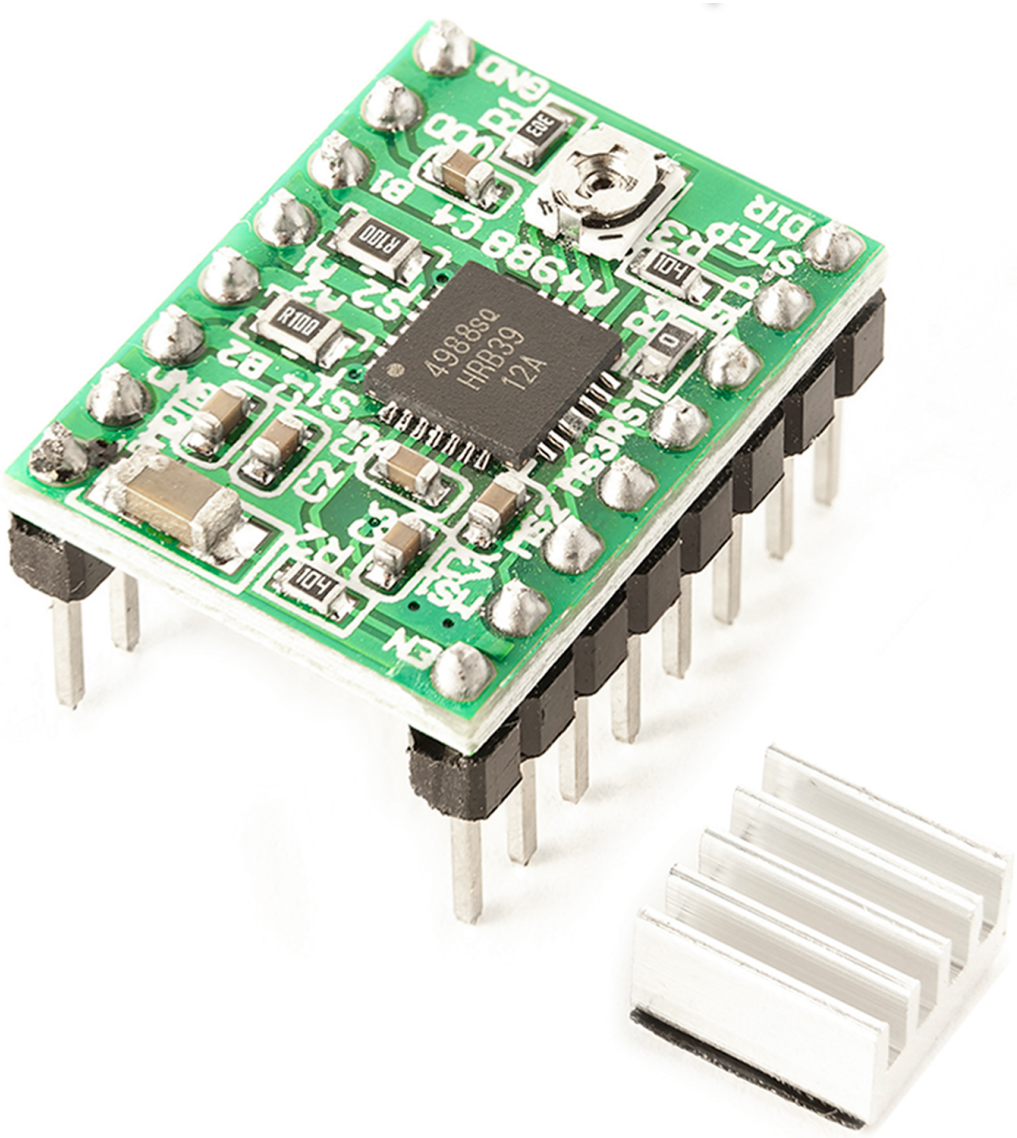
Willkommen!

Vielen Dank, dass sie sich für unser AZ-Delivery A4988 Schrittmotor-Treiber-Modul mit Kühlkörper entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

Viel Spaß!



Ein Schrittmotor ist eine Art von Motor, bei dem sich die Welle des Motors schrittweise dreht. Der Schrittmotor ist ein bürstenloser Gleichstrommotor. Die schrittweise Bewegung ermöglicht es die Welle ohne Positions-Feedback sehr präzise zu positionieren.

Alle Elektromotoren bestehen aus einem Rotor und Stator. Bei Schrittmotoren ist der Rotor standartmäßig ein Permanentmagnet, der von den Spulen des Stators umgeben ist. Damit sich der Rotor dreht, müssen wir die Spulen des Stators in einer bestimmten Reihenfolge ein- und ausschalten. Wenn Strom durch die Spulen des Stators fließt, entsteht ein elektromagnetisches Feld, das den Rotor in Richtung des elektromagnetischen Felds dreht.

Nach der Bauart gibt es drei verschiedene Arten von Schrittmotoren:

» Permanentmagnet-Schrittmotoren,
» Schrittmotoren mit variabler Reluktanz und
» Hybrid-Synchron-Schrittmotoren.

