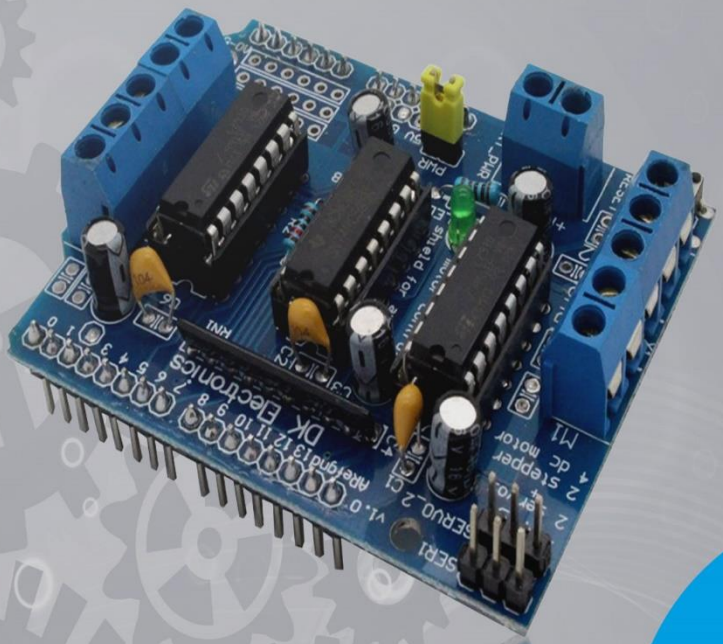
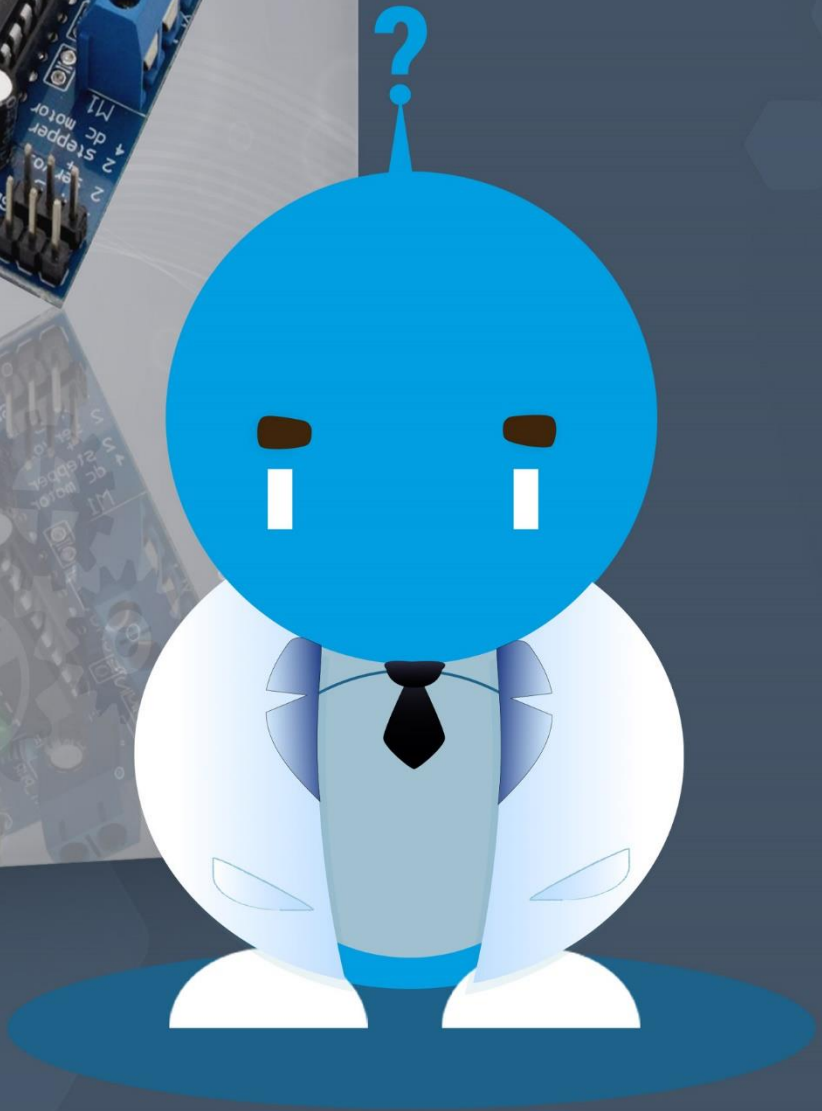
ROBOTER
Bausatz-Shop

