DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Laborversuch Digitaler Regler

Vorbereitung und Durchführung

Ausgabe 0.1, Februar 2015

S. Rupp, H. Huning

Inhaltsverzeichnis

1. Vorbereitung: Drehzahlregelung mit PID Regler

In diesen Abschnitt soll ein Gleichstromantrieb als reale Regelstrecke durch einen digitalen Regler mit vorgegebener, konstanter Drehzahl betrieben werden. Der Regler wird als Software auf eine, Mikrocontroller realisiert. Die Regelstrecke enthält den Motor mit Drehgeber. Der Regelalgorithmus soll auf dem Regler für minimalen Rechenaufwand optimiert werden. Zur Einstellung der Reglerparameter werden Regeln vorgestellt, die sich unmittelbar in der Realität überprüfen lassen.

1.1. Regelstrecke mit Regler

Die Regelstrecke ist auf einer Motorbaugruppe untergebracht. Der Aufbau entspricht dem des Laborversuchs für analoge Regler (mit Operationsverstärker). Hier soll jedoch der Regler mit Hilfe eines Mikrocontrollers realisiert werden. Die Motorbaugruppe enthält einen Drehgeber zur Messung der Regelgröße. Als Stellglied wird ein Leistungsverstärker verwendet, der sich über Pulsweitenmodulation ansteuern lässt. Folgende Abbildung zeigt den prinzipiellen Aufbau.

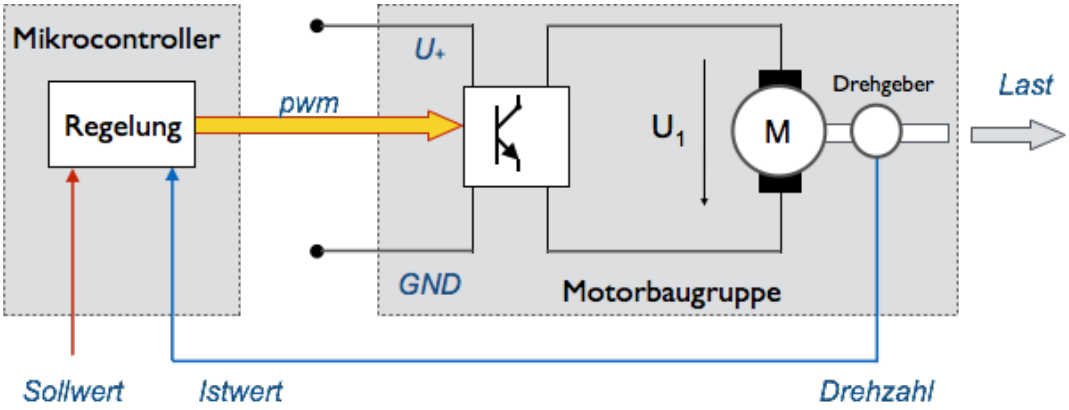


Bild 2.1 Aufbau der Drehzahlregelung mit Mikrocontroller

Somit benötigt die SPS nur digitale Eingänge und Ausgänge. Damit die Pulsweite nicht direkt vom Prozessor (CPU) der SPS erzeugt werden muss, wird ein Baustein verwendet, der zu einem gegebenen Wert eigenständig ein pulsweitenmoduliertes Signal erzeugt. Der gegebene Wert wird von der vom Regler direkt als PWM-Signal an einen Motortreiber gegeben.

Übung 2.1: Erstellen Sie ein Konzept für die Regelung. Hierzu zählt die Vorgabe der Ankerspannung über Pulsweitenmodulation (PWM), sowie das Einlesen der Messwerte des Drehgebers. Dokumentieren Sie Ihr Konzept. Hinweis: Unterlagen über den Motor mit Drehgeber (Regelstrecke), den Controller (Arduino UNO), sowie den Motor-Treiber (Arduino Motor-Shield) finden Sie im Anhang.

1.2. Regelalgorithmus

Der Regler soll als PID-Regler ausgeführt werden. Gleichung (2.1) beschreibt den Regelalgorithmus im zeitkontinuierlichen Fall. Gleichung (2.2) beschreibt den Algorithmus im zeitdiskreten Fall, wobei Δt das Abtastintervall bezeichnet.

$$u_R(t) = K_P * e(t) + K_i * \int e(\tau)\, d\tau + K_d\, de(t)/dt \qquad \text{mit } \tau = 0 \text{ bis } t \qquad (2.1)$$

$$u_R(k) = K_P * e(k) + K_i * \Delta t * \Sigma e(i) + (K_d / \Delta t) * (e(k)-e(k-1)) \qquad \text{mit } i = 0, \ldots, k \qquad (2.2)$$

