для двигателя на 1 канале (M1)
AF_DCMotor motor(1);

void setup() {
}

void loop() {
  //Задаем направление движение вперед
  motor.run(FORWARD);
  //Устанавливаем скорость 50%
  motor.setSpeed(128);
  //пауза 3 секунды - двигатель крутится
  delay(3000);
  //увеличиваем скорость до 75%, и пауза 3 сек.
  motor.setSpeed(192);
  delay(3000);
  //Снижаем скорость до 50%, и пауза 3 сек.
  motor.setSpeed(128);
  delay(3000);
  //Останавливаем двигатель, и пауза 3 сек.
  motor.run(RELEASE);
  delay(3000);
  //Переключаем направление вращения
  //Двигатель начнет вращаться со скоростью 50%
  motor.run(BACKWARD);
  //пауза 3 секунды - двигатель крутится
  delay(3000);
  //увеличиваем скорость до 75%, и пауза 3 сек.
  motor.setSpeed(192);
  delay(3000);
  //Снижаем скорость до 50%, и пауза 3 сек.
  motor.setSpeed(128);
  delay(3000);
  //Останавливаем двигатель, и пауза 3 сек.
  motor.run(RELEASE);
  delay(3000);
}
```

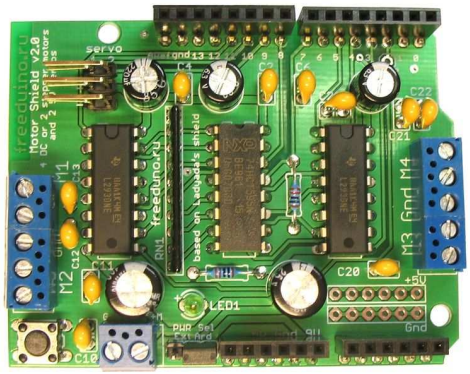
...to be continued. Планируется расширение документации примерами работы с сервоприводами и шаговыми двигателями

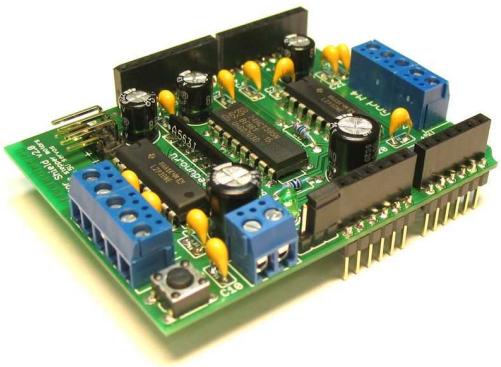
Motor Shield v2 Kit – набор для сборки

Особенности Freeduino Motor Shield v2 Kit

Модуль поставляется в виде комплекта для самостоятельной сборки, включающего печатную плату и все необходимые элементы.

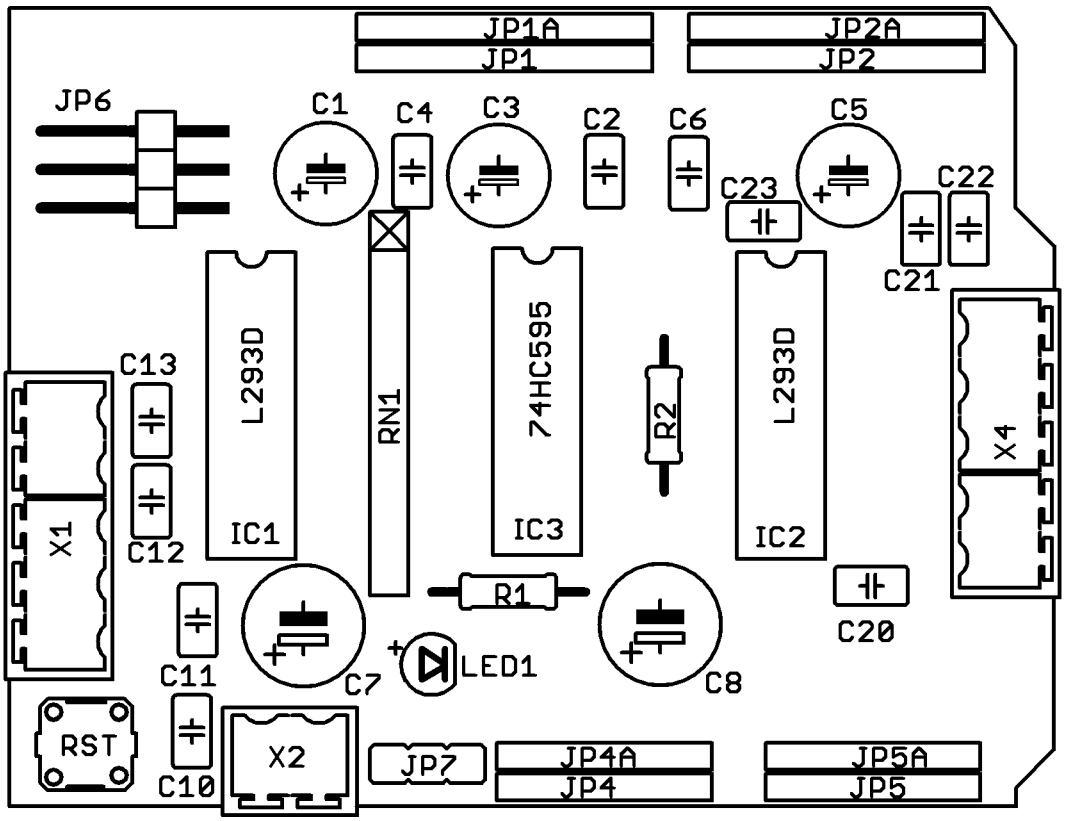
К комплекту прилагается схема сборки, и монтаж оставшихся штыревых компонентов будет вполне под силу даже начинающим радиолюбителям.





Печатная плата и все элементы комплекта проходят электроконтроль перед отправкой.

Схема сборки



Перечень элементов:

Наименование	Кол-во	Обозначение на схеме сборки, комментарий
Резистор 1 кОм	1	R1
Резистор 10 кОм	1	R2
Резисторная сборка 10х56 кОм	1	RN1
Неполярный конденсатор 100 нФ	11	C2, C4, C6, C10, C11, C12, C13, C20, C21, C22, C23
Полярный конденсатор 100 мкФ	2	C7, C8
Полярный конденсатор 220 мкФ	3	C1, C3, C5
Светодиод зеленый	1	LED1
Кнопка	1	RST
Вилка штыревая 3х1	1	JP7

Вилка штыревая 6x1	2	JP4, JP5
Вилка штыревая 8x1	2	JP1, JP2
Вилка штыревая угловая 3x2	1	JP6
Гнездо 6x1	2	JP4A, JP5A
Гнездо 8x1	2	JP1A, JP2A
Клеммник винтовой x2	3	X1, X2, X4
Клеммник винтовой x3	2	X1, X4
Перемычка (джампер)	1	Для выбора источника питания на JP7
Микросхема 74HCT595N	1	IC3
Микросхема L293DNE	2	IC1, IC2

Замечания по сборке

Обращаем Ваше внимание на следующие замечания:

1. Соблюдайте правильную ориентацию полярных компонентов (светодиод LED1, конденсаторы C1, C3, C5, C7, C8), резисторной сборки RN1, микросхем в соответствии со схемой сборки.

- ✓ У светодиода анод имеет более длинный вывод, а на схеме анод указан знаком "+".
- ✓ Отрицательные выводы полярных конденсаторов отмечены знаками "-" на корпусе.
- ✓ Общий вывод резисторной сборки отмечен точкой на корпусе, и крестиком на схеме.
- ✓ Микросхемы имеют вырез с одной из сторон.

2. Резистор R1 имеет номинал 1 кОм, а R2 - 10 кОм. Внешне резисторы различаются цветовой маркировкой - резистор R1 (1 кОм) отмечен полосками коричневая-черная-черная-коричневая-коричневая, а резистор R2 номиналом 10 кОм - коричневая-черная-черная-красная-коричневая.

3. Клеммники на 5 контактов X1 и X4 собираются из клеммников на 2 и 3 контакта состыковкой с помощью предусмотренных на корпусах пазов.

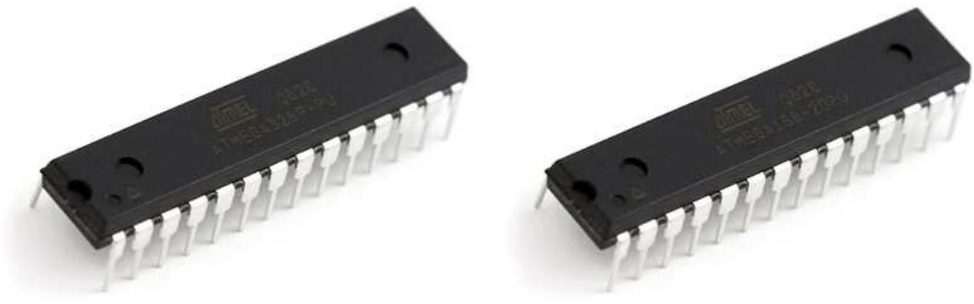
4. Для более качественного и ровного монтажа вилок JP1, JP2, JP4, JP5 можно предварительно разместить их в гнездах платы Freeduino.

5. Рекомендуем при сборке сверяться с фотографиями собранного модуля, записанными на прилагаемом компакт-диске, а также представленными на сайте.

Микроконтроллеры ATmega

Основой любой Arduino-совместимой платы является микроконтроллер ATmega семейства Atmel AVR, и технические характеристики платы определяются именно характеристиками микроконтроллера.

Приобретение дополнительного микроконтроллера может быть обусловлено, например, желанием заменить ATmega168 на уже имеющейся Arduino-совместимой плате более функциональным ATmega328.



Микроконтроллеры поставляются с прошитым Arduino загрузчиком (boot-loader), и готовы к работе сразу после установки в Вашу Arduino-совместимую плату, или в составе самостоятельно разработанного устройства.

ATmega328

Более функциональная версия микроконтроллера для Arduino-совместимых плат – имеет в 2 раза больший объем памяти программ, ОЗУ и EEPROM по сравнению с ATmega168.

- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 32 К, из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 2 Кбайт
- ✓ ПЗУ (EEPROM): 1024 байт
- ✓ Тактовая частота: до 20 МГц (на Arduino-совместимых платах обычно 16 МГц)

ATmega168

Это наиболее распространенный микроконтроллер для Arduino-совместимых плат.

- ✓ Цифровые порты ввода/вывода: 14 портов (из них 6 с ШИМ-сигналом)
- ✓ Аналоговые порты ввода: 6 портов
- ✓ ППЗУ (Flash Memory): 16 К, из них 2 К используются загрузчиком
- ✓ ОЗУ (SRAM): 1 Кбайт
- ✓ ПЗУ (EEPROM): 512 байт
- ✓ Тактовая частота: до 20 МГц (на Arduino-совместимых платах обычно 16 МГц)

Инструкция по быстрому запуску Arduino / Freeduino

Здесь дана краткая инструкция для начинающих по подключению Freeduino / Arduino, основам работы с микроконтроллером и прилагающимся программным обеспечением.

Предполагаемая последовательность действий выглядит так:

1. Приобретение Freeduino и соответствующего кабеля (для Freeduino 2009 и Through-Hole – USB кабеля типа: A–B, для Freeduino Nano – кабеля Mini-USB, для Freeduino MaxSerial – модемного кабеля типа 9M-9F);
