Willkommen!

Und herzlichen Dank für den Kauf unseres
AZ-Delivery 4-Kanal L293D Motortreiber Shield

Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Schritte von der Einrichtung bis zur Ansteuerung der Motoren.
Viel Spaß!

Das 4-Kanal L293D Motortreiber Shield wird einfach auf ein Mikrocontroller Board 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 Mikrocontroller Boardund 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.

Die wichtigsten Informationen in Kürze

- Beschriftete Anschlüsse zum einfachen Verbinden
- Kompatibel zu Mega 2560 R3, Diecimila, Duemilanove und ATmega328p R3 board
- 2 Anschlüsse für 5V Servomotoren mit Anbindung an den 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 Taster
- Abmessungen:70*55mm

Auf den nächsten Seiten findest du Informationen zur
» ***Einrichtung der Hardware***
und eine Anleitung für
» ***Ansteuern der verschiedenen Motoren***.

Diese Anleitung setzt voraus, dass du weißt, wie du Sketche auf einen Mikrocontroller hochlädst und den Serial Monitor verwendest!

Alle Links im Überblick

» Arduino IDE:
https://www.arduino.cc/en/Main/Software » Web-Editor:
https://create.arduino.cc/editor
» Erweiterung für SublimeText:
https://github.com/Robot-Will/Stino
» Erweiterung "Visual Micro" für Atmel Studio oder Microsoft Visual Studio:
http://www.visualmicro.com/page/Arduino-for-Atmel-Studio.aspx

Arduino Tutorials, Beispiele, Referenz, Community: »

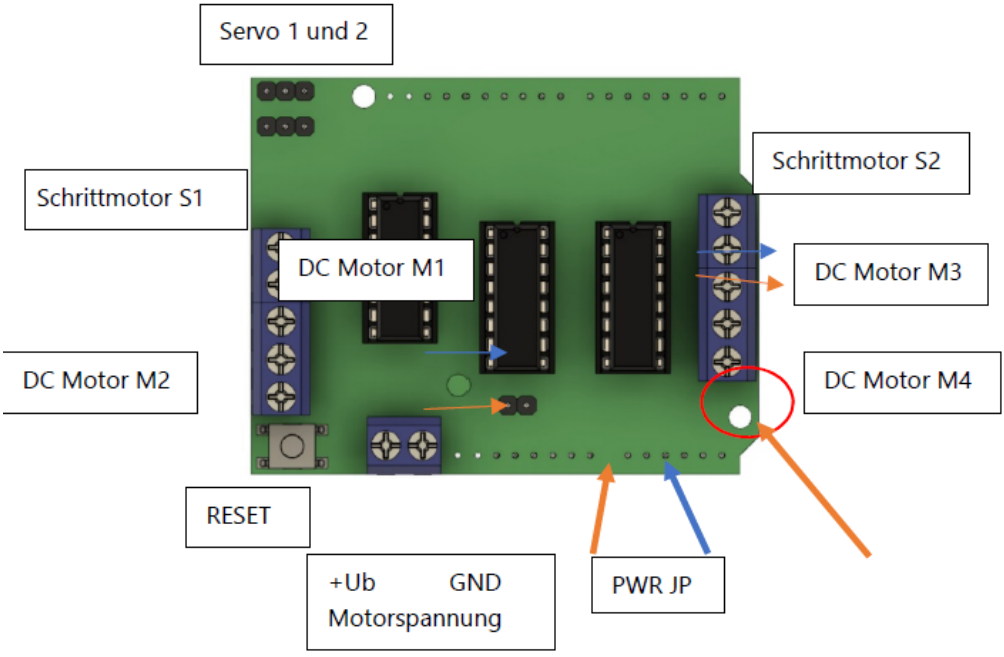
https://www.arduino.cc/en/Tutorial/HomePage
» *https://www.arduino.cc/en/Reference/HomePage*

Interessantes von AZ-Delivery

» Zubehör:
https://www.az-delivery.com/collections/weiteres-zubehoer
» AZ-Delivery G+Community:
https://plus.google.com/communities/115110265322509467732 » AZ-Delivery auf Facebook:
https://www.facebook.com/AZDeliveryShop/

Überblick

Es können 4 DC Motoren oder 2 Schrittmotoren angeschlossen werden. Der Spannungsbereich ist 4,5 - 13,5 V. Zusätzlich gibt es 2 Anschlüsse für Standard Servomotoren für 5V.



