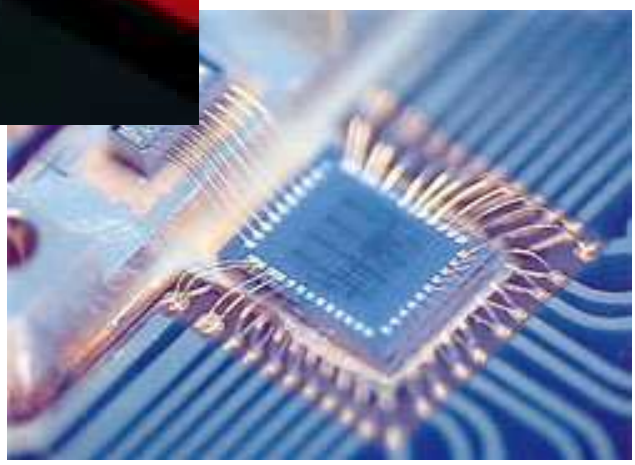
Projekt – Stirlingmotor 2

Drehzahlerfassung mit einem Microcontroller

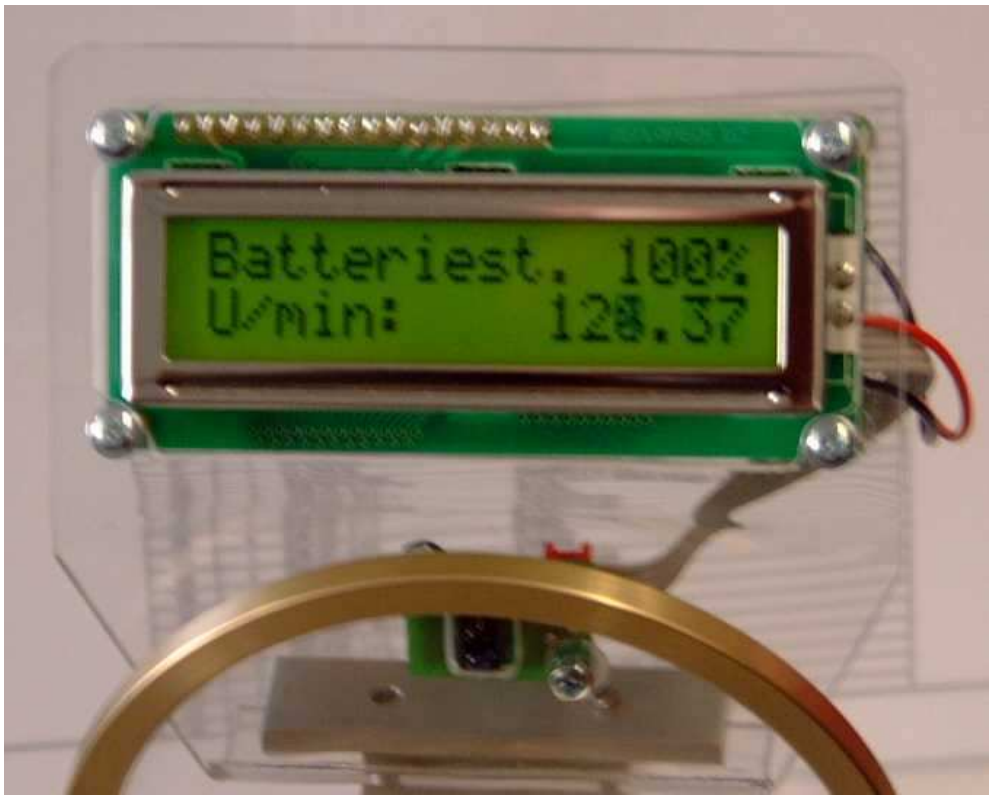


Entwicklung **Stephan Walter**

bei der Lehrfirma SICK **(D)**

(Zusammenstellung A.Schaub)

Realisierung durch die Azubis (Mechatroniker) der Berufsfachschule Emmendingen (D)