Bei der Realisierung als digitaler Regler wird der Algorithmus in einer Programmschleife zyklisch ausgeführt. Um den Rechenaufwand bzw. die Laufzeit auf dem Regler zu reduzieren, lässt sich der Algorithmus in folgenden Punkten optimieren: (1) rekursive Berechnung der Fehlersumme $\Sigma e(i)$, (2) die Division durch eine Multiplikation ersetzen, (3) die Anzahl der Operationen reduzieren. Durch Umformen erhält man einen vereinfachten Algorithmus der Form:

$$u_R(k) = u_R(k-1) + q_0\, e(k) + q_1\, e(k-1) + q_2\, e(k-2) \qquad (2.3)$$

Hierbei bedeuten

$$q_0 = K_P + K_i * \Delta t + (K_d / \Delta t) \qquad (2.4)$$

$$q_1 = - K_P - 2 * (K_d / \Delta t) \qquad (2.5)$$

$$q_2 = (K_d / \Delta t) \qquad (2.6)$$

Übung 2.2: Prüfen Sie die Korrektheit von Gleichung (2.3) in Bezug auf Gleichung (2.2).

Übung 2.3: Programmieren Sie den Regelalgorithmus auf Ihrer SPS. Testen Sie den Algorithmus an der Regelstrecke.

1.3. Einstellung der Reglerparameter

Zum Einstellen der Reglerparameter gibt es unterschiedliche Methoden. Ein guter Startpunkt ist die ungeregelten Strecke, d.h. alle Reglerparameter sind zunächst auf Null. Dann lässt sich z.B. durch Probieren eine günstige Reglereinstellung finden. Hierzu erhöht man zunächst den proportionalen Anteil K_P, bis die Dämpfung des Systems schlecht wird. Anschließend erhöht man den I-Anteil. Dann wäre zu probieren, ob ein D-Anteil die Strecke stabilisiert.

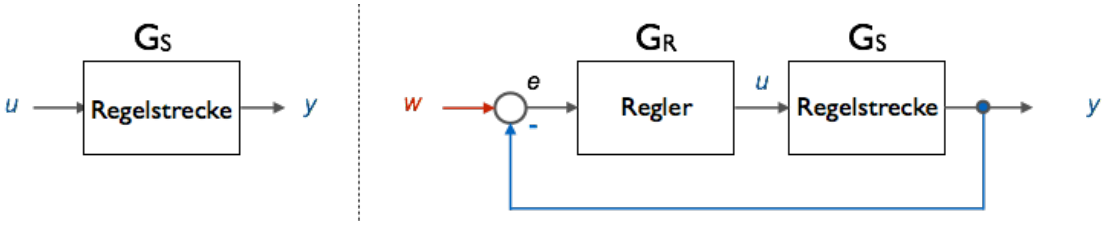
Neben der Methode des Probierens kann man auch einschlägige Einstellregeln verwenden, wie z.B. die Einstellregeln nach Ziegler/Nichols. Nach diesem Verfahren wird ausgehend vom Startpunkt Null zunächst der K_P-Anteil so weit erhöht, bis das System ins Schwingen gerät, d.h. in einen kritischen Zustand. Diese Reglereinstellung wird als K_{krit} erfasst, sowie die Periodendauer T_{krit} der Schwingung. Als Reglereinstellungen verwendet man dann: $K_P = 0{,}6\ K_{krit}$, $K_i = 1{,}2\ K_{krit}/T_{krit}$ und $K_d = 0{,}075\ K_{Krit} * T_{krit}$. Das Verfahren von Ziegler-Nichols setzt voraus, dass man die Regelstrecke in einen kritischen Zustand fahren darf.

Die Einstellung der Reglerparameter sind natürlich abhängig von den Eigenschaften der Regelstrecke. Für unterschiedliche Strecken finden sich in der Literatur Vorschläge für günstige Reglereinstellungen. Ist die Strecke bekannt, kann man jedoch mit Hilfe der Übertragungs-funktion des geregelten Systems selber günstige Reglereinstellungen ableiten, z.B. durch Vor-gabe der gewünsch-ten Zeitkonstanten bzw. der Lage der Pole.

Übung 2.4: Optimieren Sie die Einstellung Ihrer Reglerparameter nach Ihren Vorstellungen. Untersuchen Sie das Führungsverhalten und Störverhalten Ihres Reglers.

1.4. Stabilität

Durch die Einführung des Regelkreises entsteht eine Rückkopplung (engl. feed back), wie in der folgenden Abbildung gezeigt.



