Inhalt

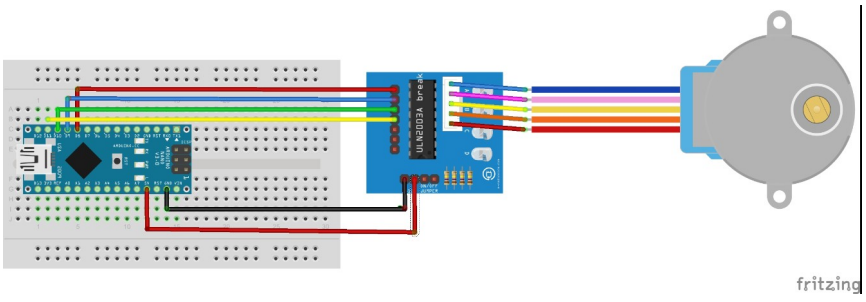
2.4. Stepper/Schrittmotoren

Schrittmotoren im Modelleisenbahnbereich kommen oft zur Anwendung, wenn es um „bewegte“ Elemente geht. Karussells, Kranbewegungen, Schiebebühnen, sich wiederholende Anwendungen und, und, und. Der Vorteil: sie lassen sich sehr genau ansteuern in Bezug auf Richtung, Winkel, Geschwindigkeit, Anzahl der Umdrehungen, was bei DC-Motoren nur schwer zu erreichen ist. Zum Einsatz kommen hier verschiedene Motoren, Motortreiberplatinen und Bibliotheken. Ich werde hier nur eine Auswahl vorstellen, das Angebot ist zu groß. Ihr werdet feststellen, dass sich die Codes bei den verschiedenen Boards und Motoren nicht gravierend unterscheiden. Deshalb habe ich dann auch darauf verzichtet sich wiederholende Befehlszeilen zu dokumentieren. Also – alles lesen!!

2.4.1. 28BYJ-48 Stepper Motor mit ULN2003 Treiber

Sehr beliebt und preiswert. Lässt sich gut verkabeln und sehr leicht installieren.
Der Motor ist ein 5-Draht-Stepper.
Im Weiteren zeige ich 3 Versionen der Anwendung,
Quelle mit vielen Zusatzinfos: https://www.makerguides.com/
Sehr zu empfehlen!! Von mir genutzt und mit deutschen Kommentaren versehen.

Verdrahtung:



Variante A: Unter Benutzung der **„stepper.h“** Bibliothek
Motor macht eine Umdrehung nach rechts, kurze Pause, eine nach links, also erst mal ganz einfach.

```
/* Beispiel 28BYJ-48 Stepper-Motor mit Motorteiber ULN2003 und Nano. Idee und Quelle:
https://www.makerguides.com */
// Einbinden der Arduino Stepper.h library:
#include <Stepper.h>
// Definition der Anzahl der Schritte für 1 Umdrehung (Vollschrittmodus):
const int stepsPerRevolution = 2048; //1024 wäre eine halbe Umdrehung, 512? richtig eine viertel
// Verdrahtung:
// Pin 8 to IN1 auf dem ULN2003 Treiberbaustein
// Pin 9 to IN2 auf dem ULN2003 Treiberbaustein
```

```
// Pin 10 to IN3 auf dem ULN2003 Treiberbaustein
// Pin 11 to IN4 auf dem ULN2003 Treiberbaustein
// Stepper-Objekt erzeugen "myStepper", Beachtet unbedingt die Pin-Reihenfolge:
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
void setup() {
  // Geschwindigkeit auf 5 U/min:
  myStepper.setSpeed(5);

  // zur Kontrolle auf seriellem Monitor mit 9600::
  Serial.begin(9600);
}
void loop() {
  // Schritt eine Umdrehung in eine Richtung:
  Serial.println("Uhrzeiger");
  myStepper.step(stepsPerRevolution);
  delay(500);

  // Schritt eine Umdrehung in die andere Richtung:
  Serial.println("gegen Uhrzeiger");
  myStepper.step(-stepsPerRevolution); // das Minus dreht Motorrichtung
  delay(500);
}
```

Fazit und zum Probieren:
Einfache Vorwärts/Rückwärtsbewegung mit geringen Einflussmöglichkeiten, aber geht.