Das Shield nutzt die Standard PWM Pins des Mikrocontroller Board zum Steuern der angeschlossenen Motoren und Servos. Das Shield ist kompatibel mit jedem Mikrocontroller (z.B.

ATmega328p, Mega2560 R3).

Pro Mikrocontroller kann immer nur 1 Motor Shield genutzt werden.

Benutzte Pins des Board

PIN 11: DC Motor 1 / Schrittmotor 1
(Aktivierung/Geschwindigkeit)
PIN 3: DC Motor 2 / Schrittmotor 1
(Aktivierung/Geschwindigkeit)
PIN 5: DC Motor 3 / Schrittmotor 2
(Aktivierung/Geschwindigkeit)
PIN 6: DC Motor 4 / Schrittmotor 2
(Aktivierung/Geschwindigkeit)

PINs 4, 7, 8 und 12 werden für die Steuerung der DC/Schrittmotoren über den 74HC595 gebraucht.

Diese PINs werden nur für die Servomotoren gebraucht.

PIN 9: Servomotor 1 Ansteuerung
PIN 10: Servomotor 2 Ansteuerung

Die 6 Analogeingänge (PIN 14 bis 19) sowie die Digitaleingänge (PIN 2 und 13) werden nicht genutzt.

Auswahl der Motoren; Hardware Motorspannung

Die meisten Motoren benötigen Spannungen von 6V bis 12V. Diese

können mit dem Mikrocontroller Shield betrieben werden.

Motoren mit Spannungen von 1,5V bis3V können nicht betrieben werden.

Strombereich

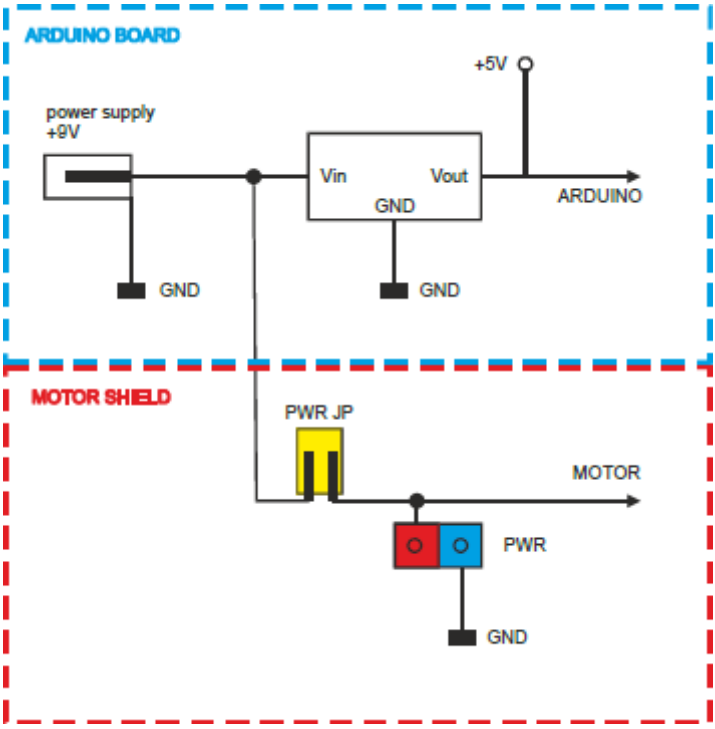
Das Mikrocontroller Shield ist ausgelegt für 0,6 A pro Motor; kurzzeitig darf der Spitzenwert bei bis zu 1,2A liegen. Bei großem Strombedarf werden die eingesetzten ICs heiß und müssen gekühlt werden.

