ThereMini
Projektrapport

av:

Ludwig Havasi, Alfred Jonasson, Ashraf Al Zain, Tonny Huynh, Leo Ekstam

EITA15 - Digitala System - Vårterminen 2023
Lärare: Bertil Lindvall

Sammantfattning

I denna rapport presenteras det projektarbete som Grupp 1 genomförde i kurs EITA15 Digitala System på Lunds Tekniska Högskola under våren 2023. Arbetet gick ut på att med hjälp av en mikroprocessor och en rad andra komponenter konstruera någonting efter en egen idé, och denna grupp valde att försöka konstruera en theremin. Detta är ursprungligen ett musikinstrument som använder sig av magnetfält som störs ut och därmed genererar en signal. Thereministen kan med ena handen styra volym, och med andra handen styra frekvensen på ljudsignalen. I detta projekt ska signalen istället styras av två potentiometrar som genom digital-analog-omvandling i mikroprocessorn genererar en sinusspänning som i en högtalare kommer att generera ljuv musik.

I det stora hela utgick projektet från en av laborationerna under kursen som gick ut på att låta en potentiometer styra en pulsviddsmodulering som i sin tur påverkade hur starkt en LED-lampa lyste. Detta ansågs vara en bra startpunkt för att istället styra frekvensen på en sinusformad spänning. Det skulle visa sig vara lättare sagt än gjort.

Även om projektet innehöll en liten mängd komponenter som gick förhållandevis snabbt och lätt att koppla ihop på korrekt sätt skulle det visa sig att programmeringen var knepigare än tänkt. Gruppen utgick från laboration 2 i kursen och försökte applicera dessa kunskaper på ThereMini, men insåg snart att det var svårt att använda två potentiometrar som styrde olika saker med de metoder som användes i laborationen. Istället insåg gruppen att de behövde angripa det hela med ett annat verktyg - *avbrottsrutiner*.

Efter denna insikt lyckades gruppen till slut med att styra både amplitud och frekvens. Ljudkvaliten blev i slutändan kanske inte ljuv musik, men det blev en maskin som kunde styra amplitud och frekvens, och för den med hög fingertoppskänsla skulle det absolut gå att spela musik.

Innehåll

Förord

När man nu ges möjligheten att få skriva lite friare så måste ju den möjligheten tas. I det här projektet har vi fem som står på framsidan av denna rapport jobbat för att gå från idé till verklighet under loppet av en läsperiod. Projektet har präglats av en trög start, lite oklarheter kring vad det egentligen var vi skulle bygga, kanske lite bristande kommunikation bitvis. I slutändan lyckades vi dock sätta oss ner och bygga ihop en pryl som gör det vi ville att den skulle göra. Om än kanske inte lika *bra* som vi hade önskat, men det är av mindre vikt.

Vi har byggt ett musikinstrument! Det är nog inte helt lätt att spela på, och det låter inte så vackert, men den producerar toner som går att reglera steglöst med en potentiometer, så vi känner oss nöjda med vår insats.

Vi kan passa på att rikta ett tack till gruppen i samma klass som byggde en synth som kunde ge värdefulla tips som gjorde att vi nådde närmre vårt mål. Tack även till de labbhandledare som orkat svara på frågor när vi suttit och kliat oss i huvudet.

1 Inledning

En theremin är ett elektroniskt musikinstrument - ett av de första - där thereministen genom att röra händerna i närheten av två antenner varierar volym och frekvens på en ljudsignal. Det är ett föhållandevis dyrt instrument, men principen är ganska enkel, och bör kunna åstadkommas med enklare medel. Med hjälp av en mikroprocessor, AD/DA-omvandlare och två potentiometrar är syftet att tillverka en theremin i mycket mindre format - en ThereMini.