Variante B: Einfache Rotation unter Benutzung der **„AccelStepper.h“** Bibliothek
Diese Bibliothek muss evt. erst nach dem bekannten Prinzip installiert werden. Download hier
https://www.makerguides.com/wp-content/uploads/2019/02/AccelStepper-1.59.zip
Danach über Sketch-Bibliothek einbinden-.zip Bibliothek hinzufügen diese installieren. Diese verfügt über mehr Funktionen, u.a. für Geschwindigkeit und Pendelmöglichkeit. Es lohnt sich die Beschreibung zu lesen. hier

Verdrahtung die gleiche, der Sketch mit Erklärung:

```
/* Beispiel 28BYJ-48 Stepper-Motor mit Motorteiber ULN2003 und Nano. Idee und Quelle:
https://www.makerguides.com */
// Einbinden der AccelStepper.h library:
#include <AccelStepper.h>

// Motorpin-Definition
#define motorPin1  8     // IN1 auf dem ULN2003 Treiberbaustein
#define motorPin2  9     // IN2 auf dem ULN2003 Treiberbaustein
#define motorPin3  10    // IN3 auf dem ULN2003 Treiberbaustein
#define motorPin4  11    // IN4 auf dem ULN2003 Treiberbaustein

// Definition Motorschnittstelle; 5-Drahtmotor im Halbschritt:
#define MotorInterfaceType 8
// entspricht 4096 Schritten
// Initialisieren Sie mit der Pin-Sequenz IN1-IN3-IN2-IN4, um die AccelStepper-Bibliothek mit dem
//Schrittmotor 28BYJ-48 zu verwenden: WICHTIG ist die Reihenfolge der Pins
AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3, motorPin2,
motorPin4);
```

```
 void setup() {
  // max. Geschwindigkeit begrenzen, mehr als 1000 sind unzuverlässig
  stepper.setMaxSpeed(1000);

  // zur Kontrolle auf seriellem Monitor mit 9600:
  Serial.begin(9600);
}
void loop() {
  // Drehzahl des Motors in Schritten pro Sekunde ein:
  stepper.setSpeed(500);
  // im Halbschrittmodus bedeutet das, 1 Umdrehung mit 500 Schritten/sek = ca. 7 U/Min.
  // Den Motor mit konstanter Drehzahl einstellen, wie von setSpeed () festgelegt:
  stepper.runSpeed();
}
```

Variant C: verbesserte Variant B

Mit dem folgenden Sketch können wir sowohl die Geschwindigkeit, Richtung als auch die Anzahl der Schritte / Umdrehungen steuern.

In diesem Fall dreht der Schrittmotor 1 Umdrehung im Uhrzeigersinn mit 500 Schritten/s, dann 1 Umdrehung gegen den Uhrzeigersinn mit 1000 Schritten/s und zuletzt 2 Umdrehungen im Uhrzeigersinn mit 1000 Schritten / s. Es wird mal wieder ein while() verwendet.

```
/* Beispiel 28BYJ-48 Stepper-Motor mit Motorteiber ULN2003 und Nano.
Idee und Quelle: https://www.makerguides.com
Steppermotor pendelnd
*/
// Einbinden der Arduino AccelStepper.h library:
#include <AccelStepper.h>

// Motorpin-Definition
#define motorPin1  8     // IN1 auf dem ULN2003 Treiberbaustein
#define motorPin2  9     // IN2 auf dem ULN2003 Treiberbaustein
#define motorPin3  10    // IN3 auf dem ULN2003 Treiberbaustein
#define motorPin4  11    // IN4 auf dem ULN2003 Treiberbaustein

// Definition Motorschnittstelle; 4 Drahtmotor im Halbschritt:
#define MotorInterfaceType 8

// Initialisieren der Pin-Sequenz IN1-IN3-IN2-IN4, um die AccelStepper-Bibliothek mit dem Schrittmotor 28BYJ-48 zu verwenden:
AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3, motorPin2, motorPin4);

void setup() {
  // max. Geschwindigkeit begrenzen:
  stepper.setMaxSpeed(1000);
}
void loop() {
  // , definierter Zähler:
  stepper.setCurrentPosition(0);
  //den Motor mit 500 Schritten / Sekunde vorwärts laufen, bis der Motor 4096 Schritte (1 Umdrehung) erreicht:
  while (stepper.currentPosition() != 4096) {
```

```
   stepper.setSpeed(500);
  // die Funktion mit voreingestellter Geschwindigkeit ausführen
   stepper.runSpeed();
  }
  delay(1000);
  //
  stepper.setCurrentPosition(0);
  //  den Motor mit 1000 Schritten / Sekunde rückwärts laufen lassen, bis der Motor -4096 Schritte (1
Umdrehung) erreicht.:
  while (stepper.currentPosition() != -4096) {
  // -800 kehrt Motorpolarität um = rückwärts
   stepper.setSpeed(-800);
   stepper.runSpeed();
  }
  delay(1000);
  // aktuelle Position auf 0 setzen:
  stepper.setCurrentPosition(0);
  // den Motor mit 1000 Schritten / Sekunde vorwärts laufen lassen, bis der Motor 8192 Schritte (2
Umdrehungen) erreicht:
  while (stepper.currentPosition() != 8192) {
  // wieder eine andere Geschwindigkeit
   stepper.setSpeed(900);
   stepper.runSpeed();
  }
  delay(3000);
}
```

