Motorsteuerung mit Microcontollern

Proseminar Mikrocontroller und eingebettete Systeme WS2014/2015

Clemens Niemeyer
Lehrstuhl für Echtzeitsysteme und Robotik
Fakultät für Informatik
Technische Universität München
E-Mail: clemens.niemeyer@campus.lmu.de

Zusammenfassung

In dieser Arbeit werden drei grundlegende Arten von Motoren vorgestellt, und es wird gezeigt wie diese mittels Mikrocontroller gesteuert werden können. Dabei wird im Einzelnen auf die Regelung eines permanent erregten Gleichstrommotors mittels H-Brücke, die Ansteuerung eines Servomotors mittels Puls-Weite-Modulation und Steuerung eines bipolaren Schrittmotors mit und ohne Motortreiber eingegangen. Abschließend wird am Beispiel einer 3-Achsen-CNC-Fräse die Steuerung von Schrittmotoren mithilfe von G-Code, grbl und einem Arduino vorgestellt.

Index Terms

arduino — motor controls — microcontroller — stepper motor — dc motor — servo — h-bridge — pwm

I. Motorsteuerung macht es möglich

Intel brachte 1971 mit dem ”Intel 4004” den ersten Mikrocontroller (MCU) auf den Markt [1]. Heute sind sie aus unserem Alltag nicht mehr wegzudenken. Vom Bürodrucker über die Kaffemaschine bis hin zum Kinderspielzeug überall finden sie Anwendung. Vermehrte öffentliche Aufmerksamkeit bekamen MCUs im Zusammenhang mit der aktuellen Maker-Bewegung und dem damit einhergehenden Trend der digitalen Herstellung [2]. Dabei wurde meist ein besonderes Augenmerk auf die verwendeten Maschinen wie z.B. Lasercutter, CNC-Fräsen und 3D-Drucker gelegt. Die genannten Maschinen zeichnen sich vor allem durch ihre Fähigkeit aus, Motoren mit höchster Präzision steuern und am Computer entworfene Modelle exakt nachbilden zu können. Im Folgenden versucht diese Arbeit, einen Überblick über die gängigsten Motortypen und ihre Steuerung zu geben, sowie am Beispiel einer 3-Achsen-CNC-Fräse die Umsetzung einer komplexen Motorsteuerung in der Praxis aufzuzeigen. Der Fokus wird dabei auf die Grundprinzipien gelegt, weswegen hier dargestellte Schaltkreise nicht eins zu eins technisch umgesetzt werden können und sollten.

II. Bewegung

Stellvertretend für alle Gleichstrommotoren wird am Beispiel eines permanent erregten Gleichstrommotors (Abb. 1) gezeigt wie dieser mithilfe eines MCUs gestartet, seine Geschwindigkeit reguliert und seine Position bestimmt werden kann. Die Unterschiede und Gemeinsamkeiten zu anderen Gleichstrommotoren sind ausführlich in [3] beschrieben.

Ein permanent erregter Gleichstrommotor besteht in der Regel aus zwei um dieselbe Achse gelagerten Bauteilen; einem um diese Achse drehbaren Teil, dem Rotor, und einem unbeweglichen Teil, dem Stator. Wie in Abb. 2 dargestellt, besteht der Stator aus einem permanenten Magneten und umschließt den Rotor. Sobald elektrischer Gleichstrom durch die Spule fließt, wirkt eine Lorentzkraft und erzeugt somit ein Drehmoment. Die Polung der Spule wird mitthilfe eines Kommutators und zwei Schleifkontakten alle halbe Umdrehung umgekehrt, sodass die Lorentzkraft immer in Drehrichtung wirkt [4, S. 37ff.].

