	Mikrocontroller-Labor	Name: Rahm Datum: 13.02.2019 6_6_Drehzahlregelung_mit_PI_Regler.docx
	Drehzahlregelung mit PI-Regler	6.6.1

Projekt: Drehzahlregelung eines Gleichstrommotors

Im Regelkreis werden Soll- und Istwert fortdauernd verglichen. Das Regelglied bildet die Stellgröße mit dem Ziel einer Angleichung des Istwertes an den Sollwert.

Die Drehzahlregelung soll mit dem abgebildeten Software-Regler mit PI-Charakteristik realisiert werden. Beim Digitalregler wird die Stellgröße y mittels der folgenden Gleichung gebildet:

```
void timer1ms_isr(void)
{
    e = w - x;
    yp = Kp * e;

    e_k += e;           // e aufsummieren !
    yi = Kp * e_k * Ta / Tn;

    y = yp + yi;
    y = y / 100;        // :100, wegen Kp * 100

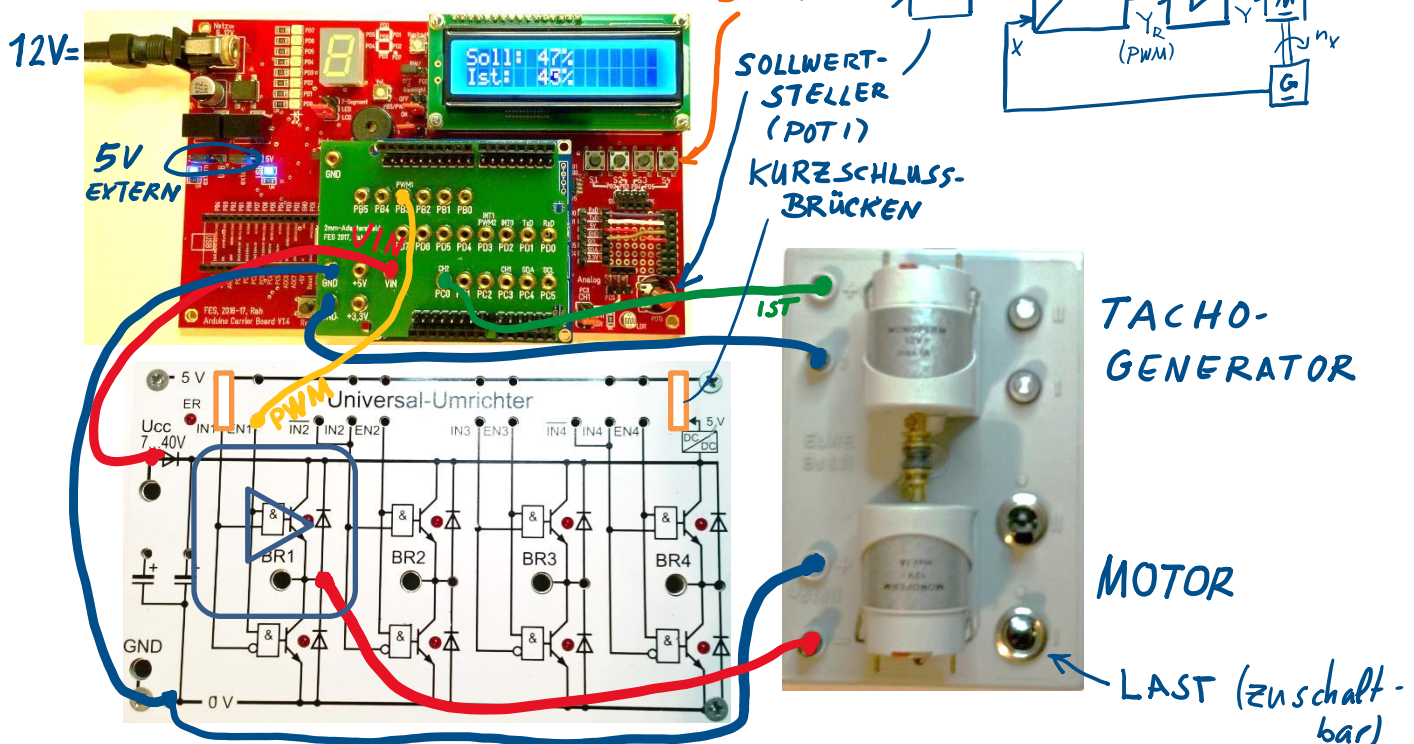
    if (y > 255) y = 255; // Begrenzung
    if (y < 0)   y = 0;

}
```




$$y_R = K_P \cdot e_n + K_P \cdot \frac{T_a}{T_N} \cdot \sum_{k=0}^n e_k$$


K_P : Proportionalbeiwert
 T_a : Abtastintervall (hier fest 1ms)
 T_N : Nachstellzeit
 e_n : Regeldifferenz ($e_n = w - x$)
 e_k : Summe aller e_n