Zu beachten wäre, dass man die Geschwindigkeiten vor dem setup definieren kann, um schneller diese zu ändern.

Variante D: Beschleunigen und verzögern/bremsen

Mit dem folgenden Sketch können wir die Bewegungen des Schrittmotors ohne komplizierte Codierung beschleunigen und verzögern. Der erste Abschnitt dieser Skizze ist der gleiche wie in Beispiel 1, aber das Setup und die Schleife unterscheiden sich.

Der Motor läuft zwei Umdrehungen mit einer Geschwindigkeit von 1000 Schritten pro Sekunde und einer Beschleunigung von 200 Schritten / Sekunde2 hin und her.

Im Setup müssen wir neben der Höchstgeschwindigkeit auch die Beschleunigung / Verzögerung definieren. Dazu verwenden wir die Funktion setAcceleration ().

Im Schleifenabschnitt des Codes habe ich eine andere Methode verwendet, um den Motor eine vordefinierte Anzahl von Schritten drehen zu lassen. Zuerst setze ich die Zielposition mit der Funktion moveTo (). Als nächstes verwenden wir einfach die Funktion runToPosition (), um den Motor mit der eingestellten Geschwindigkeit und Beschleunigung zur Zielposition laufen zu lassen. Der Motor bremst ab, bevor er die Zielposition erreicht. Näheres zu dieser Funktion in der Beschreibung der Biblio.

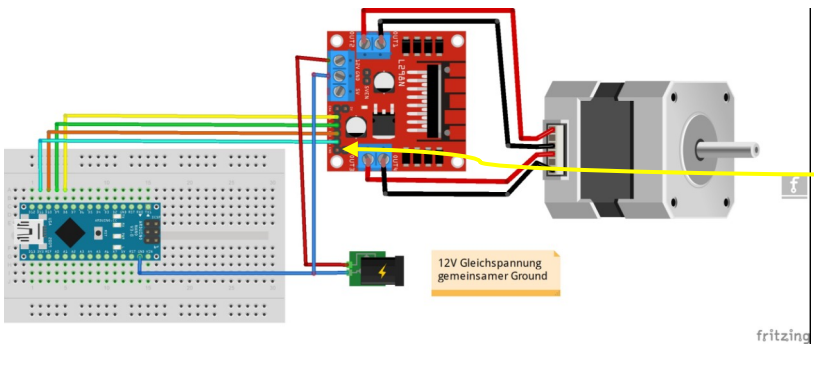
```
/* Beispiel 28BYJ-48 Stepper-Motor mit Motorteiber ULN2003 und Nano. Idee und Quelle:
https://www.makerguides.com */
// Einbinden der AccelStepper.h library:
#include <AccelStepper.h>

// Motorpin-Definition
#define motorPin1  8     // IN1 auf dem ULN2003 Treiberbaustein
#define motorPin2  9     // IN2 auf dem ULN2003 Treiberbaustein
#define motorPin3  10    // IN3 auf dem ULN2003 Treiberbaustein
```

```
#define motorPin4  11     // IN4 auf dem ULN2003 Treiberbaustein

// Definition Motorschnittstelle; 4 Drahtmotor im Halbschritt:
#define MotorInterfaceType 8
// Initialisieren der Pin-Sequenz IN1-IN3-IN2-IN4, um die AccelStepper-Bibliothek mit dem
Schrittmotor 28BYJ-48 zu verwenden:
AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3, motorPin2,
motorPin4);



void setup() {
  // Maximum begrenzen:
  stepper.setMaxSpeed(1000);
  // Einstellen der maximalen Beschleunigung in Schritten pro Sekunde ein ^ 2:
  stepper.setAcceleration(400);
}
void loop() {
  // Zielposition setzen:
  stepper.moveTo(8192);
  // Mit eingestellter Geschwindigkeit und Beschleunigung zur Position laufen:
  stepper.runToPosition();

  delay(1000);

  // und zurück zur Zielposition:
  stepper.moveTo(0);
  // Mit eingestellter Geschwindigkeit und Beschleunigung zur Position laufen:
  stepper.runToPosition();

  delay(1000);
}
```