BEDIENUNGSANLEITUNG

L293D Motortreiber Shield für Arduino





Inhaltsverzeichnis

Sicherheitshinweise

- Maximale Betriebsspannung von 36V nicht überschreiten
- Maximale Stromstärke von 600mA (1,2A Spitzenstrom) pro Kanal nicht überschreiten.
- Bei Verwendung einer externen Stromversorgung für die Motoren unbedingt den PWR-Jumper entfernen, um Schäden am Shield und Arduino zu vermeiden
- Auf ausreichende Kühlung achten, besonders bei hoher Last oder längerem Betrieb.
- Kurzschlüsse an den Motorausgängen vermeiden.
- Motoren nicht im Leerlauf mit voller Geschwindigkeit betreiben.
- Kabel und Anschlüsse regelmäßig auf festen Sitz und Beschädigungen prüfen.
- Shield nur in trockener Umgebung verwenden und vor Feuchtigkeit schützen.
- Bei ungewöhnlichen Geräuschen oder Gerüchen sofort die Stromversorgung trennen.
- Vorsicht bei rotierenden Teilen - Verletzungsgefahr!
- Außerhalb der Reichweite von Kindern aufbewahren.
- Bei Nichtgebrauch oder Wartungsarbeiten immer die Stromversorgung trennen.

Beschreibung

Arduino kompatibles Motorschield mit 4-Kanal L293D Schrittmotortreiber

Dieses Shield wird einfach auf ein Arduinoboard aufgesteckt und erlaubt es bis zu 4 DC-Motoren, 2 Schrittmotoren oder 2 Servomotoren anzusteuern. Als H-Brücke dient der leistungsfähige und zuverlässige L293D Chip, der die Last verteilt. Dadurch können Sie problemlos DC-Motoren und Netzteile bis 36V verwenden.

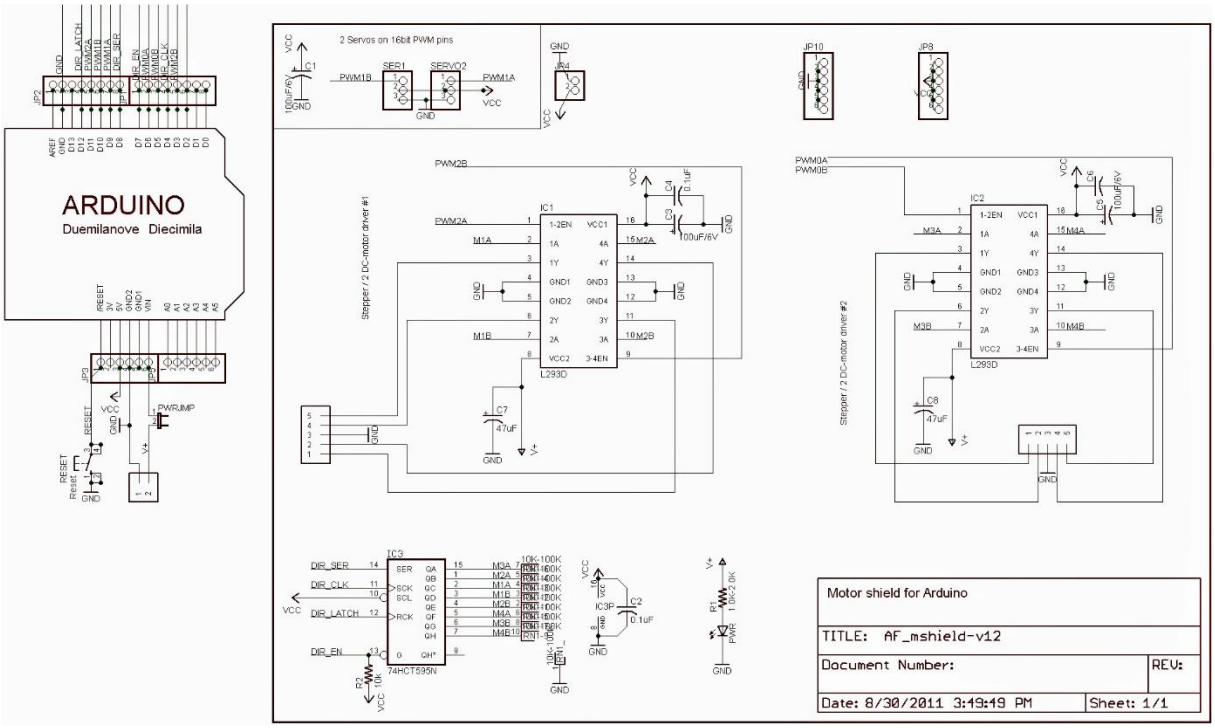
Das bewährte Design erlaubt das einfache Anschließen und Steuern von Motoren durch einen Arduino und eignet sich vor allem für schnelles Protoyping. Auch für Einsteiger ist dieses Shield bestens geeignet, so existieren zahlreiche Bibliotheken, Anleitungen und Beispiel-Sketches die das Steuern der Richtung oder Geschwindigkeit von Motoren zum Kinderspiel machen.