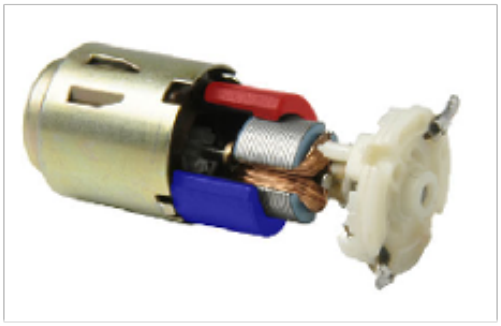

Abbildung 1: Explodierte Ansicht eines Gleichstrommotors

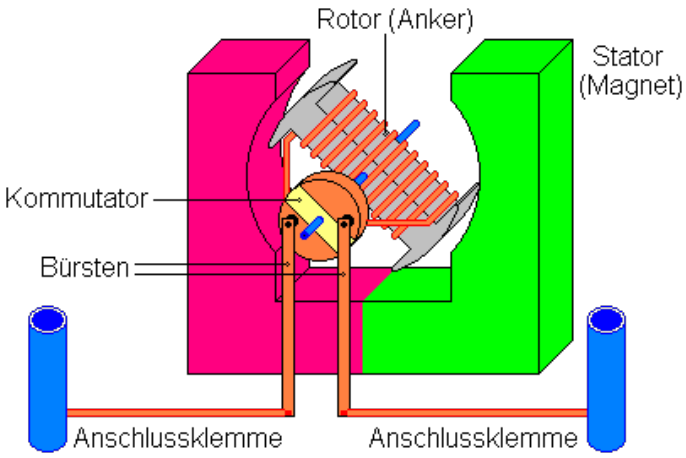

Abbildung 2: Aufbau eines permanent erregten Gleichstrommotors

A. *Motor starten*

Zum Starten des Motors wird ausschließlich Strom benötig. Dazu wird ein NPN-Bipolartransistor (Transistor) als Schalter eingesetzt. Wird an seiner Basis (B) eine ausreichend hohe[1] positive Spannung angelegt, kann Strom vom Collector (C) zum Emitter (E) fließen, siehe Abb. 3.

Wird ein Pin mit der Basis des Transistors, wie im Schaltkreis in Abb. 4 dargestellt, verbunden, kann mit dem Anlegen einer positiven Spannung - dem Setzen des Pins auf logisch 1 - der Transistor geöffnet und damit der Motor gestartet werden. Er beginnt auf seine maximale Geschwindigkeit zu beschleunigen, bis der Pin wieder auf logisch 0 gesetzt wird. In diesem Fall sperrt der Transistor, und der Motor wird langsamer. Codebeispiel in Arduino:

```
digitalWrite(PinNumber, HIGH); // Motor an
digitalWrite(PinNumber, LOW);  // Motor aus
```

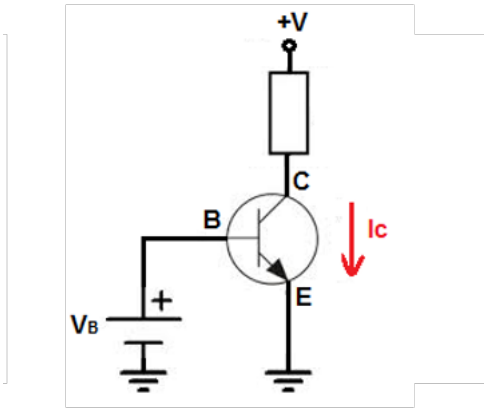

Abbildung 3: NPN-Bipolartransistor Funktionsweise [7, 124 ff.]

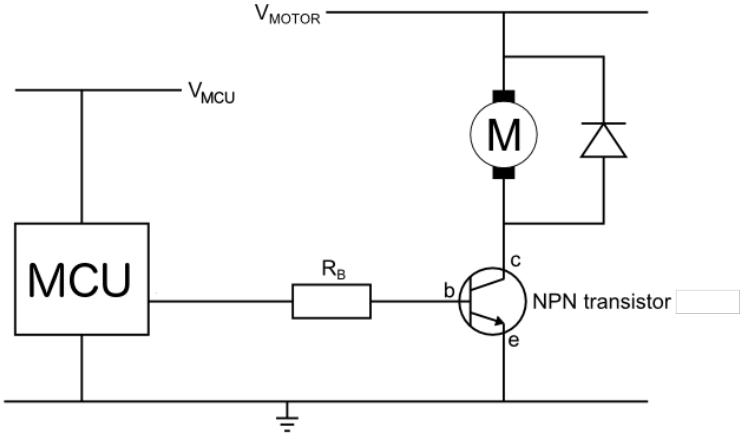

Abbildung 4: Schaltkreis zum Motor starten mittels Mikrocontroller (MCU)

[1] ist vom Verwendeten Transistor und der angelegten Spannung abhänging [5] [6, S. 80]

B. Geschwindigkeit regulieren

Die Geschwindigkeit eines Motors ist direkt proportional zur angelegten Spannung [8, S. 180]. Dies ermöglicht eine Geschwindigkeitsregulierung, indem ein Signal als Puls gesendet wird. Ein Impuls der sogenannten Puls-Weite-Modulation (PWM) besteht aus einem bestimmten Verhältnis von logisch 1 zu logisch 0, siehe 5. Liegt z.B. ein Anteil von 50 % logisch 1 vor, ist der Transistor nur die Hälfte der Zeit geöffnet und sinkt die am Motor angelegte Spannung auf die Hälfte, womit sich der Motor mit halber[2] Geschwindigkeit dreht. Die meisten MCUs stellen PWM an einigen Pins zur Verfügung [10]. Auf Arduino-Boards sind sie mit einer Tilde oder "PWM"gekennzeichnet und haben eine Auflösung von 8 Bit. Im Code wird beim Setzen des Signals eine Zahl zwischen 0 und 255 angegeben, um den Logisch-1-Anteil des Signals festzulegen. Soll sich der Motor also wie oben erwähnt nur halb so schnell drehen, wird das entsprechende Signal programmiert mit:

```
analogWrite(PinNumber,127); // Motor an mit halber Geschwindigkeit
```

