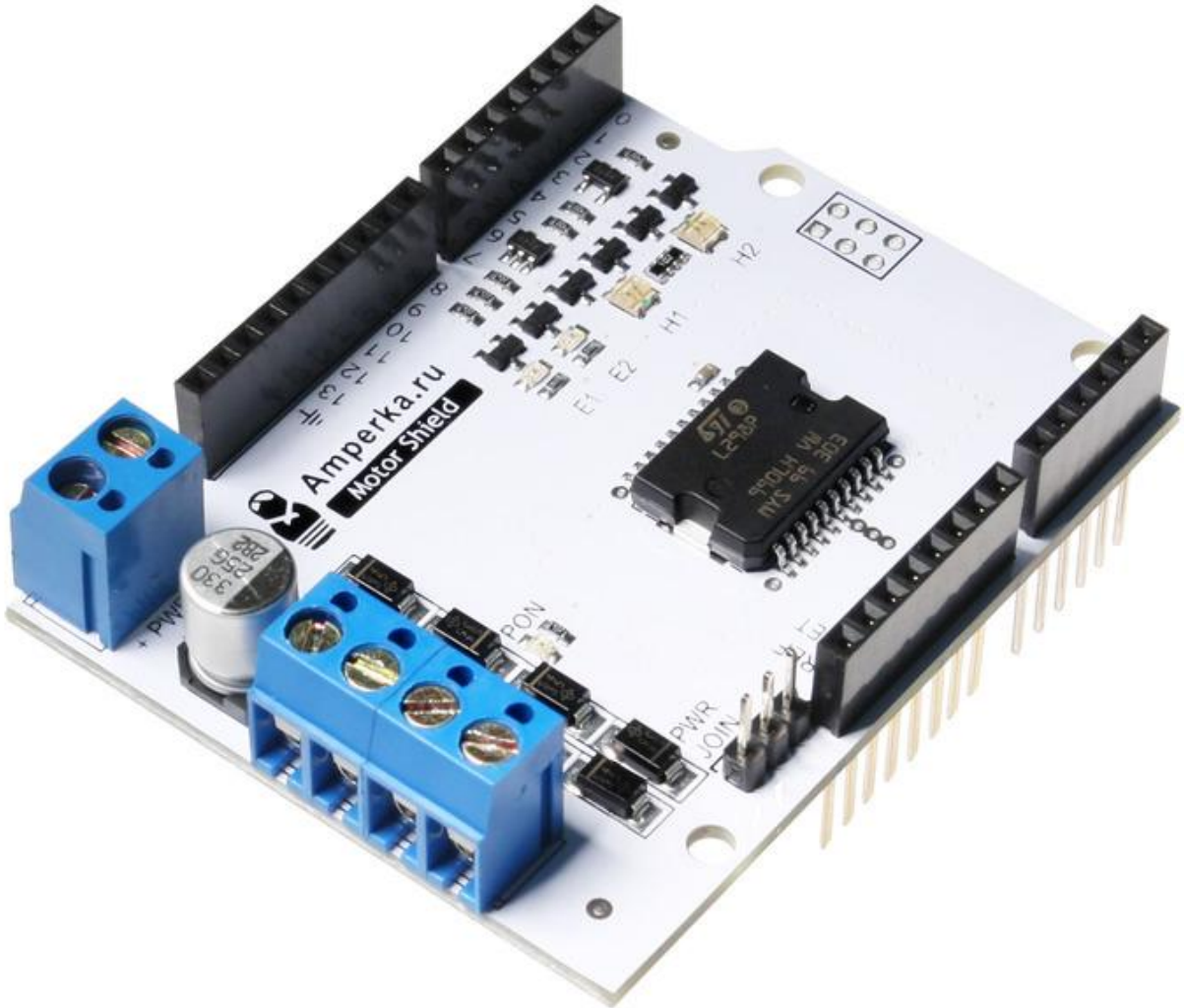
Motor Shield

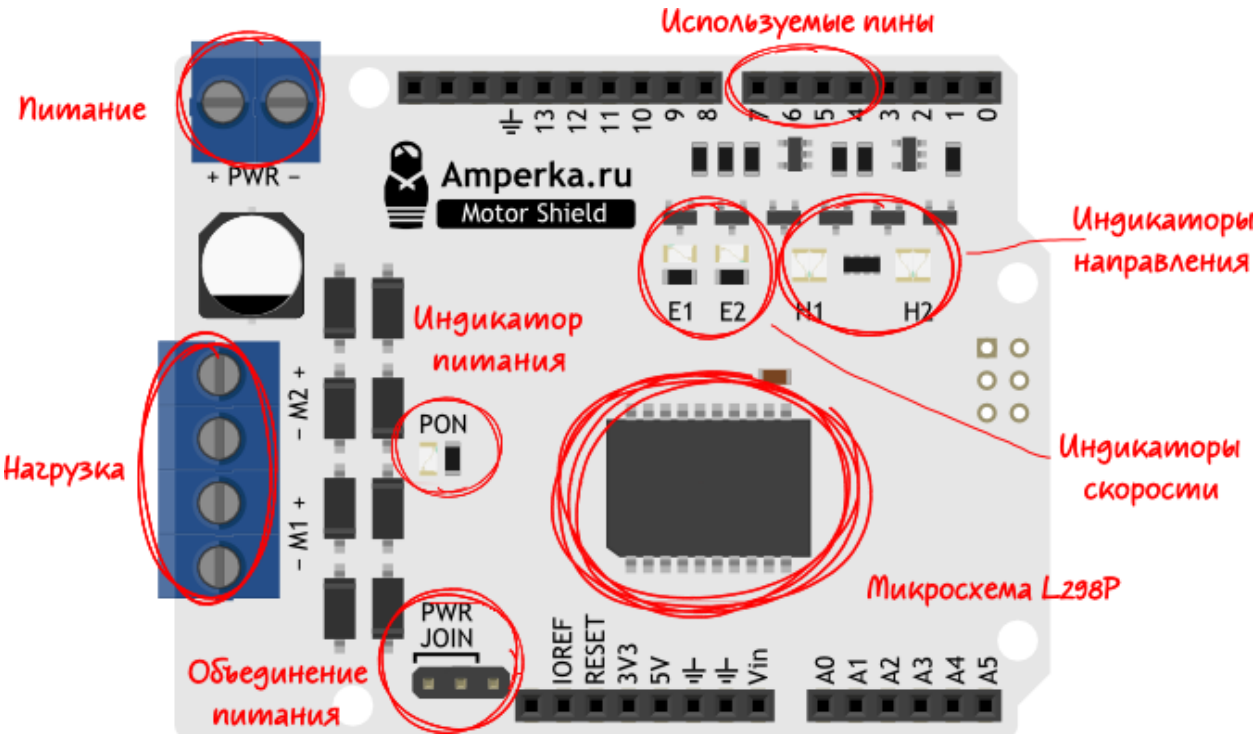
Микроконтроллер, установленный на Arduino не может непосредственно управлять большой нагрузкой на своих цифровых выходах. Максимально возможный выходной ток с ножки микроконтроллера - 40мА. Чтобы иметь возможность управлять большой нагрузкой существуют специализированные устройства. Именно к таким устройствам и относится Motor Shield.

Motor Shield - это плата расширения для Arduino, предназначенная для двухканального управления скоростью и направлением вращения коллекторных двигателей постоянного тока, напряжением 5-24В и максимальным током до 2А на канал.

Motor Shield также может быть использован для управления одним биполярным шаговым двигателем или, при совмещении двух каналов в один, для управления мощной нагрузкой с током до 4А.



Элементы платы



Питание

Клеммники под винт для подключения питания на плате обозначены как **PWR**. К ним подключается источник питания, который будет использоваться для питания моторов. Напряжение питания должно быть в пределах 5-24В постоянного тока.

Внимание!

При подключении питания соблюдайте полярность. Неправильное подключение может привести к непредсказуемому поведению или выходу из строя платы или источника питания.

При установке платы над Arduino Uno, или другой платой, обладающей высоким разъёмом вроде USB Type B или RJ45, наклейте на разъём пару слоёв изоленты, чтобы избежать замыкания дорожек на нижней стороне платы.