Für besonders empfehlenswert halten wir die Adafruit Motor Shield Library (AFM), die Sie standardmäßig in der Arduino IDE herunterladen können.

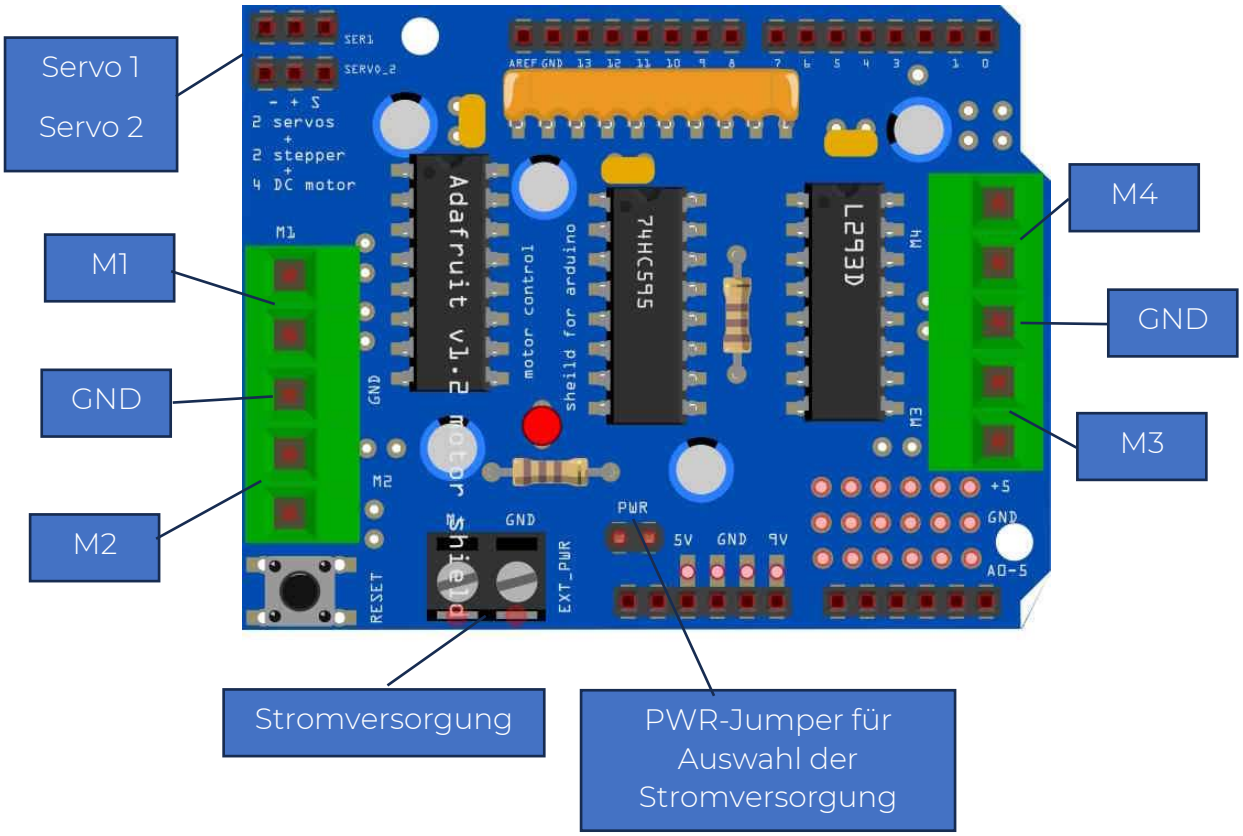
Details

- Beschriftete Anschlüsse zum einfachen Verbinden
- Kompatibel zu Arduino Mega, Diecimila, Duemilanove und Uno R3
- 2 Anschlüsse für 5V Servomotoren mit Anbindung an den Arduino Timer zur ruckelfreien Steuerung
- Zum Steuern von 4 DC-Motoren, 2 Schrittmotoren oder 2 Servomotoren.
- Bis zu 4 bidirektionale DC-Motoren mit individueller 8-Bit Steuerung
- Bis zu 2 Schrittmotoren (unipolar oder bipolar) mit single coil, double coil oder interleaved stepping
- 4 H-Brücken: 0.6A (1.2A Spitzen) mit Thermalschutz für Motoren von 4.5V bis 36V DC
- Pull-Down Widerstände, um die Motoren beim Anschalten anzuhalten
- 2 Anschlüsse für externe Stromversorgung, getrennt für Logik- und Motor-Versorgung
- Status-LED zur Betriebsanzeige
- Reset button
- Abmessungen:70*55mm