1.1 Syfte

Efter tre läsperioder av studier inom kursen EITA15 Digitala System kommer själva examensarbetet i kursen. Kunskaper inom C-programmering och en grundläggande förståelse för digitalteknik och kretsar ska sättas på prov genom att med hjälp av diverse komponenter snickra ihop en fungerande prototyp.

Under kursens gång har potentiometrar och AD/DA-omvandlare redan använts, och bör därför vara utmärkta komponenter att utforska vidare utan att introducera alltför mycket nya, svåra moment.

1.2 Målformulering

Först och främst är målet att på ett eller annat sätt få ljud ur en högtalare med hjälp av tidigare nämnda medel. Detta ska åstadkommas genom att programmera ett antal värden som kommer att rotera för att på så vis konstruera en sinuskurva. Amplituden och hastigheten med vilka värdena uppdateras kommer att ge upphov till olika volym och frekvens, och dessa parametrar ska kunna styras med de två potentiometrarna.

Som ett ytterligare mål, om tid finnes, ska enheten utrustas med en enkel display som visar upp aktuell frekvens (Hz) som just nu spelas ur högtalaren.

1.3 Problemformulering

De problem som är omedelbart uppenbara som de största hindren i detta projekt är först och främst konstruktionen av en sinuskurva, och att kunna styra frekvensen. Denna får *inte* ske med hjälp av så kallade *delays*, utan måste skötas på annat sätt.

Närmre beskrivning av metoden finns att läsa i avsnitt 3.2.2.

Utöver detta bekymmer var det även platsbrist bland pinnarna på mikroprocessorn. Efter att alla komponenter blivit inkopplade i kretsschemat insåg gruppen att det endast fanns en pinne kvar. Något snäv marginal som förhoppningsvis inte ska behöva bli ett problem så länge allt annat funkar.

2 Teknisk bakgrund

För att ThereMini ska fungera behövs ett antal komponenter och metoder. De viktigaste och mest intressanta för projektet kommer att presenteras i detta avsnitt. Det gäller dels hur den analoga signalen från potentiometrarna ska översättas till ett digitalt värde, och dels hur digitala värden genom digital-analog-omvandling ska bilda en sinusspänning.

2.1 Digitalisering, Sampling och ADC

För att en processor ska kunna bearbeta en analog signal måste den analoga informationen omvandlas till digital form, så kallad digitalisering. En komponent som gör detta heter AD-omvandlare (analog till digital omvandlare) eller ADC på engelska (analog to digital converter).

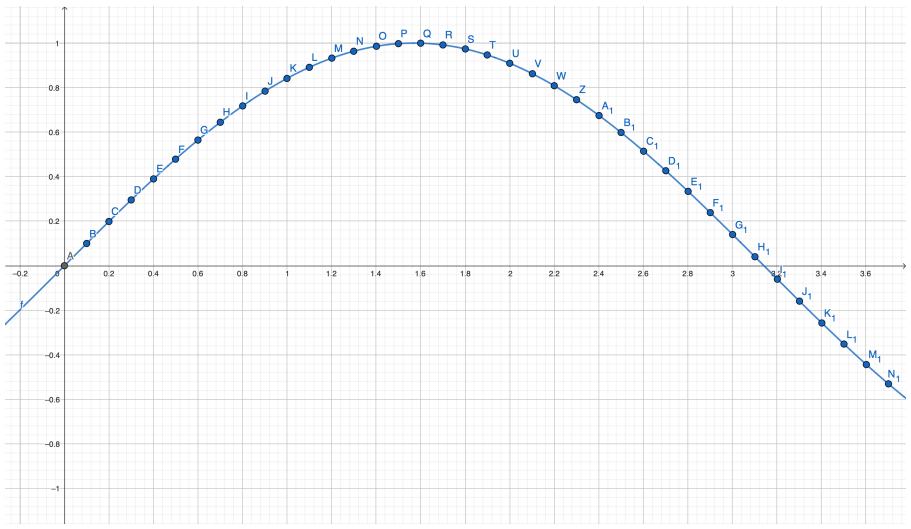
Omvandlingen från analog till digital form sker genom en process som kallas sampling (A. Andersson, Kihl 2020, 25-28). Detta innebär att en analog signals amplitud mäts upprepade gånger med ett visst tidsintervall. Varje gång en mätning görs så kvantiseras det uppmätta värdet, det vill säga värdet avrundas till närmaste förutbestämda digitala värde som kodar för en viss amplitud.