2. Установка программного обеспечения;
3. Подключение кабеля к Freeduino и компьютеру, и установка USB драйверов (для USB версий Freeduino). Пользователям ОС семейства GNU/Linux установка драйверов не требуется, поскольку поддержка данного оборудования обычно включена в ядро;
4. Запуск программного обеспечения на компьютере и его настройка;
5. Написание простейшей программы и загрузка ее в Freeduino.

Описание приводится для операционной системы Windows XP. Для других операционных систем Microsoft (Windows 2000/Vista) возможны незначительные различия. Для прочих операционных систем (Linux/FreeBSD/Mac OS X и т.д.) последовательность действий значительно отличается, и зависит от конкретного дистрибутива. При возникновении проблем рекомендуем искать решения в сети Internet как на основном сайте проекта http://www.arduino.cc , так и на других ресурсах.

1. Приобретение Freeduino и кабеля для связи с ПК

Для передачи данных с компьютера на Freeduino, необходимо приобрести соответствующий кабель.

Для вариантов Freeduino 2009 и Through-Hole – это USB кабель типа: A – B, такой кабель часто используется, например, для подключения принтера или сканера.

Для Freeduino Nano нужен кабель Mini-USB.

Для подключения Freeduino MaxSerial необходим кабель для последовательного порта 9M-9F.



В комплект поставки кабель не входит.

Для Freeduino MaxSerial также потребуется внешний источник питания напряжением от +7 до + 12 В. Можно использовать линию +12 В от компьютера. Центральный контакт разъема питания – "плюс", внешний контакт – "земля".

2. Установка программного обеспечения

На прилагаемом к модулю диске имеется архив arduino-0017.zip – это ZIP архив всего необходимого программного обеспечения. Рекомендуем разархивировать содержимое на диск C:, т.е. в итоге на диске C: должна появиться папка arduino-0017, а в ней уже папки drivers, examples, и прочие.

Программное обеспечение Arduino можно также скачать с сайта разработчика:

http://arduino.cc/en/Main/Software

http://arduino.googlecode.com/files/arduino-0017.zip

http://files.arduino.cc/downloads/arduino-0017.zip

3. Подключение USB кабеля к Freeduino и компьютеру, и установка USB драйверов

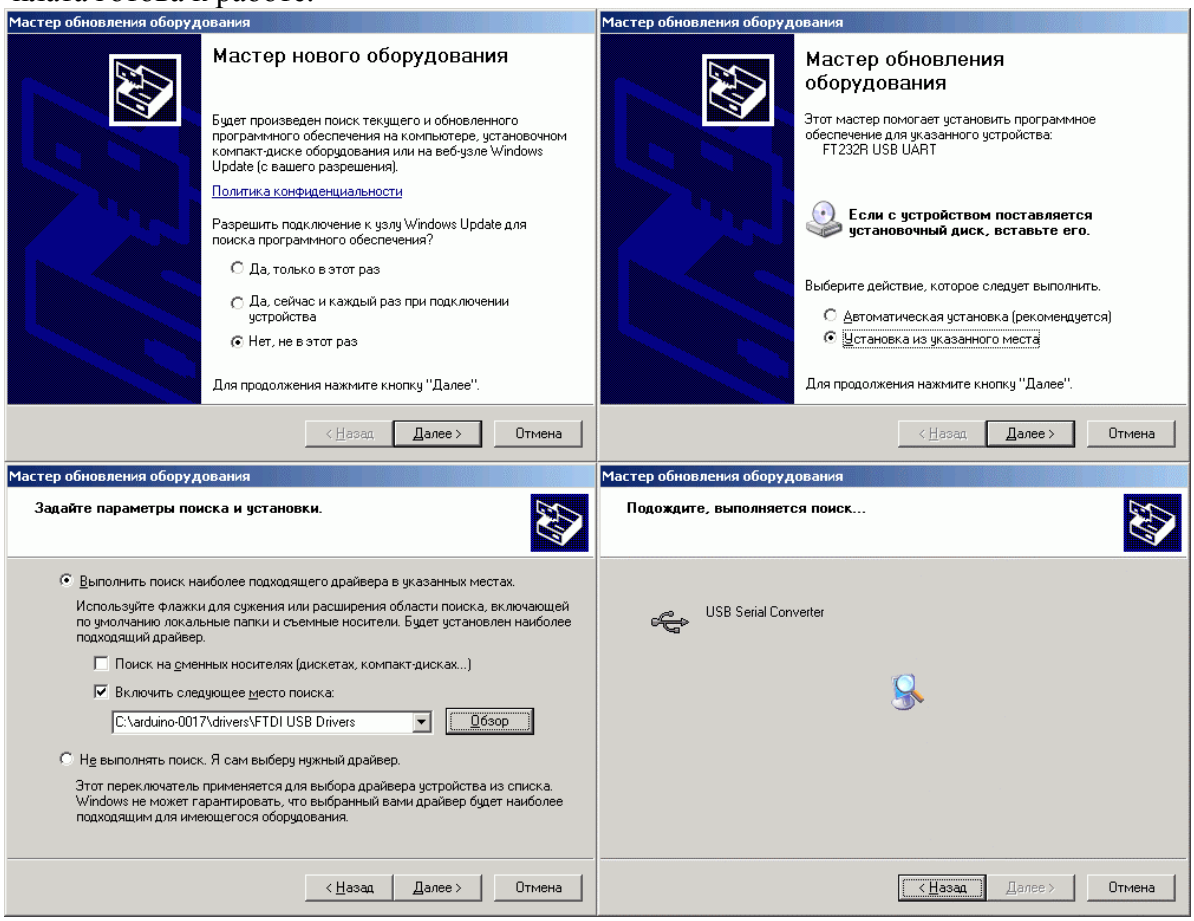
Установка драйверов требуется только для USB версий Freeduino, Freeduino MaxSerial драйверов не требует.

Подключите кабель USB к Freeduino и компьютеру при этом индикатор питания должен включиться. Для Freeduino Through-Hole предварительно нужно выбрать питание Freeduino от

разъема USB, для этого нужно установить перемычку выбора питания на те два контакта, которые расположены ближе к разъему USB (положение JUSB).

Передача данных с компьютера на микроконтроллер Atmega осуществляется через USB конвертор – встроенную микросхему типа FT232RL. Для работы с ней необходимо установить USB драйвер. Он находится в директории arduino-0017-win/drivers/FTDI USB Drivers. Последняя версия драйвера может быть найдена на официальном сайте компании FTDI по ссылке http://www.ftdichip.com/Drivers/VCP.htm, и для ОС семейства Windows версий старше XP (Windows 7) рекомендуется установка драйвера с сайта производителя.

Через некоторое время Windows обнаружит новое устройство, и запустит мастер установки новых устройств. Откажитесь от подключения к Windows Update и нажмите «Далее». В следующем окне выберите «Установка из указанного места». На следующем шаге выберите в качестве пути поиска путь до папки drivers\FTDI USB Drivers (C:\arduino-0017\drivers\FTDI USB Drivers). После подтверждения Windows установит драйвер на контроллер и последовательный порт – плата готова к работе.



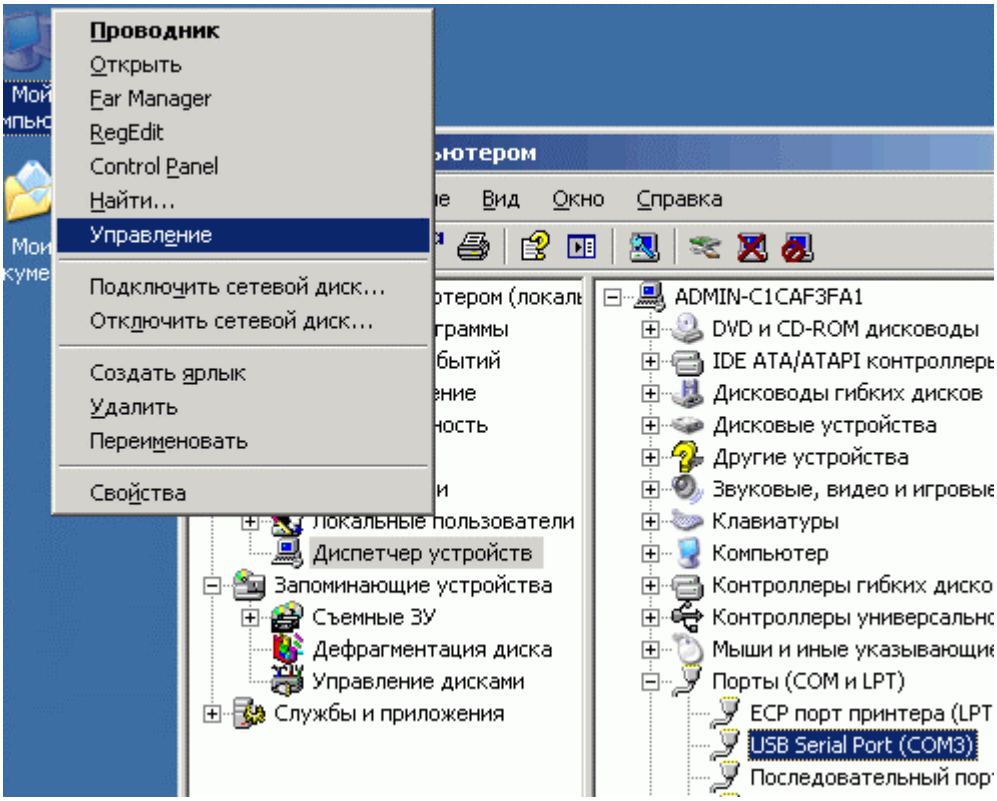
4. Запуск программного обеспечения на компьютере и его настройка

Запустите arduino.exe из папки C:\arduino-0017. Ваши проекты будут по умолчанию сохраняться в папку Мои Документы\Arduino.

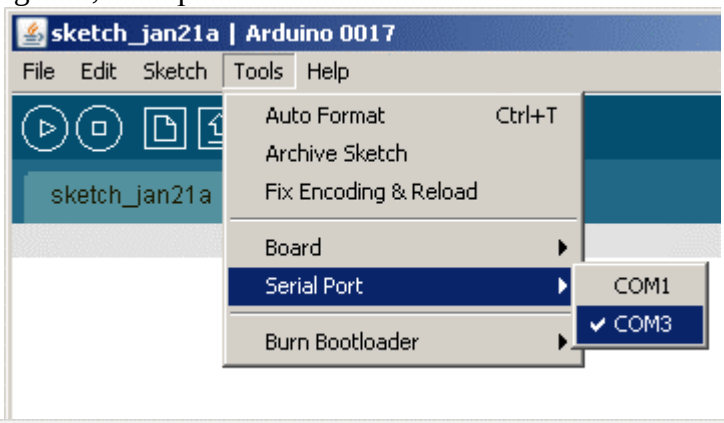
Программное обеспечение Arduino предыдущих версий некорректно работало на русских версиях Windows, что проявлялось в сообщениях об ошибках вида «Sketchbook folder disappeared», и сложностях в сохранении проектов. Для них рекомендовалось создать в папке Arduino любую папку без русских букв в имени (например Sketches), и при первом же запуске выбрав меню File\Preferences указать ее в качестве папки для хранения проектов. В ПО версии 0017 и старше эта проблема решена.

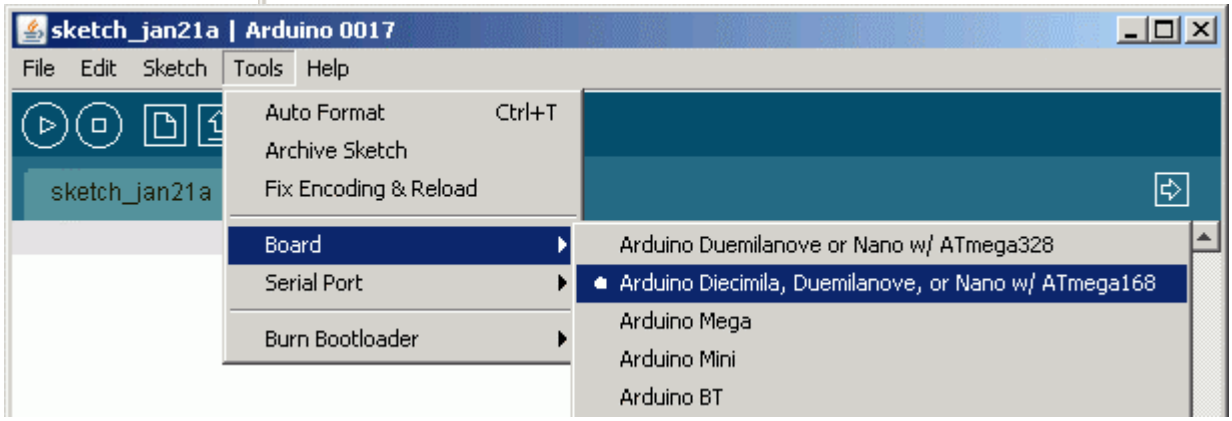
Затем укажите программному обеспечению номер COM-порта, по которому будет осуществляться передача данных с компьютера на Freeduino, используя пункт меню Tools\Serial Port.

В операционной системе Windows, номер COM-порта, который был присвоен системой USB устройству, можно определить щелкнув правой кнопкой мыши по «Мой компьютер», выбрав там Управление, и далее Диспетчер устройств/порты (Com и LPT).



Также убедитесь, что опция меню в: Tools\Board установлена соответственно Вашей плате. Для любых вариантов Freeduino с микроконтроллером ATmega168 нужно выбрать пункт «Arduino Diecimila, Duemilanove, or Nano w/ ATmega168». Если в Вашей плате используется микроконтроллер ATmega328, выбирайте «Arduino Duemilanove or Nano w/ ATmega328».





5. Написание простейшей программы и загрузка ее в Freeduino.

Для первого знакомства с микроконтроллером и прилагаемым программным обеспечением напишем простейшую программу (ее можно просто набрать в главном окне среды разработки):

