Tillhanda: Nils Svensson

Inlämnad av: Jenny Sheng

Kurs: DT544C HT14 Signaler och sensorer

Inlämnings datum: 2014-11-04

Epost addresser: qd_sheng2000@hotmail.com

Projekt. handledare: Nils Svensson, Haydar Al Attar

1. Inledning

Vi utförde denna laborationen i Signal- och Sensorkursen under Högskoleingenjörsutbildningen.

Laborationen handlar om att man ska kunna samla in och analysera mätdata från MEMS accelerometern MMA7260QT, att få en viss insikt om hur en accelerometer fungerar samt hur man behandlar och analyserar mätdata.

Programmeringsspråket som har använts är C. Man skriver koden i Atmel Studio och gör en ADC(Analog till digital omvandling), accelerometern avläser accelerationen i de tre rums dimensioner och skickar värden av dessa till X-CTU. Dessa värden bli kallades för x, y och z., det kommer att visa olika värde från olika inriktning.

3. Material och verktyg

- En Atmel mikroprocessor ATmega328P
- MEMS accelerometer MMA7260QT
- Xbee moduler för trådlös kommunikation(dessa 3 sitter på ”fjärrkontroll”)
- En arbetsstation med AVR Studio 6 och X-CTU
- Förskriven kod som hanterar överföring via Xbee
- En programmerare för Atmel-processorer
- En mätfixtur med en gradskiva (0-360°)

4. Metod och genomförande

Vi ska mäta x, y, z värde på olika känslighet i olika inriktning i accelerometer.
Genom ställer in g-select, enligt tabel nedan:

Table 3. g-Select Pin Descriptions

g-Select2	g-Select1	g-Range	Sensitivity
0	0	1.5g	800 mV/g
0	1	2g	600 mV/g
1	0	4g	300 mV/g
1	1	6g	200 mV/g

Så kunde vi ställar in olika känslighet på accelerometern.

5. Resultat

Vi har 360 grader i varje axel, $2^{10}=1024$ värden, så har jag räknat den motsvarande siffror till varje 45 grader svänger.

0	0
45	128
90	256
135	384
180	512
225	640
270	768
315	896
360	1024

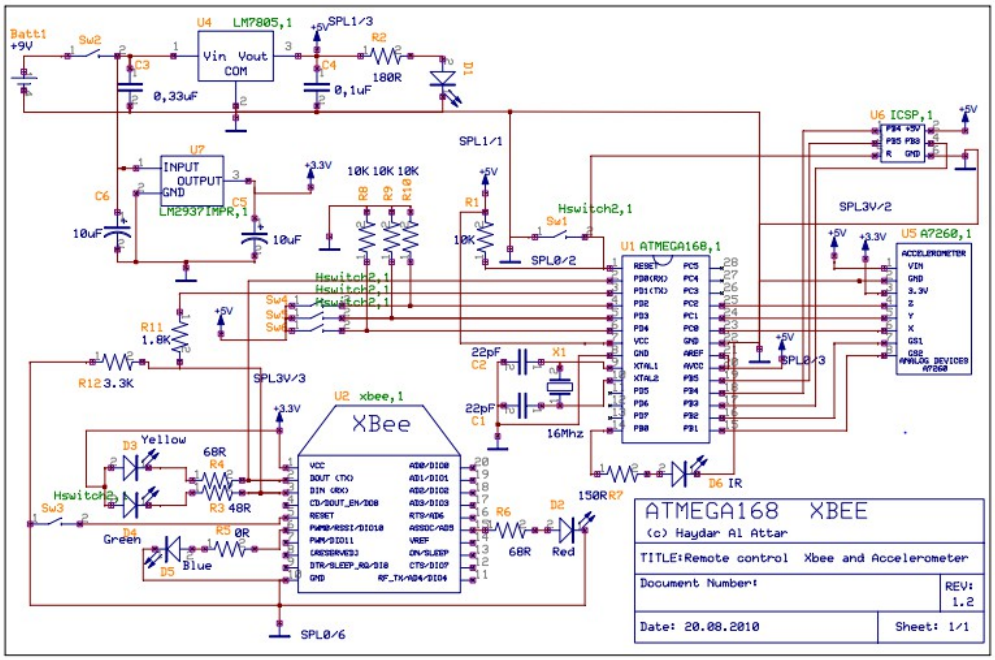
Detaljer kan man kolla på bifoga.

