Weitere Steuerungsmöglichkeiten für unsere Schrittmotor (Teil 2)

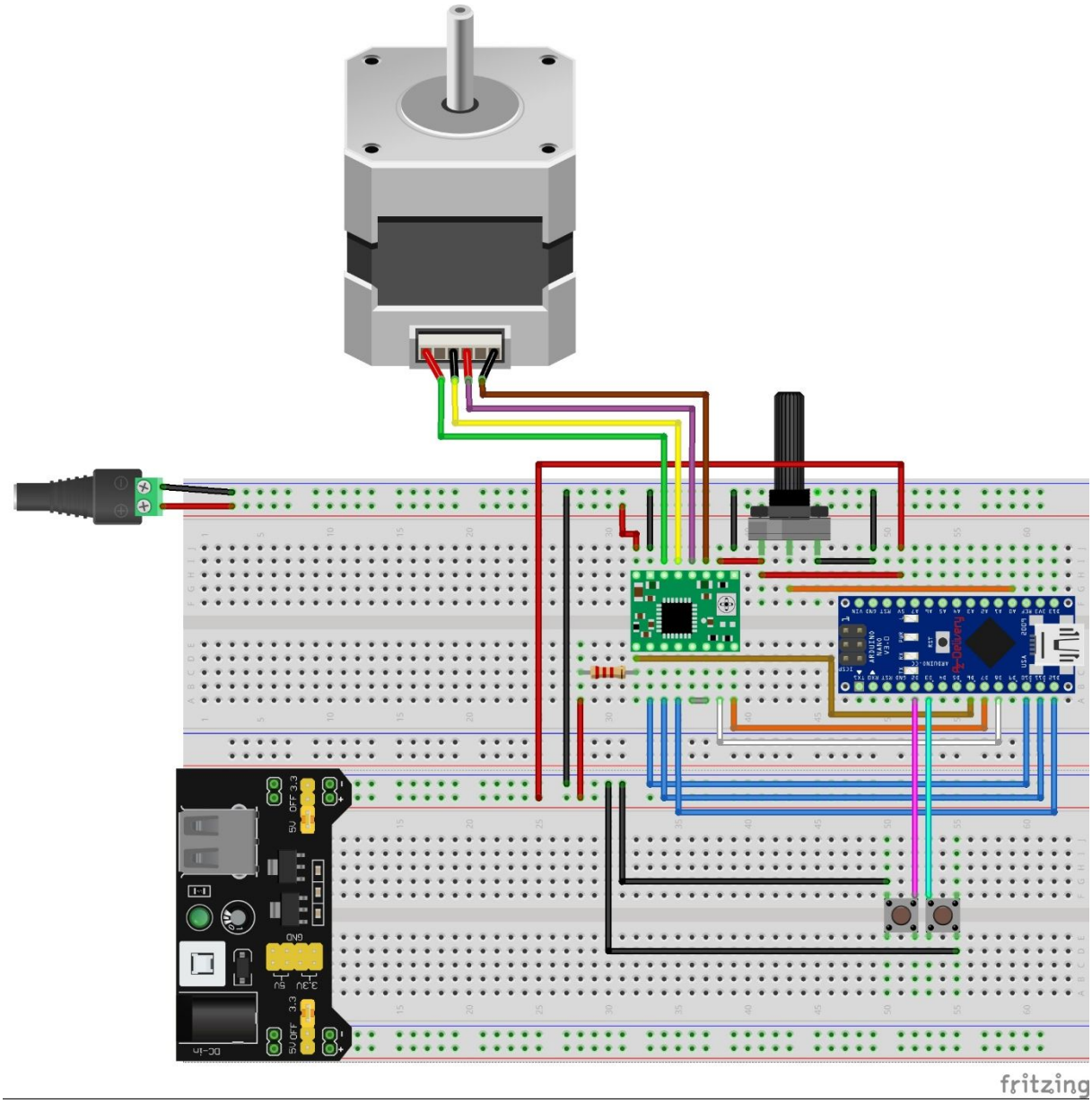
Im heutigen letzten Teil unserer Miniblogreihe zur Steuerung eines Stepper Motors erweitern wir unsere Schaltung um einige neue Steuerungsoptionen. Es soll so möglich werden, die Drehrichtung des Motors zu steuern und auch die Schrittlänge bzw. die Schritteinstellungen des Treiberchips von Vollschritt, Halbschritt, Viertelschritt, Achtelschritt und Sechzehntel Schritt zu nutzen. Alle Steuerungsmöglichkeiten, die der Treiberbaustein A4988 bietet, werden jetzt nutzbar gemacht.