Schaltplan



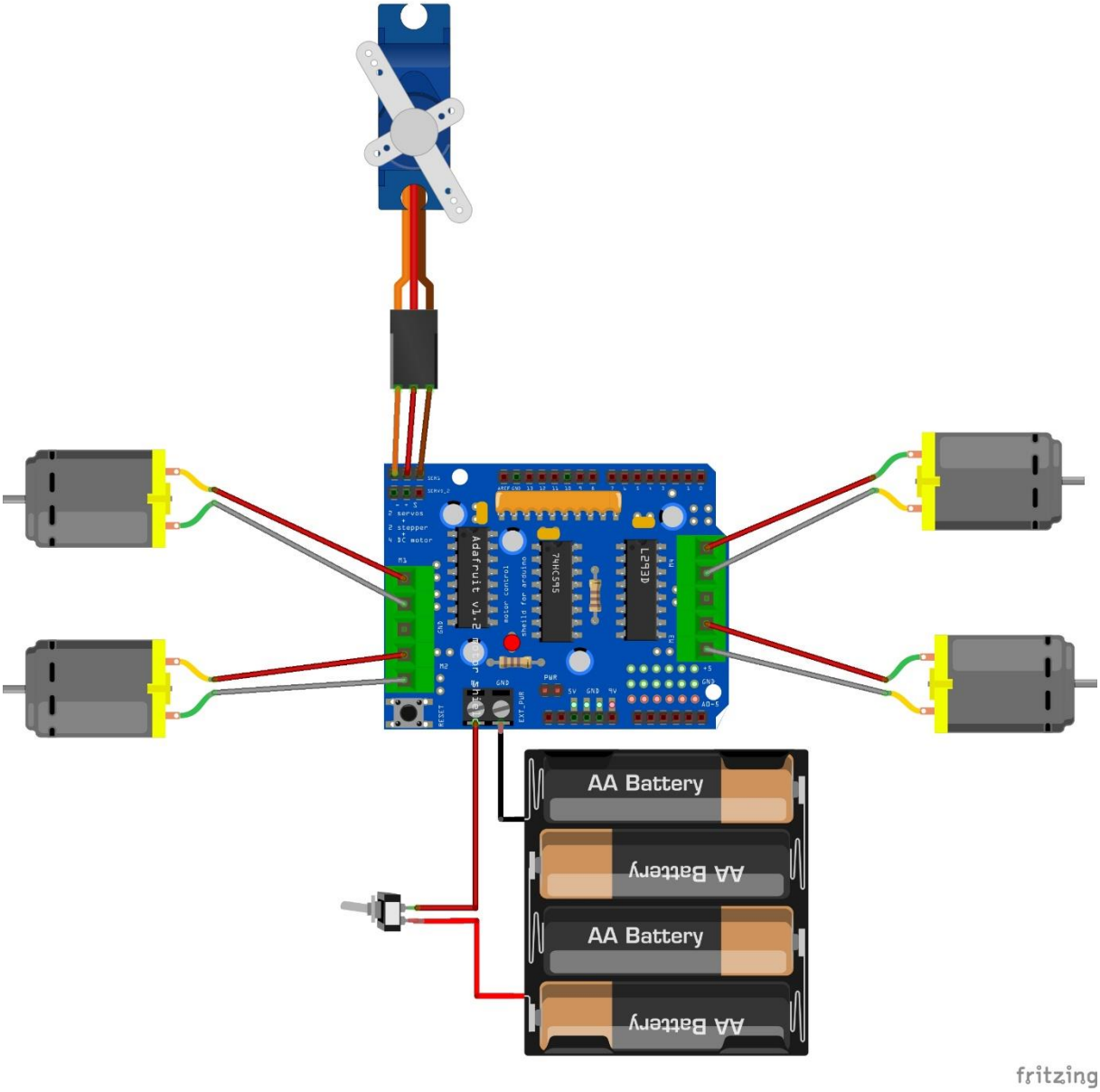
Pinbelegung



Verwendete Arduino Pins

- D3: Motor 2 / Schrittmotor 1
- D5: Motor 3 / Schrittmotor 2
- D6: Motor 4 / Schrittmotor 2
- D11: Motor 1 / Schrittmotor 1
- D9 Servomotor 1
- D10 Servomotor 2#
- Pin D4, D7, D8, D12 für Steuerung über den 74HC595

Anschlussplan



Typischer Anschluss mit externer Stromversorgung

Stromversorgung

DC-Motoren benötigen eine separate Stromversorgung, da sie oft hohe Ströme ziehen. Es ist wichtig zu beachten:

- Niemals DC-Motoren direkt an die 5V-Pins des Arduino-Boards anschließen.
- Eine direkte Verbindung könnte das Arduino-Board oder den USB-Port beschädigen.
- Bei Verwendung einer externen Stromversorgung für die Motoren unbedingt den PWR-Jumper entfernen, um Schäden am Shield und Arduino zu vermeiden!