C. Richtungswechsel

Um die Richtung des Motors zu wechseln, muss der Stromfluss in der Spule umgekehrt werden. Dies wird mithilfe von vier Transistoren, die wie in Abb. 6 miteinander verbunden werden, einer sogenannten H-Brücke, bewerkstelligt [11, 92 ff.]. Werden nur Pin A und D auf logisch 1 gesetzt, fließt der Strom von links nach rechts durch den Motor. Werden hingegen nur Pin B und C auf logisch 1 gesetzt, fließt er in die entgegengesetzte Richtung. Es dürfen niemals zwei Pins der gleichen Seite auf logisch 1 gesetzt werden, da es sonst zu einem Kurzschluss kommt. Am sichersten ist es, entsprechende Vorkehrungen auf Hardwareseite zu treffen bzw. fertige H-Brücken wie die L289 zu verwenden [11, S. 97] [12]. Code Beispiel:

```
digitalWrite(PinB, LOW);   digitalWrite(PinC, LOW);
digitalWrite(PinA, HIGH);  digitalWrite(PinD, HIGH); // Motor an CW
...
digitalWrite(PinB, HIGH);  digitalWrite(PinC, HIGH);
digitalWrite(PinA, LOW);   digitalWrite(PinD, LOW);  // Motor an CCW
```

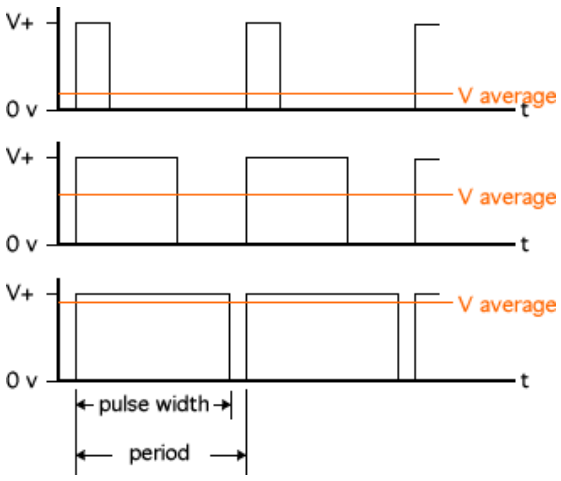

Abbildung 5: Puls Weite Modulation

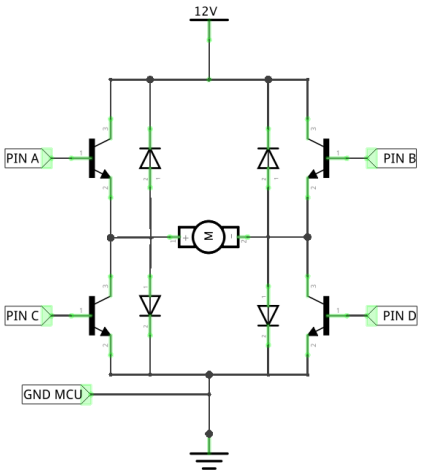

Abbildung 6: H-Brücke

[2] genaue Geschwindigkeit hängt von der Kalibrierung des Motors ab [9, S. 79]

III. Position bestimmen

Für präzise Arbeiten ist es wichtig, eine genaue Kontrolle über die Position der Motorwelle zu haben. Bisher wurden lediglich die Richtung sowie die Geschwindigkeit eingestellt, nicht aber z.B. die Anzahl an Umdrehungen. Dafür wird eine Rückkopplung mit zugehörigem Regelschaltkreis benötigt, die unserem MCU die aktuelle Position der Motorwelle mitteilt.

A. Positionsbestimmung mit optischen Sensoren

Zur Positionsbestimmung können Phototransistoren (PHT) als optische Sensoren verwendet werden. Ein PHT funktioniert wie ein NPN-Bipolartransistor, der an seiner Basis jedoch Licht im Gegensatz zu Spannung benötigt, um sich zu öffnen [13, S. 41]. Dieser PHT wird nun am Eingang (Kollektor) auf logisch 1 gezogen und am Ausgang (Emitter) mit einem Eingangspin verbunden [14, S. 70]. Mit der Motorwelle wird entweder direkt oder indirekt eine wie in Abb. 8 dargestellte Lochscheibe verbunden. Auf ihrer einen Seite wird eine Lichtquelle angebracht, auf der anderen Seite der PHT [14, S. 68ff.]. Dreht sich die Scheibe, kann genau dann eine logische 1 am Eingangspin gemessen werden, wenn der PHT gerade auf ein Loch in der Scheibe trifft. Wird nun im Programm eine Zählvariable angelegt, kann die Anzahl der passierenden Löcher gezählt und so eine Aussage über die Position der Motorwelle getroffen werden, solange sich die Drehrichtung nicht ändert.

```
int count = 0;                          // Anlegen der Zählvariablen
...                                     // bei Lichteinfall
if(digitalRead(InputPin)) count++;      // Inkrementieren von count
```

Kann sich die Drehrichtung auch ändern, benötigt man einen Quadratur-Encoder. Dabei werden zwei PHTs so angeordnet, dass sie um 90° versetzt sind, wodurch wir die in Abb. 8 dargestellten zwei Signale erhalten. Das vorauseilende Signal gibt Aufschluss darüber, in welche Richtung sich der Motor dreht. Jetzt kann die Zählvariable sowohl inkrementiert als auch dekrementiert werden [15].