Inhaltsverzeichnis

1. Aufgabenstellung

Das Partnerschulenprojekt zwischen der GHS-Emmendingen und der GIB-Liestal sah dieses Mal eine Erweiterung an ihrem Stirlingmotor vor. Die Laufeigenschaften des Stirlingmotors sollten durch mechanische Raffinessen verbessert werden und durch eine elektronische Drehzahlanzeige sollte das Projekt abgerundet werden. Die Projekteinteilung sah vor, dass die Schweizer Schüler den Maschinenbaupart übernehmen und wir Emmendinger für die Drehzahlanzeige zuständig sind.
Die Aufgabenstellung war somit klar: Entwicklung einer elektronischen Drehzahlanzeige für einen Stirlingmotor, welche möglichst kompakt und preisgünstig sein sollte. Die Drehzahlerfassung sollte am Schwungrad des Motors erfolgen, dazu standen mehrere Sensoren zur Verfügung, die auf unterschiedlichen Prinzipien der Gegenstands-Materialerfassung basieren. Als Spannungsquelle sollte mit einer 9V-Block-Batterie gearbeitet werden, um die Mobilität des Ganzen zu gewährleisten. Ihre Gestaltung, sowie die Auswahl der benötigten Bauteile waren frei wählbar.

2. Die Idee

Meine Idee war, eine einfache und kompakte Schaltung zu entwerfen, die auf dem Stand der Elektrotechnik ist. Die Schaltung sollte in der Lage sein, ständig die aktuelle Drehzahl anzuzeigen und auch Features, wie zum Beispiel den Batteriestatus. Damit man später die Werte nur noch über ein Display einfach abzulesen hat, habe ich mich für einen Microcontroller entschieden,

den ich individuell programmieren kann. Durch die Verwendung des Controllers fallen natürlich eine Menge anderer Bauteile weg, auf die man bei anderen Applikationen nicht verzichten könnte. Bei der Controllerauswahl entschied ich mich für einen Pic 18F2220 von Microchip. Er überzeugte mich hauptsächlich durch seine Bauart und einen relativ günstigen Preis. Beim Display fiel die Entscheidung auf ein 2x16 Zeichen alphanumerisches Display mit Hintergrundbeleuchtung (2x16 Zeichen bedeutet: 2 Spalten mit jeweils 16 Zeichenreihen). Zur Drehzahlerfassung habe ich mich für kleine Reflexionslichttaster entschieden, die man direkt auf eine Platine montieren kann.

3. Entwicklung und Bau

Wie bei allen Entwicklungen üblich, geht als Erstes ein Prototyp voraus, der mir hauptsächlich zur Software- und Hardwareentwicklung dient. Dies war bei diesem Projekt nicht anders.
Natürlich war zu diesem Zeitpunkt auch noch nicht klar, ob meine Idee bei den Lehrern und Klassenkollegen Anklang findet. Als dann klar war, dass meine Entwicklung 80 Mal gebaut werden sollte, wurden natürlich Layout und Software komplett neu erstellt und überarbeitet. Die Platine sollte kleiner und in einen Aufbau montiert werden. Der Aufbau sollte dann auf dem Motor mit Schrauben befestigt werden. Die Anforderungen waren jetzt, die Platine und die Aufnahme so zu konstruieren, dass alles in einem relativ kleinen Rahmen Platz findet und trotzdem ein gewisses Budget nicht überschreiten wird.

3.1 Schaltplan und Platinenlayout

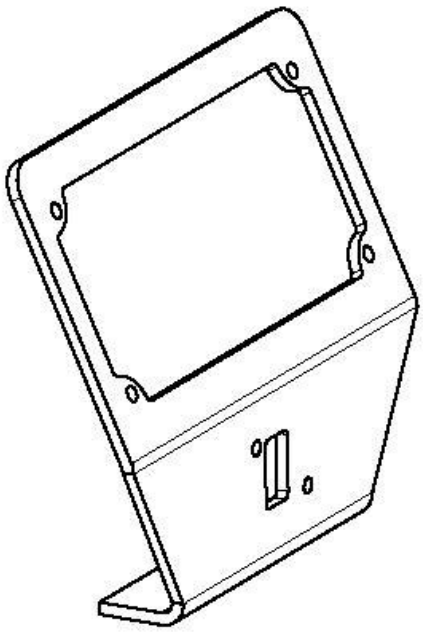
Schaltplan wie auch das Platinenlayout wurden mit Eagle erstellt. Eagle ist eine Software, in der man Schaltpläne erstellen kann, um später daraus Platinenlayouts zu fertigen. Bei unserer Platine handelt es sich um eine Zwei-Layer-Platine, sprich, es befinden sich zwei von einander getrennte Kupferschichten auf der Platine. Auf jeder dieser Layer verlaufen dann die Wires “Leitungen“, die die Bauteile miteinander verbinden.
Schaltplan und Layout: siehe Punkt 6 bis 8.