(wir werden in diesem eBook keine Schrittmotor-Konstruktionen behandeln)

Antriebsarten von Schrittmotoren

Um den Schrittmotor anzutreiben, gibt es mehrere Modi:

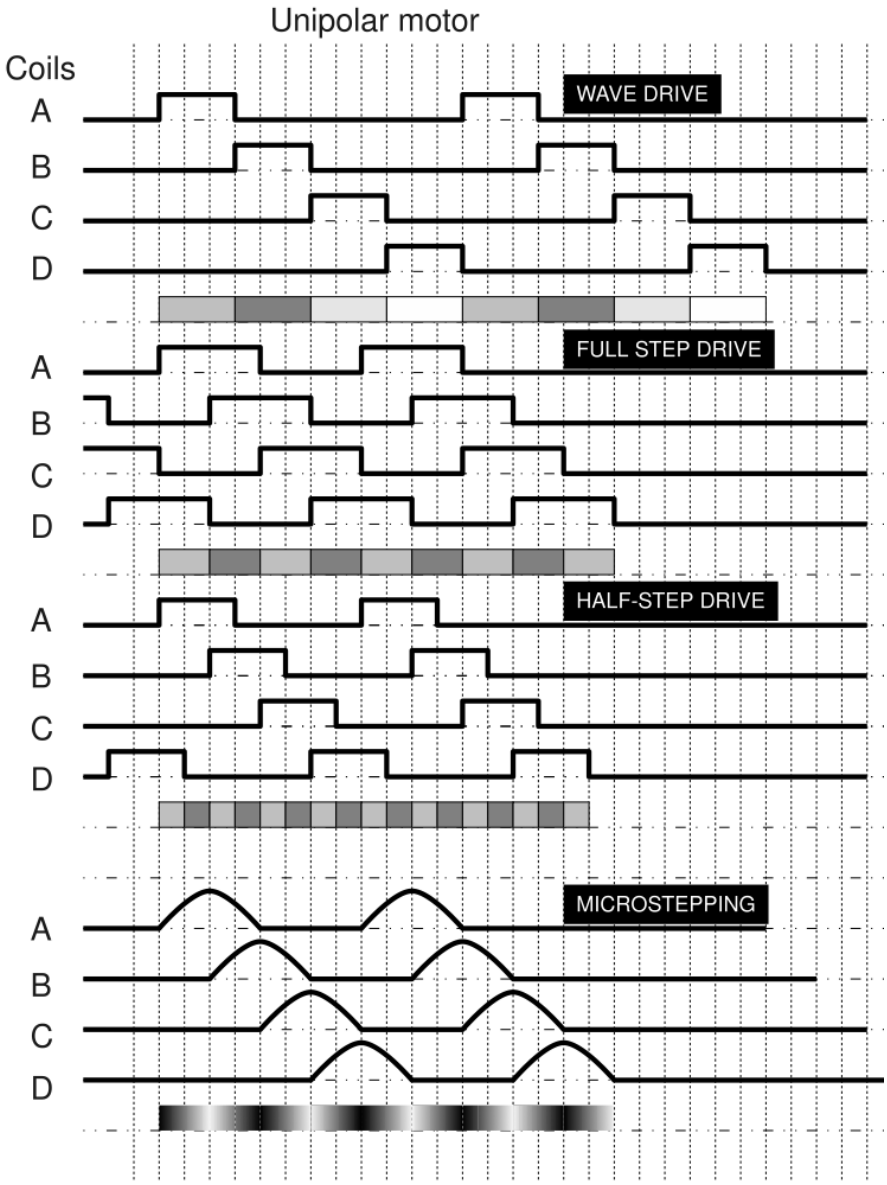
» **Wellenantriebsmodus**, in diesem Modus aktivieren wir jeweils nur eine Spule des Stators, dann die nächste, usw. In diesem Modus wird nur eine Spule aktiviert, um den Rotor zum nächsten Schritt zu bewegen. Nach und nach schalten wir die einzelnen Spulen ein. Wenn die zweite Spule eingeschaltet wird, wird die erste Spule ausgeschaltet, usw.

» **Vollschritt-Antriebsmodus**, dieser Modus bietet eine wesentlich höhere Drehmomentabgabe, da wir immer zwei Spulen zur gleichen Zeit aktiviert werden. Dieser Modus wird auf dem Bild der nächsten Seite genauer erklärt.

» **Halbschrittbetrieb,** dieser wird benutzt, um die Auflösung des Schrittmotors zu erhöhen. Dieser Modus entstand aus der Kombination der beiden vorherigen Modi. Hier aktivieren wir zuerst eine Spule, und am Ende des aktiven Zustands dieser Spule ,aktivieren wir die nächste. Wenn die zweite Spule aktiviert wurde, schalten wir die erste Spule aus, und so weiter. Mit diesem Modus erhalten wir bei gleichem Design die doppelte Auflösung.