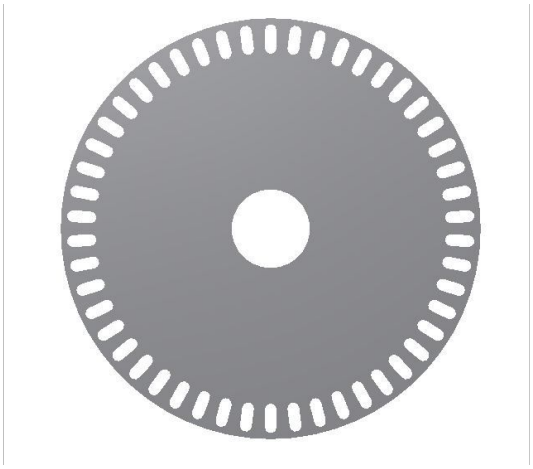

Abbildung 7: Lochscheibe

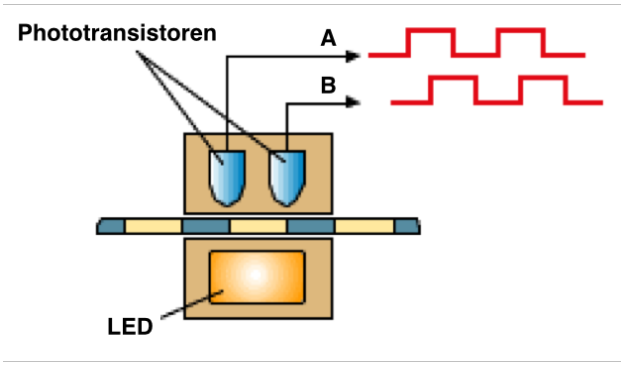

Abbildung 8: Aufbau eines Quadratur-Encoders

Ausschlaggebend für die Genauigkeit eines Encoders ist zum einen die Anzahl der Löcher in der Scheibe, diese liegen zwischen 100 und 1000, und zum anderen die Übersetzung, mit der die Scheibe an die Motorwelle gekoppelt ist [14, S. 68].

B. Positionsbestimmung mit Potentiometer

Eine weitere Möglichkeit, die Position zu bestimmen, kann mit einem Potentiometer realisiert werden. Ein Potentiometer ist ein variabler Widerstand, der eine angelegte Spannung stufenlos teilen kann. Die eingestellte Spannung kann mithilfe eines Pins mit angeschlossenem Analog/Digital-Wandler (AD-Wandler)

eingelesen werden. Die meisten Mikrocontroller stellen solche Pins zur Verfügung. Ein Arduino Leonardo zum Beispiel stellt sechs analoge Eingänge mit integriertem 10-Bit AD-Wandler zur Verfügung, d.h. das Eingangssignal wird in eine Zahl zwischen 0 und 1024 umgewandelt [16]. Wird die Motorwelle mit dem Drehregler eines Potentiometers verbunden und das Signal des Potentiometers eingelesen, kann dadurch ihre Position ermittelt werden. Ein Nachteil dabei ist, dass Potentiometer i.d.R. nur eine Reichweite von ca. 180°-360° haben und deswegen die Freiheit des Motors eingeschränkt werden muss, um das Potentiometer nicht zu demolieren.

```
int position = 0;    // 0 ≙ −90° | 512 ≙ 0° | 1024 ≙ 90°
...
position = analogRead(InputPin);
```

C. Servomotor

Ein Servomotor (Servo) besteht aus einem Gleichstrommotor mit Getriebe, einem Potentiometer und einer Recheneinheit mit integriertem AD-Wandler (RE), siehe Abb. 9. Die RE kann über das am Getriebe befestigte Potentiometer den Ist-Winkel des Servos bestimmen. Da das Potentiometer fest mit dem Getriebe verbunden ist, haben Servos meist eine Freiheit von 180°.