Alles in Allem, sehr kurzer und leicht zu händelnder Code.

2.4.2. Motortreiber L298N und Steppermotor NEM17

Hier erfahrt ihr, wie ihr einen Schrittmotor mit dem Motortreiber L298N steuert. Diese Treiberplatine wird normalerweise zur Steuerung von Gleichstrommotoren verwendet, ist aber auch eine kostengünstige Alternative zur Steuerung von Schrittmotoren! Es kann sowohl die Drehzahl als auch die Drehrichtung der meisten Schrittmotoren, wie eines NEMA 17, steuern. Zu beachten ist, dass das Board mind. 12V Versorgungsspannung, also extern, benötigt. Daraus ergibt sich aber auch ein Vorteil, die Motoren sind nicht nur sehr leise, sondern auch leistungsstark. Da kann man schon mal ordentlich was bewegen!!
Im ersten Beispiel sehen wir uns die Stepper.h Arduino-Bibliothek an. Ich empfehle dringend, auch die Beispielcodes für die AccelStepper-Bibliothek am Ende dieses Tutorials zu lesen.
Erst mal die Verkabelung:



Nicht wirklich spektakulär, aber eben mit externer Stromversorgung ab 12V aufwärts. Bei Spannung > 12V muss der Jumper entfernt werden!

Variant A: 1 Umdrehung nach recht, kurze Pause – eine nach links mit 2 unterschiedlichen Geschwindigkeiten.
Sketch dazu:

```
/* Beispiel NEMA 17 Motor mit Motorteiber L198N und Nano. Idee und Quelle:
https://www.makerguides.com */
// Einbinden der stepper.h library:
#include <Stepper.h>

// Festlegen der Schritte/Umdrehung
const int stepsPerRevolution = 200;

// Definition der Nummern der Schritte/Umdrehung
Stepper myStepper = Stepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // setzen der Motorgeschwindigkeit (U/min)
Stepper myStepper = Stepper(stepsPerRevolution, 8, 9, 10, 11);
  // setzen der Motorgeschwindigkeit (U/min)
  myStepper.setSpeed(100);
}
void loop() {
  // Eine Umdrehung in einer Richtung
  myStepper.step(200);
  delay(2000);
  // Eine Umdrehung in die andere Richtung, etwas schneller (-200)
  myStepper.step(-200);
  delay(2000);
}
```

Variant B: wieder mit der AccelStepper.h, konstante Geschwindigkeit ohne Richtungswechsel
Eigentlich ganz einfach, manchmal braucht man eben auch so was.
Der Sketch dazu:

```
/* Beispiel NEMA 17 Stepper-Motor mit Motorteiber L298N und Nano. Idee und Quelle:
https://www.makerguides.com */

// Einbinden der AccelStepper.h library:
#include <AccelStepper.h>

// Schnittstelle definieren
#define MotorInterfaceType 4

// Erstellen einer neuen Instanz der AccelStepper-Klasse:
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);

void setup() {
  // Max. Geschwindigkeit begrenzen:
  stepper.setMaxSpeed(1000);
}
```

```
void loop() {
  //  die Drehzahl des Motors in Schritten pro Sekunde einstellen:
  stepper.setSpeed(500);
  // den Motor mit konstanter Drehzahl losrennen lassen, wie mit setSpeed () eingestellt:
  stepper.runSpeed();
}
```

Geht, läuft einfach. Ändert man stepper.setSpeed(500); in (-400) geht es in die andere Richtung. Erspart das Umstöpseln am Brett (haha).