3.2 Programmierung der Software

Die Software wurde in der Programmiersprache C geschrieben, in einer Programmierumgebung, die sich MPLAB nennt. Man kann sich das so vorstellen, dass der Softwarecode in ein Programm am Computer geschrieben wird, um später innerhalb dieses Programms mit Hilfe eines Compilers in die so genannte Maschinensprache umgesetzt zu werden. Bei dem Compiler handelt es sich um den weit verbreiteten CCS- Compiler.
Der komplette Quellcode: siehe Punkt 10.

3.3 Platinenhalterung

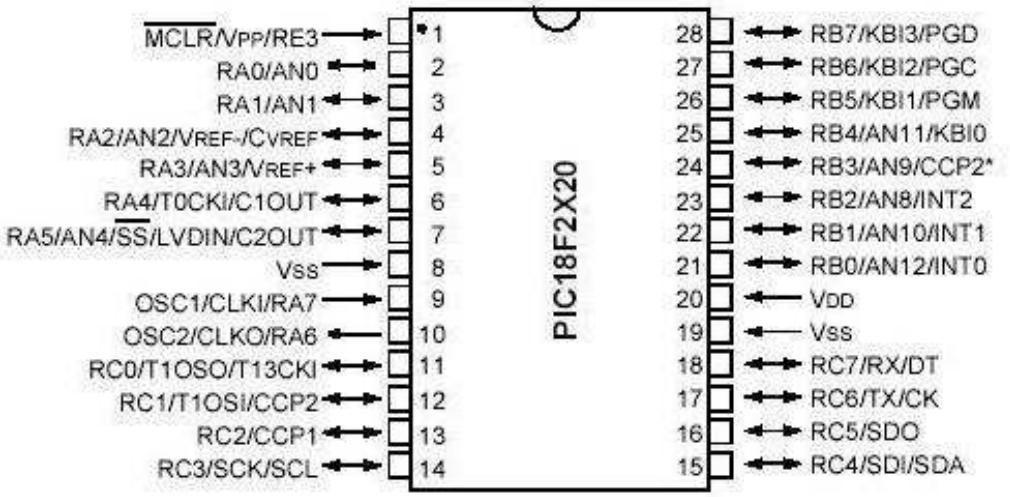
Den Platinenhalter entwickelte ich anhand einer Vorgabe der Schweizer Konstrukteure. Der Halter ist so konstruiert, dass im oberen Teil das Display und die Controllerplatine und im unteren Teil die Sensorplatine befestigt werden. Der Halter ist aus Plexiglas gefertigt und sitzt direkt auf dem Stirlingmotor, ist aber jederzeit durch Schrauben leicht abnehmbar.



4. Verwendete Bauteile

4.1 Pic 18F2220

Der Pic 18F2220 eignet sich optimal für kleinere Applikationen im Hobbybereich bis hin zu anspruchsvollen Lösungen. Der einzige Nachteil, den man nicht leugnen kann, ist, dass er nur über ein kleines ROM und RAM verfügt, was in unserem Fall aber gereicht hat, wie man am Endergebnis sehen kann.



4.2 Display

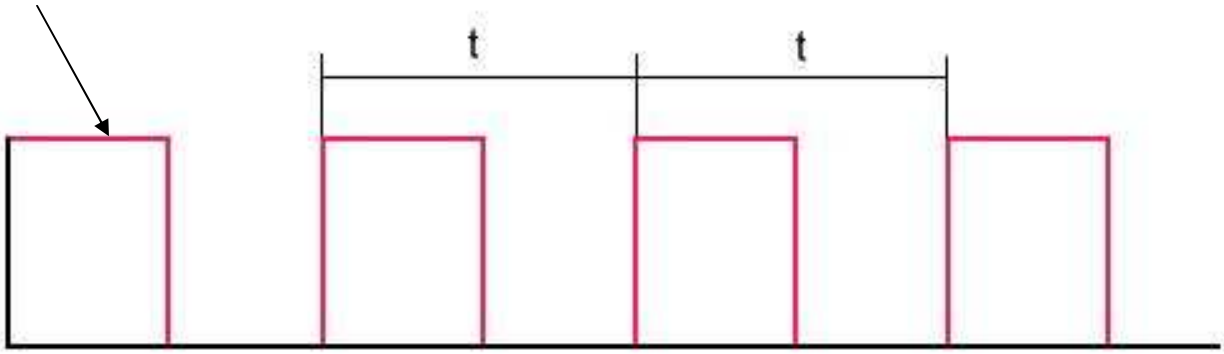
Die von uns verwendeten zweizeiligen Displays besitzen eine LED-Hintergrundbeleuchtung und können über dem ASCII-Code direkt vom Controller aus angesteuert werden. Wie man auf dieser ASCII-Tabelle nebenan sehen kann, hat jedes alphanumerische Zeichen einen binären Zahlencode, der meist hexadezimal dargestellt wird. Alphanumerische Zeichen sind im engeren Sinne entweder Buchstaben, Ziffern oder Sonderzeichen.