Abbildung 9: Servomotor

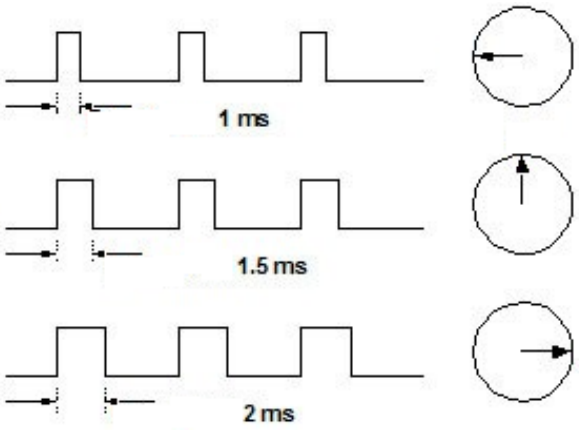

Abbildung 10: Servo Steuersignal

D. Servomotor steuern

Der RE wird mit einem in Abb. 10 dargestellten Pulssignal gesteuert. Das Signal übermittelt einen gewünschten Soll-Winkel. Es hat meist eine Periodendauer von ca. 20 ms (50 Hz), und die Dauer der Pulsweite liegt i.d.R. zwischen 1 und 2 ms. Dabei entspricht eine Pulsweite von 1.0 ms einem Winkel von -90°, 1.0 ms entsprechen der Ruhelage (0°), und 2.0 ms entsprechen 90°, siehe Abb. 10 [17, S. 152]. Diese Signale können entweder mit PWM selbst implementiert werden, oder es wird auf die von Arduino zur Verfügung gestellten Servo Bibliothek zurückgegriffen [18].

Wird nun an den Servo z.B. ein Signal von 1.8 ms gesendet, wandelt seine integrierte Recheneinheit dieses Signal in einen Soll-Winkel um und vergleicht diesen mit dem über das Potentiometer berechneten Ist-Winkel. Anschließend wird der Motor entsprechend beschleunigt, um die Differenz der beiden Werte zu minimieren. Je kleiner die Differenz, desto langsamer dreht sich der Motor. Dies soll ein sog. Überschießen verhindern, bei dem die Motorwelle aufgrund von Trägheit über den Soll-Winkel hinausschießt und sich einpendelt. Je besser der verwendete Algorithmus und die Kalibrierung des Servos, desto geringer das Überschießen [19].

E. Kontinuierlicher Servomotor

Ein Servo kann auch umgebaut werden, sodass er sich kontinuierlich dreht. Dazu werden die Sicherheitsstopper entfernt und das Potentiometer durch zwei identische feste Widerstände ersetzt. Dadurch wird der gemessene Ist-Winkel immer auf 0° gesetzt. Da die Geschwindigkeit direkt proportional zur Differenz von Soll- und Ist-Winkel ist, kann die Geschwindigkeit über das PWM Signal geregelt werden [20].

IV. Schrittmotoren

Wie der Name schon verrät, ist die Besonderheit eines Schrittmotors die, dass er sich nicht kontinuierlich dreht, sondern in diskreten Schritten. Exemplarisch wird in dieser Arbeit die Funktionsweise eines Schrittmotors anhand eines bipolaren Schrittmotors mit magnetischem Rotor vorgestellt, weitere Bauarten werden in [21] ausführlich beschrieben.

A. Ein Schritt nach dem anderen

Der hier vorgestellte bipolare Schrittmotor besteht aus einem magnetischen Rotor, der von zwei um 90° versetzten Erregerspulen umschlossen wird, wie in Abb. 12 dargestellt. Sobald eine der beiden Spulen von Strom durchflossen wird, richtet sich der Rotor entsprechend seiner Polung in dem dadurch entstandenen Magnetfeld aus. Wird wie in Abb. 13 die vertikale Spule ausgeschaltet und die horizontale Spule eingeschaltet, dreht sich der Rotor um genau einen Schritt weiter. Wird die horizontale Spule wieder ausgeschaltet und die vertikale Spule mit umgedrehter Polung wieder aktiviert, dreht sich der Rotor einen Schritt im Uhrzeigersinn weiter.

Die hier dargestellten Schritte sind als logische Schritte zu verstehen, da ein Schrittmotor pro Umdrehung nicht vier sondern normalerweise 200 Schritte benötigt [9, S. 80]. Dies wird z.B. dadurch erreicht, dass die Polschuhe der Erregerspulen gezahnt sind und der Rotor weniger Zähne besitzt als der Stator. Dadurch stehen die Rotorzähne immer genau auf Lücke mit den Zähnen der ausgeschalteten Spule. Wird diese dann eingeschaltet richten sich die Zähne wieder Zahn auf Zahn aus, und es erfolgt ein Schritt. Da es hierfür sehr viele verschiedene Implementationen gibt, wird an dieser Stelle auf [21] verwiesen.