Omvandling kan ske med olika bitdjup, alltså att den analoga signalen som mäts kan kodas digitalt med olika antal bitar. Ett högre bitdjup gör att den analoga signalen återskapas i digital form mer exakt vilket minskar kvantiseringsfelet som är avvikelsen mellan den uppmätta signalen och vilket värde signalen tilldelas i digital form.

Den digitala signalen kan även göras mer autentisk ursprungssignalen genom att öka frekvensen för mätning och kvantisering av den analoga signalen. Vilken samplingsfrekvens som väljs bygger på samplingsteoremet som anger att signalen måste samplas med åtminstone dubbla frekvensen mot ursprungssignalen för att kunna återskapa den analoga signalen väl (Lai 2004, 16-17).

Ett exempel på bitdjup och samplingsfrekvens som brukar användas i verkligheten är CD (Compact Disc)-standarden som använder ett bitdjup på 16 bitar och en samplingsfrekvens på 44,1 kHz (Nationalencyklopedin u.å.).

Ett exempel av sampling av en vanlig sinusfunktion f(x) = sin(x) med frekvensen 10 Hz under en period syns i figur 1.



Figur 1: Sampling av en sinusfunktion $f(x) = \sin(x)$ under cirka en halv period med en samplingsfrekvens på 10 Hz. X-axeln visar tidsförloppet och y-axeln amplituden på den analoga sinussignalen.

Om bitdjupet i exemplet i figur 1 hade varit till exempel 6 bitar hade det inneburit att amplituden hade kunnat kodas och kvantiseras till $2^6 = 64$ olika nivåer, från -1 till 1. Det hade inneburit att varje kvantiseringsnivå hade blivit

$$\frac{2}{2^6} = 2^{-5} = 312{,}5 \cdot 10^{-4}$$

stor. Detta innebär att det maximala kvantiseringsfelet skulle bli

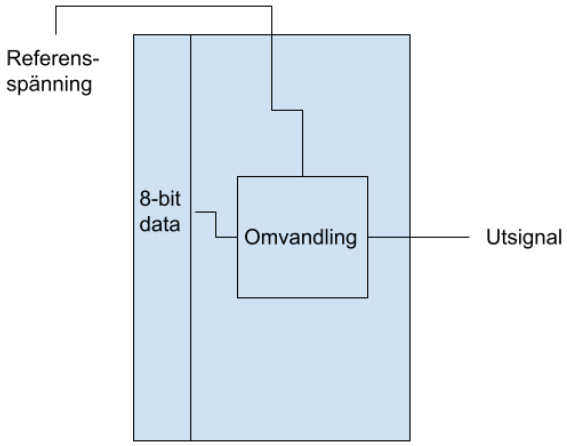
$$\frac{2^{-5}}{2} = 2^{-6} = 156{,}25 \cdot 10^{-4}.$$

Samplingsfrekvens och bitdjup bör väljas utifrån kraven som ställs på den digitaliserade signalen. Högre samplingsfrekvens och bitdjup ger en digital signal som är mer autentisk den ursprungliga analoga signalen men ställer högre krav på hårdvaran. En gräns kan också nås vid väldigt hög samplingsfrekvens och högt bitdjup eftersom vinsterna med att öka dessa värdena då börjar bli försumbara i sammanhanget.

AD-omvandlaren används i detta projekt för att få ut digitala värden från två potentiometrar, vilket beskrivs närmre i avsnitt 3.2.1.