» **Der Mikroschrittmodus** ist heutzutage der am meisten verwendete Modus zur Steuerung von Schrittmotoren. In diesem Modus stellen wir den Spulen variablen Strom in Form einer Sinuswelle zur Verfügung. Dies sorgt für eine gleichmäßige Bewegung des Rotors, verringert die Belastung und erhöht gleichzeitig die Genauigkeit des Schrittmotors.

Eine weitere Möglichkeit, die Auflösung des Schrittmotors zu erhöhen, besteht darin, die Polzahl des Rotors und die Polzahl des Stators zu erhöhen.



Das Bild ist aus einem Wikipedia-Artikel zu Schrittmotoren

https://en.wikipedia.org/wiki/Stepper_motor

Schrittmotortreiber

Wenn Sie ein Gerät konstruieren wollen, bei dem Sie die Drehung der Motorwelle präzise steuern müssen, wie z.B. in 3D-Druckern oder anderen CNC-Maschinen, oder Roboterarmen, etc., benötigen Sie mehrere Schrittmotoren und vor allem Schrittmotortreiber.

Es ist unpraktisch und in den meisten Fällen unmöglich mit dem Arduino Schrittmotoren zu steuern. Deshalb brauchen wir für jeden Schrittmotor eine elektronische Treiberschaltung.

In diesem eBook werden wir den "A4988"-Schrittmotortreiber behandeln. Dieses Gerät kann die Drehzahl und Drehrichtung der Motorwelle steuern, sowie den Schrittmotor in mehreren Anregungsmodi betreiben. Wir können mit diesem Gerät sogar Mikroschritt-Anregungsmodi verwenden. Allerdings können wir mit diesem Schrittmotortreiber nur bipolare Schrittmotoren steuern.

Der Hauptchip auf diesem Gerät ist die Schaltung "A4988", hergestellt von "Allegro". Der Treiber verfügt über einen integrierten Übersetzer für eine einfache Bedienung. Dadurch reduziert sich die Anzahl der Steuerpins auf zwei, einer zur Steuerung der Schritte und einer zur Steuerung der Drehrichtung. Der Treiber bietet fünf verschiedene Schrittauflösungen oder Anregungsmodi: Vollschritt- und vier Mikroschrittmodi: Halbschritt, Viertelschritt, Achtelschritt und Sechzehntelschritt.

Technische Daten

- » Min. - max. Logikspannung: 3V - 5,5V
- » Nennstrom pro Phase: 1A
- » Maximaler Strom pro Phase: 2A, passiver Kühlung, Alu-Kühlkörper
- » Motorausgangsspannung: 8V - 35V

Der "A4988" benötigt zwei Stromversorgungsanschlüsse. Einer für Logikpins und einer für die Motorstromversorgung:

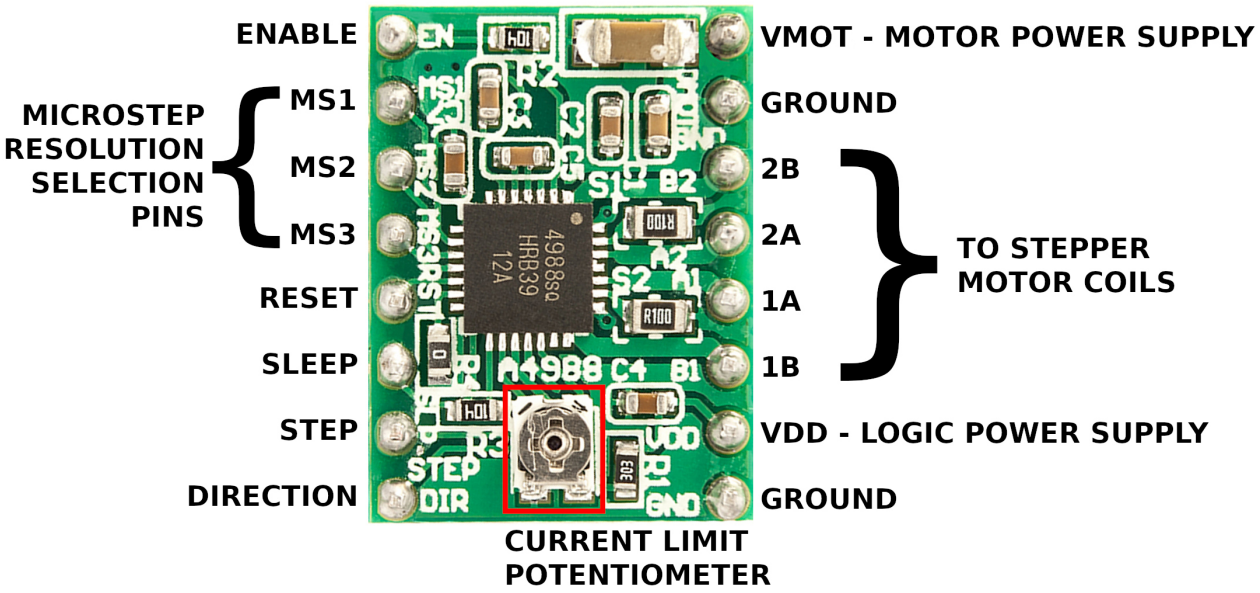
VDD und GND werden für den Betrieb der internen Logik des Treibers verwendet (von 3V bis 5,5V), so dass wir entweder den Arduino oder Raspberry Pi als Steuereinheit für dieses Gerät verwenden können.