upper 4 bit / lower 4 bit	0000	0010	0011	0100	0101	0110	0111
0000	CG RAM (1)						
0001	(2)						
0010	(3)						
0011	(4)						
0100	(5)						

5. Prinzip der Frequenz- Drehzahlmessung

Das Problem anderer Schaltungen ist, dass sie erst nach einer gewissen Zeit die Drehzahl pro Minute ausgeben können und dadurch immer einen veralteten Drehzahlwert anzeigen. Durch den µ-Controller ist es möglich, auf jede Speiche des Schwungrades einen aktuellen Drehzahlwert auf dem Display anzuzeigen. An diesem Schaubild kann man grafisch sehen, wie der Controller die Signale vom Sensor bekommt und dann selbstständig die Zeit zwischen den Impulsen misst. Anhand der dargestellten Formel rechnet er die Zeitimpulse in Umdrehungen pro Minute um.

5.1 Sensorsignal



5.2 Drehzahlbestimmung

Der Controller zählt die Impulse p vom Start bis zum Ende eines Ereignisses. Die Zeit t zwischen zwei Werten beträgt beim „Pic 18F2220“ t=51.2 µs.

Die Zeit (Periode) T für ein Ereignis berechnet sich also $\boxed{T = p \cdot t}$ (1)

für die Frequenz f_a gilt allgemein $\boxed{f_a = \frac{1}{T}}$ (2)

daraus folgt für die Frequenz f mit N=7 Speichen $\boxed{f = \frac{1}{T \cdot N}}$ (3)

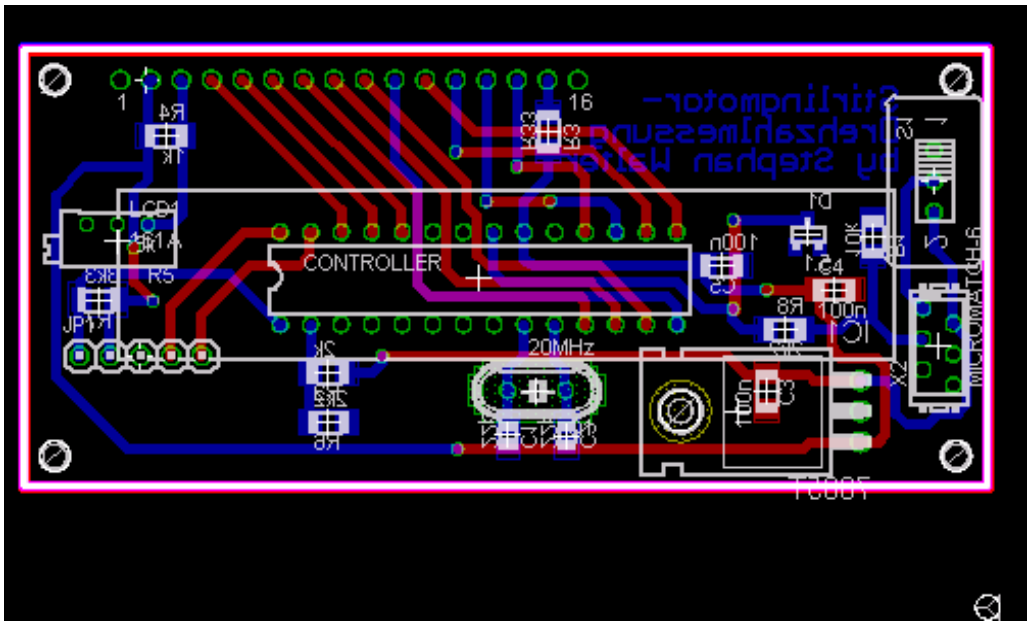
die Drehzahl n aus f berechnet sich $\boxed{n = f \cdot 60s/\min}$ (4)

