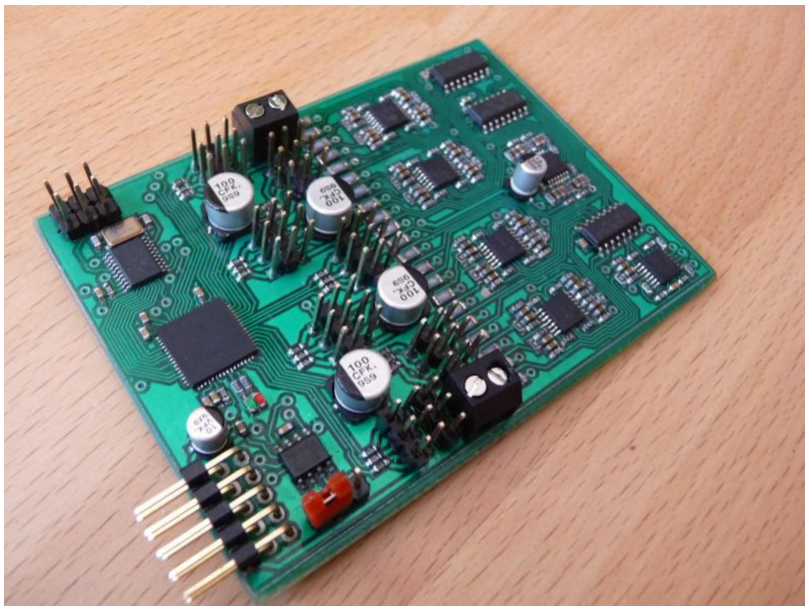
Insgesamt 18 Servos wollen mit einem Signal gefüttert werden. Um für Zukünftige Anwendungen gerüstet zu sein habe ich die Schaltung jedoch auf 21 Kanäle ausgelegt.

Die wichtigsten Funktionen im Überblick:

- Ausgabe von 21 Servo Signalen mit einer theoretischen Auflösung von 16 000 Stufen pro Servo im Bereich von 0,5 - 2,5 ms Impulsdauer. Praktisch ist die Auflösung durch sich möglicherweise überschneidende Interrupts geringer. Dies spielt jedoch kaum eine Rolle, da die Servos ohnehin nicht in der Lage sind so genau zu stellen
- Strommessung auf jedem einzelnen Servo Kanal
- Ausgabe der Messwerte auf dem CAN Bus
- 50 mal pro Sekunde die Soll- Position der Servos empfangen, umrechnen, und entsprechend stellen



Download Schaltplan als PDF

Insbesondere die Strommessung hat mir viel Kopfzerbrechen bereitet. In einer ersten Version der Schaltung habe ich die an den Shunts abfallende Spannung auf 3 Tiefpässe gemuxt,

verstärkt und dann mit dem AD Wandler eingelesen. Das Problem dabei war die große Zeitkonstante der Tiefpässe. Diese benötigen ca. 250 ms um auf ein neues Signal einzuschwingen. Durch geschickte Auswahlder Reihenfolge von gerade gemuxten und gewandelten Kanälen war es letztendlich möglich, alle Kanäle in ca. 1,2 s zu sampeln. Da ich jedoch vor habe, über den Servostrom beispielsweise festzustellen, ob ein Fuß mechanisch durch ein Hindernis blockiert wird, ist eine Ansprechzeit von über einer Sekunden nicht akzeptabel. In der endgültigen Version ist daher für jeden Kanal ein eigener Tiefpass 2. Ordnung vorgesehen. Durch die Verwendung von OPs im TSSOP Gehäuse ist hält sich der Platzbedarf noch in Grenzen. Lediglich das löten der vielen 0603 Bauteile hat sich als Geduldsprobe herausgestellt ;)

Doch warum überhaupt filtern? Nun, als es um die Dimensionierung der Stromversorgung ging, habe ich zu Testzwecken die Stromaufnahme eines einzelnen Servos oszilloskopiert. Dabei hat sich herausgestellt dass stets der volle Blockierstrom gezogen wird. Lediglich die anteilige Dauer pro Regelzyklus unterscheidet sich je nach mechanischer Last. Mit anderen Worten: man erhält am Shunt eine PWM, deren Tastverhältnis die Stromaufnahme darstellt.