Variante C: Geschwindigkeit, Richtung Anzahl Umdrehungen steuern

Jetzt wollen wir sowohl die Geschwindigkeit, Richtung als auch die Anzahl der Schritte / Umdrehungen steuern.

Dabei dreht der Schrittmotor 2 Umdrehungen im Uhrzeigersinn mit 200 Schritten / s, dann 1 Umdrehung gegen den Uhrzeigersinn mit 600 Schritten / s und zuletzt 3 Umdrehungen im Uhrzeigersinn mit 400 Schritten / s.

Der Sketch offenbart nichts Ungewöhnliches, ähnlich wie beim ULN2003 oben.

```
/* Beispiel NEMA 17 Stepper-Motor mit Motorteiber L298N und Nano. Idee und Quelle:
https://www.makerguides.com */

// Einbinden der AccelStepper.h library:
#include <AccelStepper.h>

// Schnittstelle definieren
#define MotorInterfaceType 4

// Erstellen einer neuen Instanz der AccelStepper-Klasse:
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);

void setup() {
  // Max. Geschwindigkeit begrenzen:
  stepper.setMaxSpeed(1000);
}

void loop() {
  // momentane Position auf 0 setzen
  stepper.setCurrentPosition(0);
  // Motor dreht mit 200 Schritten / Sekunde vorwärts , bis der er 400 Schritte (2 Umdrehungen)
erreicht:
  while (stepper.currentPosition() != 400)  {
    stepper.setSpeed(200);
    stepper.runSpeed();
  }
  delay(1000);
  // momentane Position auf 0 setzen
  stepper.setCurrentPosition(0);
  // Motor dreht mit 600 Schritten / Sekunde rückwärts , bis der er -200 Schritte (1 Umdrehungen)
erreicht:
    while (stepper.currentPosition() != -200)   {
    stepper.setSpeed(-600);
    stepper.runSpeed();
  }
  delay(1000);
  // momentane Position auf 0 setzen
```

```
  stepper.setCurrentPosition(0);
  // Motor dreht mit 400 Schritten / Sekunde vorwärts, bis er 600 Schritte (3 Umdrehungen) erreicht:
    while (stepper.currentPosition() != 600)  {
    stepper.setSpeed(400);
    stepper.runSpeed();
  }
  delay(3000);
}
```

Variante D: parallel zu 4.2.1 Beschleunigen und Bremsen
Der Motor läuft fünf Umdrehungen mit einer Geschwindigkeit von 200 Schritten pro Sekunde und einer Beschleunigung von 50 Schritten / Sekunde2 hin und her. Jeweils vor Erreichen der Endposition bremst er ab.
Sketch:

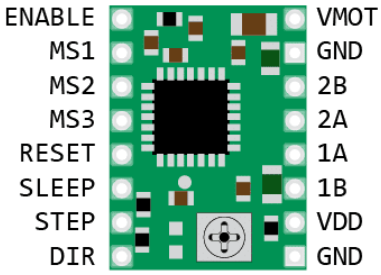
```
/* Beispiel NEMA 17 Stepper-Motor mit Motorteiber L298N und Nano.
   Beschleunigen und Bremsen, 5 U vorwärts, 5 rückwärts
   Idee und Quelle: https://www.makerguides.com
*/

// Einbinden der AccelStepper.h library:
#include <AccelStepper.h>

// Schnittstelle definieren
#define MotorInterfaceType 4

// Erstellen einer neuen Instanz der AccelStepper-Klasse:
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);

void setup() {
  // Max. Geschwindigkeit begrenzen:
  stepper.setMaxSpeed(200);
 //  maximale Beschleunigung in Schritten pro Sekunde^ 2:
  stepper.setAcceleration(50);
  }

void loop() {
  // Zielposition setzen
  stepper.moveTo(1000);
  // Motor startet zur Position mit eingstellter V
  stepper.runToPosition();
  delay(1000);
  // zurück zur Ausgangsposition:
  stepper.moveTo(0);
  // Motor startet zur Position mit eingestellter V
  stepper.runToPosition();
  delay(1000);
}
```