2.2 Digital-Analog-Omvandlig (DAC)

När något ska konverteras åt andra hållet är processen något lite enklare att beskriva. Den komponent som sköter omvandligen består av en databuss med ett antal bitar (i detta projekt används en med 8 bitar), en referensspänning, och en utsignal. Databussen bestämmer alltså hur stor del av referensspänningen som den ska mata ut.



(a) En simplifierad bild av en digital-analog-konverterare (DAC). Databussen på 8 bitar bestämmer hur stor del av en referensspänning som ska skickas till utsignalen.

DIGITAL INPUT (see Note 3) MSB LSB	ANALOG OUTPUT
1 1 1 1 1 1 1 1	$-V_{ref}$ (255/256)
1 0 0 0 0 0 0 1	$-V_{ref}$ (129/256)
1 0 0 0 0 0 0 0	$-V_{ref}$ (128/256) = $-V_{ref}/2$
0 1 1 1 1 1 1 1	$-V_{ref}$ (127/256)
0 0 0 0 0 0 0 1	$-V_{ref}$ (1/256)
0 0 0 0 0 0 0 0	0

(b) Tabell från databladet för DAC TLC7524 som visar hur stor del av referensspänningen som matas ut.

Figur 2: En schematisk bild över en DAC samt den tabell som visar utspänningen i förhållande till referensspänning för den DAC som används i projektet.

Om den förenklade DAC'n i figur 2a skulle ha en referensspänning på 5V, och de åtta bitarna skulle stå på sitt maxvärde, 255, skulle utsignalen också bli 5V. Skulle databussen istället ta ett värde på 128, skulle utspänningen bli hälften av detta, 2,5V. Styrningen av denna databuss sker sedan oftast från någon form av mikroprocessor, som den ATMega1284 som används i detta projekt.

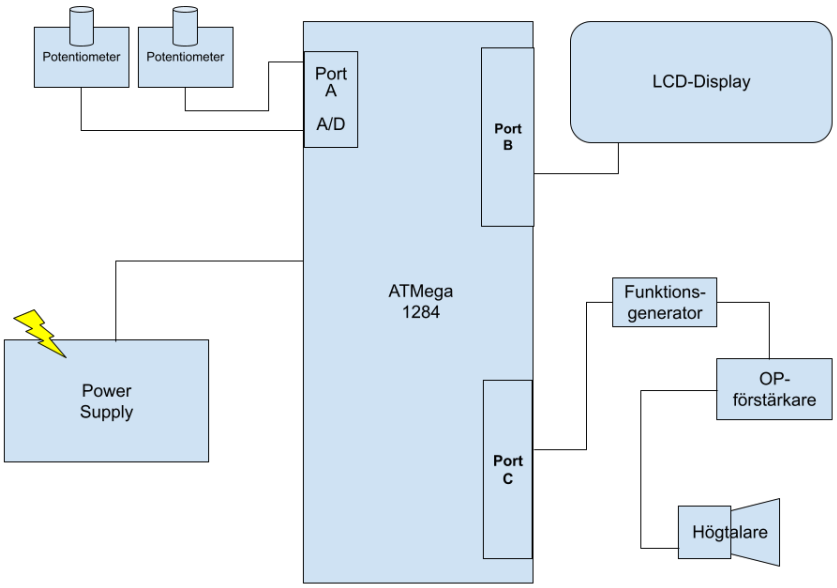
Hur DACn används för att producera en sinuskurva står beskrivet i avsnitt 3.2.2.

3 Metod

Projektet kunde delas upp i två huvudmoment - hårdvara och mjukvara. Först skulle rätt komponenter väljas, hämtas ut och kopplas ihop, och därefter skulle dessa programmeras till att göra det som krävs för att ThereMini skulle bli verklighet.