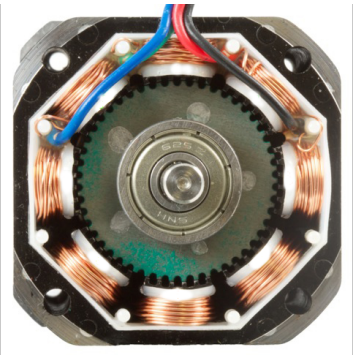

Abbildung 11: Schrittmotor aufgeschnitten

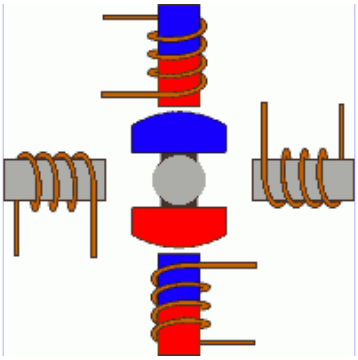

Abbildung 12: Vertikale Ausrichtung Rotor

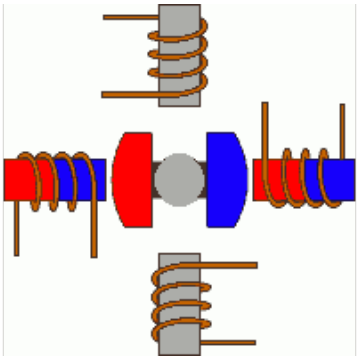

Abbildung 13: Horizontale Ausrichtung Rotor

Die benötigten Phasen, um jeweils einen Schritt zu machen, sind in Abb. 14 noch einmal in einem Impulsdiargramm aufgezeichnet. Werden zusätzlich zu dem abwechselnden Beschalten der Spulen noch Phasen durchlaufen, in denen beide Spulen aktiv sind wie in Abb. 15, können sogenannte Halbschritte vollzogen werden, die Anzahl der Schritte pro Umdrehung verdoppelt sich dadurch. Wird das Beschalten der Spulen nun weiter geglättet und ein sinuides Signal gesendet wie in Abb. 16, erhält man ein sogenanntes Mikrostepping und eine erhöhte Auflösung [22, S. 494ff.].

Das entsprechende Phasenmanagement wird mit zwei H-Brücken und einem entsprechenden Code realisiert. Die uns schon bekannte H-Brücke L289 [12] kann so implementiert werden, dass nur zwei

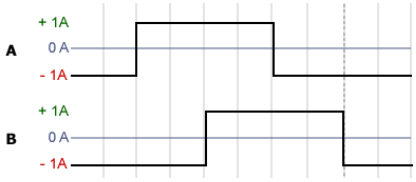

Abbildung 14: Impulsdiagramm eines bipolaren Schrittmotors für ganze Schritte

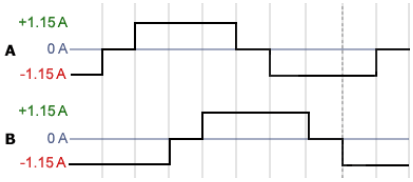

Abbildung 15: Impulsdiagramm eines bipolaren Schrittmotors für halbe Schritte

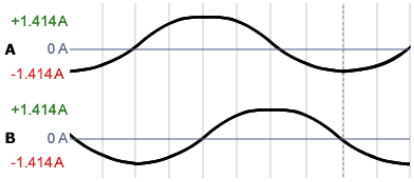

Abbildung 16: Impulsdiagramm eines bipolaren Schrittmotors für Mikrostepping

Pins pro Schrittmotor benötigt werden [23, Abb. 2]. Mit einem Pin wird die Richtung gesteuert, und mit dem anderen werden die Schritte gesteuert. Eine mögliche Implementierung wäre, dass bei steigender Flanke des Schrittsignals ein Schritt gemacht wird. Code Beispiel:

```
digitalWrite(directionPin, HIGH); // im Uhrzeigersinn (CW)
digitalWrite(stepPin, HIGH);      // Ein Schritt CW
digitalWrite(stepPin, LOW);
...
digitalWrite(directionPin, LOW);  // gegen den Uhrzeigersinn (CCW)
digitalWrite(stepPin, HIGH);      // Ein Schritt CCW
```

B. Anwendungsbeispiel Stepper

Im Folgenden wird am Beispiel einer 3-Achsen-CNC-Fräse aufgezeigt, wie die Motorsteuerung einer Maschine in der Praxis umgesetzt wird. Eine 3-Achsen-Fräse hat in der Regel vier Motoren. Einen Schrittmotor pro Achse und einen Gleichstrommotor für die Spindel.

CNC steht für computergestützte numerische Steuerung (engl. Computerized Numerical Control) und bezeichnet eine Methode zur elektronischen Steuerung und Regelung von Werkzeugmaschinen. Dabei werden Kommandos an einen sogenannten Motor Controller geschickt. Diese Kommandos werden meistens in G-Code verfasst. G-Code ist eine Maschinensprache bzw. ein Protokoll, welches an den Motor Controller gesendet wird und aus der Zeit der Lochkarten stammt [24]. Ein G-Code-Kommando beschreibt meist eine Tätigkeit mit entsprechenden Parametern. G-Code Beipiel:

```
G00 X10 Y10 F70     // Gehe nach (10,10,-) mit 70% Geschw.
M3  F100            // Mache Spindel an mit 100% Geschw.
```

Der Motor Controller besteht in diesem Beiespiel aus einem Arduino. Er muss die Kommandos interpretieren und in Arduino-Signale übersetzen können. Dafür muss er mit grbl, einem G-Code Parser, geflasht und entsprechend eingerichtet worden sein [25].

Werden dem Motor Controller nun über USB seriell G-Code Befehle gesendet, wandelt er diese in Arduino-Signale um und sendet sie an den angeschlossenen Motor-Treiber. Dieser kümmert sich um das entsprechende Phasenmanagement der Spulen in den Schrittmotoren und reguliert den Gleichstrommotor der Spindel. Aufbau siehe Abb. 17.