2.4.3. Motortreiber A4998 und Schrittmotor NEMA 17

Noch ein Treiber? Ja, und zwar ein ganz penibler. Schrittmotoren haben typischerweise eine Schrittgröße von 1,8 ° oder 200 Stufen pro Umdrehung, dies bezieht sich auf volle Stufen. Ein

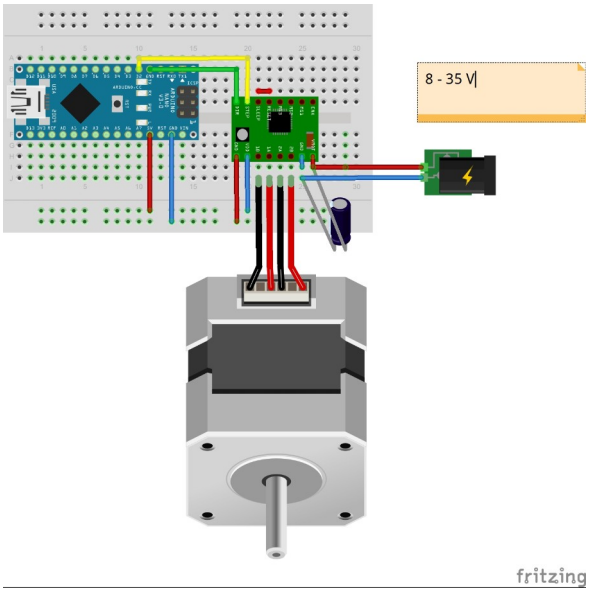
Mikroschritt-Treiber wie der A4988 ermöglicht höhere Auflösungen, indem er Zwischenschritt-positionen zulässt. Dies wird erreicht, indem die Spulen mit Zwischenstrompegeln erregt werden.

Wenn ihr beispielsweise einen Motor im Viertelschrittmodus antreibt, erhält der Motor mit 200 Schritten pro Umdrehung 800 Mikroschritte pro Umdrehung, indem vier verschiedene Strompegel verwendet werden.
Hier mal die PIN-Belegung.
Mit den Auswahlstiften für die Auflösung (Schrittgröße) (MS1, MS2 und MS3) könnt ihr eine der fünf Schrittauflösungen gemäß der folgenden Tabelle auswählen.

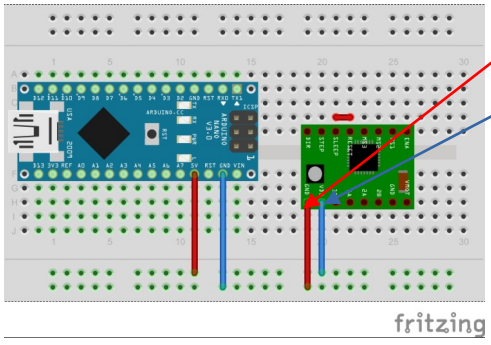


MS1	MS2	MS3	Mikroschrittauflösung
Niedrig	Niedrig	Niedrig	Voller Schritt
Hoch	Niedrig	Niedrig	1/2 Schritt
Niedrig	Hoch	Niedrig	1/4 Schritt
Hoch	Hoch	Niedrig	1/8 Schritt
Hoch	Hoch	Hoch	1/16 Schritt

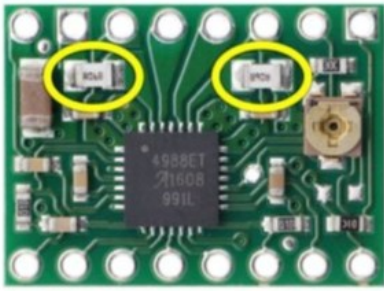
Je höher die Auflösung, umso ruhiger (und meistens auch leiser) läuft der Motor.
Alle drei Eingänge verfügen über interne 100-kΩ-Pulldown-Widerstände. Wenn ihr also die drei Mikroschritt-Auswahlstifte nicht angeschlossen lasst, erhaltet ihr einen Vollschrittmodus.
Auch bei diesem Board wir eine ext. Stromversorgung benötigt, min. 8 bis max. 35 V.
Hier die Verkabelung für Vollschritt (ohne M1-M3)



