Kakofoni 3000

En simplistisk synt som använder sig av en motor

Projekt 4,5 högskolepoäng
Bertil Lindvall, Lars-Göran Larsson
Digitala system
År 1

Gruppmedlemmar
Erik Sevelin
Ian Nguyen
Tor Lindén
Rasmus Hansson

Sammanfattning

Denna rapport beskriver projektarbetet i kursen Digitala System, EITA15. Projektet gick genom några ändringar i början, där gruppen hade olika idéer för vad projektet skulle omfatta. Målet med projektet blev att skapa och programmera ett eldrivet instrument där ljudet genereras av en motorkontrollerad siren. Instrumentet skulle kunna spela inom ett ungefärligt intervall av en oktav och kunna använda sig av olika sätt att ändra hur en ton låter. Instrumentet skulle även kunna spela in och spela upp noter inmatade av användaren. Projektet delades upp i tre moment, först att sätta upp struktur för hur arbetet skulle genomföras. Andra momentet var att rita ett kretsschema och sedan bygga den krets med givna komponenterna. Tredje momentet var att programmera kretsen att utföra tänkta funktioner. För att kontrollera hur snabbt och starkt motorn snurrade användes Pulse-width modulation. Instrumentet med siren kunde spela upp 2 noter och tog lång tid på sig att ändra frekvens, dock fungerade instrumentet väl när sirenhuvudet togs bort. Genom att låta motorhuvudet slå mot något kunde ett intervall lite större än en oktav spelas upp. Inspelning och uppspelning fungerade: tre sekunder spelas in i en array och kan sedan spelas upp. Toner kunde moduleras genom att få dem att stamma, och att låta dem spela upp stigande tonföljder.

Nyckelord

Akustik. Motor. Siren. Instrument. PWM.

Abstract

This report describes a project done in the course digital systems, EITA15. The project initially went through some changes at the beginning, where the group had devised different ideas for what the project would encompass. The project goal became creating an electricity driven instrument where sound would be generated by a motor driven siren. The instrument was supposed to be able to play notes within an approximate interval of one octave, and be able to modulate the sound of said notes. The instrument was also supposed to be able to record and play recorded notes by the user. The project was split into three parts, firstly setting up a structure for how the project work would be handled. The second part involved designing a circuit schematic and thereafter building said circuit with given components. The third part was designing the circuit to be able to perform the planned functions. Pulse-width modulation was used in order to control the velocity and the torque of the motor. In the end, using a siren head on the motor, the instrument was able to play two different notes and lagged significantly between the two. The instrument worked well when the siren head was removed. Letting the head of the motor repeatedly hit something, notes could be played in an interval a slightly bigger than an octave. Recording and playback worked, the seconds are recorded in an array and can later be played. Notes could be modulated by having them stutter, and having them play a rising sequence of frequencies.

Keywords

Acoustic. Motor. Siren. Instrument. PWM.

Innehållsförteckning

Förord

Den ursprungliga idéen med projektet var att bygga antingen ett instrument som man kunde koppla in i ett ljudchip eller i en dator och använda som MIDI-Instrument. Kort förklarat är MIDI som ett digitalt nothäfte. Tekniken kontrollerar vilka noter/knappar som matas in, samt vad som ska matas ut till programmet som sköter ljud. Detta visade sig snabbt vara för komplicerat/ta för mycket tid att implementera och vi nöjde oss istället med att använda en siren med motor, vilket gav oss åtminstone *ett* ljud att arbeta med (jämfört med de flertal i ljudchip och de oändligt många som kan kontrolleras av MIDI).

Inledning

I denna rapport beskrivs processen av att skapa en maskin som använder en siren för att göra toner. Detta gjordes då det ansågs av gruppen att vara en intressant utmaning. I praktiken blev det som en typ av elektroniskt instrument likt en synt.

Bakgrund

Under de senaste århundradet har teknologin utvecklats väldigt fort. Från 50-talet då datorer med endast några kilobytes utrymme tog upp hela rum och behövde kopplas om manuellt. Till nutiden där textfiler på moderna datorer kan vara större än det så har tekniken kommit extremt långt. Musik och instrument är inte ett undantag. Ljud består i grunden av mekaniska vågor. Dessa vågor kan skilja sig mycket och vara sinus-, fyrkantsvågor, eller komplexa sammansättningar av dem, som ljudet av ett piano.

Syfte

Projektet genomfördes för att utforska hur syntar i sin enklaste form fungerar och för att därmed göra en egen implementering av en synt. Det förväntade resultatet är att maskinen ska kunna spela upp olika toner.