6. Diskussion

När man vrider accelerometer på x-axel med olika känslighet, både x och z värde förändras. Det verkar vara rimligt.

.

Kopplingsschema:



Bilden tagit från accelerometer schematic remote control itslearning.

Källkode:

```
/*
 * signalLab1.c
 * Created: 2014-10-07
 *  Author: jenny
 */
#include<avr/io.h>

#include<util/delay.h>

#include<stdio.h>

#include<avr/interrupt.h>

#include <math.h>



#define F_CPU 16000000UL

#include "H_filen_uart.h.h"



uint16_t X; //unsigned 16-bit integer
```

```
uint16_t Y; //unsigned 16-bit integer

uint16_t Z; //unsigned 16-bit integer




char s[7];



/*****************************************************//**

 * Mainfunction

 *****************************************************/

int main(void)

{
        DDRB=0;
        PORTB &= 0b00000000;
        PORTB |= 0b00000010;//set g-select
        DDRC=0xff;
        PORTC=0xff;
```

```
initUART();
initADC();


while(1){
_delay_ms(1000);
ADC_omvandlingX();
_delay_ms(3000);
ADC_omvandlingY();
_delay_ms(3000);
ADC_omvandlingZ();
_delay_ms(3000);


                /*------------------------------ADC_0------------------------------*/

                uart_puts( "X:" );


                if (X >= 0)

                {

                                uart_puts( " " );

                                itoa( X, s, 10 );
```

```
                uart_puts( s );




                uart_puts( "\t\t" );

        }

        else

        {

                itoa( X, s, 10 );

                uart_puts( s );

                uart_puts( "\t\t" );

        }

        /*------------------------------ADC_1------------------------------*/

        uart_puts( "Y:" );
```

```
if (Y >= 0)

{

                uart_puts( " " );

                itoa( Y, s, 10 );

                uart_puts( s );

                uart_puts( "\t\t" );

}

else

{

                itoa( Y, s, 10 );

                uart_puts( s );

                uart_puts( "\t\t" );

}
```

```
/*------------------------------ADC_2------------------------------*/

uart_puts( "Z:" );

if (Z >= 0)

{

        uart_puts( " " );

        itoa( Z, s, 10 );

        uart_puts( s );

        uart_puts( "\t\t" );

}

else

{

        itoa( Z, s, 10 );

        uart_puts( s );

        uart_puts( "\t\t" );
```

```
                }
                uart_puts( "\n\r" ); //new line
                _delay_ms(10000);


}//end while

}//end main




void ADC_omvandlingX(){
                ADMUX = (ADMUX & 0xf0 | 0x00);
                ADCSRA |= 0x40;//start ADC
                while (ADCSRA & (1<<ADSC)){}
                                X=ADCL|(ADCH<<8);

}

void ADC_omvandlingY(){
                ADMUX = (ADMUX & 0xf0 | 0x01);
                ADCSRA |= 0x40;
```

```
                while (ADCSRA & (1<<ADSC)){}
   Y=ADCL|(ADCH<<8);
}

void ADC_omvandlingZ(){
                ADMUX = (ADMUX & 0xf0 | 0x02);
                ADCSRA |= 0x40;
                while (ADCSRA & (1<<ADSC)){}//start conversion bit 6
                Z=ADCL|(ADCH<<8);
}


void initADC(){
                ADMUX = 0b01000000;
                ADCSRA = 0;
                ADCSRA |= (1<<ADEN);
                ADCSRA |= (1<<ADPS0);
                ADCSRA |= (1<<ADPS1);
                ADCSRA |= (1<<ADPS2);


}
```