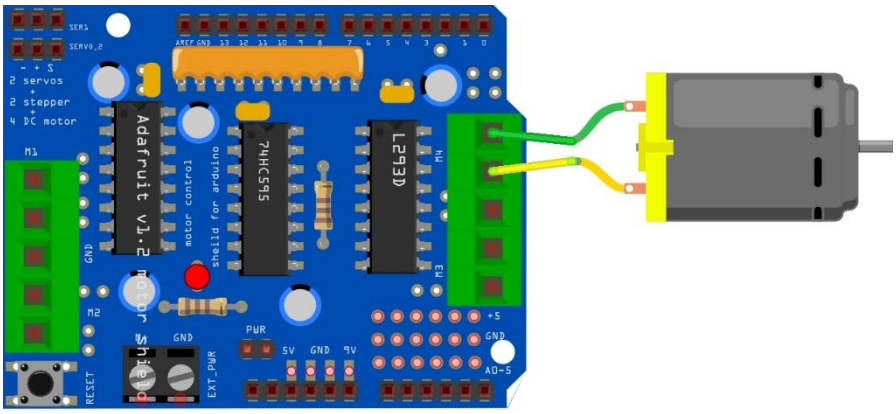
Optionen für die Stromversorgung

Es gibt zwei sichere Möglichkeiten, die Stromversorgung für die DC-Motoren anzuschließen:

1. PWR-Jumper gesteckt:
   - Die Motoren werden über den DC-Anschluss des Arduino-Boards mit Strom versorgt.
   - Arduino und Motoren teilen sich eine gemeinsame Stromquelle.
   - Diese Konfiguration ist nur für Motorspannungen unter 12V geeignet.
2. PWR-Jumper entfernt:
   - Die Motorstromversorgung wird vom Arduino getrennt.
   - Eine separate Stromquelle muss an den EXT_PWR-Anschluss des Shields angeschlossen werden.
   - Ermöglicht die Verwendung höherer Spannungen für die Motoren (bis zu 36V).