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	
2	Taster	

Herzstück der Steuerung ist auch diesmal 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. Zur Steuerung der Drehrichtung und der Schrittbreite besitzt dieser weitere TTL kompatible Pins namens Dir und MS1 bis MS3. Diese werden wir im heutigen Blog mithilfe des Arduinos und zwei zusätzlichen Tastern ansteuern.

Um die Steuerung für den Schrittmotor aufzubauen verbinden wir die Komponenten wie auf folgendem Bild gezeigt, miteinander:



Im Vergleich zu Teil 1 wurden die Schaltung um zwei Taster zur Steuerung der fehlenden Parameter, Richtung und Schrittlänge, hinzugefügt. Die Steuerung ist dabei zwischen den beiden Tastern wie folgt aufgeteilt:

Taster	Port an Arduino / Variable	Funktion
1	2 MOTORGEARSWITCH	Schaltet zwischen Schrittlängen Voll-Halb-Achtel-Sechzehtel
2	3 MOTORDIRSWITCH	Schaltet Rechts-/Linkslauf und Halt

Bevor aber nun die neuen Funktionen über die Taster steuerbar sind, muss die Firmware aktualisiert werden. Wir laden daher folgende Programm hoch:

```
/* Einfache Schrittmotor Steuerung
 *  von Tobias Kuch 2020 https://github.com/kuchto tobias.kuch@googlemail.com
 *
 *  Version: 1.0
 *
 *  Funktion:
 *  Steuert die Geschwindigkeit eines Schrittmotor in einem durch einen Taster einstellbaren
 *  Schrittmodus im Frequenzbereich von 241 bis 980 Hz.
 *  Ebenso ist durch einen weiteren Taster die Drehrichtung umschaltbar
 *  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
 *    - 2 Taster
 *
 *  Verwendete Bibliotheken:
 *    - keine
 *
 */


#define MOTORDIRPIN 7
#define MOTORSPEED 8
#define MOTORENABLE 6
#define MOTORDIRSWITCH 3
#define MOTORGEARSWITCH 2

#define A4988MS1 10
#define A4988MS2 11
#define A4988MS3 12

#define SPEEDCONTROL A0

byte valuea = 0;
byte oldvaluea = 0;
bool DirMemory = false;
bool GearMemory = false;
byte Motorstatus = 2;
byte Motorgear = 1;
byte GearConrolReturn = 0;
byte ReturnGear = 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 MotorSpinUP()
{
digitalWrite(A4988MS1,HIGH); // SixteenthStep
digitalWrite(A4988MS2,HIGH); // SixteenthStep
digitalWrite(A4988MS3,HIGH); // SixteenthStep
delay(50);
digitalWrite(A4988MS1,HIGH); // EightStep
digitalWrite(A4988MS2,HIGH); // EightStep
digitalWrite(A4988MS3,LOW); // EightStep
delay(100);
digitalWrite(A4988MS1,LOW); // QuaterStep
digitalWrite(A4988MS2,HIGH); // QuaterStep
digitalWrite(A4988MS3,LOW); // QuaterStep
delay(200);
digitalWrite(A4988MS1,HIGH); // HalfStep
digitalWrite(A4988MS2,LOW); // HalfStep
digitalWrite(A4988MS3,LOW); // HalfStep
delay(300);
digitalWrite(A4988MS1,LOW); // FullStep
digitalWrite(A4988MS2,LOW); // FullStep
digitalWrite(A4988MS3,LOW); // FullStep
}

void MotorDirectionControl(byte Actualgear)
{
bool dirw = digitalRead(MOTORDIRSWITCH);
if ((!dirw) && (!DirMemory))
 {
 Motorstatus++;
 DirMemory = true;
 Serial.print ("Motorstatus: ");
 Serial.println (Motorstatus);
 if (Motorstatus == 1)
  {
  digitalWrite(MOTORDIRPIN,HIGH);
  digitalWrite(MOTORENABLE,LOW);