Als Stromversorgung eignen sich am besten NiMH Akkus. Ein Betrieb an z.B. 9V Batterieblocks ist nicht zu empfehlen. Am besten ist es die Stromversorgung der Motoren von der Versorgung des Mikrocontroller zu trennen.
(2 Stromversorgungen)

Viele Probleme während des Betriebs eines Mikrocontroller Systems kommen von Störungen, die von Motoren auf einer gemeinsamen Stromversorgung erzeugt werde. Große Schwankungen des Stromverbrauchs, die durch unterschiedliche Belastungen der Motoren erzeugt werden führen zu Spannungsschwankungen, die ein Mikrocontroller Programm „durcheinander“ bringen können.

Anschluss von Motoren
Spannungsversorgung von Motoren

DC Motoren müssen mit einer eigenen Stromversorgung versorgt werden, da sie zum Teil hohe Ströme ziehen. Sie dürfen nicht an die 5V Pins des Mikrocontroller Bords angeschlossen werden. Das könnte das Mikrocontroller Board oder den USB Port zerstören.



Für den Anschluss gibt es zwei Möglichkeiten:

1. **Der DC Stecker (power supply) am Board.** Am Stecker befindet sich eine Schutzdiode. Damit ist der Mikrocontroller geschützt gegen falsche Spannung.
2. **Die 2-polige Schraubklemme PWR am Motor Shield.** An der Schraubklemme befindet sich ebenfalls eine Schutzbeschaltung, die eine Beschädigung des Motor Shields verhindert.

<u>Möglichkeit 1:</u> Gleiche Spannungsversorgung für Mikrocontroller Board und Motor Shield (1 Akku)

Eine Spannungsquelle 6 - 12V z.B. Akku wird an den Stecker des Mikrocontroller Boards oder an die Schraubklemmen des Motor Shields angeschlossen. **WICHTIG: Die Steckbrücke PWR JP am Motor Shield muss gesteckt sein!**

Hierbei kann es zu Störungen kommen, da die Spannung schwanken kann – abhängig von der Stromaufnahme der Motoren. Diese Betriebsart wird nur empfohlen, wenn ein starkes Akkupack genutzt wird.

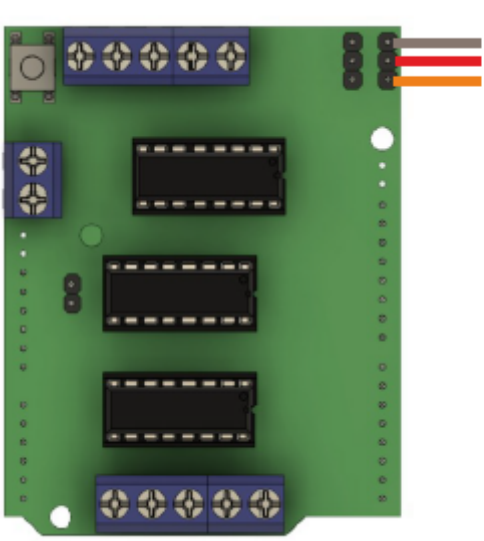
Möglichkeit 2: Unterschiedliche Spannungsversorgung für Mikrocontroller Board und Motor Shield (2 Akkus)

Der Mikrocontroller wird entweder über das USB Kabel oder über den Spannungsanschluss versorgt. An den Schraubklemmen des Motor Shields wird eine zusätzliche Spannungsversorgung angeschlossen. **WICHTIG: Die Steckbrücke PWR JP am Motor Shield darf nicht gesteckt sein**!

Das ist die empfohlene Methode, wenn Motoren oder größere Lasten verwendet werden. Logikspannung und Lastspannung sind entkoppelt.

Anschuss von Modellbau Servomotoren.

Die Anschlüsse des Mikrocontroller Shields sind gedacht für kleine Modellbau Servos. Sie werden direkt von den 5V des Mikrocontroller Boards versorgt. Werden größere Servos eingesetzt (mehr Leistung), dann müssen die Leiterbahnen von den Servo Anschlüssen zur Stiftleiste aufgetrennt werden und es müssen Kabel zur eigenen 5V Versorgung gelötet werde.



Modellbau Servomotoren sind sehr einfach zu nutzen.

Sie haben ein 3-poliges Buchsen Kabel (female). Die Signale werden als PWM übertragen; dazu gibt es des Anschluss PULS. (oft als orange oder weiße Ader ausgeführt.) Die beiden anderen Adern sind die Spannungsversorgung.
(rot = Vcc +5V; schwarz = GND)

Die Spannungsversorgung erfolgt ausschließlich über das Mikrocontroller Board.