Målformulering

Målet är att skapa en synt med fyra knappar som kan kontrollera alla tonhöjder i åtminstone en oktav (12 tonhöjder). Synten ska ha tre knappar som kan kontrollera olika ljudeffekter, inspelning och uppspelning. Komponenterna ska

vara kopplade till en Atmega1284. Med instrumentet ska melodier kunna vara hörbara av en liten publik (ca 20 personer i ett klassrum).

Problemformulering

1. Hur kan akustisk musik spelas utan att använda resonansen från strängar eller resonansen i tuber när luft flyter igenom dem.
2. På vilka sätt ska användaren kunna styra tonerna så att de kan spela melodier?
3. Vilka sätt kan instrumentet ändra hur tonerna låter, t.ex ljudeffekter?
4. Hur ska melodistycken spelas in?
5. Hur ska melodistycken spelas upp?

Teknisk bakgrund

Processorn Atmega1284 samt programmeringsspråket C används för att genomföra detta projekt. För att kunna styra motorn behöver spänningen till den kunna styras. För att kunna styra spänning till motorn används en H-brygga mellan motorn och spänningskällan som använder PWM (Pulse Width Modulation). PWM innebär att H-bryggan får signaler med varierande pulsbredd som ändrar dess spänning, och i sin tur motorns hastighet, och sirenens ton.

Själva tonen som skapas kommer genereras av en mekanisk siren. Sirenen i sig består av två bitar, en inre kärna (motorn) och en yttre bit (sirenhuvudet). Sirenhuvudet består av en inre roterande bit och ytterbit där luften flyter ut. Sirenen fungerar genom att rotera kärnan vilket får inre biten av sirenhuvudet att flytta runt luft. Luften trycks därefter genom hålen på ytterbiten. Eftersom den inre biten roterar så varierar area där luft kan flöda ut, t.ex leder långsammare hastigheter till en större area (lägre frekvens) och snabbare hastigheter, mindre area (högre frekvens). Det är detta som skapar ljudvågorna i sirenen.

På grund av motorns kraft är det möjligt att den inte är tillräckligt stark att kontrollera sirenhuvudet effektivt, i detta fall är det möjligt, på grund av formen på motorhuvudet, att använda sig av material som t.ex papper/plast för att skapa ljudvågor. Motorhuvudet har en rund form med jämna triangulära bitar som sticker ut och det kommer leda till jämna ljudvågor, dessa ljudvågor baseras på

frekvensen motorn slår mot materialet t.ex 440 gånger/sekund = 440 Hz vilket är standarden för modern västerländsk stämning. Ljudvågor blir svårare att kontrollera med denna metod då materialet måste hållas på plats, samt att detta kan stanna motorn med för mycket friktion.

Metod

Detta arbete genomfördes i 3 faser. Dessa faser var idéfasen, konstruktionsfasen, kodfasen. Under idéfasen bestämdes vad som skulle skapas och ett gantschema togs fram för att koordinera deadlines. I detta fall landade det på att göra en maskin som skulle kunna producera akustiskt ljud. Detta slutade med att det bestämdes att en siren skulle användas för att skapa ljudet. Därefter skrevs en kravspecifikation där det beslöts vilka funktioner som skulle finnas på maskinen och de rangordnades efter hur viktiga gruppen ansåg att de var för slutresultatet. Därefter påbörjades konstruktionsfasen. Denna fas innefattar konstruktionen samt planeringen kring den. Det inleddes med att ett kretsschema skapades för att lättare organisera konstruktionen och kunna undvika att kortsluta komponenter. Sedan konstruerades maskinen där gruppen ursprungligen försökte löda så lite som möjligt i syfte att lättare kunna demontera maskinen efter att projektet var färdigt. Detta blev dock problematiskt då det innebar att vissa kopplingar glappade, främst de kopplingar som gick till jord och spänningskälla. För att få bitarna till sirenen användes 3D-printing, bitarna hade designats efter ritningar som hämtats från internet och ändrats i storlek baserade på motorhuvudets dimensioner. De 3D-printade bitarna monterades på motorn. Detta gjordes som sista biten i konstruktionen. En siren användes för att producera ljud därför att det verkade simplare att arbeta med det jämfört med att till exempel använda små trummor eller buzzers, i övriga fall hade antingen hade antingen en komponent per ton behövt användas eller så hade det varit väldigt komplicerat att använda olika frekvenser. Med sirenen skulle det istället räcka med att endast använda en komponent då varvhastigheten i sirenen bestämmer tonläget.
Slutligen påbörjades programmeringen, den maskin som hade skapats var väldigt simpel och därmed var denna fasen relativt simpel. Först låg fokus på att skicka knappinmatningar till processorn och därefter att bestämma en period som puls kunde moduleras i (PWM), med dessa moduleringar kunde motorns hastighet kontrolleras. Fokus låg sedan på att kunna få ljudet ut att stamma samt att från en nedtryckt grundton spela upp tre repeterande noter som går upp i

