Guitar Tuner

EITF11 Digitala Projekt, Projektrapport
Elektro- och informationsteknik

Carolin Sundvik, Magnus Torstensson
5/9/2011

Abstract

To be able to play a guitar it needs to be tuned, and to do this easily you need to have a guitar tuner. This report presents the way to construct a guitar tuner with an ATmega16, a display and some diodes. In this particular case the tone frequency is created by an audio oscillator. C programming was used to read the different frequencies and present the correct tune and tuning on the guitar tuner.

Inledning

Detta projekt är en del i kursen Digitala projekt EITF11. Med ett gediget gitarrintresse tog vi oss an uppgiften om att göra en prototyp till en gitarrstämmare utifrån ett fåtal tillgängliga komponenter. Pga. tidsbrist avgränsade vi oss genom att simulera gitarrens tonfrekvenser med hjälp av en tongenerator.

Gitarrstämmaren visar korrekt resultat för de givna frekvenserna för standardstämningen på en gitarr.

Kravspecifikation

Kravspecifikationen upprättades för att ge en bild av hur vi vill att vår stämapparat skall fungera som färdig prototyp.

Den ursprungliga kravspecifikationen

- Stämapparaten skall ge ett tillfredställande resultat vid stämning av en semiakustisk gitarr i normal stämning, med andra ord E, A, D, G, B, E
- Stämapparaten skall kunna ange vilken närmsta regelrätta sträng är utifrån tonen som spelas
- Instrumentet ansluts via sladd
- Fem lysdioder anger felmarginalen för tonen i förhållande till närmsta förinställda frekvens
- En LCD-display visar vilken ton som är anslagen

Uppfyllda krav

- Stämapparaten ger tillfredsställande resultat för normal stämning när dessa frekvenser framkallas med hjälp av en tongenerator
- Stämapparaten anger närmsta regelrätta sträng för given frekvens
- Fem lysdioder anger felmarginal i förhållande till närmsta förinställda frekvens
- En LCD-display visar vilken ton som tongeneratorn motsvarar för en gitarr

Teori

Enligt kravspecifikationen krävdes ett antal komponenter, en display att visa den närmast anslagna tonen på samt ett antal lysdioder som indikerade hur nära man var den korrekta frekvensen.

Vi anpassade komponentvalet efter vår kravspecifikation och valde en LCD-display och en processor som lämpade sig väl för funktionerna vi avsåg att använda. Med hjälp av Powerlogic skissade vi upp följande kopplingsschema för vår konstruktion för att få en god överblick.

Till vår hjälp hade vi manualen för ATmega16[i] och manualen för LCD-displayen[ii]. Vi använde oss av Wikipedias sida för gitarrstämning[iii] för att få reda på de korrekta frekvenserna för varje ton.

Vår konstruktion innehöll följande komponenter:

- ATmega16

- 5 * LED dioder varav en grön, två gula och två röda
- 5 st motstånd
- SHARP Dot-Matrix LCD Units display
- AVR JTAG ICE

Vi använde oss sedan av AVR Studio för att skriva själva koden i C för hur dioderna skulle lysa samt vad som skulle visas på displayen för de olika frekvensintervallen.

Genomförande

Projektet inleddes med att införskaffa sig absolut grundläggande kunskaper inom datorteknik, analog elektronik och digitalteknik genom fyra föreläsningar i kursen, två laborationer samt en introduktion inom C programmering. En kravspecifikation för vår konstruktion upprättades för att ge en översikt av vårt projekt, se vilka komponenter som skulle krävas för prototypen samt veta vilka mål vi hade med vår gitarrstämmare.

Initialt gjordes ett kopplingsschema över prototypen och lämpliga komponenter införskaffades (se teori-avsnittet). Byggandet enligt kopplingsschemat började och den mesta av tiden lades på att hitta rätt port som skulle kopplas var. Denna information fann vi genom att studera de olika manualerna till komponenterna. Vi hade även mycket stor hjälp av vår handledare som ställde upp på att svara på frågor när dessa frekvent uppkom.

Först monterades grundkomponenterna, det vill säga AVR:en, LCD-displayen samt strömförsörjare, på plattan. Vi satte sedan dit de fem lysdioderna samt deras motstånd och kopplade sedan samman samtliga komponenter.