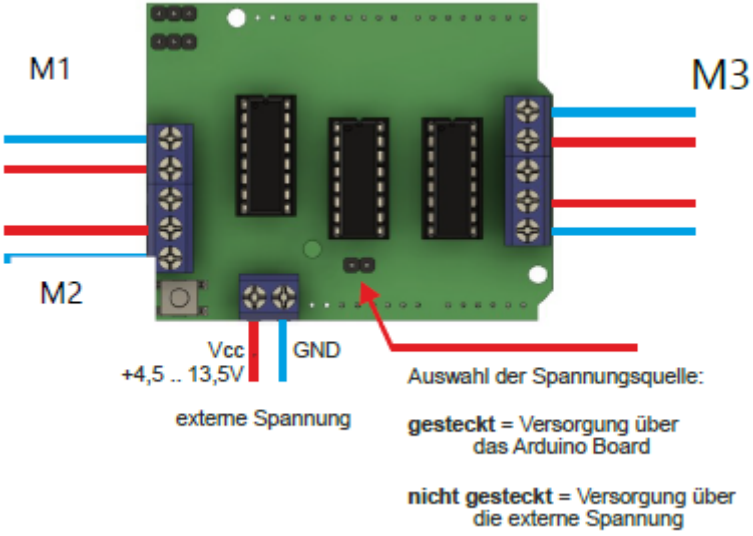
PIN Belegung:

Pin 9 PWM Servo 1
Pin 10 PWM Servo 2

Anschluss von DC Motoren (2 Dracht)

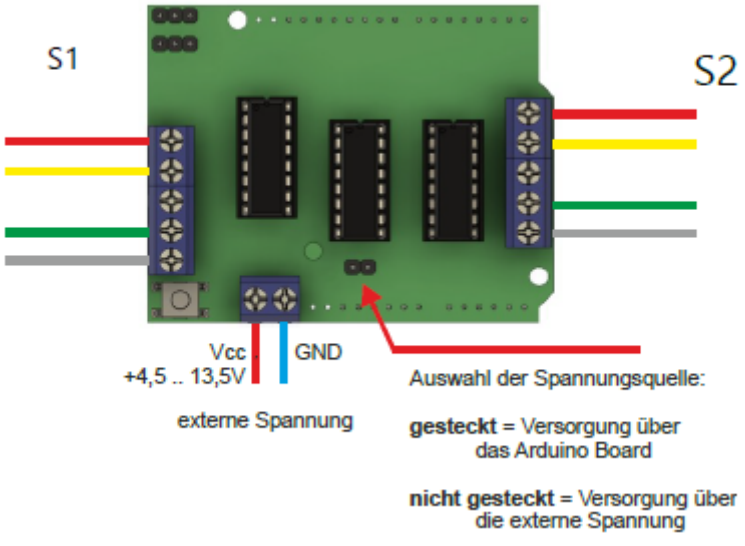
Bei größeren Motoren oder langer Laufzeit wir empfohlen einen Kühlkörper auf den Treiber IC aufzukleben.

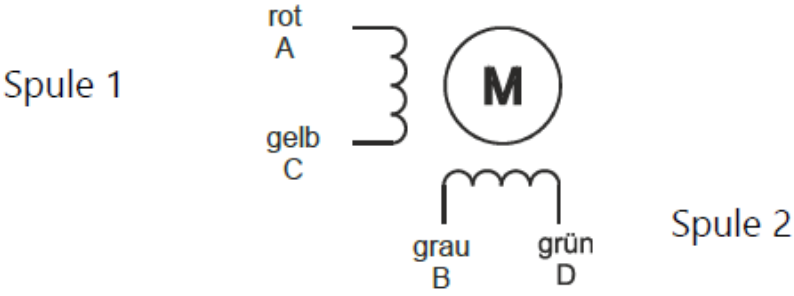
Beide Drähte des Motor s werden an die entsprechenden Klemmen angeschlossen. Durch Vertauschung der Drähte kann die Drehrichtung geändert werden. (z.B. alle Motoren soll in die gleiche Richtung laufen.)