3.1 Hårdvara

Inledningsvis ritades ett principiellt block-schema över konstruktionen upp för att få en aning om vilka komponenter som skulle behövas. Figur 3 visar den version som blev grunden för projektet, även om flera komponenter byttes ut i ett senare skede.



Figur 3: En tidig version av ett blockschema för ThereMini. Här syns både en funktionsgenerator och en OP-förstärkare som båda byttes ut mot en DA-omvandlare. Även vilka portar som saker är inkopplade i ändrades.

3.1.1 Komponenter

Komponenterna som kom att behövas var följande:

- Mikroprocessor - ATMega1284
- DA-omvandlare - TLC7524
- 2x Potentiometrar (variabla resistorer)
- Display - PD2437
- Högtalare

Utöver detta givetvis även sladdar, verktyg, kraftaggregat, multimeter etc.

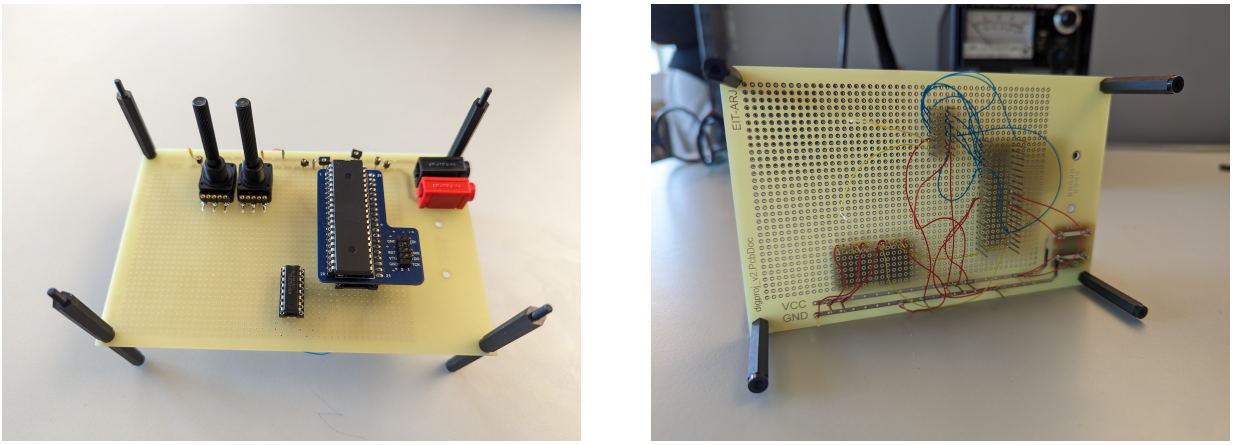
När vilka komponenter som skulle behövas var klarlagt kunde processen att rita ett mer detaljerat kretsschema ta vid. Här blev det tydligt att både LCD-display och DA-omvandlaren behövde varsin hel 8-bitars port för data, plus ett antal extra pinnar för styrning av komponenten. När kretsschemat var klart återstod endast en enda pin på ATMega1284an.

Kretsschemat finns att se som en bilaga på sida 18, avsnitt 8.1.

3.1.2 Inkoppling av komponenter

Efter att komponenterna var gruppen tillhanda kunde dessa placeras på ett kopplingsbord, kontakter lödas på, och sladdar viras runt komponenternas pinnar för att koppla dessa enligt kretsschemat. Detta gjordes med ett manuellt virningsverktyg där man stoppar in ena änden av sladden, trär över aktuell kretspinne, och vrider.

Därefter kunde ström kopplas på, och programmeringen kunde påbörjas.



(a) Enhetens ovansida

(b) Enhetens baksida

Figur 4: Dessa två bilder visar hur Theremini såg ut när alla komponenter utom LCD-displayen blivit virade med sladdar och kunde börja programmeras.