Für die Formel (5), wie sie im Listing auf Seite 12 zu finden ist, werden die Formeln (1) bis (3) in Formel (4) eingesetzt und es folgt für die Drehzahl n als Funktion vom Zählwert p:

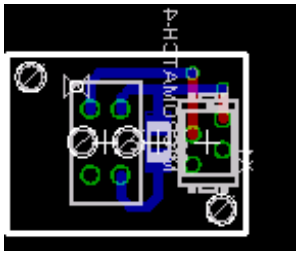
$$\boxed{n = f(p) = \frac{60s/\min}{T \cdot N} = \frac{60s/\min}{p \cdot t \cdot N} = \frac{60s/\min}{p \cdot 0.000'0512s \cdot 7} = \frac{1/\min}{p \cdot 0.000'0512 \cdot 7} \cdot 8.571} \quad \left[\frac{U}{\min}\right] \quad (5)$$

T	[s]	Periodendauer, Zeit für ein Ereignis
p	[ ]	gezählte Impulse des Controllers
t	= 51.2 µs	Zeitintervall zwischen zwei Zählwerten
f_a	[Hz=s^{-1}]	Frequenz in Herz allgemein
f	[Hz=s^{-1}]	Frequenz in Herz des Schwungrades
N	= 7	Anzahl Speichen im Schwungrad
n	**[min^{-1}]**	**Drehzahl in U/min**