VMOT und GND werden für die Stromversorgung des Motors von 8V bis 35V verwendet. Gemäß dem Datenblatt erfordert die Motorversorgung einen geeigneten Entkopplungskondensator in der Nähe der Platine, der in der Lage ist, 4A-Strom zu liefern.

WARNUNG: Dieser Treiber hat Keramikkondensatoren mit niedrigem ESR-Wert verbaut, was ihn anfällig für Spannungsspitzen macht. In einigen Fällen können diese Spitzen die 35V (maximale Nennspannung von A4988) überschreiten, was zu dauerhaften Schäden an der Platine und/oder dem Schrittmotor führen kann.

Eine Möglichkeit, den Treiber vor solchen Spitzen zu schützen, besteht darin, einen großen 100µF-Elektrolytkondensator (oder mindestens 47µF) über die Versorgungsanschlüsse des Motors zu legen.

Pinbelegung des “A4988”



Auswahl der Mikroschrittpins

Der "A4988"-Treiber ermöglicht Microstepping durch kleinere Schrittpositionen. Dies wird erreicht, indem die Spulen mit a variablem Strom versorgt werden. Wenn Sie beispielsweise den Schrittmotor "NEMA17" betreiben, hat er einen Schrittwinkel von 1,8° oder macht 200 Schritte pro voller Umdrehung. Im Viertelstufenmodus macht dieser Motor 800 Mikroschritte pro voller Umdrehung.

Der A4988-Treiber hat drei Eingangspins für Mikroschrittauflösungen:

» MS1

» MS2

» MS3

Durch das Einstellen geeigneter Logikpegel auf diesen Pins können wir den Antriebsmodus des Motors auf den einen dieser fünf Modi einstellen:

MS1	MS2	MS3	Mikroschrittauflösung
LOW	LOW	LOW	Vollschritt
HIGH	LOW	LOW	Halbschritt
LOW	HIGH	LOW	Viertelschritt
HIGH	HIGH	LOW	Achtelschritt
HIGH	HIGH	HIGH	Sechzehntelschritt

Diese drei Mikroschrittpins werden von internen Pull-Down-Widerständen auf LOW geschalten, so dass der Motor im Vollschrittmodus arbeitet, wenn wir alle trennen.

Der Unterschied zwischen Vollschritt und Mikroschritt

Mikroschritt-Erregungsmodi sind alle Modi, bei denen sich die Welle des Motors zwischen den Hardware-Schritten bewegt. Diese Modi positionieren die Welle des Motors zwischen den Schritten, wodurch weitere Schritte und eine gleichmäßige Bewegung der Welle entstehen.

Der halbstufige Anregungsmodus ist eine Kombination aus Vollschritt und Wellenantrieb. Dies ergibt die Hälfte des Grundschrittwinkels. Dieser kleinere Schrittwinkel sorgt durch die erhöhte Auflösung des Winkels für einen ruhigeren Betrieb. Der Halbschritt erzeugt etwa 15% weniger Drehmoment als der Vollschritt, jedoch eliminiert der modifizierte Halbschritt den Drehmomentverlust, indem es den Strom erhöht, der an den Motor geliefert wird, wenn eine einzelne Spule unter Spannung steht.

Durch Mikroschritte kann der Grundschritt eines Motors auf 256 geteilt werden, wodurch kleine Schritte noch kleiner werden. Ein Mikroschritt-Treiber verwendet zwei variable Strom-Sinuswellen im Abstand von 90°, wodurch der Motor gleichmäßig läuft. Sie werden feststellen, dass der Motor leise und ohne wirklich erkennbare Schrittfunktion läuft.

Man kann die Auflösung erhöhen, indem man die Richtung und Amplitude des Stromflusses in jeder Spule des Stators steuert. Die Eigenschaften des Motors verbessern sich auch, was zu weniger Vibration und einem ruhigeren Betrieb führt. Da die Sinuswellen zusammenwirken, entsteht ein fließender Übergang von einer Spule zur anderen. Wenn der Strom in einer Spule ansteigt, nimmt er in der nächsten Spule ab, was zu einem sanften Schrittverlauf und einem konstanten Drehmoment führt.

Schritt- und Richtungsstifte

Der "**STEP**"-Eingangspin steuert die Schritte des Motors. Jeder "HIGH"-Impuls, der an diesen Pin gesendet wird, steuert den Motor um die Anzahl der Mikroschritte, die durch die Mikroschrittauswahlpins eingestellt werden. Je schneller die Impulse, desto schneller dreht sich der Motor.

Der "**DIR**"-Eingangspin steuert die Drehrichtung der Motorwelle. Stellt man ihn auf HIGH wird die Welle im Uhrzeigersinn bewegt. Stellt man dagegegen auf LOW wird die Welle gegen den Uhrzeigersinn bewegt. Wenn Sie möchten, dass sich die Welle in eine einzelne Richtung dreht, können Sie den "DIR"-Stift direkt mit VCC oder GND verbinden.

