Eine Schrittmotorsteuerung mit dem Arduino Nano (Teil 1)

Im heutigen Blog möchte ich Ihnen eine einfache Möglichkeit vorstellen, einen Schrittmotor präzise in seiner Geschwindigkeit zu steuern. Ein Schrittmotor unterscheidet sich von einem regulären Motor insofern, dass ein Schrittmotor präzise sowohl im Weg als auch in der Geschwindigkeit bei gleichbleibendem Drehmoment gesteuert werden kann. Eingesetzt werden kann der Schrittmotor und die heutige Steuerung zb. in einer Präsentationsdrehscheibe für Kunst- oder Wertobjekte, bei der der Schrittmotor die Drehscheibe steuert oder auch bei jedem anderen Anwendungszweck bei dem es auf gleichmäßige Kraft (Drehmoment) bei gleichmäßiger und präziser Geschwindigkeit ankommt.

Die vorgestellte Schaltung bzw. Schrittmotorsteuerung kann mit jedem 4 phasigen Schrittmotor im Vollschrittbetrieb (200 Schritte pro Umdrehung) zwischen 72 Umdrehungen/Min und 294 Umdrehungen /Min steuern. Die Steuerfrequenz bewegt sich dabei zwischen 241 Hz bis 980 Hz.

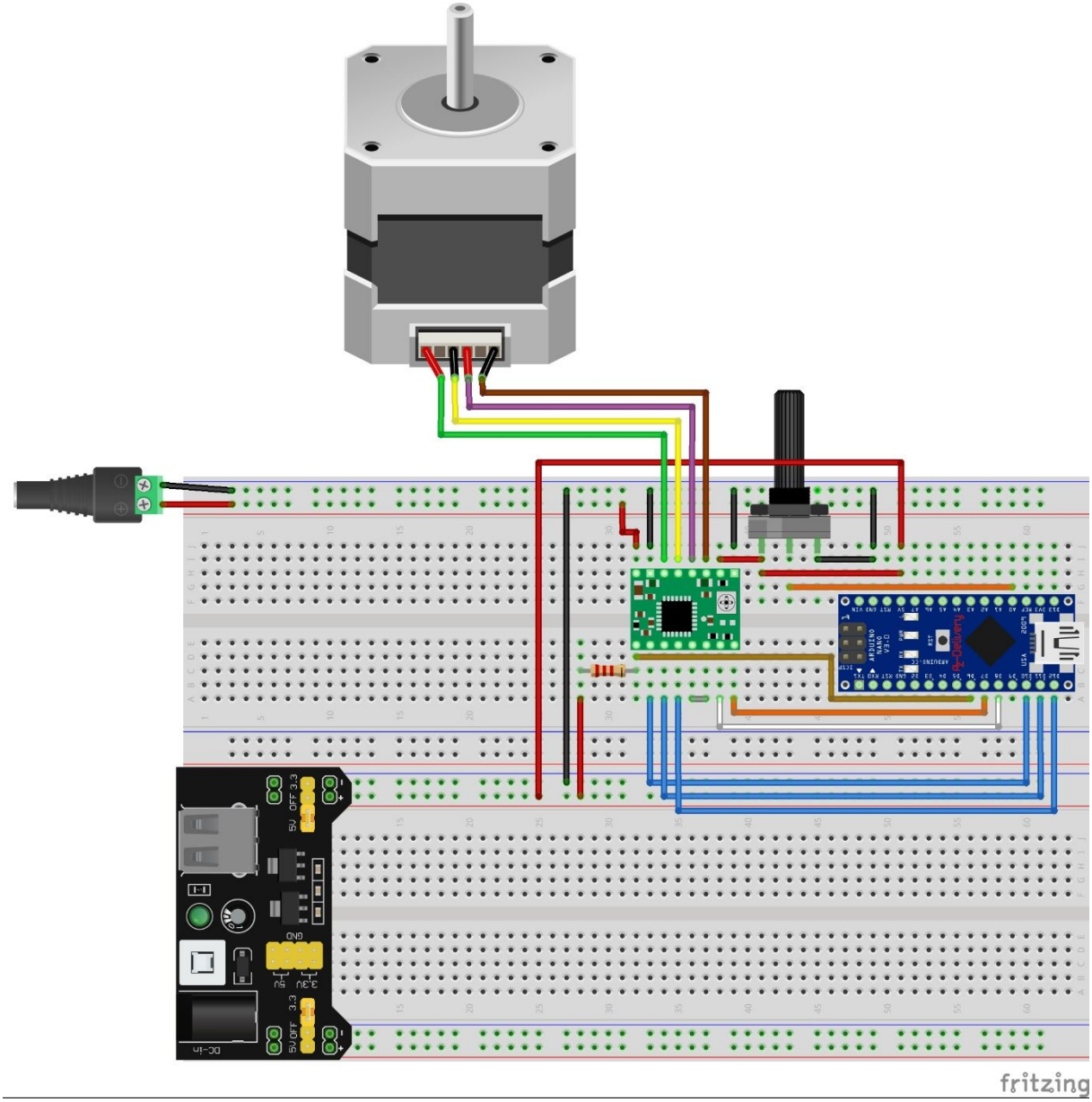
Wir brauchen für unser heutiges Projekt folgende Hardware:

Anzahl	Bezeichnung	Anmerkung
1	10 K Ohm Drehpotentiometer	
1	Arduino Nano	
1	Schrittmotor Nema 17 45Ncm	1.5A 12V 4-Draht 1.8 Deg
1	Widerstand 2,2 KOhm	
1	A4988 Schrittmotor-Treiber-Modul	Kühlkörper benutzen
1	YwRobot Power Supply	

Herzstück der Steuerung ist der Baustein A4988. Dieser elektronische Schrittmotortreiber erzeugt aus einem digitalen Takt- und einem Richtungssignal die nötige Phasenabfolge (Sequenz) im Betriebsspannungsbereich des Motors (12 Volt), um das für den Schrittmotor nötige Drehfeld zu erzeugen. Da der Baustein A4988, wie alle elektronischen Schrittmotortreiber im Schaltbetrieb eine H-Brücke schaltet bzw. damit arbeitet, ist die Verlustleistung des Bausteins sehr gering. Alternativen zu dem Baustein A4988 sind die Bausteine L298P, L6219DS, UC3717AQ, A4988, TMC2100 und TMC2208. Diese arbeiten nach dem gleichen Prinzip. Empfehlenswert ist hierzu die Lektüre unseres E-Books für den Baustein A488. Dieses kann hier heruntergeladen werden. Schrittmotoren, wie diesen, den wir in unserem heutigen Blog verwenden, haben zwei galvanisch getrennte Spulen A und B, die in 4 verschiedenen Schritten mit dem Baustein A4988 angesteuert werden können- Diese sind:

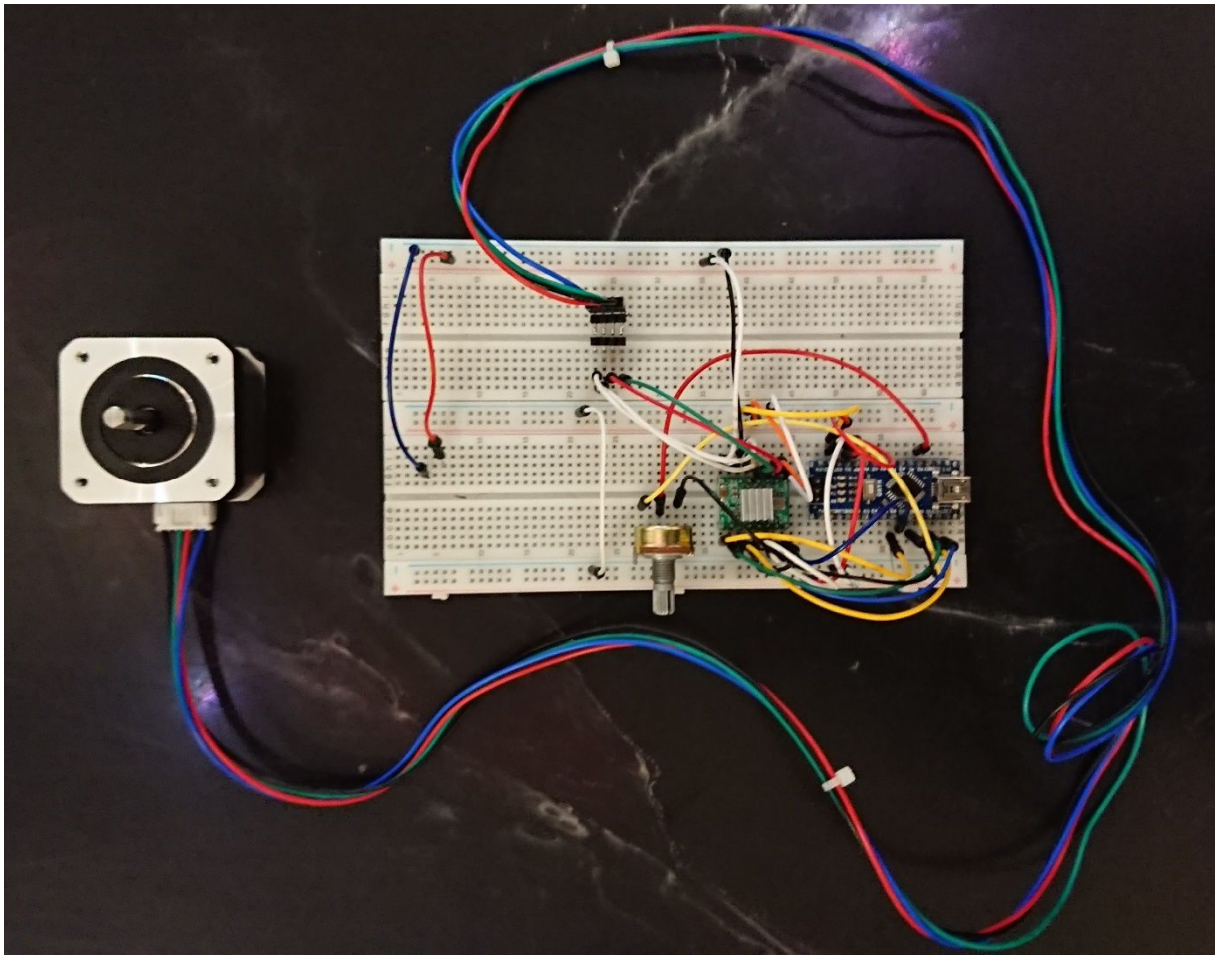
Schrittgröße	Schritte pro Umdrehung
Vollschritt	200
Halbschritt	400
Viertelschritt	800
Achtelschritt	1600