6. Controllerplatine

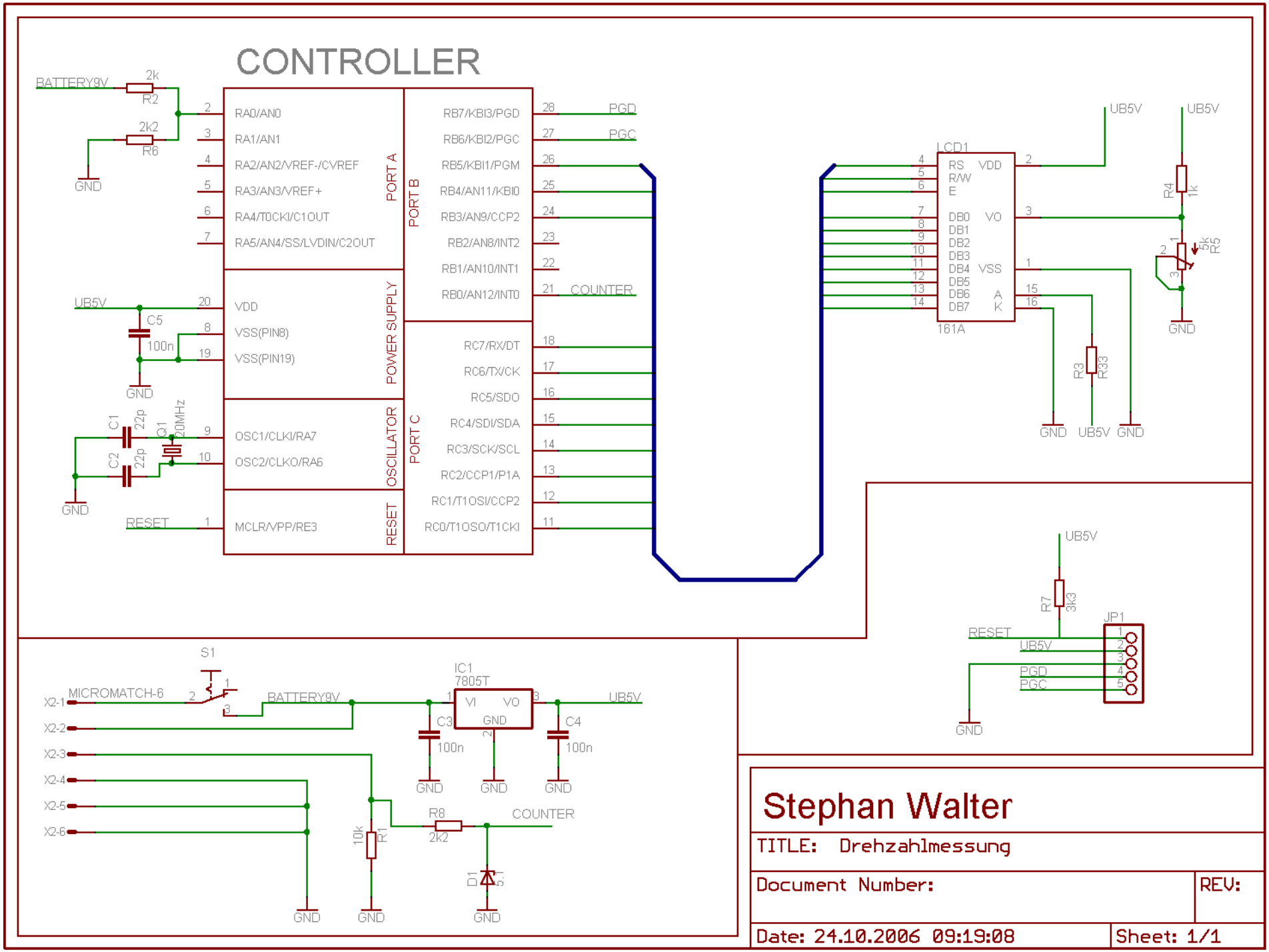


7. Sensorplatine



8. Schaltpläne

Schaltpläne von Controller und Sensor auf folgender Seite



Stephan Walter

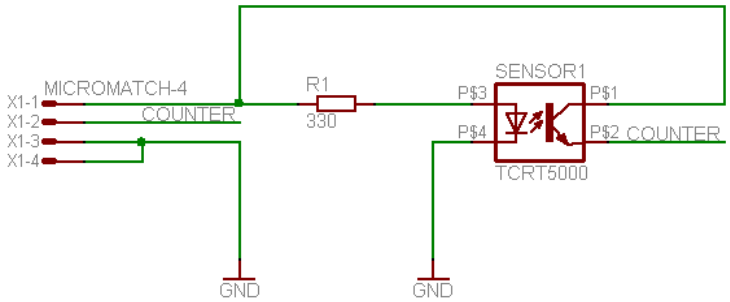
TITLE: Drehzahlmessung

Document Number:

REV:

Date: 24.10.2006 09:19:08

Sheet: 1/1



Stephan Walter	
TITLE: Sensor_Platine	
Document Number:	REV:
Date: 24.10.2006 09:16:58	Sheet: 1/1

9. Partlist

9.1 Controllerplatine

Menge	Wert	Device	Bauteile	Preise	Bestellnr.	Lieferant
1		PIC18F2220-I/SP	Controller	6.21 €	siehe Internet	Farnell
0,5*			Stiftleiste 36pol.	1.04 €	972-9038	Farnell
0,5*			Buchsenl. 36pol.	4.09 €	972-8910	Farnell
1	5pol	PINHD-1X5	JP1	0.18 €	SL 1X40G 2,54	Reichelt
1	1k	R-EU_R1206	R4	0.10 €	SMD 1/4 1,0k	Reichelt
1	2k	R-EU_R1206	R2	0.10 €	SMD 1/4 2,0k	Reichelt
2	2k2	R-EU_R1206	R6, R8	0.10 €	SMD 1/4 2,2k	Reichelt
1	3k3	R-EU_M1206	R7	0.10 €	SMD 1/4 3,3k	Reichelt
1	D 5,1	ZENER-DIODESOT23	D1	0.09 €	SMD ZD 5,1	Reichelt
1	5k	R-TRIMM64X	R5	0.64 €	64-5,0k	Reichelt
1	10k	R-EU_R1206	R1	0.10 €	SMD 1/4W 10K	Reichelt
1	20MHz	CRYTALHC49S	Q1	0.23 €	20-HC49U-S	Reichelt
2	22p	C-EUC0805	C1, C2	0.05 €	NPO-G1206 22P	Reichelt
3	100n	C-EUC1206	C3, C4, C5	0.09 €	X7R-G1206 100N	Reichelt
1	162C	161C	LCD1	6.90 €	LCD 162C LED	Reichelt
1	7805T	7805T	IC1	0.17 €	µA 7805	Reichelt
1		MICROMATCH-6	X2	0.29 €	MM FL 6G	Reichelt
1		MICROMATCH-6	X2	0.35 €	MM SL 6SK	Reichelt
1	R33	R-EU_R1206	R3	0.10 €	SMD 1/4W 33	Reichelt
1		IC-Sockel		0.33 €	GS 28P-S	Reichelt