Vergleich der ungeregelten mit der geregelten Strecke

Wenn diese Rückkopplung zu einer Mitkopplung wird, gerät das geregelte System ins Schwingen. Dieser Effekt ist von Veranstaltungen bekannt, wenn ein Mikrophon in die Nähe eines Lautsprechers gerät. Die Übertragungsfunktion des Regelkreises beträgt

$$G(s) = G_R(s)\, G_S(s) / (1 + G_R(s)\, G_S(s)) \qquad (2.7)$$

Hierbei bedeutet $G_S(s)$ die Übertragungsfunktion der Strecke, und $G_R(s)$ die Übertragungsfunktion des Reglers. Aus der Lage der Pole der Übertragungsfunktion lässt sich die Stabilität des Systems ableiten. Gibt es Pole in der rechten komplexen Halbebene, so ist das System instabil. Die Wahl der Reglerparameter ist so zu wählen, dass es solche Pole nicht gibt.

Haben alle Pole der Übertragungsfunktion negative Realteile, so ist das System asymp-totisch stabil, d.h. es schwingt sich auf einen stabilen Zustand ein. Je weiter die Pole der Übertragungsfunktion in der linken Halbebene von der imaginären Achse entfern sind, desto geringer sind die zeitkonstanten und desto höher ist die Dämpfung des Systems.

Ist der Realteil eines Poles gleich Null, so ist das System ungedämpft. Ein Beispiel hierfür wäre ein idealer Schwingkreis (mit einem konjugiert komplexen Polpaar auf der imaginären Achse). Ein solches System bezeichnet man als grenzstabil. Pole mit positivem Realteil bedeuten, dass das System sich aufschwingt, also instabil ist. Zu den instabilen Systemen rechnet man auch die grenzstabilen Systeme.

2. Laborarbeit

Die Laborarbeit gibt Ihnen Gelegenheit, sich individuell und praktisch mit der Regelungs-technik zu beschäftigen. Sie erhalten eine Aufgabe, die Sie alleine bzw. in einer Zweiergruppe bearbeiten. Die Aufgabe wird zum Teil innerhalb der Vorlesungszeit gelöst, wo Ihnen der Dozent für Fragen zur Verfügung steht, zum anderen Teil in Heimarbeit bzw. in Vorlesungslücken.

Die benötigte Hardware wird Ihnen für die Dauer der Laborarbeit zur Verfügung gestellt. Das Material ist transportabel, so dass Sie sich die Bearbeitung im Verlauf des Semesters flexibel gestalten können.

2.1. Laborbericht

Die Ergebnisse der Laborarbeit werden formlos in Art eines Laborberichtes dokumentiert und zum Ende des Semesters an den Dozenten gegeben. Die Bewertung der Laborarbeit erfolgt durch ein Testat. Das Testat für den erfolgreichen Abschluss der Laborarbeit ist Teil der Prüfungsleistung. In Absprache mit Ihrem Dozenten kann die Laborarbeit auch benotet werden und somit zusammen mit der Klausur in die Gesamtnote einfließen.

Ziel der Laborarbeit ist die praktische Auseinandersetzung mit einem Problem der Regelungs-technik. Die Laborarbeit sollen Sie in die Lage versetzen, das in der Vorlesung erworbene Wissen in der Praxis anzuwenden.

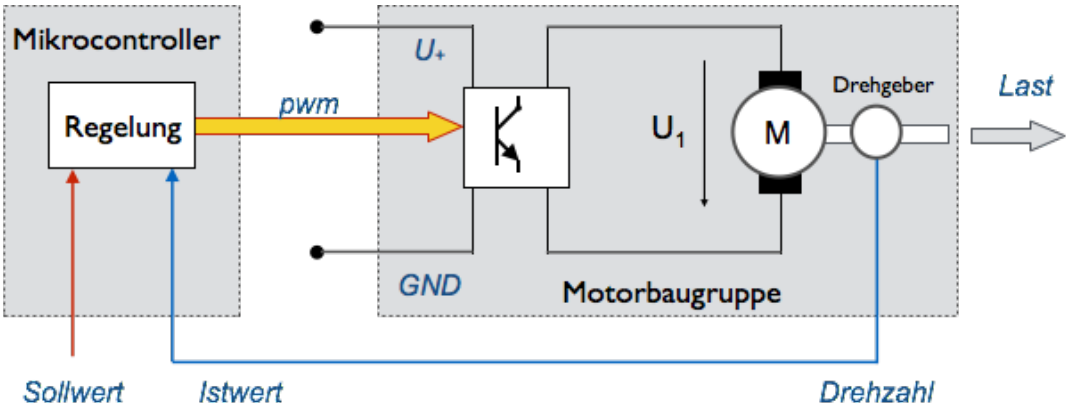
2.2. Laboraufgabe: PID-Regler für einen Gleichstromantrieb

Realisieren Sie auf Ihrem Mikrocontroller einen PID-Regler für die Motorbaugruppe. Programmieren Sie den Regelalgorithmus. Stellen Sie die Reglerparameter geeignet ein. Untersuchen Sie das Führungsverhalten, Störverhalten sowie die Stabilität des Systems. Dokumentieren Sie Ihre Ergebnisse.