Anschluss von Schrittmotoren

Es können 2 Schrittmotoren angeschlossen werden; sowohl uni-polare als auch bi-polare. Schrittmotoren verfügen über 2 Spulen:





Beim Anschluss von 2 Schrittmotoren ist zu beachten, dass immer die gleichen Spulen mit einem Anschluss verbunden sind. (Die verwendeten Farben dienen zur Zuordnung. Sie sind bei jedem Schrittmotor anders.)

z.B.

Spule 1 von Schrittmotor 1 -> M1
Spule 1 von Schrittmotor 2 -> M3

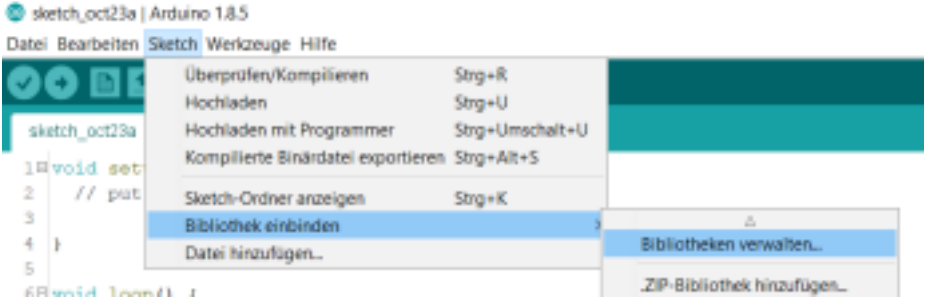
Spule 2 von Schrittmotor 1 -> M2
Spule 2 von Schrittmotor 2 -> M4

Programmierung; Software
Laden der Bibliothek

Zur Programmierung des Motor Shields sollte die **„Adafruit Motorshield v1 library“** genutzt werden.

Sie wird in der Arduino IDE über:

<Sketch -> Bibliothek einbinden -> Bibliothek verwalten> geholt.

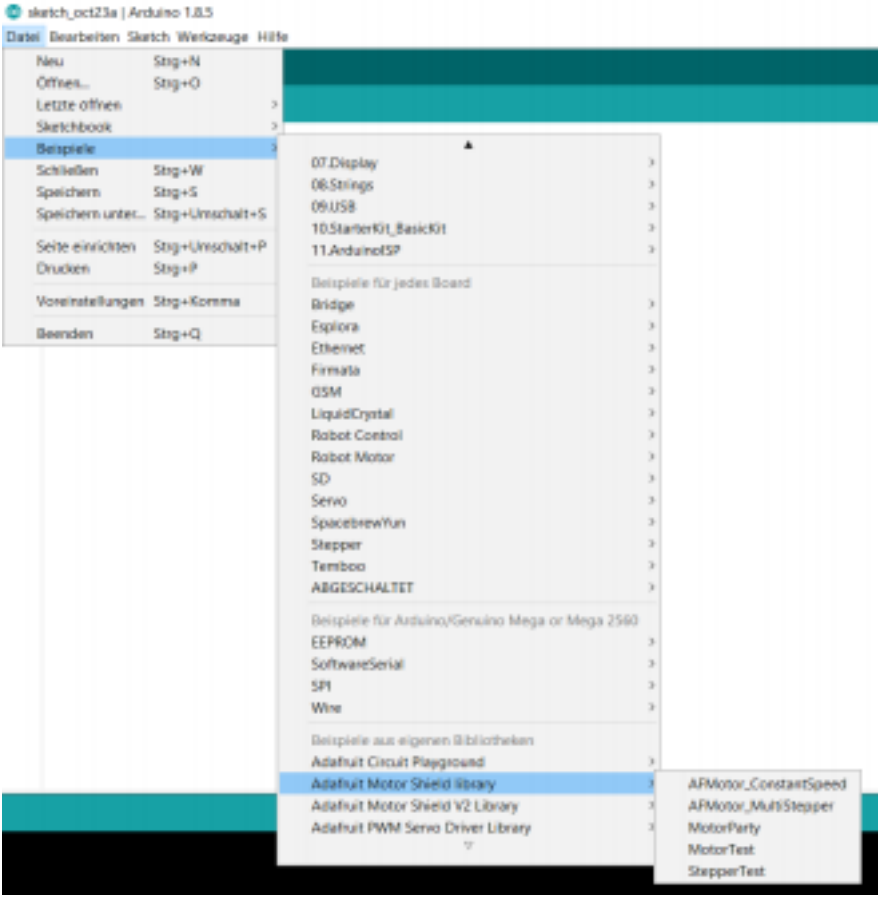


Jetzt kommt das Fenster mit allen verfügbaren Bibliotheken. In der obersten Zeile kann man einen Suchbegriff eingeben.