Da das Thema Servo Ansteuerung wohl auch für viele interessant ist, hier der relevante Code Ausschnitt:

```
#include "globals.h"
#include "servo.h"
#include <util/delay.h>
#include <avr/interrupt.h>

/* Timer 2 erzeugt eine 500µs ISR. In dieser IST wird eine variable hochgezählt, ablauf wie folgt:

0:      es werden 2 16 bit timer gestartet, um nach ablauf von 0 - 2ms die servopins des letzten
 durchgangs zu deaktivieren
1:      nop
2:     nop
3:      es werden die pins für die nächsten beiden servos aktiviert, dadurch ergibt sich beim
 nächsten durchlauf (0) bereits eine implusdauer von 500µs, die durch die 16 Bit timer noch
 verlängert werden kann.

Dadurch ist es möglich, trotz des Zeitfensters von 2ms pro Servo, eine maximale Impulslänge
von
2,5ms zu realisieren, und somit den Stellbereich der Servos zu vergrößern.

f_Servo ~ 48Hz
```

```
*/

/*#############################################################################
*/

void servo_init(void){
 TCCR2 |= (1 << CS22);    //Timer 2 (2,048ms duch preload mit 128) Prescaler = 256
 TIMSK |= (1 << TOIE2) | (1 << TOIE1);    //Enable Overflow Interrupt at Timer 2 und 1
 ETIMSK |= (1 << TOIE3);                  //Enable Timer 3 Overflow Interrupt

 //Sämtliche Servo Signal Pins auf Ausgang setzen

 DDRC = 0xff;
 DDRG |= (1 << PG0) | (1 << PG1) | (1 << PG2);
 DDRA |= (1 << PA1) | (1 << PA2) | (1 << PA3) | (1 << PA4) | (1 << PA5) | (1 << PA6) | (1 <<
PA7);
 DDRD |= (1 << PD5) | (1 << PD6) | (1 << PD7);


 for (unsigned char i = 0; i<18; i++){
 servo_position[i] = 49536;       // == 1,5ms Pulsbreite (Mittelstellung) //41536
 }
 servo_number = 0;
 servo_timer_step_counter = 0;

 TCNT2 = 225;    //Int alle 500µs
}

/*#############################################################################*/

ISR(TIMER2_OVF_vect){                          //500µs Routine

 servo_timer_step_counter++;
 if(servo_timer_step_counter > 3){
 servo_timer_step_counter = 0;
 }

 if (!servo_timer_step_counter){       //Bei 0

 ms2 = 1;       //Globales Timebase Flag

 TCNT1 = servo_position[servo_number];    //Timer 1 vorladen
 servo_number++;
 TCNT3 = servo_position[servo_number];    //Timer 3 vorladen
 servo_number++;
```

```
 timer_hast_to_turn_off = servo_number;

 if (servo_number >= 20){
 servo_number = 0;
 }else{
 TCCR1B |= (1 << CS10);              //Timer 1 aktivieren
 TCCR3B |= (1 << CS30);              //Timer 3 aktivieren
 }
 }

 if (servo_timer_step_counter == 3){
 switch (servo_number){
 case 0: PORTG |= (1 << PG0) | (1 << PG1); ms2_time_slot = 0; break;
 case 2: PORTC |= (1 << PC0) | (1 << PC1); break;
 case 4: PORTC |= (1 << PC2) | (1 << PC3); break;
 case 6: PORTC |= (1 << PC4) | (1 << PC5); break;
 case 8: PORTC |= (1 << PC6) | (1 << PC7); break;
 case 10: PORTG |= (1 << PG2); PORTA |= (1 << PA7); break;
 case 12: PORTA |= (1 << PA6) | (1 << PA5); break;
 case 14: PORTA |= (1 << PA4) | (1 << PA3); break;
 case 16: PORTA |= (1 << PA2) | (1 << PA1); break;
 case 18: ms2_time_slot = 1; break;    //Zeitraum in dem Ints deaktiviert sein dürfen
 }

 }

 TCNT2 = 225;    //Int alle 500µs
}

/*#########################################################################*
/

ISR(TIMER1_OVF_vect){                  //Dynamische Impulsgenerierung Timer1 (Gerade Pins)
unsigned char servo = servo_number - 2;
 switch (servo){
 case 0: PORTG &= ~(1 << PG0); break;
 case 2: PORTC &= ~(1 << PC0); break;
 case 4: PORTC &= ~(1 << PC2); break;
 case 6: PORTC &= ~(1 << PC4); break;
 case 8: PORTC &= ~(1 << PC6); break;
 case 10:PORTG &= ~(1 << PG2); break;
 case 12:PORTA &= ~(1 << PA6); break;
 case 14:PORTA &= ~(1 << PA4); break;
 case 16:PORTA &= ~(1 << PA2); break;
 case 18:break;
 }
```

```
 TCCR1B &= ~((1 << CS10));           //Timer 1 aus
}

/*#############################################################################
*/

ISR(TIMER3_OVF_vect){                   //Dynamische Impulsgenerierung Timer3  (Ungerade
Pins)
unsigned char servo = servo_number - 2;
 switch (servo){
 case 0: PORTG &= ~(1 << PG1); break;
 case 2: PORTC &= ~(1 << PC1); break;
 case 4: PORTC &= ~(1 << PC3); break;
 case 6: PORTC &= ~(1 << PC5); break;
 case 8: PORTC &= ~(1 << PC7); break;
 case 10:PORTA &= ~(1 << PA7); break;
 case 12:PORTA &= ~(1 << PA5); break;
 case 14:PORTA &= ~(1 << PA3); break;
 case 16:PORTA &= ~(1 << PA1); break;
 case 18:break;
 }
 TCCR3B &= ~(1 << CS30);                    //Timer 3 aus
}

/*#############################################################################
*/
```