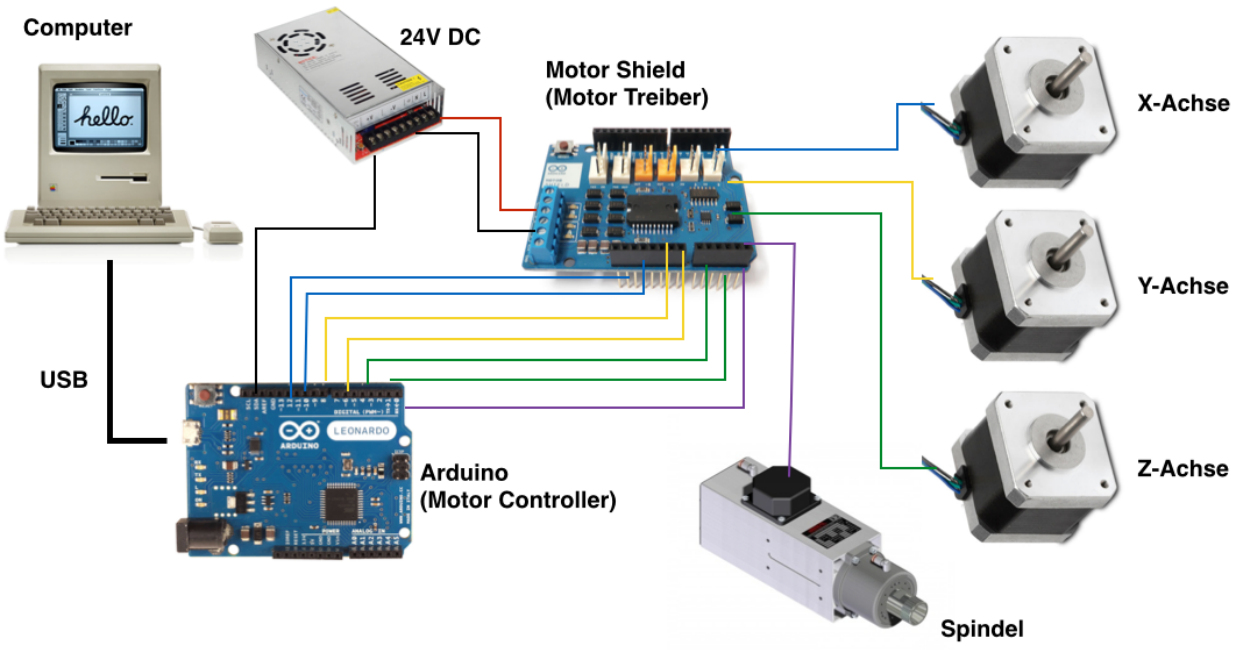

Abbildung 17: Aufbau zur Steuerung einer 3-Achsen-CNC-Fräse

V. Zusammenfassung und Ausblick

In dieser Arbeit wurden drei Motorttypen vorgestellt und grundlegende Prinzipien der Motorsteuerung erläutert. Wie gezeigt wurde, kann ein Motor mit sehr einfachen Mitteln gestartet werden. Für eine wirklich präzise Steuerung gibt es die einfache Variante eines Servomotors mit schon eingebauter Rückkopplung und Steuereinheit. Eine präzise Steuerung ohne Rückkopplung ist mit Schrittmotoren möglich.

Dem Umfang der Arbeit geschuldet, konnte auf viele weitere Themen nicht eingegangen werden. Dazu zählen die unterschiedlichen Eigenschaften der verschiedenen Motorklassen eines Motortyps. So gibt es bei Gleichstrommotoren, Servomotoren und Schrittmotoren eine große Anzahl an verschiedenen Bauweisen mit komplett unterschiedlichen Eigenschaften wie Drehmoment, Geschwindigkeit, Verbrauch und Zuverlässigkeit, die an spezielle Anforderungen angepasst sind.

Für weitere Arbeiten wäre es durchaus interessant, speziellere Anwendungsgebiete und die damit verbundenen Schwierigkeiten und Lösungsansätze zu beleuchten, sei es z.B. die Optimierung der Motorsteuerung von 3D-Druckern oder die virtuelle Steuerung von Maschinen mit entsprechenden Frameworks wie z.B. Gestalt [26].