HINWEIS

Die STEP- und DIR-Pins werden intern nicht auf eine bestimmte Spannung geschalten, deswegen sollten Sie sie nicht in Ihrer Anwendung treiben lassen.

Spannungsfreigabestifte

Wenn Sie den ***EN***-Pin auf *LOW* schalten, wird der Treiber aktiviert. Standardmäßig ist dieser Pin *LOW* geschaltet, so dass der Treiber immer aktiviert ist. Schalten Sie ihn *HIGH*, um den Treiber zu deaktivieren.

Wenn Sie den ***SLP***-Pin *LOW* schalten, wird der Treiber in den Ruhezustand versetzt, wodurch der Stromverbrauch minimiert wird.

Wenn sie den ***RST***-Pin *LOW* schalten, werden alle *STEP*-Pins ignoriert, bis Sie ihn HIGH schalten. Außerdem wird der Treiber zurückgesetzt, wenn der interne Übersetzer in einen vordefinierten "Home"-Zustand" versetzt wird. Der "Home"-Zustand ist die Ausgangsposition, von der aus der Motor startet und unterscheidet sich je nach Mikroschrittauflösung.

HINWEIS

Wenn Sie den RST-Pin nicht verwenden, können Sie ihn mit dem SLEEP-Pin verbinden, um ihn HIGH zu schalten und den Treiber zu aktivieren.

Ausgangspins

Die Ausgangspins des Treibers sind 1A, 1B, 2A und 2B. Jeder Ausgangspin kann Strom von bis zu 2A liefern. Die dem Motor zugeführte Strommenge hängt jedoch von der Stromversorgung, dem Kühlsystem und den Einstellungen zur Strombegrenzung ab.

Wenn Sie die Belegung Ihres bipolaren Schrittmotors nicht kennen, können Sie ihn mit dem Multimeter testen. Schalten Sie das Multimeter ein für einen Durchgangstest (mit dem Schallzeichen gekennzeichnet). Prüfen Sie zunächst, ob das Multimeter funktioniert, indem Sie beide Knotenpunkte des Multimeters miteinander verbinden. Man sollte einen Piepton hören, wenn es funktioniert. Der bipolare Schrittmotor hat zwei Spulen und vier Drähte, zwei Drähte pro Spule (für den Eingang und den Ausgang der Spule). Eine Spule ist nur ein gewickelter Draht. Wir überprüfen die Kontinuität zwischen zwei Drähten des Motors. Verbinden Sie einfach einen Knoten des Multimeters mit einem beliebigen Draht des Motors und den zweiten Knoten mit einem beliebigen anderen Draht des Motors. Wenn Sie einen Piepton hören, sind diese beiden Drähte Teil einer Spule.

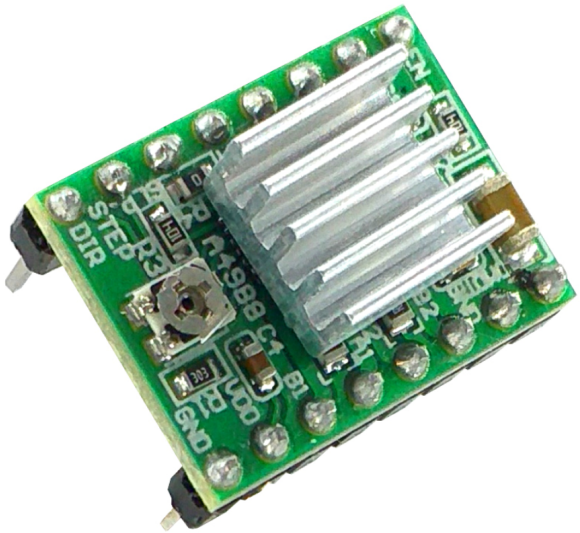
Die Kabelnamen sind von Hersteller zu Hersteller unterschiedlich. Hier sind einige Beispiele für Kabelnamen und wie diese verbunden sind:

Schrittmotorenpins					**>**	**Treiberpins**
A’	A	1	A1	A+	>	1A
A”	C	3	A2	A-	>	1B
B'	B	2	B1	B+	>	2A
B”	D	4	B2	B-	>	2B

(1A und 1B sind für die erste Spule, 2A und 2B für die Zweite)

Kühlsystem - Aluminum Kühlkörper

Ein übermäßiger Verlust der Leistung des Treiberchips führt zu einem Temperaturanstieg, der über die Kapazität des Chips hinausgehen kann. Das könnte den Chip beschädigen. Selbst wenn der Treiber-IC eine maximale Nennstromstärke von 2A pro Spule hat, kann der Chip nur 1A pro Spule liefern, ohne zu überhitzen. Um mehr als 1A pro Spule zu erreichen, ist ein Kühlkörper oder ein anderes Kühlsystem erforderlich. Unser "A4988"-Treibermodul wird mit einem Aluminiumkühlkörper geliefert. Wir raten Ihnen es vor der Verwendung des Moduls zu installieren.