Schaltungsschema



Arbeitsauftrag

-  Bauen Sie die Versuchsschaltung auf.
-  Erstellen Sie den umseitig abgedruckten Reglercode und übertragen Sie ihn auf den Mikrocontroller.
-  Untersuchen Sie den Regler durch Aufnahme der Sprungantwort des Regelkreises (U_{Tacho}) mit dem PicoScope. Variieren Sie die Reglereinstellwerte (K_P und T_N) und versuchen Sie das Regelverhalten zu verbessern.

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller-Labor	Name: Rahm Datum: 13.02.2019 6_6_Drehzahlregelung_mit_PI_Regler.docx
	Drehzahlregelung mit PI-Regler	6.6.2

Stellungsalgorithmus nach Technischer Richtlinie FA205

```
#include "controller.h"
#define Taster _PORTB_
#define Start 5

#define Kp 20 // Kp = 20 / 100 = 0.2
#define Tn 1300 // Tn = 1300ms
#define Ta 1 // Ta = 1ms (Timer1ms)

uint8_t status_start = 0;
volatile int32_t x,w,e,y,yp,yi,e_k;

void setup (void) /* Initialisierungen */
{
  lcd_init();
  adc_init();
  pwm_init();
  bit_init(Taster,Start,IN);
  timer1ms_init(timer1ms_isr);

  lcd_clear();
  lcd_setcursor(1,1);
  lcd_print("Soll:");
  lcd_setcursor(2,1);
  lcd_print("Ist:");
}
```

```
void timer1ms_isr(void)
{
  e = w - x;
  yp = Kp * e;

  e_k += e; // e aufsummieren !
  yi = Kp * e_k * Ta / Tn;

  y = yp + yi;
  y = y / 100; // :100, wegen Kp * 100

  if (y>255) y=255;
  if (y<0) y = 0;
}
```

Die Technische Richtlinie kennt keine Fließkommaberechnung. Daher müssen die Zahlen groß gewählt werden (hier Kp). Am Ende erfolgt dann die Division durch den Multiplikator. Der Rundungsfehler der Stellgröße y ist dann ≤ 1 .

```
int main(void)
{
  uint8_t temp;

  setup();

  while(1) // Endlosschleife
  {
    bit_toggle(Taster,Start,&status_start);


    lcd_setcursor(1,6);
    if (status_start)
    {
      timer1ms_enable();
      pwm_start();
    }
    else
    {
      timer1ms_disable();
      pwm_stop();
      e_k = 0;
    }

    w = adc_in1();
    x = adc_in2();

    pwm_duty_cycle(y); // PWM Tastgrad

    lcd_setcursor(1,6); // Sollwert in %
    temp = w*100/255;
    lcd_byte(temp);
    lcd_char('%');

    lcd_setcursor(2,6); // Istwert in %
    temp = x*100/255;
    lcd_byte(temp);
    lcd_char('%');
  }
}
```

	Mikrocontroller-Labor	Name: Rahm Datum: 13.02.2019 6_6_Drehzahlregelung_mit_PI_Regler.docx
	Drehzahlregelung mit PI-Regler	6.6.3



Nur für Profis: Das Verhalten des Reglers lässt sich durch Verwenden des sogenannten Geschwindigkeits-Algorithmus verbessern. Dabei wird nur die Änderung der Stellgröße neu berechnet und zum letzten Wert aufaddiert.

$$y_R = y_{alt} + K_P \cdot \left[e - e_{alt} \cdot \left(1 - \frac{T_a}{T_N} \right) \right] = y_{alt} + K_P \cdot e - \left(K_P \cdot e_{alt} - K_P \cdot e_{alt} \cdot \frac{T_a}{T_N} \right)$$

Da die Regeldifferenz nicht mehr aufsummiert wird, ergeben sich wesentlich kleinere Zahlenwerte. Die Berechnung lässt sich nur noch sinnvoll mit Fließkomma-Berechnung (keine Technische Richtlinie!!) durchführen.