Aufgrund der bereits hohen Schrittzahl (200) des Schrittmotors bereits im Vollschrittbetrieb bewegt sich der Rotor des Schrittmotors mit jedem Schritt nur mit einem Winkel von 1,8 Grad vorwärts. Im Schrittmotorenbereich sind heute 24 bis 400 Schritte pro Umdrehung üblich bzw. erhältlich. Wegen des relativ hohen Drehmoments von Schrittmotoren im Allgemeinen lassen sich viele Anwendungen ohne zusätzlich notwendiges Getriebe realisieren. Um die Steuerung für den Schrittmotor aufzubauen verbinden wir die Komponenten wie auf folgendem Bild gezeigt, miteinander:



Für die Versorgung der Schaltung werden zwei Spannungen (12 Volt und 5 Volt) benötigt, da die Logik mit 5 Volt und der Motor mit 12 Volt arbeitet. Im oberen Bild ist die 12 Volt Versorgung des Motors in der oberen Stromschiene mit Hilfe einer Buchse zum Anschluss eines Netzteils realisiert, die 5 Volt Seite dagegen mit einem YwRobot Power Supply Board, das aus einer Eingangsspannung von 12 Volt die benötigte Spannung von 5 Volt Mithilfe eines Linearreglers erzeugt. Es kann somit die ganze Schaltung an ein Netzteil mit einer Ausgangsspannung von 12 Volt und einer Ausgangsleistung von 24 Watt angeschlossen werden.

Zur Visualisierung zeige ich hier den Aufbau der Schaltung auf einem Breadboard. Da bei mir die Stromversorgung ein Labornetzteil übernimmt, das parallel verschiedene Spannungen und Ströme zur Verfügung stellen kann, verzichte ich auf nachfolgendem Foto auf die Benutzung des YwRobot Power Supply Boards.



(Aufbau Schrittmotorsteuerung auf dem Breadboard ohne des YwRobot Power Supply)

