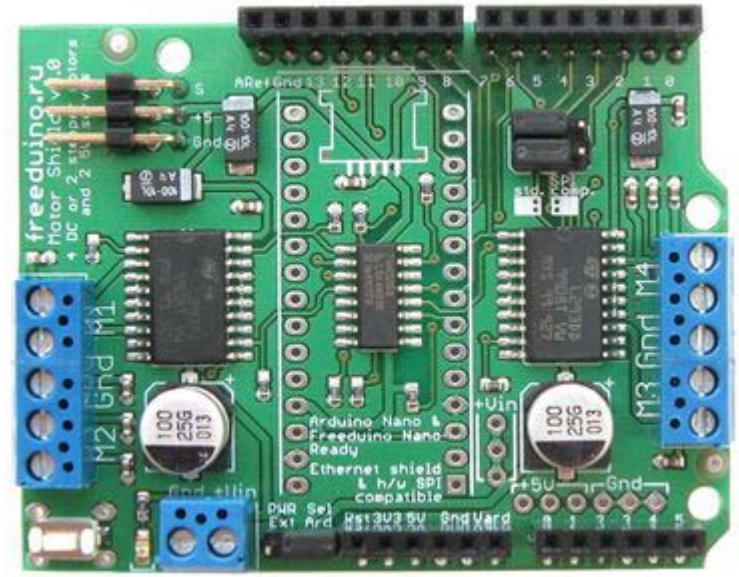
Описание Motor Shield

Motor Shield (далее M–Shield) – силовой модуль управления двигателями для микроконтроллеров серии Freeduino/Arduino. Модуль предназначен для упрощения разработки моторизированных и робототехнических устройств .

Модуль существует в двух версиях: v2 со штыревыми компонентами, и пришедшая ей на замену v3 с поверхностным монтажом компонентов и дополнительным функционалом.

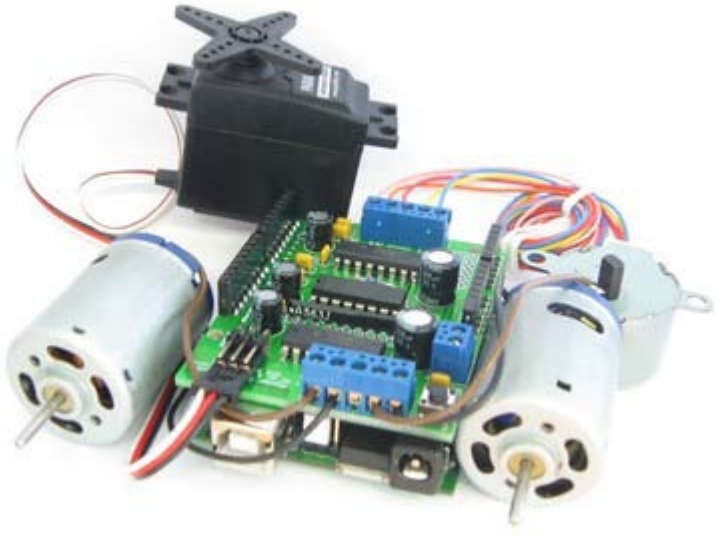
Модуль подключается к Freeduino с помощью установленных на нем разъемов.

Обновленная версия M-Shield v3:



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 одинарной 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.

Обновленная версия v3 может быть переключена в режим, совместимый с аппаратным SPI (а значит и с модулями типа Ethernet Shield), в котором не используются выводы 10, 11, 12, 13. Также, для v3 возможно простое подключение к базовой плате в формате Nano.

^^^

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: MaxSerial, Through-Hole, Diecimila, 2009, Duemilanove, а также Arduino Mega. Версия v3 может удобно стыковаться с Freeduino/Arduino Nano.

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.