♦ Ändern Sie das Programm folgendermaßen ab:

```
#define Kp    0.2      // Kp = 0.2
#define Tn   1300     // 1300ms
#define Ta    1       // 1ms (Timerinterrupt)

uint8_t status_start = 0;
volatile float x,w,e,y,yp,yi;
volatile float y_alt, e_alt;
```

```
// Geschwindigkeitsalgorithmus (PI-Regler)
void timer1ms_isr(void)
{
    e = w - x;

    yp = Kp * e;
    yi = Kp * e_alt - Kp * e_alt * Ta / Tn;

    y = y_alt + yp - yi;

    y_alt = y;
    e_alt = e;

    if (y>255) y=255;      // Begrenzung
    if (y<0)   y = 0;
}
```

♦ Nehmen Sie die Sprungantwort des Regelkreises mit und ohne zugeschalteter Belastung auf. Experimentieren Sie auch mit verschiedenen Einstellungen für K_P und T_N . Welches sind die besten Einstellwerte?



Nur für absolute Cracks:

♦ Schreiben Sie ein Programm zur Aufnahme der Sprungantwort der Motor-Generator-Strecke. Der Sollwert (Eingangssprung) soll mit dem Poti vorgegeben werden. Soll- und Istwert werden wieder auf dem Display als Prozentwerte angezeigt.

♦ Ermitteln Sie die Sprungantwort bei einem Eingangssprung von 10% unter Belastung der Strecke. Messen Sie dazu die Tachospannung und die Spannung am Poti mit dem PicoScope. Führen Sie eine Reglereinstellung nach CHR-Verfahren durch. Zur Berechnung der Reglerparameter können Sie das Programm **CHR-Reglereinstellung**¹ verwenden. Beurteilen Sie das Ergebnis der Regler-Einstellung.