Strombegrenzungspotentiometer

Vor dem Einsatz des Motors müssen wir eine kleine Änderung vornehmen. Wir müssen die maximale Strommenge, die durch die Schrittmotorspulen fließt, begrenzen und verhindern, dass sie den Nennstrom des Motors überschreitet. Am "A4988"-Treiber befindet sich ein kleines Trimmer-potentiometer, mit dem die Stromgrenze eingestellt werden kann. Um die aktuelle Grenze einzustellen, müssen Sie die nächsten Schritte ausführen:

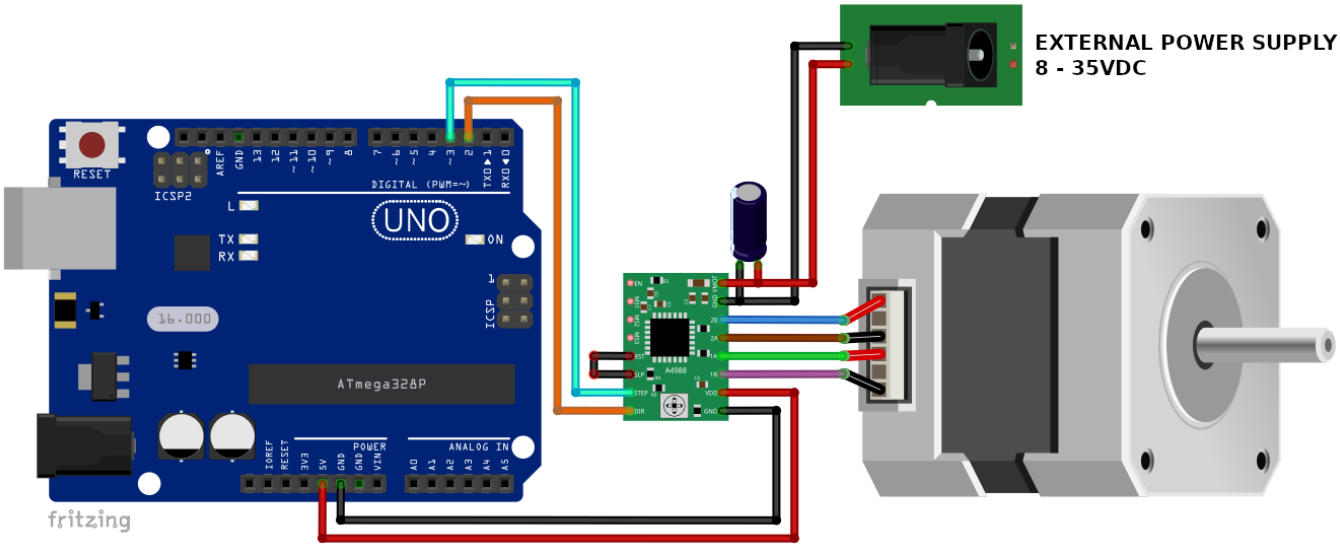
» Werfen Sie einen Blick auf das Datenblatt Ihres Schrittmotors. Notieren Sie sich den Nennstrom des Motors. In unserem Fall verwenden wir "*NEMA17*" 200 Schritte/Umdrehung, 12V, 350mA.
» Setzen Sie den Treiber in den Vollschrittmodus, indem Sie die drei Mikroschrittpins getrennt lassen.
» Halten Sie den Motor an einer festen Position, indem Sie den *STEP*-Eingang NICHT taktieren. Lassen Sie den *STEP*-Eingang nicht frei, sondern schließen Sie ihn an die Logikversorgung an (5V für Arduino oder 3,3V für Raspberry Pi).
» Stellen Sie das Amperemeter in der Reihe mit einer der Spulen auf Ihrem Schrittmotor und messen Sie den tatsächlichen Strom.
» Stellen Sie mit einem kleinen Schraubenzieher das Potentiometer so ein, dass der Nennstromwert aus dem Datenblatt des Motors erreicht wird.

HINWEIS: Sie müssen diese Einstellung erneut durchführen, wenn Sie die Logikspannung (VDD) ändern.

WARNUNG: Das Anschließen oder Trennen eines Schrittmotors bei eingeschaltetem Treiber kann diesen beschädigen!!!

Verbindung des Treibers mit dem Arduino Uno

Verbinden Sie den Treiber mit dem Arduino wie unten abgebildet:



Treiberpin	**>**	**Arduino-Pin**	
VDD	>	5V	**Roter Draht**
GND	>	GND	**Schwarzer Draht**
STEP	>	D3	**Cyaner Draht**
DIR	>	D2	**Oranger Draht**

VMOT	>	+ der externen Stromversorgung	**Roter Draht**
GND	>	GND der externen Stromversorgung	**Schwarzer Draht**

Verbinden Sie den *RST-P*in mit dem *SLEEP-P*in, damit der Treiber aktivert bleibt (**Schwarzer Draht**).