3.2 Mjukvara

Till grund för funktionen tittade gruppen på en av laborationerna (Laboration 2 i Datorteknik) som gjordes under Digitala System-kursen där potentiometrar användes tillsammans med en pulsviddsmodulering för att styra ljusstyrkan på en ljusdiod. Detta då ThereMini använder potentiometrar på samma sätt. Även sättet att använda timrar för att styra en signal var relevant för projektet.

3.2.1 Hämta värden från potentiometrarna

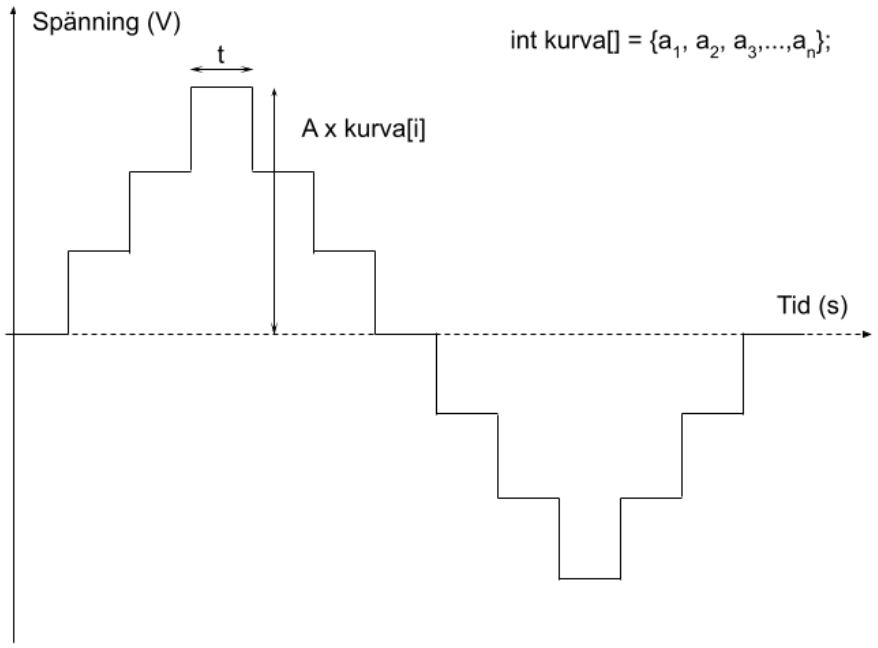
Genom användning av ATMega1284ans inbyggda AD-omvandlare (som man kan läsa mer om i avsnitt 2.1 på sidan 6) kunde ett 8-bitars värde inhämtas från potentiometrarna. Då en DA-omvandlare

behöver ett antal cykler på sig för att mäta värdet från en potentiometer, och det samtidigt inte är önskvärt att "stå och vänta" på att denna processen sker, används såkallade *interrupts*, eller avbrott. Det innebär att processen för att mäta startas, sedan fortsätter processorn med annat under tiden, och när mätningen är färdig sker ett avbrott. Då exekveras en särskild avbrottsrutin som lagrar detta värde, och startar nästa mätning. Därefter återgår processorn till vad den höll på med.

Då projektet innehåller två potentiometrar alternerar denna avbrottsmetod mellan de två potentiometrarna. För att avbrotten inte ska ske alltför ofta sattes en *prescaler* på AD:ns klocka, vilket gör att den bara jobbar i $\frac{1}{128}$ av processorns klockfrekvens.

3.2.2 Att tillverka en sinuskurva

Genom att använda en DAC (som beskrivs i avsnitt 2.2) skulle en sinuskurva tillverkas genom att låta ett antal värden i en vektor rotera med en viss hastighet, och denna hastighet skulle avgöra frekvensen på utsignalen. Denna hastighet, samt amplituden på kurvan, skulle bestämmas av potentiometrarna. I figur 5 syns en grafisk förklaring av hur denna kurva skulle tillverkas.