  if ((Actualgear == 5) && (oldvaluea < 40))
 //if (Actualgear == 5)
    {
    MotorSpinUP();
    Serial.println ("MotorSpinUp");
    }
  }
 if (Motorstatus == 2)
  {
```

```
   digitalWrite(MOTORENABLE,HIGH);  // Motor Stop
   }
  if (Motorstatus == 3)
   {
   digitalWrite(MOTORDIRPIN,LOW);
   digitalWrite(MOTORENABLE,LOW);
   if ((Actualgear == 5) && (oldvaluea < 40))
   //if (Actualgear == 5)
    {
    MotorSpinUP();
    Serial.println ("MotorSpinUp");
    }
   }
  if (Motorstatus == 4)
   {
   digitalWrite(MOTORENABLE,HIGH); // Motor Stop
   Motorstatus = 0;
   }
  delay(200); // Entprellung Taster
  }
if ((dirw) && (DirMemory)) { DirMemory = false; }
}


byte MotorGearControl()
{
bool gearw = digitalRead(MOTORGEARSWITCH);
if ((!gearw) && (!GearMemory))
 {
 Motorgear++;
 GearMemory = true;
 Serial.print ("Motorgang: ");
 Serial.println (Motorgear);
 if (Motorgear == 1)
  {
  digitalWrite(A4988MS1,HIGH); // SixteenthStep
  digitalWrite(A4988MS2,HIGH); // SixteenthStep
  digitalWrite(A4988MS3,HIGH); // SixteenthStep
  ReturnGear = 1;
  }
 if (Motorgear == 2)
  {
  digitalWrite(A4988MS1,HIGH); // EightStep
  digitalWrite(A4988MS2,HIGH); // EightStep
  digitalWrite(A4988MS3,LOW); // EightStep
  ReturnGear = 2;
  }
 if (Motorgear == 3)
  {
  digitalWrite(A4988MS1,LOW); // QuaterStep
  digitalWrite(A4988MS2,HIGH); // QuaterStep
  digitalWrite(A4988MS3,LOW); // QuaterStep
  ReturnGear = 3;
```

```
   }
  if (Motorgear == 4)
   {
   digitalWrite(A4988MS1,HIGH); // HalfStep
   digitalWrite(A4988MS2,LOW); // HalfStep
   digitalWrite(A4988MS3,LOW); // HalfStep
   ReturnGear = 4;
   }
  if (Motorgear == 5)
   {
   digitalWrite(A4988MS1,LOW); // FullStep
   digitalWrite(A4988MS2,LOW); // FullStep
   digitalWrite(A4988MS3,LOW); // FullStep
   ReturnGear = 5;
   Motorgear = 0;
   }
delay(200); // Entprellung Taster
  }
if ((gearw) && (GearMemory)) { GearMemory = false; }
return ReturnGear;
}

void setup() {

 Serial.begin(9600);
 pinMode(MOTORSPEED, OUTPUT);
 pinMode(MOTORDIRPIN, OUTPUT);
 pinMode(MOTORENABLE, OUTPUT);
 pinMode(MOTORDIRSWITCH,INPUT_PULLUP);
 pinMode(MOTORGEARSWITCH,INPUT_PULLUP);
 pinMode(A4988MS1, OUTPUT);
 pinMode(A4988MS2, OUTPUT);
 pinMode(A4988MS3, OUTPUT);
 digitalWrite(A4988MS1,HIGH); // SixteenthStep
 digitalWrite(A4988MS2,HIGH); // SixteenthStep
 digitalWrite(A4988MS3,HIGH); // SixteenthStep
 digitalWrite(MOTORENABLE,HIGH);
 digitalWrite(MOTORDIRPIN, HIGH);
 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
  }
 GearConrolReturn = MotorGearControl();
 MotorDirectionControl(GearConrolReturn);

}
```

Ich wünsche viel Spaß beim Nachbauen und beim Experimentieren mit dem spannenden Feld der Schrittmotoren. Eingesetzt kann die Schrittmotorsteuerung im Modellbau, in der Dekorationstechnik (z.B. als Antrieb für einen Präsentationsdrehteller) oder auch zum Bau eigener Modellautos. In Verbindung mit zusätzlichen Endschaltern und eine kleine Erweiterung des Codes währe z.B auch eine weitere Anwendungsmöglichkeit als Vorhangsteuerung vorstellbar.