frekvens och tillbaka till grundtonen. Detta gjordes genom att låta en knapp bestämma stammandet och att sätta pulsen på 0 för ett bestämt intervall T, samt därefter sätta pulsen på det användaren matade in under intervall T, och repetera tills användaren släppte knappen. För de repeterande stigande noterna bestämde en annan knapp funktionen skulle användas, samma intervall T användes mellan varje not, och tonföljden upprepades. Slutligen skapades inspelnings- och uppspelningsfunktionen. Det åstadkoms genom att ha använt en array i C som lagrade de värden som skickades till processorn under ett kort intervall. Det spelades in i tidsramar, t.ex som en film (spelas ofta in och upp i 30 ramar per sekund). Maskinen kunde sedan skicka samma array till motorn igen och på så sätt spela upp den.
Eftersom det standardiserade västerländska 12-noter-per-oktav-systemets frekvenser är exponentiella och motorns frekvenser är linjära behövde knapp-inmatningarna omvandlas till rätt hastighet för att få rätt frekvens. Om perioden till motorn var för stor blev det stammande i ljudet, därför blev omvandlingen avrundad och förenklad. Inte alla noter stämde helt med de uträknade frekvenserna och behövde ändras om, läggas till/raderas pga att pulsen bara kunde använda sig av några bitar.

Analys

För att kunna göra så att motorn kunde ändra hastighet användes en H-brygga. Detta gjordes för kunna bestämma spänningen till motorn. Själva motorn styrs med hjälp av PWM vilket möjliggjorde att motorns hastighet kunde ändras. Motorn var dock inte tillräckligt stark för att effektivt kontrollera sirenens komponenter och bara 2 toner kunde höras. Sirenen blev en för stor last på motorn vilket gjorde att det blev nästintill omöjligt att skapa en konstant ton. Ett ytterligare problem var att 3d-skrivningen inte blev helt jämn vilket ledde till att inre biten slog in i yttre biten och var mycket svår att få upp till höga hastigheter. Detta ledde till att den redan stora lasten blev ännu tyngre och det var svårt att både nå och behålla den hastighet som krävdes för få sirenen att göra en ton. För att lösa detta problem hade motorn behövt bytas till en starkare motor men detta fanns dessvärre inte tillgängligt. Motorn kunde kontrolleras med hjälp av att ta motorhuvudet mot material som papper, och små melodier kunde spelas upp, även om de inte hade exakt rätt ton. 15 olika tonhöjder, och ett intervall av 12 tonhöjder kunde spelas upp. Då motorhuvudet hölls på rätt sätt och slog mot ett material kunde dess ljud höras över ett område ca. storleken på en laborationssal på Campus Helsingborg, Lunds universitet.

C valdes för att programmera maskinen då gruppen hade tidigare erfarenheter av det samt all kunskap som kan hittas relativt lätt på nätet. Om koden hade skrivits i assembly hade det tagit längre tid, varit svårare att genomföra, samt hade det inte haft någon större effekt på vad användaren skulle spela upp och ha hört.

Resultat

Gruppen lyckades skapa en maskin som kunde spela, samt ändra varvtalet på motorn. Dessutom kunde sirenen skapa en ton men detta fungerade endast då motorn var på sin högsta hastighet och därmed lyckades den inte skapa den musikmaskin som gruppen hoppades skapa. Maskinen fungerade dock som tänkt då motorhuvudet stötte mot annat material. Inspelningsfunktionen i maskinen lyckades, den kunde spela in ungefär tre sekunder som kan justeras i programkoden.
Sirenen var problematisk då den endast kunde göra ljud när motorn var på högsta eller näst intill högsta varvtalen vilket innebar att den kunde ej få så många toner som gruppen hade hoppats på. För att lösa detta problem hade det behövt att skapa en antingen en simpel växel eller använda en annan motor då den som valt att använda helt enkelt inte hade styrka nog för att kunna driva sirenen. Detta var också problematiskt då det hade hoppats på att kunna ändra ton relativt snabbt men då lasten för stor för motorn så tog dessutom lång tid att kunna nå tongivande hastighet vilket innebar att den helt enkelt inte var praktisk som ett instrument eftersom man vill kunna ändra tonen snabbt för många låtar. Trots att gruppen inte lyckades använda en siren för att spela upp ljud så kunde motorn åtminstone använde motorljuden som ton, då den ändrade intensitet så ändrades tonen också. Alla extrafunktionerna fungerade som planerat.