Denken Sie daran einen großen 100µF Entkopplungs-Elektrolytkondensator über die Pins der Motorstromversorgung zu legen. Möglichst nah an der Platine, wie auf dem Anschlussplan oben dargestellt.

Halten Sie die Mikroschrittlpins getrennt, um den Motor im Vollschrittmodus zu betreiben, oder schließen Sie den entsprechenden MS-Pin an die VDD-Spannung an, um einen anderen Erregungsmodus zu benutzen, wie bereits besprochen. Allerdings würde die Motorwelle dann für den gleichen *STEP*-Takt langsamer arbeiten, wenn Sie einen anderen Anregungsmodus als den Vollschrittmodus verwenden. Das liegt daran, dass wir in einem anderen Anregungsmodi mehr Schritte verwenden. Mehr Schritte = mehr Zeit, um sie zu vollenden. Um also einen Motor mit gleicher Drehzahl zu verwenden, müssen Sie die Taktung am STEP-Pin erhöhen. Mit dem Schrittmotor "NEMA17" haben wir 950us für den eingeschalteten Zustand und 950us für den ausgeschalteten Zustand des Taktsignals auf dem STEP-Pin verwendet. Das ist das Minimum im Vollschrittmodus (alle MS-Pins sind getrennt). Aber mit dem Sechzehntelschrittmodus (alle drei MS-Pins sind mit dem VDD verbunden) gehen diese Werte auf 35us runter. Das werden Sie im Code sehen.

Wir benötigen keine Library, damit dieser Treiber funktioniert. Wir werden unsere eigene Skizze erstellen.

Arduino-Code:

```
uint8_t stepPin = 2;
uint8_t dirPin = 3;
int steps = 1000;    // you should increase this if you are using
                     // some of microstepping modes
int usDelay = 950;   // minimal is 950 for full step mode and NEMA15 motor
                     // minimal is 35 for sixteenth step mode

void setup() {
  Serial.begin(9600);
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop() {
  digitalWrite(dirPin, HIGH); // motor direction cw
  for(int x = 0; x < steps; x++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(usDelay);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(usDelay);
  }
  delay(1000);
  digitalWrite(dirPin, LOW); // motor direction ccw
  for(int x = 0; x < steps; x++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(usDelay);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(usDelay);
  }
  delay(1000);
}
```

Wir beginnen die Skizze mit der Definition der *STEP*- und *DIR*-Pins, die mit dem Arduino verbunden sind. Wir definieren eine Variable namens *steps*, die wir für die Anzahl der Schritte für die Motorwelle verwenden. In der Setup-Funktion bezeichnen wir *STEP*- und *DIR*-Pins als digitale Ausgänge. Im Loop-Bereich drehen wir den Motor im Uhrzeigersinn und drehen ihn dann im Abstand von zwei Sekunden gegen den Uhrzeigersinn.

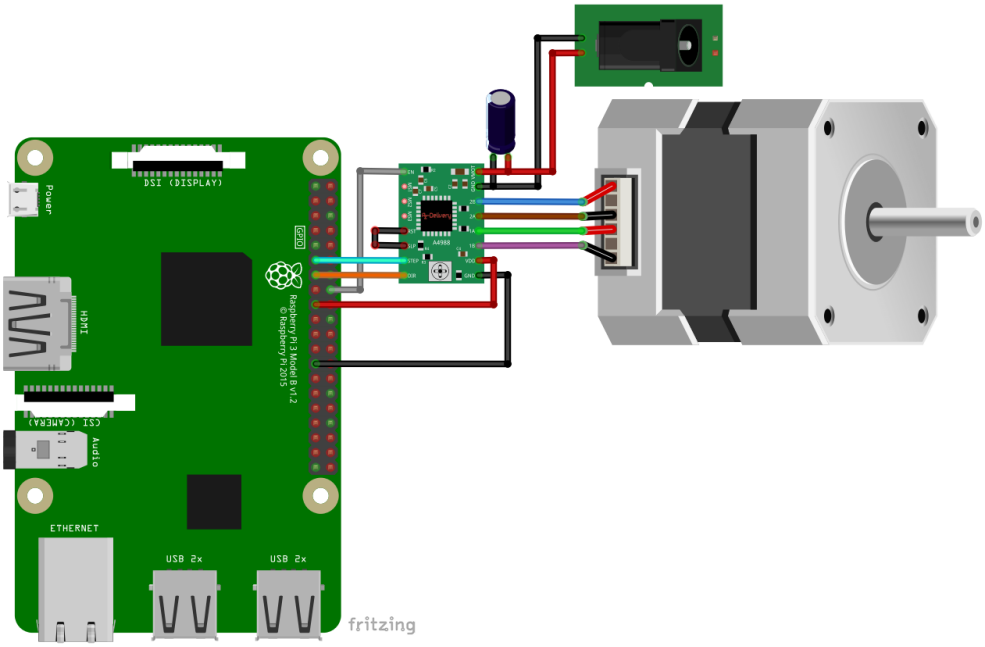
Um die Drehrichtung eines Motors zu steuern, stellen wir den DIR-Pin entweder HIGH oder LOW ein. HIGH dreht den Motor im Uhrzeigersinn und LOW dreht ihn gegen den Uhrzeigersinn.