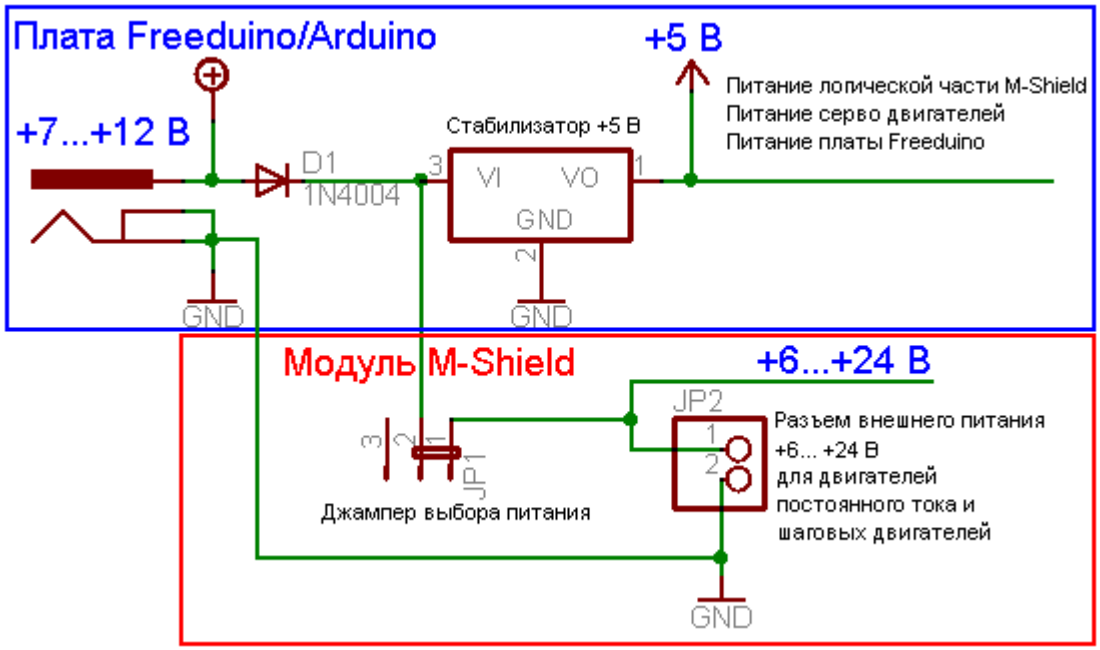
Выбор источника питания силовой части модуля осуществляется при помощи джампера питания на M-shield: либо от источника внешнего питания +7...+12 В управляющей платы Freeduino, либо от дополнительного источника постоянного напряжения +6...+ 24 В, подключаемого к двум клеммам разъема питания на M-Shield.

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.



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.

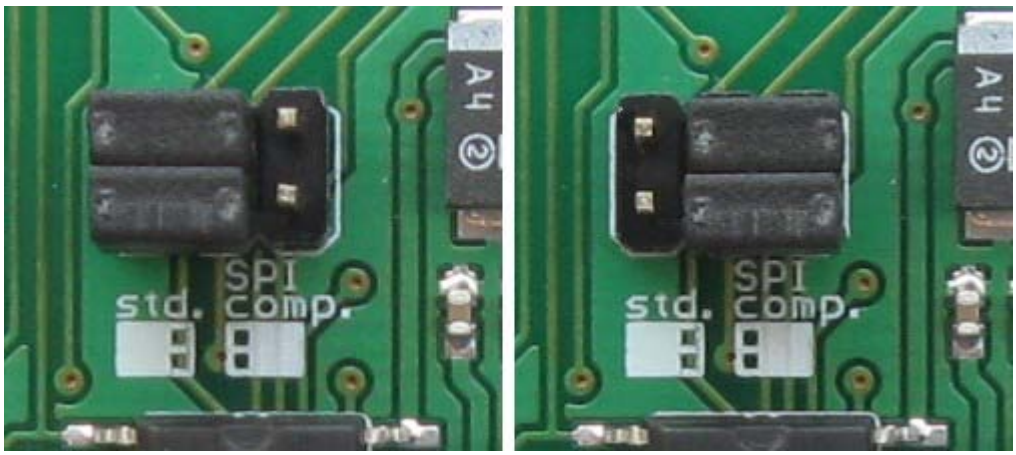
^^^

Особенности версии v3

Обновленная версия v3 модуля M-Shield в основном подобна предыдущей версии v2, но выполнена с применением поверхностного монтажа.