На Motor Shield два контура питания.

- Первый — *силовой*, напряжение на который приходит с клеммника PWR. От этого контура запитана микросхема H-моста L298P и нагрузка.
- Второй контур используется для питания вспомогательной *цифровой логики* управления микросхемой L298P и светодиодов индикации. Этот контур получает питание от пина 5V Arduino. Если по какой-то причине напряжения на этом пине не оказалось, или напряжение на нём оказалось ниже 5 В, Motor Shield работать не будет.

Индикация питания

При правильном подключении питания силового контура Motor Shield, загорается светодиод индикации питания **PON**. Если полярность питания перепутана, или питание по какой-то причине не подано, светодиод гореть не будет.

Из-за большой ёмкости фильтрующего конденсатора, установленного на плате, светодиод PON в некоторых случаях может кратковременно продолжать гореть и после отключения питания.

Нагрузка

Клеммники под винт для подключения нагрузки на плате обозначены как **-M1+ -M2+**.

Нагрузка разделена на 2 канала. К одному каналу можно подключить один коллекторный мотор. Первый канал на плате имеет обозначение «M1», второй канал имеет обозначение «M2». Каждый канал управляется независимо.

Обозначения «+» и «-» показывают воображаемые начало и конец обмотки: если подключить два коллекторных двигателя таким образом, чтобы их одноимённые контакты щёточного узла соответствовали одному и тому же обозначению на плате, то, при подаче на плату одинаковых управляющих импульсов, моторы будут вращаться в одну и ту же сторону.

Объединение питания

Штырьки с джампером для объединения питания Arduino и Motor Shield на плате обозначены как **PWR JOIN**.

Когда джампер находится в положении PWR JOIN, происходит объединение контакта Vin Arduino и положительного контакта клеммника PWR.

Этот режим используется для обеспечения питания и Arduino и силовой нагрузки от *одного источника питания*. При этом питание может быть подано как на клеммник PWR Motor Shield, так и на разъём питания Arduino. Выбор для питания клеммника PWR предпочтительнее, так как при работе двигателей по цепи питания может проходить очень большой ток, на который цепь Vin Arduino не рассчитана.

В режиме совместного питания Arduino и Motor Shield, рекомендуемое напряжение питания — 7–12 В.

Третий штырёк нужен для хранения джампера при работе с раздельным питанием. Так он не потеряется.

Важно!

Не все источники питания подходят для режима работы с объединённым питанием Arduino и Motor Shield.

Источники питания должны быть способны обеспечить стабильное напряжение при резких скачках нагрузки. Даже кратковременная просадка напряжения при броске нагрузки приведёт к перезагрузке управляющего контроллера Arduino, и

связанным с этим неадекватным поведением двигателей. Этому требованию соответствуют только **литий-ионные** и **никель-металлгидридные** аккумуляторы или лабораторные блоки питания.

Если вы используете другие источники питания, лучше всего воспользоваться раздельной схемой питания Arduino и Motor Shield.

Используемые пины

Motor Shield использует пины номер 4, 5, 6 и 7 Arduino.

Назначение	Канал 1	Канал 2
Скорость	5	6
Направление	4	7

Пины управления скоростью вращения двигателей

Управление скоростью происходит при помощи ШИМ, за счёт быстрого включения и выключения нагрузки. Напряжение на нагрузку подаётся при высоком логическом уровне на пине скорости, поэтому можно запустить двигатель на полную скорость просто выставив на этом пине логическую единицу. Или выключить двигатель, выставив на этом пине логический ноль.

Индикация скорости

Светодиоды индикации скорости на плате обозначены как **E1, E2**.

- E1 показывает состояние пина скорости канала M1
- E2 показывает состояние пина скорости канала M2.

Чем выше скорость вращения двигателя, тем ярче горит светодиод индикации скорости.

Пины управления направлением вращения двигателей

Пины направления отвечают за направление вращения двигателей. Смена направления вращения коллекторных двигателей достигается за счёт изменения полярности приложенного к ним напряжения.