Um das A4988-Board vor zerstörerischen Spannungsspitzen zu schützen, insbesondere wenn Stromkabel verwendet werden, die länger als 5 cm sind, solltet ihr einen Elektrolytkondensator zwischen VMOT und GND anschließen (47 -100 µF) .
WICHTIG: Strombegrenzung einstellen!!
Bevor ihr mit der Programmierung des Arduino beginnt und den Treiber verwendet, müsst ihr eine wichtige Sache tun, die viele Leute vergessen: Stellt das aktuelle Limit ein!
Dieser Schritt ist nicht sehr kompliziert, aber unbedingt erforderlich, um den Schrittmotor und das Board zu schützen. Wenn ihr das nicht tut, kann der Motor mehr Strom ziehen, als er oder das Board verarbeiten kann. Dies kann wahrscheinlich einen oder beide beschädigen.
Um die Strombegrenzung einzustellen, müsst ihr eine Referenzspannung messen und das integrierte Potentiometer entsprechend einstellen. Dazu folgende Verkabelung:



Die ext. Stromversorgung abstöpseln, Messung zwischen Poti und Ground. Berechnet wird nach folgender Formel: Amper des Motor x 0.7 x 8 x Rs ohm = V. Der Rs ist in der Regel 0,02 bis 0,068 Ω. Evt. messen.



Beispiel:
Nema 17 1.2A Motor x 0.7 x8 x Stepper Driver A4988 (0.2 Rs ohm) = 1.344V
Nachdem an dem kleinen Poti der Wert eingestellt wurde, ein Sketch dazu:
Dieser steuert sowohl die Geschwindigkeit, die Anzahl der Umdrehungen als auch die Drehrichtung des Schrittmotors.
/* Beispielsketch zur Steuerung eines Schrittmotors mit A4988-Schrittmotortreiber und Arduino **ohne Bibliothek**. Weitere Informationen: https://www.makerguides.com * /

```
// Schrittmotoranschlüsse und Schritte pro Umdrehung definieren:

#define dirPin 2
#define stepPin 3
#define stepsPerRevolution 200

void setup() {
  // definition der Pins als output:
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop() {
  // Motordrehrichtung auf Uhrzeiger:
  digitalWrite(dirPin, HIGH);

  // Motor dreht 1 Umdrehung langsam:
  for (int i = 0; i < stepsPerRevolution; i++) {
    // Diese vier Zeilen ergeben einen Schritt:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }

  delay(1000);

  // Motordrehrichtung gegen Uhrzeiger:
  digitalWrite(dirPin, LOW);

  // Motor dreht 1 Umdrehung schnell:
  for (int i = 0; i < stepsPerRevolution; i++) {
    // Diese vier Zeilen ergeben einen Schritt:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(1000);
  }

  delay(1000);

  // Motordrehrichtung auf Uhrzeiger:
  digitalWrite(dirPin, HIGH);

  // Motor dreht 5 Umdrehung schneller:
```

```
 for (int i = 0; i < 5 * stepsPerRevolution; i++) {
  // Diese vier Zeilen ergeben einen Schritt:
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(500);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(500);
 }

 delay(1000);

 // Motordrehrichtung gegen Uhrzeiger:
 digitalWrite(dirPin, LOW);

 // Motor dreht 5 Umdrehung schneller:
 for (int i = 0; i < 5 * stepsPerRevolution; i++) {
  // Diese vier Zeilen ergeben einen Schritt:
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(500);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(500);
 }

 delay(1000);
}
```

Mit den integrierten Kommentaren sollte jetzt jeder klarkommen. Auch hier empfhele ich das verändern der Werte zum Probieren und „learning bei machen“. Die berühmte Methode **MUP** (Methode des unbekümmerten Probierens oder auch: mal guggen, was passiert).

So, das war viel (Arbeit). Ich habe alles selbst getestet und hoffe, ihr kommt klar. Wenn sich Fehler einschlichen haben, so war das gewollt ☺. Hinweise, Kritiken und Erfahrungen sind ausdrücklich erwünscht.
Das nächste Kapitel wird WiFi, Bluetooth und Infrarot…