Первое важное отличие – возможность работы в режиме совместимости с аппаратным SPI. Оригинальный модуль от Adafruit Industries, как и версия M-Shield v2, в числе прочих задействует вывод 12 для управления микросхемой 74HCT595 и вывод 11 для ШИМ, и поэтому несовместимы с аппаратным SPI, и в частности с Ethernet Shield.

В версии v3 можно с помощью перемычек вместо вывода 12 использовать вывод 2, а вместо вывода 11 – вывод 9.

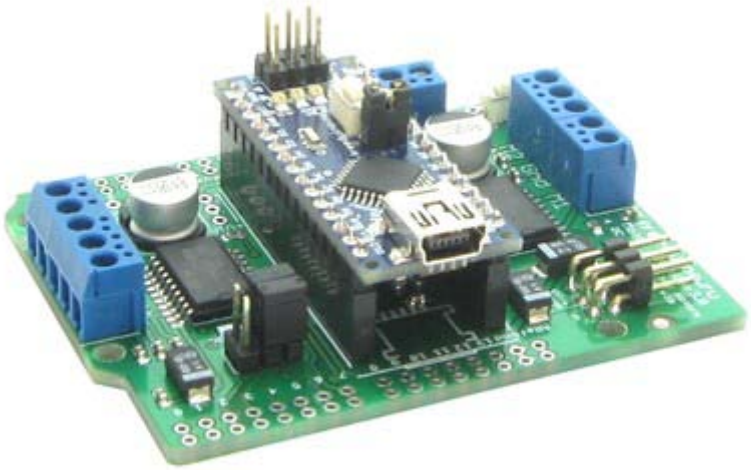


В таком режиме возможно совместное использование модуля M-Shield и Ethernet Shield, но нужно отметить, что выведенные на модуле штыревые разъемы для сервоприводов уже недоступны, т.к. один из них находится на занятом выводе 9, а второй на используемом SPI выводе 10.

Данный модифицированный режим работы требует изменений в управляющей библиотеке. Нами модифицирована библиотека AFMotor-08_12_2009, но ее работа протестирована только в режимах управления двигателями постоянного тока:

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

Второе отличие версии v3 – возможность стыковки не только с полноразмерной Arduino платой, но и с платами в формате Nano. В законченных решениях можно даже впаять плату Freeduino Nano вместо гнезд, и получить очень компактное устройство. Важно только не перепутать ориентацию платы.



Важно отметить, что модуль не может использоваться как полноценный переходник с формата Nano на полноразмерный формат Arduino, т.к. между разъемами Nano и Arduino разведены только используемые M-Shield выводы.

В комплект поставки входит M-Shield v3, разъемы Arduino Nano и разъемы «классической» Arduino. При заказе монтажа соответствующие разъемы монтируются. Если монтаж разъемов не заказывать, то при необходимости их можно будет смонтировать самостоятельно.

^^^

Библиотека 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

Для работы в SPI-совместимом режиме необходима модифицированная библиотека:

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

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

^^^

Управление двигателями постоянного тока

Для управления двигателями постоянного тока используется класс AF_DCMotor. Ниже рассмотрены его основные методы и приведены примеры работы.

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.

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.