- Если выставить на пин направления логическую единицу, то полярность напряжения на клеммниках нагрузки будет соответствовать обозначению «+» и «-» на плате.
- При подаче на пин логического нуля, напряжение на клеммнике изменится на противоположное.

Индикация направления

Светодиоды индикации направления вращения на плате обозначены как **H1, H2**.

- H1 показывает состояние пина направления канала M1
- H2 показывает состояние пина направления канала M2.

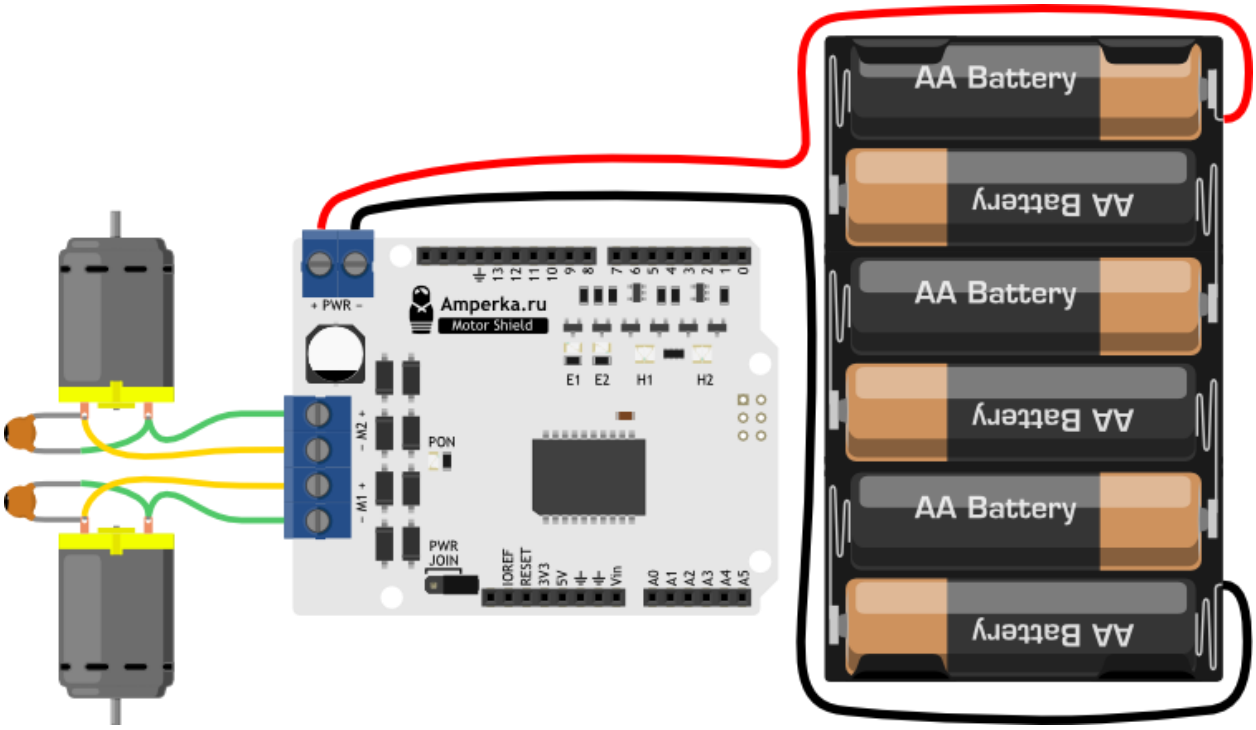
При высоком логическом уровне на пине управления направлением вращения, т.е. вращении вперёд, индикатор светится зелёным светом. При низком уровне, т.е. при реверсе — красным.

Характеристики

Параметр	Мин.	Номинал.	Макс.	Ед. изм.
Напряжение питания силовой части	4,8	—	24	В
Напряжение питания силовой части с PWR JOIN	7	—	12	В
Напряжение питания логической части	4,5	5	7	В
Допустимый уровень управляющих сигналов	3	5	7	В
Продолжительный ток на канал *			2	А
Пиковый прерывистый ток на канал (80% включение, до 10 мс)			2,5	А
Пиковый непрерывный ток на канал (до 100 мкс)			3	А

* для достижения максимума необходимо дополнительное охлаждение чипа L298P

Управление коллекторными двигателями



Приступим к демонстрации возможностей. Схема подключения — на картинке. Джампер PWR JOIN установлен в положение «Раздельное питание». Arduino запитана через USB.