In diesem Fenster soweit nach unten scrollen bis die **„Adafruit Motor Shield library Version 1.0.0“** zu sehen ist. Ein Klick auf den Eintrag für die Bibliothek zur IDE hinzu.

ACHTUNG: Nicht die Bibliothek für das Motor Shield V2 laden. Sie ist mit dem Motor Shield nicht kompatibel.

Nach dem Laden kann man sich die Beispiel Sketche ansehen.



Beispiel für die Ansteuerung eines Servomotors Der Sketch ist bei den Beispielen zu finden.

```
// Adafruit Motor shield library copyright Adafruit Industries LLC,
2009 // this code is public domain, enjoy!


#include <AFMotor.h>
#include <Servo.h>
// DC hobby servo

Servo servo1;

void setup() {
 Serial.begin(9600); // set up Serial library at 9600 bps
```

```
Serial.println("Servo Test!");
 // servo anschalten
 servo1.attach(9);
}

int i;
// Servo testen

void loop() {
 for (i=0; i<255; i++) {
 servo1.write(i);
 delay(3);
 }
 for (i=255; i!=0; i--) {
 servo1.write(i-255);
 delay(3);
 }
}
```

Beispiel für die Ansteuerung eines DC Motors Die Bibliothek einbinden:

#include <AFMotor.h>

Das DC Motor Objekt erzeugen:

AF_DCMotor motor(nummer, frequenz)

Der Befehl hat 2 Argumente:

nummer Motor Anschluss 1,2,3 oder 4.

Frequenz Geschwindigkeit des Control Signals
Für Motor 1 oder 2 kann MOTOR12_64KHZ, MOTOR12_8KHZ, MOTOR12_2KHZ oder MOTOR12_1KHZ gewählt werden.
Für Motors 3 oder 4 kann nur 1KHz gewählt werden.
Alle anderen Einstellungen werden ignoriert.

Hinweise: Hohe Geschwindigkeiten z.B. 64KHz werden nicht gehört. Die Wahl einer niedrigen Geschwindigkeit z.B. 1KHz führt

aber zu einem geringeren Energieverbrauch.

Geschwindigkeit des Motors einstellen.

setSpeed(geschwindigkeit)

geschwindigkeit 0 (Motor stop) bis 255 (maximal)

Den Motor bewegen

run(richtung)

richtung FORWARD (vorwärts),
BACKWARD (rückwärts) oder
RELEASE (anhalten)

Die Richtungen „vorwärts“ und „rückwärts“ sind nicht festgelegt. Sie sind von der Verdrahtung der einzelnen Motoren abhängig. Sie können durch einfaches Vertauschen der Anschlüsse geändert werden.

Beispiel für die Ansteuerung eines DC Motors:

Es wird ein Objekt mit dem Namen **„motor“** erzeugt. Alle Befehle beziehen sich auf **<motor>**. Deshalb steht immer zuerst **„motor.“**. Das Sketch stammt aus den Beispielen der „Adafruit Motor shield library“.

```
// Adafruit Motor shield library copyright Adafruit Industries LLC,
2009 // this code is public domain, enjoy!

#include <AFMotor.h>
AF_DCMotor motor(2, MOTOR12_64KHZ); // erzeuge objekt motor an  port
2, 64KHz pwm

void setup() {
Serial.begin(9600); // Serielles Interface mit  9600 bps
Serial.println("Motor test!");
motor.setSpeed(200); // Geschwindigkeit 200  (maximal 255
 möglich)
 }
void loop() {
Serial.print("tick");
motor.run(FORWARD); // Motor dreht vorwärts delay(1000);
```

```
Serial.print("tock");
motor.run(BACKWARD); // Motor dreht rückwärts delay(1000);
Serial.print("tack");
motor.run(RELEASE); // Motor stopp delay(1000); // warte
1000ms (1s) }
```