Nachdem wir die Schaltung aufgebaut haben, können wir nachfolgenden Code mithilfe der Arduino IDE auf unseren Arduino hochladen:

```
/*  Einfache Schrittmotor Steuerung
 *  von Tobias Kuch 2020 https://github.com/kuchto  tobias.kuch@googlemail.com
 *
 *  Version: 1.0
 *
 *  Funktion:
 *  Steuert die Geschwindigkeit eines Schrittmotor im Full Step Modus im Frequenzbereich von 241 bis 980 Hz.
 *  Bei einem Schrittmotor mit einem Schrittgrad von 1,8 Grad und von 200 Schritten pro Umdrehung ergibt sich eine
 *  Geschwindigkeitseinstellbarkeit von min. 72 U/Min bis max. 294 U/Min
 *
 *  Verwendete Hardware:
 *    - Arduino Nano V3
 *    - Potentiometer (10KOhm)
 *    - externe Spannungsversorgung 5V und 12 Volt für Steppermotor
 *    - Steppermotor
 *    - Stepper Motor IC
 *
 *  Verwendete Bibliotheken:
 *    - keine
 *
 */


#define MOTORDIRPIN 7
#define MOTORSPEED 8
#define MOTORENABLE 6
#define A4988MS1 10
#define A4988MS2 11
#define A4988MS3 12

#define SPEEDCONTROL A0

byte valuea = 0;
byte oldvaluea = 0;


ISR(TIMER1_COMPA_vect){ //Timer1 interrupt. Schaltet Pin 8 um.
 // Weitere Infos auf: https://www.mikrocontroller.net/articles/AVR-Tutorial:_Timer
 digitalWrite(MOTORSPEED,!(digitalRead(MOTORSPEED)));// Aufruffrequenz maximal: 2.604 Khz
 //Frequenz an Pin 8(Hz) = (Arduino clock speed 16,000,000Hz) / (prescaler * (compare match register + 1)) / 2
}

void setup() {
 pinMode(MOTORSPEED, OUTPUT);
 pinMode(MOTORDIRPIN, OUTPUT);
 pinMode(MOTORENABLE, OUTPUT);
 pinMode(A4988MS1, OUTPUT);
 pinMode(A4988MS2, OUTPUT);
 pinMode(A4988MS3, OUTPUT);
 digitalWrite(A4988MS1,LOW); // FullStep
```

```
 digitalWrite(A4988MS2,LOW); // FullStep
 digitalWrite(A4988MS3,LOW); // FullStep
 digitalWrite(MOTORENABLE,LOW);
 cli();                          //stoppe alle Interrupts
 TCCR1A = 0;                     // set entire TCCR1A register to 0 TCCR - Timer/Counter Control
Register
 TCCR1B = 0;                     // Setze Timer/Counter Control Register TCCR1B auf 0
 TCCR1B |= (1 << WGM12);             // Schalte Clear Timer on Compare (CTC) Modus ein
 // TCCR1B |= (1 << CS12) | (1 << CS10);  // Setze CS10 und CS12 Bit auf 1 für den 1024 Prescaler.
Maximalfrequenz: 7.812 Khz
 TCCR1B |= (1 << CS12);              // Setze CS12 Bit auf 1 für den 256 Prescaler.
 TCNT1  = 0;                     // Initialisiere Zähler/Zeitgeber Register Wert auf 0
 OCR1A = 130;//  Aufruffrequenz Timer 1  241 Hz * 2
 TIMSK1 |= (1 << OCIE1A); // Erlaube Timer compare interrupt TIMSK - Timer/Counter Interrupt
Mask Register
 sei();//allow interrupts
}

void loop() { // Hauptschleife - Abfragen des Potentiometers
 int Gb = analogRead(SPEEDCONTROL);
 byte valuea = map(Gb,0,1023,31,130);
 if (valuea != oldvaluea)
  {
  oldvaluea = valuea ;
  cli();//stop interrupts
  OCR1A = valuea;
  if ( TCNT1 > OCR1A )
    {
    TCNT1 = OCR1A -1;//initialize counter value to 0
    }
  sei();//allow interrupts
  }
}
```

Funktionsweise des Codes:

Um die Funktionsweise zu beschreiben, setze ich voraus das die Funktionsweise von Interrupts bei Micro Controllern im Grundprinzip bekannt ist. Der Baustein A488 hat für die Steuerung eines Schrittmotors die wichtigen Anschlüsse „Step“ und „Dir.“ Diese sind an dem Arduino an den Pins 7 und 8 angeschlossen. Im Code verwenden wir dafür die Definitionen MOTORDIRPIN (Pin 7) und MOTORSPEED (Pin 8). Die Geschwindigkeit des Schrittmotors ist also in unserem Falle proportional zu der Frequenz (F) die eir an den Pin 8 anlegen. Zur Generierung der notwendigen Frequenz setzen wir den Timer 1 Interrupt des Arduinos im Clear Timer on Compare Match (CTC Mode) ein. In diesem Interrupt Modus wird das Register OCR1A bei jeder Erhöhung des Zählers verglichen. Stimmen beiden Werte, die des Zählers und des Registers OCR1A überein, wird ein Interrupt ausgelöst.

Dies nutzen wir in unserem Programm aus um eine gleichmäße Frequenz auf dem Pin 8 auszugeben. Die Höhe der ausgegebenen Frequenz bestimmt sich dabei durch den Wert des Registers OCR1A.

Dieser Wert wird zur Laufzeit des Programmes an die Einstellung des Potentiometers zur Geschwindigkeitssteuerung angepasst.

Im zweiten Teil der Reihe werden wir Steuerungsmöglichkeiten für die Drehrichtung des Schrittmotors als auch der Schritttiefe (Vollschritt, Halbschritt) hinzufügen, und so den Schrittmotor noch weiter kontrollierbar machen. Ich wünsche Ihnen nun viel Spaß beim nachbauen und freue mich auf einen Besuch auf meiner GitHub Seite unter https://github.com/kuchto.