Для начала простейший код для демонстрации всех основных функций Motor Shield:

2DC_Motor_Shield.ino

```
// Моторы подключаются к клеммам M1+, M1-, M2+, M2-
// Motor shield использует четыре контакта 4, 5, 6, 7 для управления моторами
// 4 и 7 — для направления, 5 и 6 — для скорости
#define SPEED_1      5
#define DIR_1        4

#define SPEED_2      6
#define DIR_2        7


void setup()
{
    // Настраивает выводы платы 4, 5, 6, 7 на вывод сигналов
    for(int i = 4; i < 8; i++)
        pinMode(i, OUTPUT);
}

void loop()
{
  // Для коллекторного мотора можно выбрать значение скорости от 0 до 255.
  // Покрутим в течении секунды M1 на средней скорости сначала в одну сторону...
  analogWrite(SPEED_1, 126);
  digitalWrite(DIR_1, LOW);
  delay(1000);

  // ... а затем в другую.
  digitalWrite(DIR_1, HIGH);
  delay(1000);

  // После чего остановим мотор 1
  analogWrite(SPEED_1, 0);

  // А теперь заставим мотор 2 медленно разгоняться до максимума
  for (int i=0; i <= 255; ++i)
  {
    analogWrite(SPEED_2, i);
    delay(50);
  }

  // Теперь он будет крутится до нажатия на Reset или выключения питания
  while (true)
    ;
}
```

Следующий код подойдёт для двухколёсной платформы:

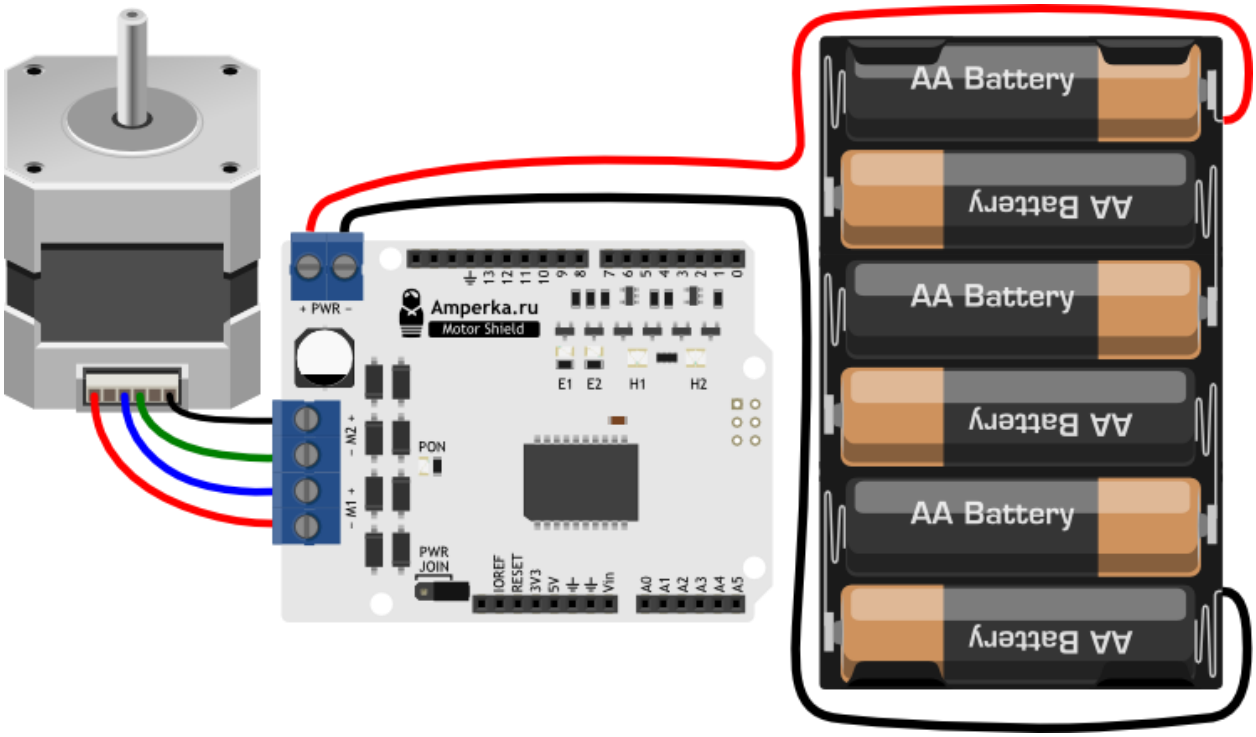
2DC_Motor_Shield_Platform.ino

```
// Моторы подключаются к клеммам M1+, M1-, M2+, M2-
// Если полюса моторов окажутся перепутаны при подключении,
// можно изменить соответствующие константы CON_MOTOR с 0 на 1
#define CON_MOTOR1     0
#define CON_MOTOR2     0
```

```
// Motor shield использует четыре контакта 4, 5, 6, 7 для управления
моторами
// 4 и 7 — для направления, 5 и 6 — для скорости
#define SPEED_1      5
#define DIR_1        4

#define SPEED_2      6
#define DIR_2        7

// Возможные направления движения робота
#define FORWARD   0
#define BACKWARD  1
#define LEFT      2
#define RIGHT     3

/*
 * В функции `go` мы управляем направлением движения и скоростью
 */
void go(int newDirection, int speed)
{
  boolean motorDirection_1, motorDirection_2;

  switch ( newDirection ) {

    case FORWARD:
        motorDirection_1 = true;
        motorDirection_2 = true;
        break;
    case BACKWARD:
        motorDirection_1 = false;
        motorDirection_2 = false;
        break;
    case LEFT:
        motorDirection_1 = true;
        motorDirection_2 = false;
        break;
    case RIGHT:
        motorDirection_1 = false;
        motorDirection_2 = true;
        break;
  }

  // Если мы ошиблись с подключением - меняем направление на обратное
  motorDirection_1 = CON_MOTOR1 ^ motorDirection_1;
  motorDirection_2 = CON_MOTOR2 ^ motorDirection_2;

  // Поехали! Скорость может меняться в пределах от 0 до 255.
  analogWrite(SPEED_1, speed);
  analogWrite(SPEED_2, speed);

  digitalWrite(DIR_1, motorDirection_1);
  digitalWrite(DIR_2, motorDirection_2);
}

void setup()
{
    // Настраивает выводы платы 4, 5, 6, 7 на вывод сигналов
    for(int i = 4; i <= 7; i++)
        pinMode(i, OUTPUT);
}

void loop()
{
    // Задержка 5 секунд после включения питания
```

```
    delay(5000);

    // Медленный разгон до максимальной скорсти
    for (int i=50; i<=250; ++i)
    {
      go(FORWARD, i);
      delay(30);
    }

    // Едем секунду вперёд на максимальной скорости
    go(FORWARD, 255);
    delay(1000);

    // быстро крутимся влево полторы секунды
    go(LEFT, 250);
    delay(1500);

    // медленно едем назад полторы секунды
    go(BACKWARD, 70);
    delay(1500);

    // медленно крутимся вправо полторы секунды
    go(RIGHT, 80);
    delay(1500);

    // Остановка. Скорость равна нулю
    go(FORWARD, 0);

    // Всё, приехали. Стоим до нажатия Reset или отключения питания
    while (true)
      ;
}
```

Управление биполярным шаговым двигателем



Управление в разных режимах без библиотек

Существует несколько режимов управления биполярным шаговым двигателем при помощи Motor Shield:

- Однофазный полношаговый (wave drive mode)
- Двухфазный полношаговый (full step mode)
- Полушаговый (half step mode)

Разницу между ними можно понять, протестировав на Motor Shield следующий скетч:

StepMotor_mode.ino

```
//Определяем пины для управления Motor Shield
#define E1    5
#define E2    6
#define H1    4
#define H2    7

// Три режима управления шаговым мотором
#define WAVE_DRIVE  0 // Однофазный режим
#define FULL_STEP   1 // Двухфазный режим
#define HALF_STEP   2 // Полушаговый режим

// Задержка между переключением обмоток. Определяет скорость вращения
int delayTime = 10;

unsigned long startTime;

// Эта функция непосредственно выставляет значение на пинах
void doStep(boolean E1State, boolean H1State, boolean E2State, boolean
H2State)
{
  digitalWrite(E1, E1State);
  digitalWrite(H1, H1State);
  digitalWrite(E2, E2State);
  digitalWrite(H2, H2State);
  delay(delayTime);
}

// Здесь определяются комбинации управляющих импульсов в зависимости от
режима
void rotate(byte rotateMode)
{
  switch (rotateMode)
  {
    case WAVE_DRIVE :
      doStep(1, 1, 0, 0);
      doStep(0, 0, 1, 1);
      doStep(1, 0, 0, 0);
      doStep(0, 0, 1, 0);
      break;

    case FULL_STEP :
      doStep(1, 1, 1, 1);
      doStep(1, 0, 1, 1);
      doStep(1, 0, 1, 0);
      doStep(1, 1, 1, 0);
      break;

    case HALF_STEP :
```

```
      doStep(1, 1, 0, 0);
      doStep(1, 1, 1, 1);
      doStep(0, 0, 1, 1);
      doStep(1, 0, 1, 1);

      doStep(1, 0, 0, 0);
      doStep(1, 0, 1, 0);
      doStep(0, 0, 1, 0);
      doStep(1, 1, 1, 0);
      break;
  }
}

void setup()
{
  // Настраиваем ножки на выход
  for (int i = 4; i < 8; ++i)
    pinMode(i, OUTPUT);

  startTime = millis();
}

void loop()
{
  // Узнаём время прошедшее с начала работы в миллисекундах
  unsigned long loopTime = millis() - startTime;

  // Переведём его в секунды
  unsigned int timeInSecond = loopTime / 1000;

  // Шагаем! Каждую секунду режим управления будет меняться.
  rotate(timeInSecond % 3);
}
```

Скорость вращения шагового двигателя очень сильно влияет на развиваемый двигателем момент. Чтобы в этом убедиться, запустите этот скетч с разными значениями `delayTime`. Обратите внимание, что двигатель в однофазном полношаговом режиме позволяет развить гораздо меньший момент, чем в двухфазном полношаговом режиме.

Управление через готовую библиотеку

Легко управлять шаговым двигателем в своём проекте вам поможет библиотека StepperAmperka. С помощью неё можно управлять биполярным шаговым двигателем в однофазном, в двухфазном или полушаговом режиме.

Пример работы в различных режимах:

StepperAmperka_example.ino

```
#include <StepperAmperka.h>

// Параметр конструктора — количество шагов на 1 оборот.
// Фикированная характеристика используемого шагового двигателя
StepperAmperka motor = StepperAmperka(200);

// Если использовать Motor Shield на нестандартных пинах,
// конструктор будет выглядеть иначе.
// Например, для пинов 8, 9, 10, 11 конструктор будет таким:
// StepperAmperka motor = StepperAmperka(200, 8, 9, 10, 11);
```

```
void setup()
{
  // Устанавливаем скорость вращения 30 оборотов в минуту.
  motor.setSpeed(30);
}

void loop()
{
  // 180° по часовой стрелке в двуфазном режиме
  motor.step(100, FULL_STEP);
  delay(1000);

  // 180° против часовой стрелки в однофазном режиме
  motor.step(-100, WAVE_DRIVE);
  delay(1000);

  // 180° по часовой стрелке в полушаговом режиме
  motor.step(200, HALF_STEP);
  delay(1000);

  // 180° против часовой стрелки в двуфазном режиме
  // этот режим используется по умолчанию, если не передан
  // второй аргумент
  motor.step(-100);
  delay(1000);
}
```