Beispiel für die Ansteuerung eines Schrittmotors Die Bibliothek einbinden:

#include <AFMotor.h>

Das Schrittmotor Objekt erzeugen:

AF_Stepper name(schritte, stepper#)

name: Name des Motorobjektes z.B. „motor“; wird bei allen weiteren Befehlen angegeben

schritte: Auflösung des Schrittmotors.
z.B. ein 7,5 Grad Schrittmotor hat 360/7,5 = 48 Schritte Auflösung
z.B. ein 1,8 Grad Schrittmotor hat 360/1,8 = 200 Schritte Auflösung

stepper#: Nummer des Anschlusses für den Schrittmotor.
Anschluss 1 = Klemmen M1 und M2
Anschluss 2 = Klemmen M3 und M4

Geschwindigkeit des Motors einstellen:

name.setSpeed(rpm)

rpm: Anzahl der Umdrehungen / Minute des Schrittmotors name: Name des erzeugten Objektes

Den Motor bewegen

name.step(#steps, direction, steptype)

#steps: Anzahl der Schritte
direction: Richtung; FORWARD (vorwärts) oder

BACKWARD (rückwärts)
steptype Art es Schrittes; SINGLE, DOUBLE,
INTERLEAVE oder MICROSTEP
SINGLE nur eine Spule ist aktiv
DOUBLE beide Spulen sind gleichzeitig aktiv (höheres Drehmoment)
INTERLEAVE Wechsel zwischen SINGLE und DOUBLE um die Auflösung zu verdoppeln; dabei halbiert sich die Geschwindigkeit
MICROSTEP Die Spulen werden mit einem PWM Signal angeregt. Dadurch ergibt sich eine „weiche“ Bewegung wischen den einzelnen Schritten.
name: Name des erzeugten Motorobjektes

Den Motor anhalten:

Standardmäßig hält der Motor seine Position aktiv nach abgeschlossenem Schritt. (hoher Stromverbrauch) Wenn der Moter sich frei bewegen soll (geringer Stromverbrauch), dann muss man es extra befehlen.

name.release()

name: Name des erzeugten Motorobjektes

Programmablauf

Die Schrittbefehle „blockieren“ den Programmablauf; sie werden nicht im Hintergrund ausgeführt. Es wird so lange gewartet, bis die Motorbewegung abgeschlossen ist. (Kein Multitasking !)

Beispiel für die Ansteuerung eines Schrittmotors:

Es wird ein Objekt mit dem Namen **„motor“** erzeugt. Alle Befehle

beziehen sich auf **<motor>**. Deshalb steht immer zuerst **„motor.“**.

Das Sketch stammt aus den Beispielen der **„Adafruit Motor shield library“.**

```
// Adafruit Motor shield library copyright Adafruit Industries LLC,
2009 // this code is public domain, enjoy!


#include <AFMotor.h>

AF_Stepper motor(48, 2); // 48 Schritte / Umdrehung, port 2, M3+M4
                                    // Objektname: motor
void setup() {
Serial.begin(9600); // Serielles Interface mit 9600 bps
Serial.println("Stepper test!");
motor.setSpeed(10); // 10 Umdrehungen / Minute motor.step(100,
FORWARD, SINGLE); // 100 Schritte, vorwärts, eine  Spule
motor.release(); // Motor frei; geringer  Stromverbrauch
delay(1000); // warte 1000 ms (1s) }

void loop() {
motor.step(100, FORWARD, SINGLE); // je 100 Schritte in verschiedenen
Modies
motor.step(100, BACKWARD, SINGLE);
motor.step(100, FORWARD, DOUBLE);
motor.step(100, BACKWARD, DOUBLE);
motor.step(100, FORWARD, INTERLEAVE);
motor.step(100, BACKWARD, INTERLEAVE);
motor.step(100, FORWARD, MICROSTEP);
motor.step(100, BACKWARD, MICROSTEP);
}
```

Impressum

https://az-delivery.de/pages/about-us