Beispiel mit DC-Motor

Für diesen Code wird die Adafruit Motor Shield V1 Library benötigt. Bei diesem Code wird ein Motor an Anschluss M4 in eine Richtung beschleunigt und anschließend bis zum Stillstand abgebremst. Nach einer Umdrehung kehrt der Motor seine Drehrichtung um und Widerholt den Vorgang.




```
#include <AFMotor.h>

// Motor definieren 1 fuer M1, 2 fuer M2 usw.
AF_DCMotor motor(4); // Motor an M4 angeschlossen


void setup()
{
  //Initiale Motorgeschwindigkeit einstellen
  motor.setSpeed(200);
  motor.run(RELEASE);
}
void loop()
{
  uint8_t i;
```

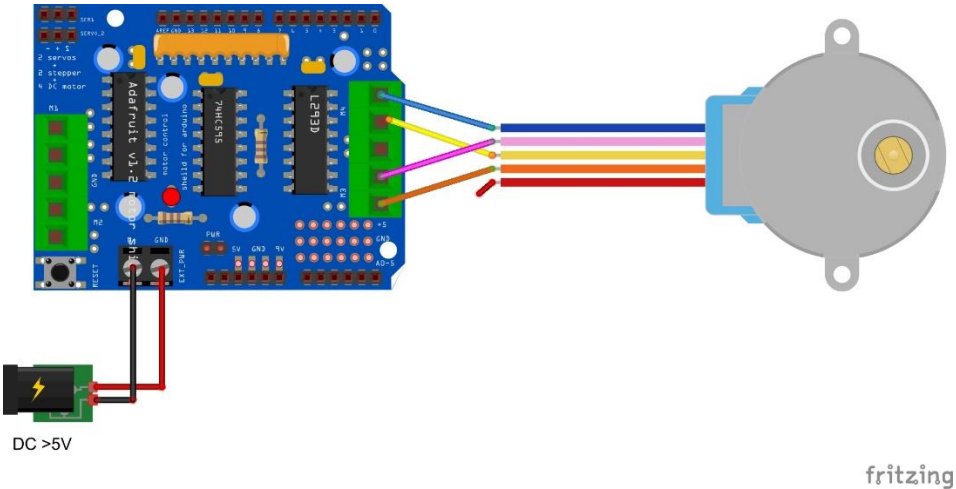
```
  // Motor einschalten
  motor.run(FORWARD);
  // Beschleunige von 0 bis maximale Geschwindigkeit
  for (i=0; i<255; i++)
  {
    motor.setSpeed(i);
    delay(10);
  }
  // Bremse von maximaler Geschwindigkeit auf 0
  for (i=255; i!=0; i--)
  {
    motor.setSpeed(i);
    delay(10);
  }

  // Drehrichtung aendern
  motor.run(BACKWARD);
  // Beschleunige von 0 bis maximale Geschwindigkeit
  for (i=0; i<255; i++)
  {
    motor.setSpeed(i);
    delay(10);
  }
  // Bremse von maximaler Geschwindigkeit auf 0
  for (i=255; i!=0; i--)
  {
    motor.setSpeed(i);
    delay(10);
  }
  // Motor ausschalten
  motor.run(RELEASE);
  delay(1000);
}
```