```
void setup()                    //функция начальных установок - вызывается
                                //при включении микроконтроллера
{                               //фигурная скобка открывает тело функции
  pinMode(13, OUTPUT);          //устанавливаем 13 контакт в режим вывода
}                               //конец функции начальных установок
```

```
void loop()                     //основная функция - вызывается по кругу
                                //в процессе работы микроконтроллера
{
  digitalWrite(13, HIGH);       //включение светодиода на 13 выходе
  delay(1000);                  //задержка 1000 миллисекунд (1 сек)
  digitalWrite(13, LOW);        //выключение светодиода на 13 выходе
  delay(1000);                  //задержка 1000 миллисекунд (1 сек)
}                               //конец функции. По завершении она снова
                                //будет вызвана
```

Разберем исходный текст программы. Первая строка объявляет обязательную функцию инициализации – она будет вызвана при включении микроконтроллера. В ней происходит вызов стандартной функции pinMode, с помощью которого 13й контакт микроконтроллера переводится в режим вывода, чтобы затем можно было управлять напряжением на этом контакте.

Затем объявляется функция loop() – ядро микроконтроллера будет постоянно вызывать ее в процессе работы. Здесь мы последовательно выставляем на 13м контакте «высокое» значение – это +5 Вольт, делаем паузу в тысячу миллисекунд (одну секунду), выставляем «низкое» значение – это 0 Вольт, и опять делаем паузу в одну секунду.

После завершения функции loop ядро микроконтроллера вызовет ее снова.

Итак, мы получили программу, переключающую 13й контакт с +5В на 0В каждую секунду. С учетом того, что к 13му контакту на плате подключен тестовый индикатор (светодиод), получается, что он будет мигать с периодом 2 секунды.

Чтобы загрузить программу в микроконтроллер просто нажмите кнопку «Upload» на панели инструментов: 

. В процессе загрузки программы будут загораться индикаторы чтения и записи (RX и TX), затем произойдет программный сброс микроконтроллера, один раз вспыхнет тестовый индикатор, и затем начнет выполняться наша программа.

Подобная программа есть и в стандартных примерах – ее можно загрузить через меню File\Sketchbook\Examples\Digital\Blink. Штатный пример отличается лишь тем, что номер контакта заносится в специальную переменную, и при дальнейших вызовах pinMode и digitalWrite происходит обращение к этой переменной.

Описание основных функций языка Arduino

Введение

Freeduino/Arduino программируется на специальном языке программирования – он основан на C/C ++, и позволяет использовать любые его функции. Строго говоря, отдельного языка Arduino не существует, как и не существует компилятора Arduino – написанные программы преобразуются (с минимальными изменениям) в программу на языке C/C++, и затем компилируются компилятором AVR-GCC. Так что фактически, используется специализированный для микроконтроллеров AVR вариант C/C++.

Разница заключается в том, что Вы получаете простую среду разработки, и набор базовых библиотек, упрощающих доступ к находящейся «на борту» микроконтроллера периферии.

Согласитесь, очень удобно начать работу с последовательным портом на скорости 9600 бит в секунду, сделав вызов одной строчкой:

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

А при использовании «голого» C/C++ Вам бы пришлось разбираться с документацией на микроконтроллер, и вызывать нечто подобное:

```
  UBRR0H = ((F_CPU / 16 + 9600 / 2) / 9600 - 1) >> 8;
  UBRR0L = ((F_CPU / 16 + 9600 / 2) / 9600 - 1);
  sbi(UCSR0B, RXEN0);
  sbi(UCSR0B, TXEN0);
  sbi(UCSR0B, RXCIE0);
```

Здесь кратко рассмотрены основные функции и особенности программирования Arduino. Если Вы не знакомы с синтаксисом языков C/C++, советуем обратиться к любой литературе по данному вопросу, либо Internet-источникам.

С другой стороны, все представленные примеры очень просты, и скорее всего у Вас не возникнет трудностей с пониманием исходных текстов и написанием собственных программ даже без чтения дополнительной литературы.

Более полная документация (на английском языке) представлена на официальном сайте проекта – http://www.arduino.cc . Там же есть форум, ссылки на дополнительные библиотеки и их описание.

По аналогии с описанием на официальном сайте проекта Arduino, под «портом» понимается контакт микроконтроллера, выведенный на разъем под соответствующим номером. Кроме того, существует порт последовательной передачи данных (COM-порт).

Структура программы

В своей программе Вы должны объявить две основных функции: setup() и loop().

Функция setup() вызывается один раз, после каждого включения питания или сброса платы Freeduino. Используйте её, чтобы инициализировать переменные, установить режимы работы цифровых портов и т.д.

Функция loop() последовательно раз за разом исполняет команды, которые описаны в ее теле. Т.е. после завершения функции снова произойдет ее вызов.

Разберем простой пример:

```
void setup()          // начальные установки
{
  beginSerial(9600);  // установка скорости работы серийного порта на 9600 бит/сек
  pinMode(3, INPUT);  // установка 3-его порта на ввод данных
}

// Программа проверяет 3-ий порт на наличие на нём сигнала и посылает ответ в
// виде текстового сообщения на последовательный порт компьютера
void loop() // тело программы
{
  if (digitalRead(3) == HIGH) // условие на опрос 3го порта
    serialWrite('H');         // отправка сообщения в виде буквы «H» на COM-порт
  else
    serialWrite('L');         // отправка сообщения в виде буквы «L» на COM-порт
  delay(1000);                // задержка 1 сек.
}
```

Константы

Константы – предопределенные значения. Они используются, чтобы делать программы более легкими для чтения. Мы классифицируем константы в группах.

Уровни сигналов порта HIGH и LOW

При чтении или записи к цифровому порту применимо только два возможных значения – порт может быть установлен как HIGH (высокий уровень) или LOW (низкий уровень).

Уровень HIGH соответствует 5 вольтам на выходе. При чтении значения на цифровом порте, начиная с 3 вольт и выше, микропроцессор воспримет это напряжение как HIGH. Эта константа представлена целым числом 1.

Уровень LOW соответствует 0 вольтам на выходе порта. При чтении значения на цифровом порте, начиная с 2 вольт и меньше, микропроцессор воспримет это напряжение как LOW. Эта константа представлена целым числом 0.

Таким образом, оба следующих вызова будут эквивалентны:

```
digitalWrite(13, HIGH);   // можно так,
digitalWrite(13, 1);      // а можно и так
```

Считается, что первый вариант более нагляден.

Настройка цифровых портов на ввод (INPUT) и вывод (OUTPUT) сигналов

Цифровые порты могут использоваться на ввод или вывод сигналов.

Изменение порта с ввода на вывод производится при помощи функции pinMode().

Порты, сконфигурированные на ввод сигналов, имеют большое входное сопротивление, что позволяет подключать к ним источник сигнала, и порт не будет потреблять большой ток.

Порты, сконфигурированные на вывод сигналов, имеют малое выходное сопротивление. Это означает, что такие порты могут обеспечивать подключенные к ним элементы электроэнергией. В этом состоянии порты поддерживают положительное или отрицательное направление тока до 40 мА (миллиампер) на другие устройства или схемы. Это позволяет подключить к ним какую-либо нагрузку, например светодиод (через резистор, ограничивающий ток). Порты, сконфигурированные как выводы, могут быть повреждены, если их замкнуть накоротко на «землю» (общая шина питания), на источник питания +5 В, или подсоединить к мощной нагрузке с малым сопротивлением.

Пример:

```
pinMode(13, OUTPUT);  //13й вывод будет выходом
pinMode(12, INPUT);   //а 12й - входом
```

Специфичные для Freeduino/Arduino функции и объекты

Цифровой ввод/вывод

pinMode

Вызов:

pinMode (порт, режим);

Описание:

Конфигурирует указанный порт на ввод или вывод сигнала.

Параметры:

порт – номер порта, режим которого Вы желает установить (значение целого типа от 0 до 13).

режим – либо INPUT (ввод) либо OUTPUT (вывод).

Пример:

```
pinMode(13, OUTPUT);  //13й вывод будет выходом
pinMode(12, INPUT);   //а 12й - входом
```

Примечание:

Аналоговые входы могут использоваться как цифровые входы/выходы, при обращении к ним по номерам с 14 (аналоговый вход 0) по 19 (аналоговый вход 5)

digitalWrite

Вызов:

digitalWrite(порт, значение);

Описание:

Устанавливает высокий (HIGH) или низкий (LOW) уровень напряжения на указанном порте.

Параметры:
порт: номер порта
значение: HIGH или LOW
Пример:

```
digitalWrite(13, HIGH);  // выставляем 13й вывод в «высокое» состояние
```

digitalRead

Вызов:
value = digitalRead (порт);
Описание:
Считывает значение на указанном порту
Параметры:
порт: номер опрашиваемого порта
Возвращаемое значение: возвращает текущее значение на порту (HIGH или LOW) типа int
Пример:

```
int val;
val = digitalRead(12);  // опрашиваем 12й вывод
```

Примечание:
Если к считываемому порту ничего не подключено, то функция digitalRead () может беспорядочно возвращать значения HIGH или LOW.

Аналоговый ввод/вывод сигнала

analogRead

Вызов:
value = analogRead(порт);
Описание:
Считывает значение с указанного аналогового порта. Freeduino содержит 6 каналов, аналого-цифрового преобразователя на 10 битов каждый. Это означает, что входное напряжения от 0 до 5В преобразовывается в целочисленное значение от 0 до 1023. Разрешающая способность считывания составляет: 5 В/1024 значений = 0,004883 В/значение (4,883 мВ). Требуется приблизительно 100 нС (0.0001 С), чтобы считать значение аналогового ввода, так что максимальная скорость считывания - приблизительно 10000 раз в секунду.
Параметры:
порт: номер опрашиваемого аналогового входа
Возвращаемое значение: возвращает число типа int в диапазоне от 0 до 1023, считанное с указанного порта.
Пример:

```
int val;
val = analogRead(0);  // считываем значение на 0м аналоговом входе
```

Примечание:
Аналоговые порты по умолчанию определенны на ввод сигнала и в отличие от цифровых портов их не требуется конфигурировать с помощью вызова функции pinMode.

analogWrite

Вызов:
analogWrite(порт, значение);
Описание:
Выводит на порт аналоговое значение. Эта функция работает на: 3, 5, 6, 9, 10, и 11 цифровых портах Freeduino.
Может применяться для изменения яркости светодиода, для управления двигателем и т.д. После вызова функции analogWrite, соответствующий порт начинает работать в режиме широтно-импульсного модулирования напряжения до тех пор, пока не будет следующего вызова функции analogWrite (или функций digitalRead / digitalWrite на том же самом порте).
Параметры:
порт: номер опрашиваемого аналогового входа
значение: целочисленное между 0 и 255. Значение 0 генерирует 0 В на указанном порте; значение 255 генерирует +5 В на указанном порте. Для значений между 0 и 255, порт начинает

быстро чередовать уровень напряжения 0 и +5 В - чем выше значение, тем, более часто порт генерирует уровень HIGH (5 В).

Пример:

```
analogWrite(9, 128);// устанавливаем на 9 контакте значение эквивалентное 2,5В
```

Примечание:

Нет необходимости вызвать функцию pinMode, чтобы установить порт на вывод сигналов перед вызовом функции analogWrite.

Частота генерирования сигнала – приблизительно 490 Гц.

Работа со временем

millis

Вызов:

time = millis();

Описание:

Возвращает число миллисекунд, с момента исполнения Freeduino текущей программы. Счетчик переполнится и обнулится приблизительно через 9 часов.

Возвращаемое значение: возвращает значение типа unsigned long

Пример:

```
unsigned long time; // объявление переменной time типа unsigned long
time = millis(); // передача количества миллисекунд
```

delay

Вызов:

delay(время_мс);

Описание:

Приостанавливает программу на заданное число миллисекунд.

Параметры:

время_мс – время задержки программы в миллисекундах

Пример:

```
delay(1000);  //пауза 1 секунда
```

delayMicroseconds

Вызов:

delayMicroseconds(время_мкс);

Описание:

Приостанавливает программу на заданное число микросекунд.

Параметры:

время_мкс – время задержки программы в микросекундах

Пример:

```
delayMicroseconds(500);  //пауза 500 микросекунд
```

pulseIn

Вызов:

pulseIn(порт, значение);

Описание:

Считывает импульс (высокий или низкий) с цифрового порта и возвращает продолжительность импульса в микросекундах.

Например, если параметр «значение» при вызове функции установлен в HIGH, то pulseIn() ожидает, когда на порт поступит высокий уровень сигнала. С момента его поступления начинается отсчет времени до тех пор, пока на порт не поступит низкий уровень сигнала. Функция возвращает длину импульса (высокого уровня) в микросекундах. Работает с импульсами от 10 микросекунд до 3 минут. Обратите внимание, что эта функция не будет возвращать результат, пока импульс не будет обнаружен.

Параметры:

порт: номер порта, с которого считываем импульс

значение: тип импульса HIGH или LOW

Возвращаемое значение: возвращает длительность импульса в микросекундах (тип int)

Пример:

```
int duration; // объявление переменной duration типа int
```

```
duration = pulseIn(pin, HIGH); // измеряем длительность импульса
```

Последовательная передача данных

Freeduino имеет встроенный контроллер для последовательной передачи данных, который может использоваться как для связи между Freeduino/Arduino устройствами, так и для связи с компьютером. На компьютере соответствующее соединение представлено USB COM-портом.

Связь происходит по цифровым портам 0 и 1, и поэтому Вы не сможете использовать их для цифрового ввода/вывода если используете функции последовательной передачи данных.

Serial.begin

Вызов:

Serial.begin(скорость_передачи);

Описание:

Устанавливает скорость передачи информации COM порта битах в секунду для последовательной передачи данных. Для того чтобы поддерживать связь с компьютером, используйте одну из этих нормированных скоростей: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, или 115200. Также Вы можете определить другие скорости при связи с другим микроконтроллером по портам 0 и 1.

Параметры:

скорость_передачи: скорость потока данных в битах в секунду.

Пример:

```
Serial.begin(9600); //устанавливаем скорость 9600 бит/сек
```

Serial.available

Вызов:

count = Serial.available();

Описание:

Принимаемые по последовательному порту байты попадают в буфер микроконтроллера, откуда Ваша программа может их считать. Функция возвращает количество накопленных в буфере байт. Последовательный буфер может хранить до 128 байт.

Возвращаемое значение:

Возвращает значение типа int – количество байт, доступных для чтения, в последовательном буфере, или 0, если ничего не доступно.

Пример:

```
if (Serial.available() > 0) { // Если в буфере есть данные
  // здесь должен быть прием и обработка данных
}
```

Serial.read

Вызов:

char = Serial.read();

Описание:

Считывает следующий байт из буфера последовательного порта.

Возвращаемое значение:

Первый доступный байт входящих данных с последовательного порта, или -1 если нет входящих данных.

Пример:

```
incomingByte = Serial.read(); // читаем байт
```

Serial.flush

Вызов:

Serial.flush();

Описание:

Очищает входной буфер последовательного порта. Находящиеся в буфере данные теряются, и дальнейшие вызовы Serial.read() или Serial.available() будут иметь смысл для данных, полученных после вызова Serial.flush().

Пример:

```
Serial.flush(); // Очищаем буфер - начинаем прием данных «с чистого листа»
```

Serial.print

Описание:

Вывод данных на последовательный порт.

Параметры:

Функция имеет несколько форм вызова в зависимости от типа и формата выводимых данных.

Serial.print(b, DEC) выводит ASCII-строку - десятичное представление числа b.

```
int b = 79;
Serial.print(b, DEC); //выдаст в порт строку «79»
```

Serial.print(b, HEX) выводит ASCII-строку - шестнадцатиричное представление числа b.

```
int b = 79;
Serial.print(b, HEX); //выдаст в порт строку «4F»
```

Serial.print(b, OCT) выводит ASCII-строку - восьмеричное представление числа b.

```
int b = 79;
Serial.print(b, OCT); //выдаст в порт строку «117»
```

Serial.print(b, BIN) выводит ASCII-строку - двоичное представление числа b.

```
int b = 79;
Serial.print(b, BIN); //выдаст в порт строку «1001111»
```

Serial.print(b, BYTE) выводит младший байт числа b.

```
int b = 79;
Serial.print(b, BYTE); //выведет число 79 (один байт). В мониторе
                       //последовательного порта получим символ «O» - его
                       //код равен 79
```

Serial.print(str) если str – строка или массив символов, побайтно передает str на COM-порт.

```
char bytes[3] = {79, 80, 81};    //массив из 3 байт со значениями 79,80,81
Serial.print("Here our bytes:"); //выводит строку «Here our bytes:»
Serial.print(bytes);             //выводит 3 символа с кодами 79,80,81 –
                                 //это символы «OPQ»
```

Serial.print(b) если b имеет тип byte или char, выводит в порт само число b.

```
char b = 79;
Serial.print(b); //выдаст в порт символ «O»
```

Serial.print(b) если b имеет целый тип, выводит в порт десятичное представление числа b.

```
int b = 79;
Serial.print(b); //выдаст в порт строку «79»
```

Serial.println

Описание:

Функция Serial.println аналогична функции Serial.print, и имеет такие же варианты вызова. Единственное отличие заключается в том, что после данных дополнительно выводятся два символа – символ возврата каретки (ASCII 13, или '\r') и символ новой линии (ASCII 10, или '\n').

Пример 1 и пример 2 выведут в порт одно и то же:

Пример 1:

```
int b = 79;
Serial.print(b, DEC); //выдаст в порт строку «79»
Serial.print("\r\n"); //выведет символы "\r\n" – перевод строки
Serial.print(b, HEX); //выдаст в порт строку «4F»
Serial.print("\r\n");//выведет символы "\r\n" – перевод строки
```

Пример 2:

```
int b = 79;
Serial.println(b, DEC); //выдаст в порт строку «79\r\n»
Serial.println(b, HEX); //выдаст в порт строку «4F\r\n»
```

В мониторе последовательного порта получим:

```
79
4F
```

Пример обмена данными с компьютером

Загрузим стандартный пример «Physical Pixel» через меню File \ Sketchbook \ Examples \ Communication \ PhysicalPixel. Эта программа ждет данные от компьютера. При получении символа ‘H’ тестовый индикатор загорается, при получении символа ‘L’ – гаснет. Разберем ее исходный код:

```
int outputPin = 13;                //здесь храним номер контакта
int val;                           //здесь будет храниться принятый символ

void setup()
{
  Serial.begin(9600);              //установка порта на скорость 9600 бит/сек
  pinMode(outputPin, OUTPUT);      //устанавливаем 13 контакт в режим вывода
}

void loop()
{
  if (Serial.available()) {         //если есть принятый символ,
    val = Serial.read();            //  то читаем его и сохраняем в val
    if (val == 'H') {               //  если принят симовол 'H',...
      digitalWrite(outputPin, HIGH);//  то включаем светодиод
    }
    if (val == 'L') {               //  если принят симовол 'L',
      digitalWrite(outputPin, LOW); //    то выключаем светодиод
    }
  }
}
```

Обратите внимание на вложенные условия и порядок следования отрывающих и закрывающих фигурных скобок. Для удобства чтения кода программы каждый следующий уровень вложенности сдвинут вправо. Кроме того, редактор помогает читать код – если Вы поставите курсор справа от скобки, он выделит соответствующую ей парную скобку.

Как проверить работу этой программы после того, как Вы загрузите ее в микроконтроллер? Нужно найти способ отправлять символы на COM-порт компьютера, чтобы микроконтроллер принимал и обрабатывал их. Существует множество вариантов решения этой задачи.

Используем встроенный в среду разработки монитор COM-порта

Это наиболее простой, и понятный начинающим метод.

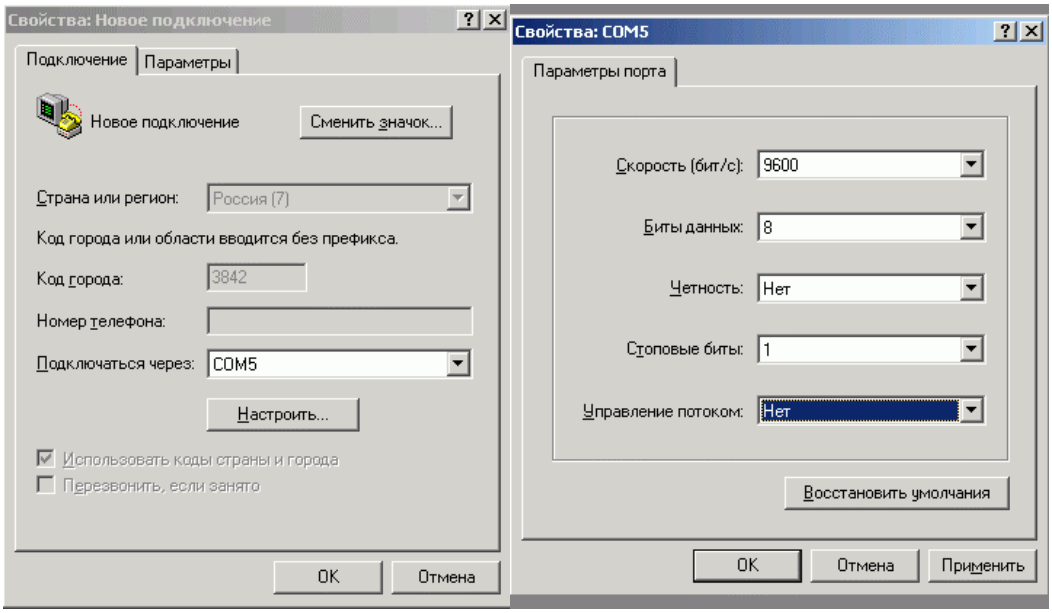
На панели инструментов есть кнопка монитора COM-порта 

. Вызвав монитор убедитесь, что выбрана та же самая скорость обмена, что и в программе микроконтроллера. Теперь можно вводить любые символы в поле ввода справа, и нажимать кнопку «Send» – введенные символы будут отправлены в порт, и там их примет Ваша программа. Введите там латинскую букву «H», нажмите «Send» – тестовый светодиод загорится. Если послать «L» – погаснет. Кстати, все данные, которые Ваша программа будет посылать на COM-порт будут выводиться в черном окне снизу.

Используем программу эмуляции терминала HyperTerminal

Это немного более сложный в реализации вариант обмена.

В состав Windows обычно включена программа эмуляции терминала HyperTerminal. В Windows XP ее можно найти в меню Пуск \ Все программы \ Программы \ Стандартные \ Связь \ HyperTerminal. При запуске нужно отказаться от создания подключения, выбрать меню Файл \ Свойства. В появившемся диалоге выбрать свой COM-порт, нажать «Настроить», и настроить параметры связи в соответствии с рисунком:



Нажмите «OK» в обоих окнах, и попав в основное окно программы, любую клавишу на клавиатуре – HyperTerminal подключится к COM-порту. Теперь все набираемые на клавиатуре символы попадают через COM-порт в микроконтроллер, а все, что отправляет микроконтроллер, попадает на экран. Понажимайте клавиши «H» и «L» (следите за выбранным языком, и регистром) – тестовый светодиод должен загораться и гаснуть.

Напишем собственную программу для ПК !

Этот вариант для настоящих энтузиастов, желающих программировать не только Freeduino, но и ПК. А почему бы и нет? Нам не потребуется изучать детали программирования последовательного порта под Windows, или еще какие-то сложные вещи. Специально для решения подобных простых задач существует язык Processing (http://processing.org), очень похожий синтаксисом и даже средой разработки на программное обеспечение Arduino.

На прилагаемом диске есть архив processing-0150.zip – рекомендуем разархивировать содержимое на диск C:. Запустите файл C:\processing-0150\processing.exe – Вы увидите среду разработки, похожую на Arduino.

Исходный код программы для языка Processing есть в комментариях ниже основного текста примера Physical Pixel. Здесь он приведен с минимальными изменениями – мы исправили открытие порта, чтобы можно было легко заменить его номер:

```
import processing.serial.*;
Serial port;
void setup()
{
  size(200, 200);
  noStroke();
  frameRate(10);
  port = new Serial(this, "COM5", 9600); // !!! Здесь прописать свой COM-порт !!!
}
boolean mouseOverRect() //Возвращает истину, если курсор внутри квадрата
{
  return ((mouseX >= 50)&&(mouseX <= 150)&&(mouseY >= 50)&(mouseY <= 150));
}
void draw()
{
  background(#222222);
  if(mouseOverRect())      // Если курсор внутри квадрата….
  {
    fill(#BBBBB0);         //   сменить цвет на поярче
    port.write('H');       //   послать 'H' в микроконтроллер
  } else {                 // если не внутри...
    fill(#666660);         //   сменить цвет на потемнее
    port.write('L');       //   послать 'L' в микроконтроллер
  }
  rect(50, 50, 100, 100);  // нарисовать квадрат
}
```

Запустите программу (через меню Sketch \ Run) – появится окно с квадратом, при помещении в который курсора мыши, будет загораться светодиод на Freeduino.

Описание языка Processing и его возможностей выходит за рамки этого простого повествования, но во многих примерах для Arduino в комментариях ниже основного текста программы представлен код Processing для ПК, взаимодействующий с Freeduino.

Пример Fading – аналоговый выход (ШИМ)

Настоящего аналогового выхода в данном микроконтроллере нет, но на выводы 3, 5, 6, 9, 10, 11 можно подавать сигнал с широтно-импульсной модуляцией (ШИМ). Здесь мы не будем разбирать точные, математически строгие формулировки, а попробуем проще всего объяснить, о чем идет речь.

Мы уже научились устанавливать на выходе микроконтроллера одно из двух значений – 0, либо +5 Вольт, и какие-то другие значения постоянного напряжения микроконтроллер формировать не может. Но что делать, если мы захотим получить +2,5 Вольта? Строгого решения этой задачи у нас не получится, но можно с большой частотой переключать выход из низкого в высокое состояние и обратно. В таком случае, светодиод, подключенный к такому выводу, будет гореть в половину яркости (на самом деле он будет очень часто мигать), а среднее значение сигнала за период будет +2,5 Вольта. Меняя соотношение между временем нахождения выхода в высоком и низком состоянии, можно получать различные средние значения напряжения.

Хорошая новость заключается в том, что нам предоставляется функция analogWrite, а все сложности реализации скрыты от наших глаз. Функция принимает 2 параметра – номер вывода (не забываем, что ШИМ сигнал можно выводить только на выводы 3, 5, 6, 9, 10, 11), и величину, пропорциональную среднему значению напряжения. 0 соответствует 0 Вольтам, а 255 - +5 Вольтам.

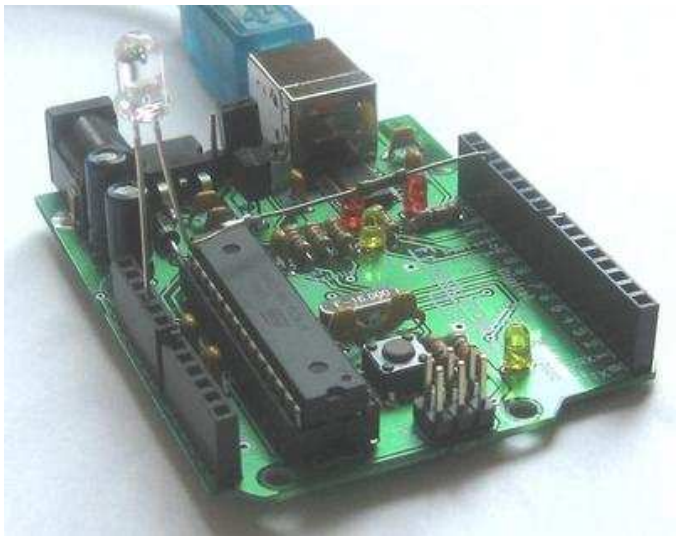
Таким образом, чтобы получить на 3м выводе +2,5 Вольта, а на 5м +1 Вольт, мы сделаем так:

```
analogWrite(3, 128);  // 255/2 = 128
analogWrite(5, 51);   // 255/5 = 51
```

Чтобы визуально наблюдать изменение выходного сигнала, нам потребуется подключить внешний светодиод, поскольку 13й вывод, к которому подключен встроенный, не поддерживает ШИМ. Светодиод нужно подключить через ограничительный резистор номиналом от 200 до 1500 Ом, и с соблюдением полярности – иначе он просто не загорится. Обычно у светодиода один из выводов чуть длиннее – это анод (плюс), а второй короче – эта катод (минус). Кроме того, со стороны катода пластиковый обод немного срезан.

Соединение должно быть таким – 9й вывод микроконтроллера соединен с резистором, второй вывод резистора соединен с анодом светодиода, катод светодиода соединен «землей» (любой из выводов, обозначенный Gnd).

У нас это получилось вот так:



Загрузим стандартный пример «Fading» через меню File \ Sketchbook \ Examples \ Analog \ Fading.

```
int value = 0;                  //здесь храним значение яркости
int ledpin = 9;                 //светодиод подключим к 9 контакту

void setup()
{
  pinMode(ledpin, OUTPUT);      //устанавливаем 9 контакт в режим вывода
}

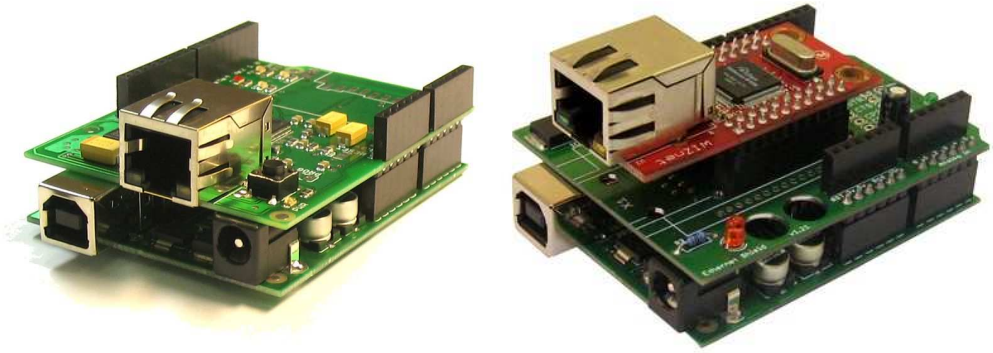
void loop()
{
```

```
  for(value = 0 ; value <= 255; value+=5) //плавно зажигаем светодиод
  {
    analogWrite(ledpin, value);           //выставляем значение
    delay(30);                            //небольшая пауза
  }
  for(value = 255; value >=0; value-=5)   //плавно гасим
  {
    analogWrite(ledpin, value);
    delay(30);
  }
}
```

Программа очень простая, и по исходному тексту и комментариям все должно быть понятно. В программе используется циклы (оператор for). В первом цикле переменная value изменяется от 0 до 255 с шагом 5, во втором – от 255 до 0 с тем же шагом 5.

Пример Web-сервера (с модулем E-Shield)

Данный пример демонстрирует, как с помощью модуля E-Shield и стандартной библиотеки Ethernet можно реализовать на Freeduino простой Web-сервер. Предполагается версия ПО Arduino не ниже 0.12 – с этой версии в состав входит библиотека Ethernet.



Наш Web-сервер будет очень простым – при получении запроса он будет измерять значения на всех шести аналоговых входах, и выдавать их клиенту. Таким образом, набрав в любом браузере (Firefox/Opera/Internet Explorer) IP адрес устройства мы увидим значения аналоговых входов.

Разберем исходный код:

```
#include <Ethernet.h>          //Подключаем описание библиотеки ethernet

//Здесь задается MAC адрес и IP адрес устройства.
//В данном примере предполагается, что IP адрес компьютера, с которого
//будем обращаться к устройству будет любым из сети 192.168.1.x,
//и маска 255.255.255.0
//MAC адрес можно задать любой, лишь бы он не пересекался с имеющимися в сети
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 1, 177 };

Server server(80);   //Создаем сервер, слушающий 80й порт (80 - это порт HTTP)

void setup()
{
  Ethernet.begin(mac, ip);   //Инициализируем Ethernet модуль
  server.begin();            //Начинаем ожидать соединений на 80 порту
}