Ett steg i taget testades hårdvaran för att se att varje komponent fungerade innan den riktiga programmeringen påbörjades. Först testade vi att dioderna fungerade som de skulle. Vi upprättade sedan rätt sekvenser för LCD-displayen för dess basala funktioner och undersökte hur vi sedan skulle skriva ut bokstäverna motsvarande den närmast anslagna tonen. När vi sedan skulle programmera avbrott för displayen fungerade inte detta som det skulle. Vi felsökte därför väldigt länge på själva avbrottet och på själva displayens hårdvara då vi trodde att problemet fanns där. Bland annat använde vi oss av ett oscilloskop för att kontrollera hur ofta ett avbrott genererades. Istället upptäckte vi att vi hade missat att programmera delay för displayen och när vi lade till detta fungerade allt då själva problemet ej låg i avbrottet.

När all hårdvara fungerade övergick arbetet till själva programmeringen. Vi hämtade, enligt teori-avsnittet, lämpliga frekvenser för gitarrtonerna. Idén var att i mjukvaran "räkna" antalet pulser med en variabel count som genereras mellan två avbrott och eftersom att vi vet längden på själva avbrottet kan vi då avgöra vilken frekvens som skickas in. Under empiriska undersökningarn med hjälp av ett oscilloskop och vår tongenerator avgjordes vilken count som motsvarade vilken av tonfrekvenserna från en gitarr.

Nedan visas vilken frekvens som motsvarar vilken ton, vilken count som i vår mjukvara representerar denna frekvens samt intervallet för då den givna tonen visas på displayen.

Ton	Frekvens	Count	Intervall
E	82 Hz	21	> 24
A	110 Hz	28	25 - 34
D	147 Hz	39	53 - 45
G	196 Hz	51	46 - 57
B	247 Hz	64	58 - 75
e	330 Hz	85	75 +

Vi programmerade sedan efter ovanstående tabell så att vid exakt rätt count lyser den gröna dioden, vid +/- 1 count lyser den gula på motsvarande side och för de övriga counts i intervallet för varje ton lyser den motsvarande röda dioden. På samma sätt visas motsvarade bokstav för tonen på displayen när count befinner sig i dess intervall.

Resultat

Resultatet blev en stämapparat som kan identifiera en frekvens från en tongenerator. På så sätt saknar prototypen möjligheten att faktiskt uppfatta en frekvens från en gitarr. Prototypen kan även uppfatta felmarginal utifrån de givna definierade frekvenserna som motsvarar en gitarrs standardstämning, EADGBe. Den visar alltså dels vilken ton som är den närmsta standardtonen, men även hur nära insignalsfrekvensen är den närmsta standardtonen.

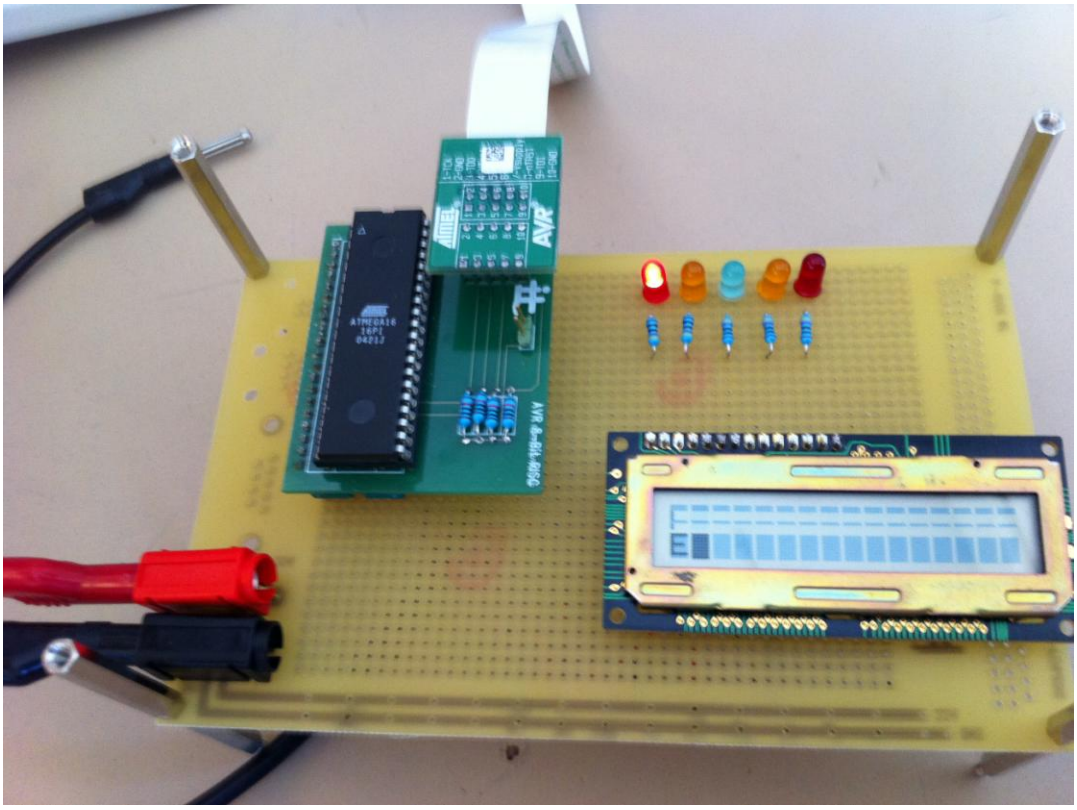
Stämmaren är även kalibrerad efter den tongenerator som användes vid kalibreringen. Dessa kan ge en marginell felmarginal utifrån de teoretiskt exakta värdena för en sträng på en gitarr.