Die Drehzahl einer Motorwelle wird durch die Frequenz der Impulse (Taktsignal) bestimmt, die wir an den *STEP*-Pin senden. Je mehr Impulse, desto schneller läuft der Motor. Für einen Impuls müssen Sie nur den Ausgang *HIGH* schalten. Warten Sie ein wenig und schalten dann auf *LOW*. Diese Wartezeit wird durch die Variable *usDelay* definiert. Ändern Sie die Verzögerung zwischen den beiden Impulsen, um die Frequenz dieser Impulse zu ändern und damit die Drehzahl der Motorwelle.

Wir haben den Code mit dem Schrittmotor "NEMA17" getestet. Im Vollschrittmodus ist der Minimalwert der Variablen *usDelay 950*. Wenn Sie tiefer gehen, bewegt sich die Welle des Motors nicht. Das bedeutet, dass der Motor nicht in der Lage ist, schnellere Taktraten zu verarbeiten. Wenn Sie den Sechzehntelschrittmodus einstellen, erhalten Sie das Sechzehnfache an Schritten. Um die Welle des Motors in die gleiche Position wie im Vollschrittmodus zu bringen, müssen wir die variablen Schritte um das Sechzehnfache erhöhen. Und um die gleiche Drehzahl wie beim Vollschritt (usDelay=950) zu erreichen, müssen wir den Wert von usDelay auf 35 ändern (Minimum).

Verbindung des Treibers mit dem Raspberry Pi

Verbinden Sie den Treiber mit dem Raspberry Pi wie unten abgebildet:



Treiberpin	**>**	**Raspberry Pi-Pin**		
VDD	>	3.3V	[Pin 17]	**Roter Draht**
GND	>	GND	[Pin 30]	**Schwarzer Draht**
STEP	>	GPIO17	[Pin 11]	**Cyaner Draht**
DIR	>	GPIO27	[Pin 15]	**Oranger Draht**
EN	>	GPIO23	[Pin 16]	**Grauer Draht**

VMOT	>	+	externe Stromversorgung	**Roter Draht**
GND	>	GND	externe Stromversorgung	**Schwarzer Draht**

Verbinden Sie den *RST-P*in mit dem *SLEEP-P*in, um den Treiber aktiviert zu lassen (**Schwarzer Draht**). **Denken Sie daran,** einen großen 100µF Entkopplungs-Elektrolytkondensator so nah wie möglich an der Platine zu platzieren, wie in der Skizze oben dargestellt.

Wie beim Arduino müssen wir für dieses Treibermodul keine Library installieren. Wir werden unser Python-Skript erstellen. Hier ist der Code:

```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
STEP = 17 # step pin
DIR = 27 # direction pin
EN = 23 # enable pin
GPIO.setup(STEP, GPIO.OUT)
GPIO.setup(DIR, GPIO.OUT)
GPIO.setup(EN, GPIO.OUT)
steps = 5000 # number of steps
usDelay = 950 # number of microseconds
uS = 0.000001 # one microsecond
GPIO.output(EN, GPIO.LOW)
print("[press ctrl+c to end the script]")
try: # Main program loop
    while True:
        GPIO.output(DIR, GPIO.HIGH) # cw direction
        for i in range(steps):
            GPIO.output(STEP, GPIO.HIGH)
            sleep(uS * usDelay)
            GPIO.output(STEP, GPIO.LOW)
            sleep(uS * usDelay)
        sleep(2)
        GPIO.output(DIR, GPIO.LOW) # ccw direction
        for i in range(steps):
            GPIO.output(STEP, GPIO.HIGH)
            sleep(uS * usDelay)
            GPIO.output(STEP, GPIO.LOW)
            sleep(uS * usDelay)
        sleep(2)
# Scavenging work after the end of the program
except KeyboardInterrupt:
    GPIO.output(EN, GPIO.HIGH)
```

Wir haben gerade die Arduino-Skizze in Python-Code umgewandelt. Der einzige Unterschied ist der Code für den EN-Pin. Dieser Pin wird benutzt, um den Treiber zu aktivieren. Wenn er auf LOW steht, ist der Treiber aktiviert, wenn er auf HIGH steht, ist der Treiber deaktiviert. Wir brauchen dies, weil die DIR- und STEP-Pins frei stehen, und wenn wir das Skript beenden, ohne diese Pins mit GND oder VDD zu verbinden, würde der Treiber ausarten. Die Welle des Motors würde beginnen sich merkwürdig zu bewegen. Also müssen wir den Treiber deaktivieren, wenn das Skript endet. Wir tun dies im *except*-Block des Codes.

Der Rest des Codes ist derselbe wie in der Arduino-Skizze, so dass er nicht näher zu behandeln ist.

Sie haben es geschafft. Sie können jetzt unser Modul für Ihre Projekte nutzen.

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart-Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

Falls Sie nach noch weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, E-Books, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.

https://az-delivery.de

Viel Spaß! I

mpressum

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