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

LITERATUR

[1] The Story of the Intel 4004. http://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html. (Visited on 11/23/2014).
[2] Maker-Bewegung: Komm, wir bauen eine neue Welt - SPIEGEL ONLINE. http://www.spiegel.de/netzwelt/reeperbahnfestival/a-992843.html. (Visited on 11/23/2014).
[3] J. T. Humphries, *Motors and Controls*. Columbus, Ohio: Merrill Publishing Company, 1988.
[4] J. Weidauer and R. Messer, *Electrical Drives - Principles, Planning, Applications, Solutions*. New York: John Wiley & Sons, 2014.
[5] Datasheet: NPN-Bipolartransistor 2N4401-D. http://www.onsemi.com/pub/Collateral/2N4401-D.PDF. (Visited on 11/23/2014).
[6] C. Platt, *Make: Elektronik*. Köln: O'Reilly Germany, 2010.
[7] H. Linse and R. Fischer, *Elektrotechnik für Maschinenbauer*, 12th ed. Berlin Heidelberg New York: Springer-Verlag, 2005.
[8] H.-W. Huang and L. Chartrand, *PIC Microcontroller - An Introduction to Software and Hardware Interfacing*, new. ed. Clifton Park, NY: Cengage Learning, 2005.
[9] T. Braunl, *Embedded Robotics - Mobile Robot Design and Applications with Embedded Systems*, 3rd ed. Berlin Heidelberg: Springer Science & Business Media, 2008.
[10] Freescale, MCU Report. http://cache.freescale.com/files/training/doc/dwf/DWF13_APF_IND_T0806.pdf. (Visited on 11/23/2014).
[11] J.-D. Warren, J. Adams, and H. Molle, *Arduino Robotics*, 1st ed. New York: Apress, 2011.
[12] Datasheet, L298 Dual H-Bridge. https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf. (Visited on 11/24/2014).
[13] A. A.P.Godse and U. U.A.Bakshi, *Basic Electronics*. Pune, India: Technical Publications, 2009.
[14] C. Nwagboso, *Automotive Sensory Systems*, 1993rd ed. Berlin Heidelberg: Springer Science & Business Media, 1993.
[15] Arduino Playground - Rotary Encoders. http://playground.arduino.cc/Main/RotaryEncoders#Example1. (Visited on 11/29/2014).
[16] Arduino - Arduino Board Leonardo. http://arduino.cc/en/pmwiki.php?n=Main/arduinoBoardLeonardo. (Visited on 11/29/2014).
[17] A. K. Solutions, *Robotics*. Sudbury, Massachuetts: Jones & Bartlett Learning, 2010.
[18] Arduino - Servo library. http://arduino.cc/en/Reference/Servo. (Visited on 11/30/2014).
[19] Understanding Servo Tune - National Instruments. http://www.ni.com/white-paper/2923/en/#toc2. (Visited on 11/29/2014).
[20] Servohacking Tutorial PWM Version. http://www.electronicsplanet.ch/Roboter/Servo/hacking/pwmhack.htm. (Visited on 11/29/2014).
[21] V. V. Athani, *Stepper Motors : Fundamentals, Applications And Design*. New Delhi: New Age International, 1997.
[22] J. Sanchez and M. P. Canton, *Embedded Systems Circuits and Programming*. Boca Raton, Fla: CRC Press, 2012.
[23] Connecting the Arduino to a L298N H-Bridge. http://www.bristolwatch.com/L298N/L298N_arduino.htm. (Visited on 11/30/2014).
[24] G-code. http://en.wikipedia.org/wiki/G-code. (Visited on 11/30/2014).
[25] grbl Wiki. https://github.com/grbl/grbl/wiki. (Visited on 11/30/2014).
[26] I. Moyer. A Gestalt Framework for Virtual Machine Control of Automated Tools. http://www.pygestalt.org/VMC_IEM.pdf. (Visited on 11/30/2014).
[27] "Mecatronica - Actuatori," http://cursuri.flexform.ro/courses/L2/document/Cluj-Napoca/grupa4/Narosi_Sanda/site/mcc.html, (Visited on 12/03/2014).
[28] Algos, "Gleichstrommaschine - Gleichstrommaschine Wikipedia," http://de.wikipedia.org/wiki/Gleichstrommaschine#mediaviewer/File:Gleichstrommaschine.svg, (Visited on 12/03/2014).
[29] "Difference Between an NPN and a PNP Transistor," http://www.learningaboutelectronics.com/Articles/Difference-between-a-NPN-and-a-PNP-transistor, (Visited on 12/03/2014).

[30] “Actuators — Robotics 3.1 DT009 /3,” http://roboted.wordpress.com/actuators/, (Visited on 12/03/2014).
[31] “Pulse Width Modulation,” http://powerelectronicsdesign.blogspot.de/2014/04/pulse-width-modulation-darbe-genislik.html, (Visited on 12/03/2014).
[32] “Encoder - how to build one? — model engineer,” http://www.model-engineer.co.uk/forums/postings.asp?th=75630, (Visited on 12/03/2014).
[33] “How Servo Motors Work,” http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html, (Visited on 12/03/2014).
[34] “Remote Control for Robots,” http://www.superdroidrobots.com/shop/custom.aspx/remote-control-rc-support/40/, (Visited on 12/03/2014).
[35] “Phidgets,” http://www.phidgets.com/documentation/Phidgets/3300_0_Primer_Stepper_Motors.pdf, (Visited on 12/03/2014).
[36] “How Stepper Motors Work,” http://www.pcbheaven.com/wikipages/How_Stepper_Motors_Work/, (Visited on 12/03/2014).