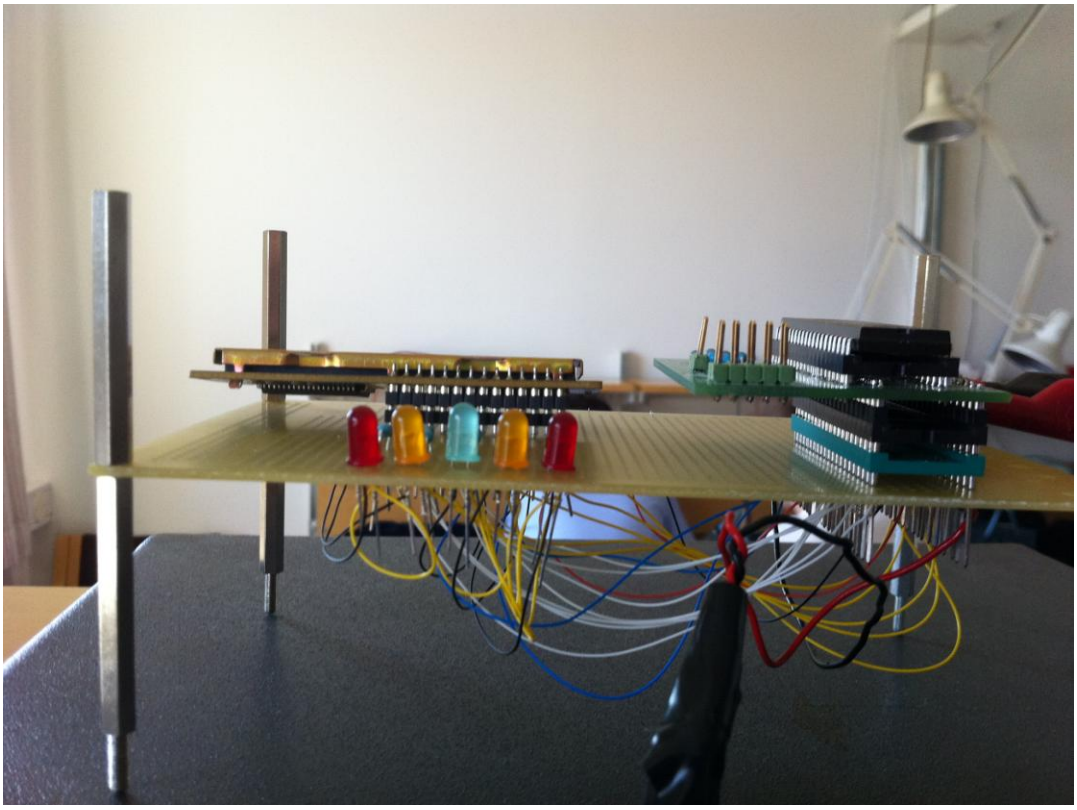
Diskussion

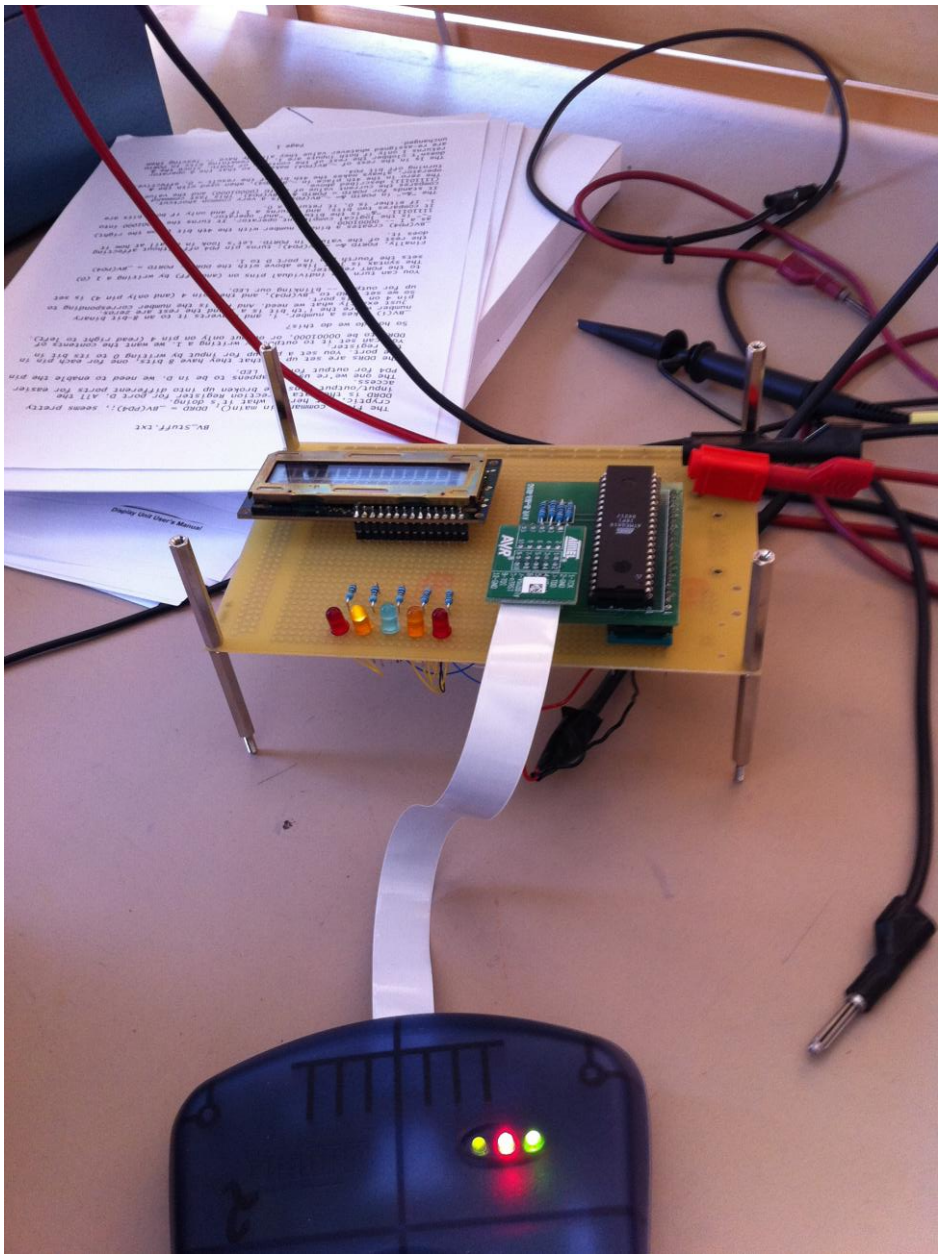
Att stämmaren inte kan användas för att stämma en gitarr kan anses som ett misslyckande. Det hade varit mycket tillfredsställande att kunna stämma sin egna gitarr med hjälp av en egengjord stämmapparat. Dock hade det vid tillfället som beslutet togs att bryta projektet, varit för lite tid kvar att påbörja en utökning av prototypen. Det saknades även resurser såsom en mikrofon som fungerade för den låga spänningen på 5 volt. Vi är mycket nöjda med de resultaten som vi uppnått med tanke på hur låga grundkunskaper som fanns i ämnet vid projektets början.

Vi är mycket nöjda med val av sätt att lösa problemet med hjälp av en counter som motsvarade en viss frekvens. Det blev förhållandevis enkelt att programmera och det behövdes inga onödiga komponenter.