Hinweis: Die Bibliothek des Mikrocontrollers enthält Beispiele zum Anschluss des Motortreibers über PWM sowie die Schnittstellen für die Messung der Drehzahl. Erstellen Sie zunächst ein Konzept, bevor Sie an die Realisierung gehen. Gehen Sie schrittweise vor, z.B. erst Betrieb als Steuerung, dann Messung der Drehzahl, dann Regelkreis mit einfachem Regler (P-Regler). Dokumentieren Sie Zwischenergebnisse. Unterlagen zur Motorbaugruppe finden Sie im Anhang.

Anhang - Unterlagen zur Laborarbeit

Übersicht



Regler (Mikrocontroller):

- Arduino UNO (aus MCT 1 im 3. Semester)
- Entwicklungsumgebung zum Mikrocontroller (aus MCT 1 im 3. Semester)

Regelstrecke (Motorbaugruppe)

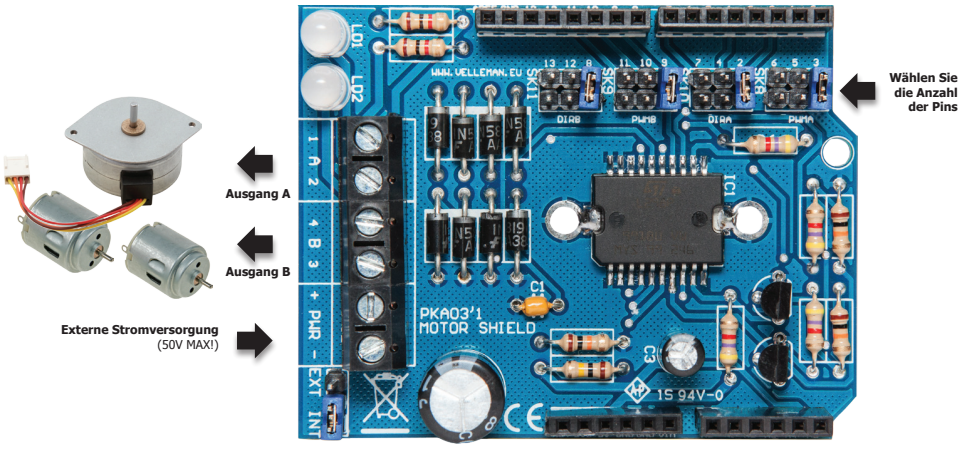
- Motortreiber: Arduino Motor Shield
- Motor: Gleichstrommotor mit Getriebe und Drehgeber, sowie einstellbarer Last

Technische Daten:
[1] zum Motor: http://nodna.de/Metall-Getriebemotor-6V-130rpm-mit-Encoder-4mm-Achse-High-Power
[2] zum Arduino Motor Shield (inkl. Code Sample): http://www.velleman.eu/products/view/?id=412174

Belegung der Motor-Pins:

Color	Function
Black	motor power
Red	motor power
Blue	Hall sensor Vcc (3.5 – 20 V)
Green	Hall sensor GND
Yellow	Hall sensor A output
White	Hall sensor B output

Motor Shield (Quelle siehe [2] oben):



Musterprogramm zur Ansteuerung des Motors und zum Einlesen der Drehgeber

```
int pwm_b = 9;  // PWM control for motor outputs 3 and 4
int dir_b = 8;  // direction control for motor outputs 3 and 4

volatile int count = 0; // counter for rotary encoders

void setup() // initial set up
{
  pinMode(pwm_b, OUTPUT);  // set motor control pin b to be output
  pinMode(dir_b, OUTPUT);  // set direction pin b to be output
  attachInterrupt(0, tacho, RISING);
                          // encoder to be attached to pin 2 (interrupt 0)
  Serial.begin(9600);     // serial monitor for control outputs
}

void loop() // main program
{
  digitalWrite(dir_b, LOW);
  analogWrite(pwm_b, 200);
  delay(1000);

  analogWrite(pwm_b, 0);
  Serial.print("counter: ");
  Serial.println(count);
  delay(5000);

  digitalWrite(dir_b, HIGH);
  analogWrite(pwm_b, 180);
  delay(1000);

  analogWrite(pwm_b, 0);
  Serial.print("counter: ");
  Serial.println(count);
  delay(5000);

}

void tacho() // interrupt service routine
{
  count = count + 1;
}
```