Slutsats

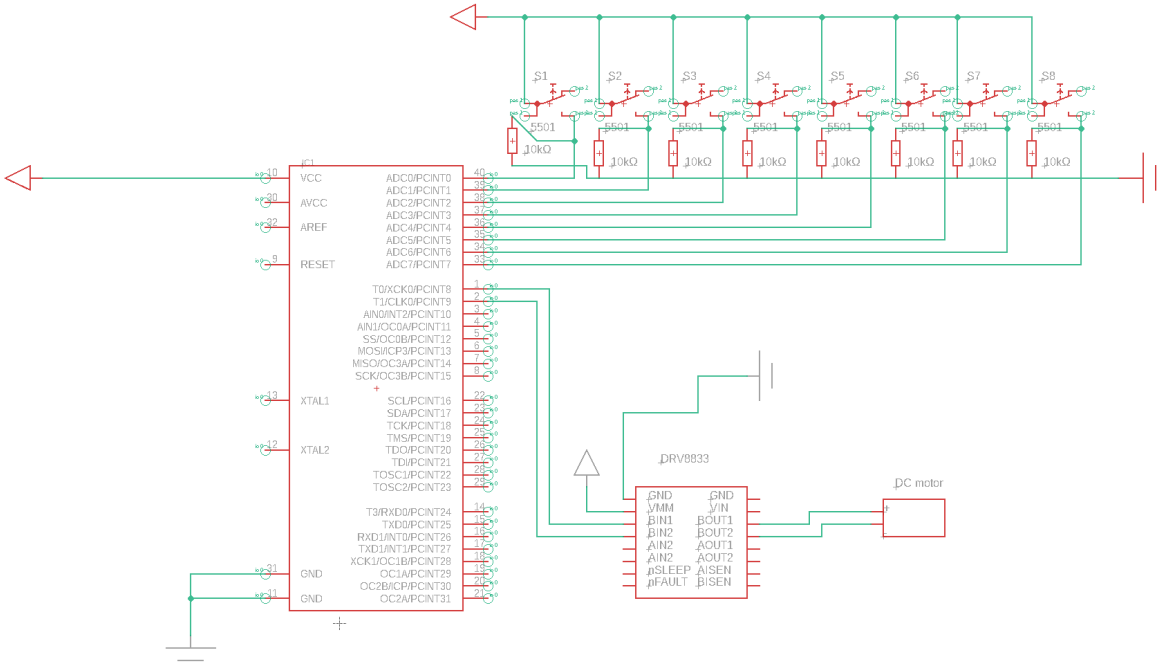
Akustisk musik kunde spelas upp genom att använda stötar mellan motorn och annat material, dock inte med sirenen. Instrumentet kan användas med fyra knappar för noter, samt tre för inspelning/uppspelning, stammande och stigande tonhöjd. Användaren kunde spela in noter under tre sekunder och spela upp det som de spelat in under den tiden. 15 toner kunde spelas upp även om de inte helt överensstämde med en oktav, de kunde även höras av en liten publik på ca 30 personer i ett klassrum.

Terminologi

PWM = pulse width modulation.
oktav: 12 noter, där not n = Grundfrekvens*(2)^(n/12). t.ex är en oktavs avstånd om A4 = 440 Hz: A3 = 220 Hz, A5 = 880 Hz där A är den första noten.

Appendix

KRETSSCHEMA:



PROGRAMKOD:

```
#define __DELAY_BACKWARD_COMPATIBLE__

#include <avr/io.h>

#include <util/delay.h>

#include <stdint.h>

#include <stdio.h>

#include <unistd.h>

#define F_CPU 16000000UL

uint16_t settings, changetime, stop;

uint16_t tid = 1;

uint16_t hasChanged = 0;

uint16_t recArray[30];
```

```
uint16_t musicButton(uint16_t buttons)
{
	buttons &= 0b00001111;

	/*
	while (buttons == 0b0000) {
		return 0;
	}
	while (buttons == 0b0001) {
		return (0b01101110 >> 3);
	}
	while (buttons == 0b0010) {
		return (0b01110101 >> 3);
	}
	while (buttons == 0b0011) {
		return (0b01111011 >> 3);
	}
	while (buttons == 0b0100) {
		return (0b10000011 >> 3);
	}
	while (buttons == 0b0101) {
		return (0b10001001 >> 3);
	}
	while (buttons == 0b0110) {
		return (0b10010011 >> 3);
	}
	while (buttons == 0b0111) {
		return (0b10011010 >> 3);
	}
	while (buttons == 0b1000) {
		return (0b10100101 >> 3);
	}
	while (buttons == 0b1001) {
		return (0b10101111 >> 3);
```

```
}
while (buttons == 0b1010) {
        return (0b10111001 >> 3);
}
while (buttons == 0b1011) {
        return (0b11000100 >> 3);
}
while (buttons == 0b1100) {
        return (0b11010000 >> 3);
}
while (buttons == 0b1101) {
        return (0b11011100 >> 3);
}
while (buttons == 0b1111) {
        return (0b11111111 >> 3);
}*/

while (buttons == 0b0000) {
        return 0;
}
while (buttons == 0b0001) {
        return (0b01011001 >> 4);
}
while (buttons == 0b0010) {
        return (0b01101110 >> 4);
}
while (buttons == 0b0011) {
        return (0b01110101 >> 4);
}
while (buttons == 0b0100) {
        return (0b10000011 >> 4);
}
while (buttons == 0b0101) {
        return (0b10100101 >> 4);
```

```
        }
        while (buttons == 0b0110) {
                return (0b11000100 >> 4);
        }
        while (buttons == 0b0111) {
                return (0b01110101 >> 3);
        }
        while (buttons == 0b1000) {
                return (0b10000011 >> 3);
        }
        while (buttons == 0b1001) {
                return (0b10100101 >> 3);
        }
        while (buttons == 0b1010) {
                return (0b11000100 >> 3);
        }
        while (buttons == 0b1011) {
                return (0b11010000 >> 3);
        }
        while (buttons == 0b1100) {
                return (0b11111111 >> 3);
        }
        while (buttons == 0b1101) {
                return (0b10010111 >> 2);
        }
        while (buttons == 0b1110) {
                return (0b10100101 >> 2);
        }
        while (buttons == 0b1111) {
                return (0b11010000 >> 2);
        }

        return buttons;
}
```

```
int main(void)
{

    DDRA &= 0x00;
       DDRB |= 0b11000000;
       timer3_init();

       set_period(0b00111111);
       //set_period(0b00001111);
    while (1)
    {
               settings = PINA;
               changetime = settings;
               changetime &= 0b01110000;
                      timechange(changetime);

               stop = settings;
               stop &= 0b01110000;
               if(stop == 0x0010){
                      stammra();
               } else if(stop == 0b00100000) {
                      arpeggio();
               } else if(stop == 0b01010000){
                      record();
               } else if(stop == 0b01100000){
                      playback();
               } else {
                      set_pulse(musicButton(PINA));
               }
    }
}

void timer3_init()
```

```
{

        TCCR3A |= 0b10000010;

        TCCR3B |= 0b00011101;


}


void set_pulse(uint16_t pulse) {

        OCR3A = pulse;

}


void set_period(uint16_t period) {

        ICR3 = period;

}


void timechange(uint16_t buttonpress){

        if(buttonpress == 0){

                hasChanged = 0;

        }

        if(buttonpress == 0b00110000 && hasChanged == 0 && tid > 1){

                tid = tid - 1;

                hasChanged = 1;

        }


        if(buttonpress == 0b01010000 && hasChanged == 0){

                tid = tid + 1;

                hasChanged = 1;

        }

}


void arpeggio() {

        set_pulse(musicButton(PINA));

        uint16_t delay = tid * 100;

        _delay_ms(delay*2);
```

```
        set_pulse(musicButton(PINA + 1));
        _delay_ms(delay*2);
        set_pulse(musicButton(PINA + 3));
        _delay_ms(delay*2);
}

void stammra()
{
        set_pulse(0);
        uint16_t delay = tid * 100;
         _delay_ms(delay);
         set_pulse(musicButton(PINA));
         _delay_ms(delay);
}

void record(){

        for(int i=0; i < sizeof(recArray); i++){

                recArray[i] = musicButton(PINA);
                set_pulse(musicButton(PINA));
                _delay_ms(100);
        }

}

void playback(){

        for(int i=0; i < sizeof(recArray); i++){
                set_pulse(recArray[i]);
                _delay_ms(100);
        }
}
```