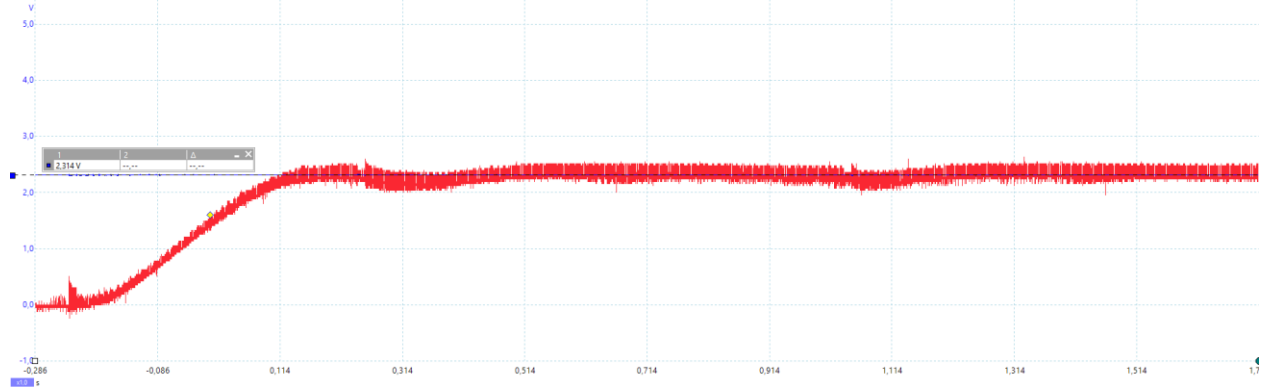
¹ <http://pcmess.rahm-home.de>

Ergebnis: Stellungsalgorithmus

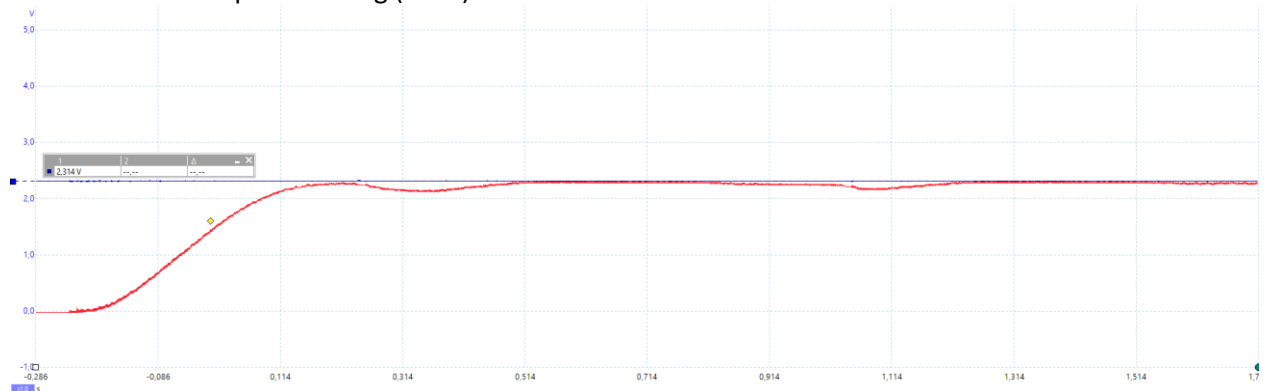
$K_p = 0,2$

$T_n = 1300\text{ms}$

Sprungantwort des Regelkreises

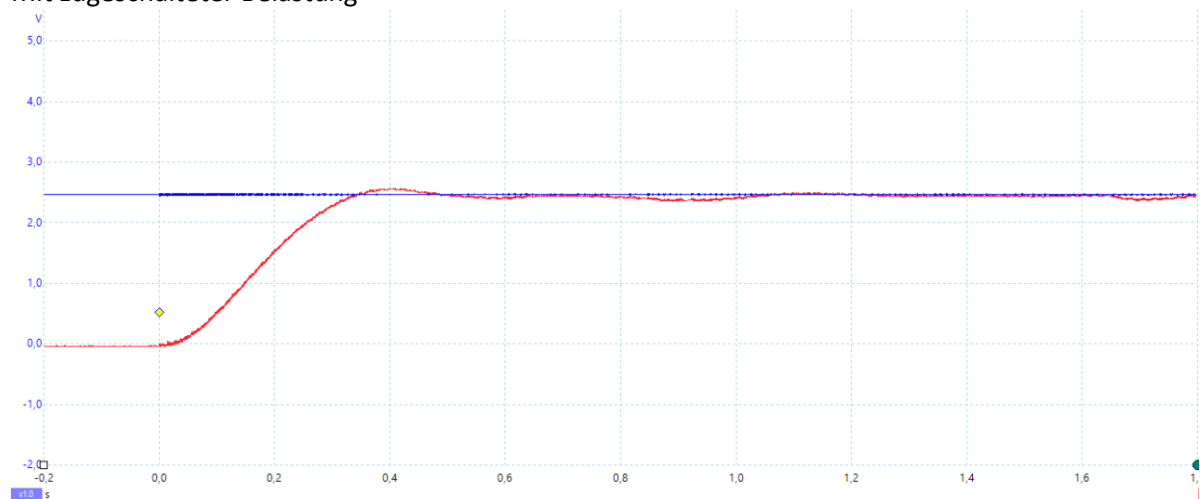


Mit aktivierter Tiefpassfilterung (1kHz)

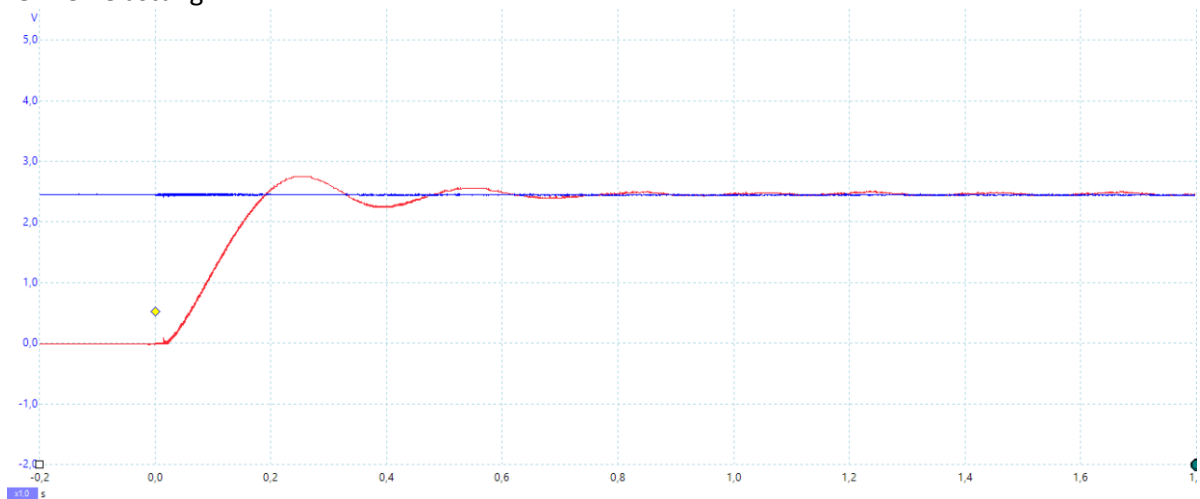


Ergebnis: Geschwindigkeitsalgorithmus

Mit zugeschalteter Belastung



Ohne Belastung



Sprungantwort des Antriebs mit Belastung und Berechnung der Reglerparameter nach CHR-Verfahren