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);
}
```

^^^

Управление шаговыми двигателями

Благодаря тому, что вращение ротора осуществляется фиксированными шагами, без необходимости обратной связи по положению, шаговые двигатели часто применяются в различных устройствах и механизмах.

Подключение шаговых двигателей

M-Shield поддерживает одновременное подключение двух шаговых двигателей. Управление шаговыми двигателями осуществляется с помощью той же библиотеки AFMotor, что и для работы с двигателями постоянного тока.

Для подключения однополярного шагового двигателя к M-shield необходимо определиться, какой из выводов двигателя подключен к его соответствующей обмотке. Если у двигателя пять выводов, то один из них является средней точкой двух его обмоток, которая подключается к электрической цепи GND модуля. Остальные выводы это начало, и конец обмоток, которые подключаются к портам M-shield: M1 и M2 или к M3 и M4.

Подключение двуполярного шагового двигателя к M-Shield производится аналогично однополярному за исключением того, что у него отсутствует вывод средней точки обмоток двигателя.

AF_Stepper – конструктор объекта

Вызов:

AF_Stepper имя_объекта(число шагов, канал);

Создает экземпляр класса AF_Stepper, принимает число шагов на один оборот двигателя и номер канала. При подключении двигателя к портам M1 и M2 указывайте 1 в качестве номера канала, при подключении к портам M3 и M4 указывайте 2.

Метод AF_Stepper::setSpeed – задание скорости

Вызов:

имя_объекта.setSpeed(скорость)

Задает частоту вращения ротора в оборотах в минуту. При указании частоты вращения больше рекомендованной для используемого двигателя с учетом напряжения питания и момента сопротивления на валу, возможно пропускание двигателем шагов.

Метод AF_DCMotor::step – вращение на нужное число шагов

Вызов:

имя_объекта.step(число шагов, направление, тип шага)

число шагов – требуемое число шагов;

направление – либо FORWARD (вперед), либо BACKWARD (назад);

тип шага – один из 4 вариантов: SINGLE, DOUBLE. INTERLEAVE или MICROSTEP.

SINGLE – активация одной обмотки двигателя для совершения шага;

DOUBLE – активация двух обмоток двигателя, что обеспечивает больший вращающий момент;

INTERLEAVE – чередование между SINGLE и DOUBLE режимом, с двойной точностью и половинной скоростью вращения.

MICROSTEP – применение ШИМ для управления шаговым двигателем двигателем.

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

Метод AF_Stepper::release – отключение двигателя

Вызов:

имя_объекта.release ()

По умолчанию шаговый двигатель удерживает свое положение после завершения шага, однако вызов release() отключает обмотки двигателя для осуществления свободного вращения ротора.

^^^

Пример управления шаговым двигателем

Ниже приведен пример простой программы, осуществляющей управление шаговым двигателем ST28, подключенным к портам M3 и M4 модуля.

У шагового двигателя ST28 32 шага на оборот, а также встроенный редуктор с передаточным числом 64, что дает 32*64=2048 шагов двигателя на один оборот выходного вала. В приведенном примере осуществляется поворот на один оборот в прямом направлении в режиме SINGLE, на один оборот в обратном в режиме DOUBLE, на пол-оборота в прямом направлении в режиме INTERLEAVE, и на пол-оборота в обратном в режиме MICROSTEP.

```
#include <AFMotor.h>  //Подключаем заголовочный файл библиотеки
//Создаем объект для двигателя на 2 канале (M3 и M4)
AF_Stepper motor(2048, 2);
void setup() {
  motor.setSpeed(10);  // 10 оборотов в минуту
}

void loop() {
  motor.step(2048, FORWARD, SINGLE);     //1 оборот
  motor.step(2048, BACKWARD, DOUBLE);    //1 оборот
  //в режиме INTERLEAVE скорость в 2 раза ниже, а шаги в
  //2 раза меньше. То же число полушагов даст 0.5 оборота
  motor.step(2048, FORWARD, INTERLEAVE); //0.5 оборота
  motor.step(1024, BACKWARD, MICROSTEP); //0.5 оборота
  motor.release();
  delay(1000);
}
```

...to be continued. Планируется расширение документации примерами работы с сервоприводами