Beispiele mit Schrittmotoren

Unipolarer Schrittmotor

In diesem Beispiel verwenden wir einen 28BY-48 Schrittmotor mit 48 Schritten pro Umdrehung. Die Schrittmotoren können mit Anschluss #1 (M1 und M2) oder Anschluss #2 (M3 und M4) verbunden werden.



```
#include <AFMotor.h>

// Anschluss eines Schrittmotors mit 48 Schritte pro Umdrehung (7.5 grad)
// An Motoranschluss #2 (M3 und M4)
AF_Stepper motor(48, 2);

void setup() {
  Serial.begin(9600);
  Serial.println("Stepper test!");

  motor.setSpeed(10);  // Geschwindigkeit auf 10 rpm
}

void loop() {
  Serial.println("Single coil steps");
  motor.step(100, FORWARD, SINGLE);
  motor.step(100, BACKWARD, SINGLE);

  Serial.println("Double coil steps");
  motor.step(100, FORWARD, DOUBLE);
  motor.step(100, BACKWARD, DOUBLE);

  Serial.println("Interleave coil steps");
  motor.step(100, FORWARD, INTERLEAVE);
  motor.step(100, BACKWARD, INTERLEAVE);

  Serial.println("Micrsostep steps");
  motor.step(100, FORWARD, MICROSTEP);
  motor.step(100, BACKWARD, MICROSTEP);
}
```

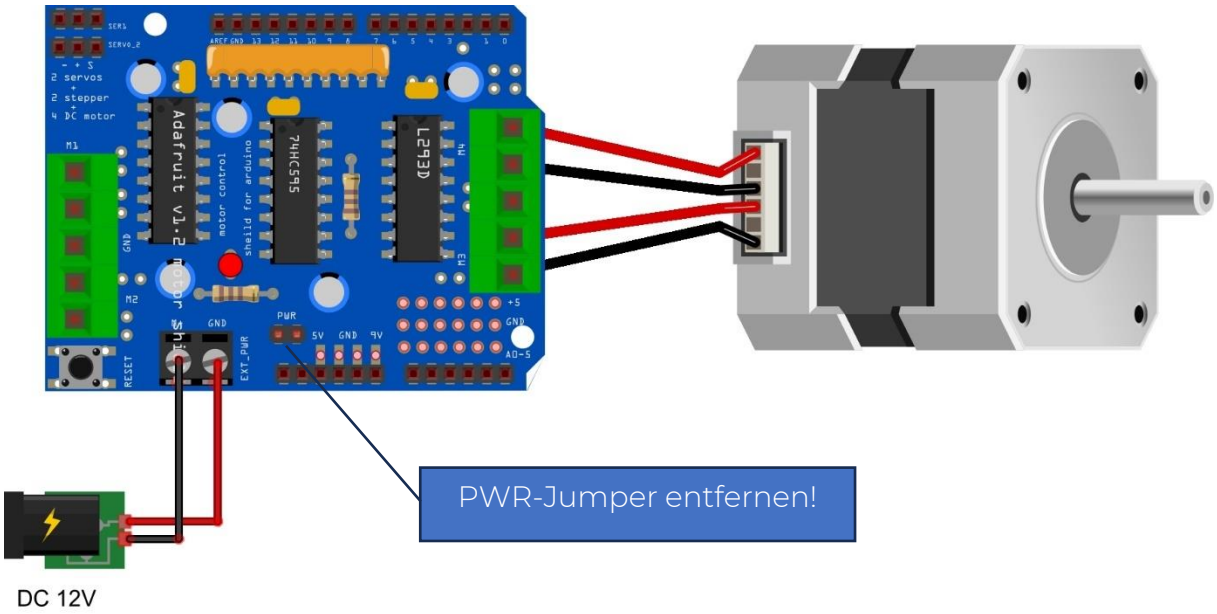
Die Funktion step(#steps, direction, steptype) wird jedes Mal aufgerufen, wenn du den Motor bewegen möchtest. #steps gibt die Anzahl der Schritte an, die der Motor ausführen soll. Die Richtung kann entweder FORWARD (vorwärts) oder BACKWARD (rückwärts) sein, und der Schrittstil (stepstyle) kann einer der folgenden sein:

- SINGLE – Eine Spule wird jeweils einzeln aktiviert.
- DOUBLE – Zwei Spulen werden gleichzeitig aktiviert, was mehr Drehmoment liefert.
- INTERLEAVE – Es wird zwischen SINGLE und DOUBLE abwechselnd, wodurch ein "Halbschritt" entsteht. Dies sorgt für einen gleichmäßigeren Betrieb, verringert aber die Geschwindigkeit um die Hälfte aufgrund der zusätzlichen Halbschritte.
- MICROSTEP – Zwischen jedem Vollschritt werden benachbarte Spulen stufenweise auf- und abgeregelt, um mehrere Mikro-Schritte zu erzeugen.