Bilder







Appendix – Källkod

```
#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
//#include <avr/signal.h>


   unsigned char num;
        int count;
        int tune;
        short int test;
        short int ozzy;

        #define rLow 0x02;
        #define yLow 0x04;
        #define green 0x08;
        #define yHigh 0x10;
        #define rHigh 0x20;

//      #define short int E 0b01000001;



        void clearDisp() {
                short int val = 0b00000001;
                writeCmd(val);
                return;
        }

        void writeCmd(short int val){
                PORTA = val;
                PORTC &= ~_BV(0);
                PORTC = _BV(7);
                PORTC &= ~_BV(7);
                _delay_ms(10);                          // Här läses val in (datan vi vill ge som
instruktion)
                return;
        }

        void writeData(short int val){
                //clearDisp();
                writeCmd(0xC0);
                PORTA = val;
                PORTC = 0b00000001;                             // enda skillnaden är att Rs är 1
denna gången
                PORTC = 0b10000001;                             // E ska vara 1   0b 0000 0 101;
                //PORTC &= ~_BV(7);                             // Här läses val in (datan vi vill
skicka till displayen)
                PORTC = 0b00000001;
                _delay_ms(1);
                return;
        }
```

```
	void functionSet(){
		short int val = 0b00111000;
		writeCmd(val);
		return;
	}

	void entryModeSet() {
		short int val = 0b00000110;
		writeCmd(val);
		return;
	}

	void cursorHome() {
		short int val = 0b00000011;
		writeCmd(val);
		return;
	}

	void dispOff(){
		short int val = 0b00001000;
		writeCmd(val);
		return;
	}

	void dispOn(){
		short int val = 0b00001111;
		writeCmd(val);
		return;
	}

	void init(){
		clearDisp();
		functionSet();
		dispOn();
		entryModeSet();
		return;
	}

//	void setInst(){
//		clearDisp();
//		cursorHome();
//		entryModeSet();
//		dispOn();
//	}

	void writeA(){
		writeData(0b01000001);
		tune = 28;
		//
		return;
	}
```

```
void writeB(){
	writeData(0b01000010);
	tune = 64;
	return;
}

void writeD(){
	writeData(0b01000100);
	tune = 39;
	return;
}

void writeBigE(){
	writeData(0b01000101);
	tune = 21;
	return;
}

void writeSmallE(){
	writeData(0b01100101);
	tune = 85;
	return;
}

void writeG(){
	writeData(0b01000111);
	tune = 51;
	return;
}

void decideDiode(){
	if(tune == count){
		PORTD = _BV(PD3);
	}

	else if(tune == count+1){
		PORTD = _BV(PD2);
	}
	else if(tune == count-1){
		PORTD = _BV(PD4);
	}
	else if(tune <= count-1){
		PORTD = _BV(PD5);
	}
	else if(tune >= count+1){
		PORTD = _BV(PD1);
	}


}

ISR(TIMER0_OVF_vect){
	count = TCNT1;
```

```
                TCNT1 = 0;
                TCNT0 = 0;
                //clearDisp();

        //spara tcnt värder i en int
        //sätt den till noll
        //sätt andra timer till noll
        //cli();
        }

/*

        ISR(TIMER0_COMP_vect){

        }
*/



void main(void)
 {

   //set PORT A for out
   DDRD = 0xff;
        DDRC = 0b11000001;
        DDRA = 0xff;
        num = 0x02;
        TIMSK = 0x01;                                   // Enable Overflow Interrupt Enable
        TCNT1 = 0;
        TCNT0 = 0;                                                      // Initialize Counter
        TCCR0|=(1<<CS02)|(1<<CS00);       // Prescaler = FCPU/1024
        TCCR1B |= (1<<CS12)|(1<<CS11)|(1<<CS10);
        init();
  //Port C[3,2,1,0] as out put
  //DDRC|=0x0F;
  //Enable Global Interrupts
        sei();



   while(1)
        {
                if(count > 0 && count <= 24) {
                        writeBigE();
                }

                if(count >= 25 && count <= 34){
                        writeA();
                }

                if(count >= 35 && count <= 45){
                        writeD();
                }
```

```
                if(count >= 46 && count <= 57){
                        writeG();
                }

                if(count >= 58 && count <= 75){
                        writeB();
                }

                if(count >= 76){
                        writeSmallE();
                }

                decideDiode();

                 //0.26112;

                //if(count > 0){
                        //cli();
                //      PORTD = _BV(PD3);
                        //init();
                //      TCNT1 = 0;
                //      TCNT0 = 0;
                        //sei();
        //      }


        }
  return;
}
```

Källor

[i] http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf
[ii] http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
[iii] http://en.wikipedia.org/wiki/Guitar_tunings