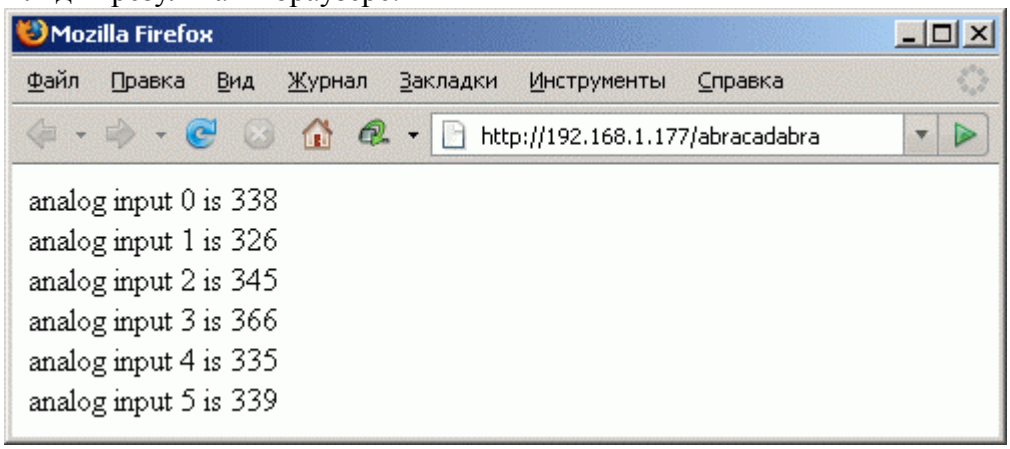
void loop()
{
  //Если кто-то установил соединение с нашим сервером, следующая строчка
  //создаст объект-клиент.
  Client client = server.available();
  if (client) {  //если client не нулевой (т.е. соединение есть)
    //...значит кто-то подключился. Согласно протокола HTTP клиент
    //шлет довольно сложный запрос, но мы не будет его разбирать -
    //нам достаточно дождаться окончания запроса.
    //Запрос заканчивается пустой строкой, поэтому мы просто дождемся
    //получения символа '\n', перед которым тоже были получены '\n' и '\r'.

    //current_line_is_blank - это переменная-флаг. Она равна true, если
    //вновь полученная от клиента строка пустая, т.е. в полученных данных
    //не встретилось символов отличных от '\n' и '\r'.
    //Будем выставлять переменную current_line_is_blank в false при
    //получении любого отличного от '\n' и '\r' символа
    boolean current_line_is_blank = true;
    while (client.connected()) {   //пока клиент подключен
      if (client.available()) {    //если от него пришел символ
        char c = client.read();    //читаем этот символ
        //Если получен перевод строки ('\n') и current_line_is_blank == true,
        //значит мы получили пустую строку, т.е. запрос клиента окончен -
        //можно слать ответ
        if (c == '\n' && current_line_is_blank) {
          //шлем стандартный HTTP заголовок
          client.println("HTTP/1.1 200 OK");
```

```
          client.println("Content-Type: text/html");
          client.println();     //он тоже заканчивается пустой строкой!

          //выдаем поочередно значения 6 аналоговых входов
          for (int i = 0; i < 6; i++) {
            client.print("analog input ");
            client.print(i);
            client.print(" is ");
            client.print(analogRead(i));
            client.println("<br>");   //HTML тэг <br> - перевод строки
          }
          break;   //работа окончена - можно выходить из цикла while
        }
        if (c == '\n') {  //если получен перевод строки, значит началась новая
          //строка. Выставим флаг current_line_is_blank = true
          current_line_is_blank = true;
        } else if (c != '\r') { //если получен любой другой символ, отличный
          //от возврата каретки ('\r'), значит получаемая строка непустая
          current_line_is_blank = false;
        }
      }
    }
    //небольшая пауза, чтобы данные успели уйти
    delay(1);
    //разрываем соединение с клиентом
    client.stop();
  }
}
```

Вот как выглядит результат в браузере:



Здесь мы специально запросили у устройства ресурс "abracadabra ", чтобы показать, что содержимое HTTP запроса игнорируется, и всегда генерируется страничка со значениями аналоговых входов.

Пример подключения кнопок

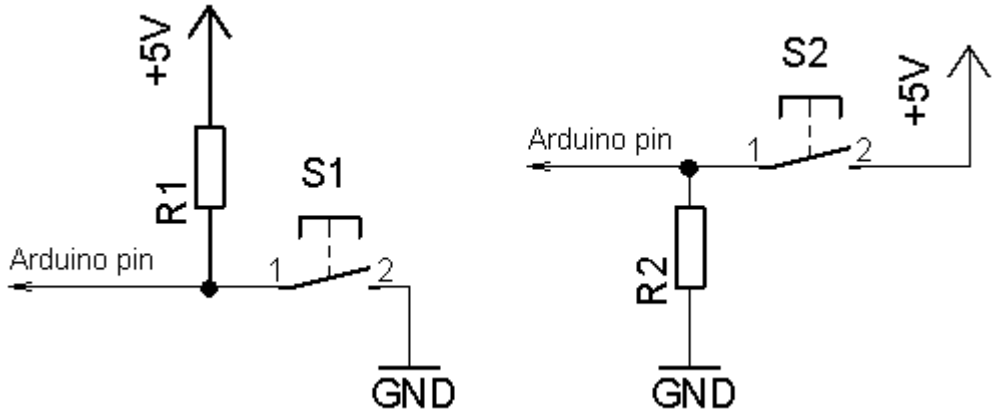
Распространенной задачей является подключение кнопок к микроконтроллеру. Несмотря на кажущуюся простоту, эта задача имеет некоторые, возможно неочевидные особенности.

Будем рассматривать наиболее простой и распространенный вариант кнопки, зачастую называемый тактовой кнопкой, имеющую два нормально разомкнутых контакта, замыкаемых при нажатии на кнопку.

Подключение кнопки к микроконтроллеру

Если мы подключим один из контактов, например, к общему проводу («земле»), а второй к выбранному выводу микроконтроллера, переключенного в режим входа, то выяснится, что такой метод не работает. При нажатии кнопки вывод микроконтроллера соединяется с землей, и программа будет считывать (с помощью функции digitalRead) логический 0 с этого вывода, но при отпущенной кнопке вывод микроконтроллера не будет соединен ни с какой линией, что часто называют «висит в воздухе». В таком режиме программа будет считать с вывода и 0 и 1 совершенно случайным образом.

Правильное подключение предполагает, что в разомкнутом состоянии вывод микроконтроллера должен через резистор быть соединен например с шиной питания, а в замкнутом - с землей, либо наоборот. Сопротивление резистора не должно быть слишком маленьким, чтобы ток, текущий через него при замкнутых контактах кнопки не был слишком большим. Обычно используют значения порядка 10-100 кОм. Оба варианта подключения можно изобразить следующим образом:



Первый вариант предпочтительнее, поскольку подтягивающие к +5В резисторы уже есть внутри микроконтроллера – их нужно только программно включить. Кнопка будет либо соединять вывод микроконтроллера с землей, либо разъединять, и тогда он "притянется" резистором к +5В.

После того, как вывод микроконтроллера установлен в режим входа, чтобы включить на нем подтягивающий резистор нужно "записать" в него 1.

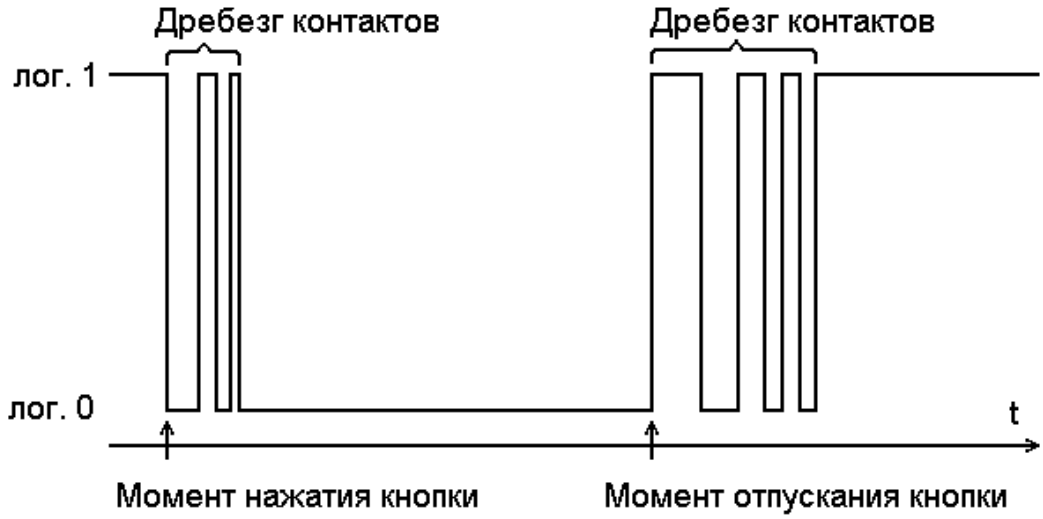
Пример программы, зажигающей светодиод на 13 выводе при нажатии кнопки на 2 выводе будет выглядеть примерно так:

```
void setup() {
  pinMode(13, OUTPUT);   //13й вывод - выход
  pinMode(2, INPUT);     //2й - вход. Здесь кнопка, замыкающая на землю
  digitalWrite(2, HIGH); //включаем подтягивающий резистор
}
void loop() {
  digitalWrite(13, !digitalRead(2));
}
```

Обращаем внимание на то, что значение, прочитанное с 2 вывода, инвертируется с помощью оператора «!», поскольку при нажатии на кнопку будет считываться «0», а при ее отпускании «1».

Дребезг контактов

На практике зачастую приходится бороться с таким явлением, как дребезг контактов, которое заключается в том, что при соприкосновении или расхождении контактов в механических переключающих устройствах, таких, как реле или кнопка, происходит многократное замыкание и размыкание. Схематично это может быть представлено следующим образом:



Чтобы микроконтроллер не обработал такую пачку переключений как множественное нажатие и отпускание кнопки нужно либо применить специальную схему, либо побороть дребезг программно. Идея программной борьбы проста – после того, как произошло переключение (от момента нажатия или отпускания кнопки), в течение некоторого защитного интервала времени игнорировать любые другие переключения.

Существует специальная библиотека Bounce, упрощающая борьбу с дребезгом контактов, и имеющая дополнительные возможности:

http://www.arduino.cc/playground/Code/Bounce

http://www.arduino.cc/playground/uploads/Code/Bounce.zip

Как и в большинстве случаев, установка библиотеки сводится к распаковке архива в подпапку \hardware\libraries\ папки с ПО Arduino.

Полная документация на библиотеку может быть найдена на сайте разработчика, а здесь рассмотрим только наиболее важное.

Bounce – конструктор объекта

Вызов:

Bounce имя_объекта = Bounce(вывод, интервал);

Создает экземпляр класса Bounce, принимает номер вывода, с которого будет считываться сигнал, и длительность защитного интервала в миллисекундах. После создания объекта можно вызывать его методы.

Метод Bounce::update

Вызов:

имя_объекта.update()

Возвращает значение типа int – истину, если состояние вывода изменилось, ложь, если нет.

Метод Bounce::read

Вызов:

имя_объекта.read()

Возвращает значение типа int – состояние вывода.

Метод Bounce::rebounce

Вызов:

имя_объекта.rebounce(время_повтора)

Повторяет последнее произошедшее событие через заданное время в миллисекундах.

Рассмотрим исходный код программы, посылающей в последовательный порт сообщение «pressed» при нажатии кнопки, сообщение «released» при ее отпускании, и повторяющей сообщение «pressed» каждые пол секунды при удержании кнопки.

```
#include <Bounce.h>

Bounce bouncer = Bounce(2, 40); //создаем экземпляр класса Bounce для 2 вывода

void setup()
{
  pinMode(2, INPUT);   //переключаем 2 вывод в режим входа
```

```
  digitalWrite(2, 1);  //включаем на нем подтягивающий резистор
  Serial.begin(9600);  //установка порта на скорость 9600 бит/сек
}

void loop()
{
  if (bouncer.update()) {     //если произошло событие
    if (bouncer.read()==0) {    //если кнопка нажата
      Serial.println("pressed");  //вывод сообщения о нажатии
      bouncer.rebounce(500);      //повторить событие через 500мс
    } else {
      Serial.println("released"); //вывод сообщения об отпускании
    }
  }
}
```

Пример работы с EEPROM

Arduino-совместимые платы имеют небольшое ПЗУ (EEPROM), размер которого зависит от используемого микроконтроллера. Для ATmega168 это 512 байт, для ATmega328 – 1024 байта.

Библиотека EEPROM

Для работы с EEPROM существует стандартная библиотека EEPROM, позволяющая записать байт в EEPROM по нужному адресу, и прочесть его оттуда.

Рассмотрим простой пример. Пусть в самом первом байте EEPROM (по адресу 0) хранится счетчик числа сбросов микроконтроллера. При каждом сбросе будет вызываться функция setup(), в которой мы будем считывать счетчик, увеличивать его на единицу и записывать обратно. Также, чтобы увидеть значение счетчика в мониторе последовательного порта, будем выводить число туда.

```
#include <EEPROM.h>
//подключаем заголовочный файл библиотеки EEPROM

void setup()
{
  uint8_t counter; //Здесь будем хранить счетчик
  //uint8_t - тип данных, занимающий 1 байт,
  //и хранящий значение от 0 до 255

  //Инициализируем последовательный порт на 9600бит/сек:
  Serial.begin(9600);

  //Читаем счетчик из EEPROM:
  counter = EEPROM.read(0);

  counter++;  // увеличиваем его на единицу

  //Записываем значение счетчика
  EEPROM.write(0, counter);

  //Выводим счетчик в порт:
  Serial.print("It's reset #");
  Serial.println(counter, DEC);
}

void loop()
{
}
```

Поэкспериментируйте с программой, и убедитесь, что значение счетчика сохраняется и после пропадания питания. Фирма ATMEL – производитель микроконтроллеров ATmega, декларирует порядка 100’000 (ста тысяч) успешных циклов записи в EEPROM и хранение данных до 100 лет при температуре 25 градусов по Цельсию.

Библиотека EEPROM2

Рассмотренная программа имеет один недостаток, связанный с тем, что для хранения счетчика используется всего один байт. После достижения счетчиком значения 255 следующее увеличение даст значение 0, из-за переполнения разрядной сетки.

Таким образом, требуется применение счетчика большего размера – 2 байта дадут верхнюю границу в 65’535, а 4 байта – 4’294’967’295.

Стандартная библиотека имеет функции только для работы с однобайтными данными типа uint8_t, и чтение 4-байтного числа будет выглядеть примерно так:

```
...
  *((uint8_t*)&counter + 0) = EEPROM.read(0);
  *((uint8_t*)&counter + 1) = EEPROM.read(1);
  *((uint8_t*)&counter + 2) = EEPROM.read(2);
  *((uint8_t*)&counter + 3) = EEPROM.read(3);
...
```

Такой исходный код, конечно, будет понятен компилятору, но далеко не каждому начинающему программисту, и поэтому мы написали свою версию библиотеки для работы с EEPOM:

http://www.freeduino.ru/arduino/files/EEPROM2.zip

Как и в большинстве случаев, установка библиотеки сводится к распаковке архива в подпапку \hardware\libraries\ папки с ПО Arduino.

Теперь рассмотрим аналогичную первому примеру программу, хранящую счетчик в 4-байтном числе с помощью библиотеки EEPROM2:

```
#include <EEPROM2.h>
//подключаем заголовочный файл библиотеки EEPROM2

void setup()
{
  unsigned long counter; //Здесь будем хранить счетчик
  //unsigned long - тип данных, занимающий 4 байта,
  //и хранящий значение от 0 до 4'294'967'295

  //следующие 2 строчки нужны только при первом запуске,
  //чтобы обнулить значение счетчика:
  counter = 0;
  EEPROM_write(0, counter);

  //Инициализируем последовательный порт на 9600бит/сек:
  Serial.begin(9600);

  //Читаем счетчик из EEPROM:
  EEPROM_read(0, counter);

  counter++;  // увеличиваем его на единицу

  //Записываем значение счетчика
  EEPROM_write(0, counter);

  //Выводим счетчик в порт:
  Serial.print("It's reset #");
  Serial.println(counter, DEC);
}

void loop()
{
}
```

Загрузите программу в микроконтроллер, удалите строчки, обнуляющие счетчик, и загрузите программу еще раз.

Комментариев в исходном коде вполне достаточно для понимания программы, а синтаксис работы с библиотекой отличается от оригинальной незначительно.

Работа с EEPROM через boot-loader

Процесс старта Arduino-совместимой платы начинается со старта загрузчика, часто называемого boot-loader. Он занимает последние 2 К памяти программ, и именно из-за него Ваши проекты ограничены размером 14 К для ATmega168 и 30 К для ATmega328, при объеме памяти программ у микроконтроллеров 16 К и 32 К соответственно.

После старта загрузчик некоторое время ожидает, что управляющий ПК начнет передачу новой программы, и если этот процесс начинается, то именно загрузчик и «прошивает» Вашу программу в микроконтроллер. Если загрузка программы не началась, то загрузчик передает управление на ранее загруженную Вами в микроконтроллер программу, в результате чего выполняется функция setup(), а затем в бесконечном цикле функция loop().

Кстати, именно загрузчик мигает светодиодом на 13 выводе при сбросе микроконтроллера.

При обмене данными с ПК загрузчик использует протокол stk500v1, и может работать, в том числе и с EEPROM.

Для чтения содержимого EEPROM в файл «0.bin» необходимо в командной строке

- ✓ Перейти в подпапку \hardware\tools\avr\bin\ папки с ПО Arduino;

✓ Выполнить команду (в одну строку):

```
avrdude -q -C ..\etc\avrdude.conf -p m168 -c stk500v1 -P COM3 -b 19200 -U
         eeprom:r:0.bin:r
```

Для записи содержимого файла 0.bin в EEPROM команда такая:

```
avrdude -q -C ..\etc\avrdude.conf -p m168 -c stk500v1 -P COM3 -b 19200 -U
         eeprom:w:0.bin
```

Параметр «COM3» нужно заменить на Ваш номер порта.

Ссылки

Ссылки на полезные ресурсы в сети

http://www.freeduino.ru/ – наша страничка. Здесь документация на русском языке, примеры применения, форум и прочее.

http://ru.wikipedia.org/wiki/Arduino – страничка на русском языке из Википедии – свободной энциклопедии, описывающая Arduino.

http://www.arduino.cc/ – официальный сайт проекта Arduino (на английском языке). Здесь новые версии среды разработки, документация, примеры, форум и многое другое.

http://www.freeduino.org/ – официальная страничка проекта Freeduino (на английском языке). Здесь кроме технической информации есть огромное количество ссылок на различные проекты, так или иначе связанные с Freeduino/Arduino.

Контакты

ООО "Микромодульные технологии"

e-mail: freeduino@freeduino.ru
Адрес: Россия, 650033, г. Кемерово, ул. Аллейная, 11.
Для почты: Россия, 650033, г. Кемерово, а/я 3060.
Тел.(моб): 8-961-700-81-59
Факс: (8-3842) 62-19-49
WEB:http://www.freeduino.ru