Dies führt zu einer feineren Auflösung und sanfterer Drehung, allerdings auf Kosten des Drehmoments.

Bipolarer Schrittmotor

In diesem Beispiel verwenden wir einen Nema 17 Schrittmotor mit 200 Schritten pro Umdrehung (1.8 Grad). Der Schrittmotor wird an M3 und M4 angeschlossen. Der Code ist gleiche wie im vorherigen Beispiel, aber mit 200 Schritten pro Umdrehungen eingestellt.




```
#include <AFMotor.h>

// Anschluss eines Schrittmotors mit 200 Schritte pro Umdrehung (1.8 grad)
// An Motoranschluss #2 (M3 und M4)
AF_Stepper motor(200, 2);

void setup() {
  Serial.begin(9600);
  Serial.println("Stepper test!");

  motor.setSpeed(10);  // Geschwindigkeit auf 10 rpm
}

void loop() {
  Serial.println("Single coil steps");
  motor.step(100, FORWARD, SINGLE);
  motor.step(100, BACKWARD, SINGLE);

  Serial.println("Double coil steps");
  motor.step(100, FORWARD, DOUBLE);
  motor.step(100, BACKWARD, DOUBLE);

  Serial.println("Interleave coil steps");
```

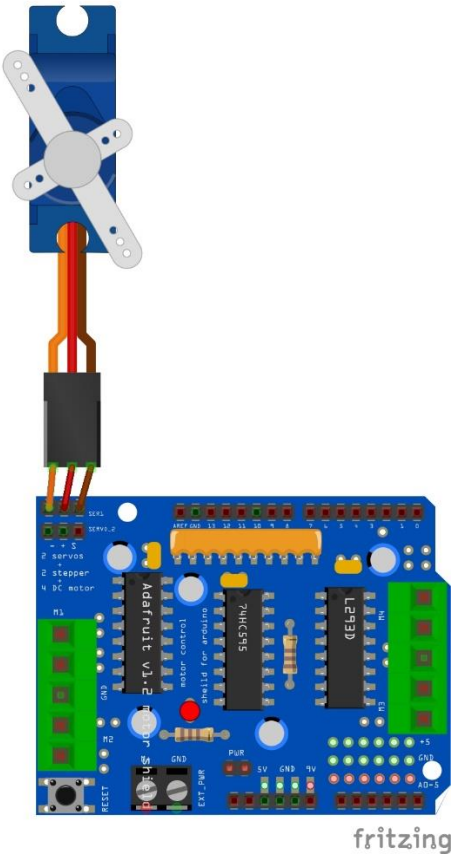
```
  motor.step(100, FORWARD, INTERLEAVE);
  motor.step(100, BACKWARD, INTERLEAVE);

  Serial.println("Micrsostep steps");
  motor.step(100, FORWARD, MICROSTEP);
  motor.step(100, BACKWARD, MICROSTEP);
}
```

Beispiel mit Servomotor

Mit dem L293D Motor Shield können Sie die Servo.h Bibliothek verwenden. Dies ist der Beispielcode „Sweep“ aus der Bibliothek, bei dem sich der Servo-arm um 180° hin- und herbewegt.

Verwenden Sie Pin 9 für Servo #1 und Pin 10 für Servo #2




```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
int pos = 0;  // variable to store the servo position

void setup()
{
  // attaches the servo on pin 9 to the servo object
  myservo.attach(9);
}
```

```
void loop()
{
  // sweeps from 0 degrees to 180 degrees
  for(pos = 0; pos <= 180; pos += 1)
  {
    myservo.write(pos);
    delay(15);
  }
  // sweeps from 180 degrees to 0 degrees
  for(pos = 180; pos>=0; pos-=1)
  {
    myservo.write(pos);
    delay(15);
  }
}
```