* Je 1 Stift- und Buchsenleiste für 2 Geräte

9.2 Sensorplatine

Menge	Wert	Device	Bauteile	Preise	Bestellnr.	Lieferant
1	330R	R-EU_R1206	R1	0.10 €	SMD 1/4W 330	Reichelt
1		MICROMATCH-4	X1	0.25 €	MM FL 4G	Reichelt
1		MICROMATCH-4	X1	0.29 €	MM SL 4SK	Reichelt
1	TCRT5000	TCRT5000	SENSOR1		siehe Internet	Distributor
1	3Polig	Flachbandkabel				Reichelt
1		Batterieclip				Reichelt

9.3 Mechanische Teile

Menge	Wert	Device	Bauteile	Preise	Bestellnr.	Lieferant
8	M 2,5x5	Zyl.-Kopfschrauben für Displaypl.				
2	M 2,5	Muttern für Sensorpl.				
2	M 2,5x8	Zyl.-Kopfschrauben für Sensorpl.				
4	M2,5	U-Scheiben				
8		Bolzen M2,5 L=5		0.22 €	473-6140	Farnell
4		Bolzen M2,5 L=10		0.11 €	474-3398	Farnell

10. Quellcode

```
//++++++++++++Drehzahlmessung by Stephan Walter+++++++++++++


//++++++++++++++++++Initialisierung++++++++++++++++++++++++++


#device PIC18F2220 *=16 adc=10
#include<18f2220.h>
#use delay(clock=20000000)
#include<lcd2x16.h>
#include"lcd2x16.c"


//++++++++++++++++++Globale-Variablen++++++++++++++++++++++++++


unsigned int16 merker1=0;
unsigned int16 merker2=0;
unsigned int16 merker3=0;
float drehz=0;
unsigned int8 intmerker=0;
char Zero[7];
char Zerobat[4];
unsigned int8 merker4=0;
unsigned int16 sysvolt=0;


//++++++++++++++++++Funktionsprototypen++++++++++++++++++++++


void Batteriestatus();
int16 get_system_voltage();


//++++++++++++++++++Hauptprogramm++++++++++++++++++++++++++++


void main(void)
{

        int8 n=0;
        int8 s=0;
        set_tris_B (0x01);
        setup_adc(ADC_CLOCK_INTERNAL);
        setup_adc_ports(AN0_ANALOG);
        setup_timer_0(RTCC_DIV_256|RTCC_INTERNAL);
        ext_int_edge(L_to_H);
        enable_interrupts(int_ext);
        enable_interrupts(int_RTCC);
        enable_interrupts(global);
        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
        enable_interrupts(int_timer1);
        enable_interrupts(global);
        lcdinit();
        sprintf(STR,"GHSE-Emmendingen");
        lcdprint(STR,1,1);

        sprintf(STR,"feat.GIB-Liestal");
        lcdprint(STR,2,1);
        delay_ms(3000);
```

```
        sprintf(STR,"   coded by     ");
        lcdprint(STR,1,1);
        sprintf(STR," Stephan Walter ");
        lcdprint(STR,2,1);
        delay_ms(3000);


while(1)
{
                Batteriestatus();


                drehz=merker1;
                drehz=(1/(drehz*0.0000512)*8.571);


        if (merker2==0)
        {
                sprintf(Zero,"%0f",drehz);

        for (n=0;Zero[n]==0x30;n++)
                Zero[n]=' ';

                sprintf(STR,"U/min:   %s",Zero);
                lcdprint(STR,2,1);

        }
        else
        {
                sprintf(STR,"Waerme zufuehren");
                lcdprint(STR,2,1);
        }



        if (merker3<28&&merker4<3)
        {
                sprintf(STR,"Stirlingmotor-  ");
                lcdprint(STR,1,1);
        }

        if (merker3>28&&merker4<3)
        {
                sprintf(STR,"Drehzahlmessung ");
                lcdprint(STR,1,1);
        }


        if (merker4==3)
        {
                sprintf(Zerobat,"%03lu",sysvolt);

        for (s=0;Zerobat[s]==0x30;s++)
                Zerobat[s]=' ';

                sprintf(str,"Batteriest. %s%%",Zerobat);
                lcdprint(str,1,1);
        }
}
}


#int_ext
```

```
void Counterinterrupt()
{

                merker1=get_timer0();
                set_timer0(0x00);

        if (merker2==1&&drehz>2.3)
                {

                }

        if (intmerker==3)
                {
                intmerker=0;
                merker2=0;
                }

        intmerker++;

}


#int_RTCC
void RTCC()
{

        merker2=1;

}


#int_timer1

void timer1()

{
        merker3++;



        if (merker3>56)
                {
                merker3=0;
                merker4++;
                }

        if (merker4==4)
                {
                merker4=0;
                }
}
```

```
//++++++++++++++++++++++++Unterprogramm++++++++++++++++++++++++++++


void Batteriestatus()

{
        sysvolt = get_system_voltage();
        delay_ms(100);

        if (sysvolt>860)
                sysvolt=100;

        else if (sysvolt<540)
                sysvolt=0;

        else
                sysvolt=((sysvolt-540)/3.2);
}

int16 get_system_voltage ()

{
        int8 n = 0;
        unsigned int32 sample=0;
        int16 result;
        set_adc_channel(0);
        for (n=0;n<32;n++)

        {
                delay_us(10);
                sample = (sample+read_adc());
        }

        sample=sample/32;
        result = sample;
        return result;
}



//++++++++++++++++++++++++++LCD.Header++++++++++++++++++++++++++++++++++

//LCD-Treiber Alphanumerische Displays

//Pindefinitionen

#DEFINE LCD_RW          PIN_B4
#DEFINE LCD_CMD         PIN_B5
#DEFINE LCD_CLK         PIN_B3
#DEFINE LCD_Dataport output_c

char STR[20];

//Funktionsprototypen

void lcdclk (void);
void lcdinit (void);
void lcdprint (char string[], int zeile, int spalte);
void lcdputch (char in, int zeile, int spalte);
void setlcdpos (int x, int y);

//++++++++++++++++++++++++++LCD.Code++++++++++++++++++++++++++++++++++
```

```
//LCD-Treiber Alphanumerische Displays

void lcdinit (void)
{
	//set_tris_c (0x00);
	set_tris_c (0x00);
	output_low (LCD_RW);
	output_low (LCD_CMD);
	LCD_Dataport (0x3C);
	lcdclk ();
	delay_ms (4);
	LCD_Dataport (0x14);
	lcdclk ();
	delay_ms (4);
	LCD_Dataport (0x0C);
	lcdclk ();
	delay_ms (4);
	LCD_Dataport (0x06);
	lcdclk ();
	delay_ms (4);
	LCD_Dataport (0x02);
	lcdclk ();
	delay_ms (4);
	LCD_Dataport (0x01);
	lcdclk ();
	delay_ms (4);
}


void lcdprint (char string[], int zeile, int spalte)
{
	int i = 0;
	set_tris_c (0x00);
	set_tris_b (0x01);
	setlcdpos (spalte, zeile);
	output_low (LCD_RW);
	output_high (LCD_CMD);
	while ((string[i] != 0x00)&&(i<20))
	{
		LCD_Dataport (string[i]);
		lcdclk();
		i++;
	}
}


void lcdputch (char in, int zeile, int spalte)
{
	set_tris_c (0x00);
	set_tris_b (0x01);
	setlcdpos (spalte, zeile);
	output_low (LCD_RW);
	output_high (LCD_CMD);
	LCD_Dataport (in);
	lcdclk();
}



void setlcdpos (int x, int y)
{
	int error = 0;
```

```
        output_low (LCD_RW);
        output_low (LCD_CMD);
        delay_us (200);
        if ((x < 1) | (x > 20) | (y < 1) | (y > 4))
                error = 1;
        if ((error == 0) && (y == 1))
                LCD_Dataport (0x7F+x);
//              LCD_Dataport (0x7E+x);

        if ((error == 0) && (y == 2))
                LCD_Dataport (0xBF+x);
//              LCD_Dataport (0xBF+x);

        if ((error == 0) && (y == 3))
                LCD_Dataport (0x93+x);
//              LCD_Dataport (0x8F+x);

        if ((error == 0) && (y == 4))
                LCD_Dataport (0xD3+x);
//              LCD_Dataport (0xCF+x);
        delay_us (1000);
        lcdclk();
        delay_us (200);
}


void lcdclk (void)
{
        output_low (LCD_CLK);
        delay_us (100);
        output_high (LCD_CLK);
        delay_us (100);
        output_low (LCD_CLK);
        delay_us (100);
}
```