Figur 5: De två potentiometrarna kommer var för sig att styra de två parametrarna t samt A. Grunden i funktionen är en vektor med värden som motsvarar en sinuskurva, och storleken på dessa värden manipuleras av A, och tiden mellan det att nya värden skickas ut modifieras av t. Ju kortare t, ju snabbare gås ett varv av vektorn igenom, och frekvensen kommer öka. Ju högre A, desto högre amplitud, vilket ger högre volym.

Antag att vektorn i figur 5 innehåller 20 värden. Det betyder att en period av sinusfunktionen har gåtts igenom efter att vektorn loopats igenom ett varv. För att styra frekvensen behövs därför möjlighet att styra hur ofta programmet stegar vidare till nästa steg. Önskas en frekvens på 440Hz (tonen A) skulle vektorn behöva gås igenom 440 gånger på en sekund, vilket skulle ge tiden mellan värdena som

$$\frac{1}{20 \cdot 440} \approx 0,1ms. \tag{1}$$

Detta kan uppnås på en rad olika sätt. Ett av sätten är att helt enkelt tvinga fram en paus i processorn, en så kallad *delay*, men då detta bokstavligen sätter hela processorn på paus är detta inte önskvärt. Istället används även här en avbrottsrutin.

Genom att använda ATMega1284ans egna interna räknare kunde en rutin programmeras så att räknaren då den nått ett förutbestämt värde triggar en avbrottsrutin. Denna rutin är den som sedan skickar ut nästa värde i vektorn. Samtidigt bestäms också, beroende på de senaste värdena som är inhämtade från potentiometrarna, amplitud samt vilket nästa värde timern ska räkna till blir. Ju mindre värde, desto snabbare skickas värdena ut, och desto högre blir frekvensen på signalen.

Hade det funnits ett behov av att ha exakta frekvenser hade enkel matte kunnat räkna ut specifika värden på räknaren för att uppnå en exakt frekvens. En av egenskaperna hos en theremin är dock den helt steglösa variationen i frekvens, och därför gjordes inga sådana exakta beräkningar. Istället sattes en *prescaler* på räknaren så att den tillsammans med de 8-bitars-värdena som kom från potentiometrarna levererade ett spann av frekvenser som var inom ett område lämpligt för musik.

4 Analys

Även om de valda metoderna för att tillverka ThereMini till en början såg enkla ut skulle det visa sig att vissa saker var svårare än andra. Det gällde både val av komponenter, och hur programmeringen skulle gå till.

4.1 Val av komponenter

Inledningsvis stod gruppen i valet mellan att använda en färdig funktionsgenerator för att generera sinuskurvan, men denna valdes bort efter att den bedömdes vara något för komplicerad för ThereMinins ganska begränsade funktion. Huruvida detta var rätt val eller inte är svårt att säga, eftersom den manuella kontruktionen av sinuskurvan visade sig vara projektets knepigaste del.

Utöver det sades det från början att ThereMini skulle ha en display för att ge projektet en något lite högre nivå av komplexitet. Då projektet påbörjats blev det dock snabbt uppenbart att det skulle krävas tillräckligt med ansträngning för att att få ljud i prototypen över huvud taget. Displayen fick hamna i listan över “nice to have”, snarare än “need to have”.

4.2 Kretsschema och inkoppling av komponenter

Kretsschemat innehöll samtliga planerade komponenter för ThereMini, även om alla i slutändan inte användes. Detta gjorde att det bitvis blev lite svårläsligt. När det var dags för inkopplingen av komponenterna kunde det vara svårt att hitta vad som skulle till vilken plats.

Ett annat bekymmer var det faktum att det var väldigt lätt att räkna fel, framförallt på ATMega1284 som har 40 pinnar. Dessutom viras sladdarna på undersidan av komponenterna, så det gäller att hålla koll på att aktuell pinne är på motsatt sida än på bilderna i databladet.

Trots noggrannhet blev kopplingarna fel ett antal gånger innan allting hamnade rätt, men detta var ingenting som egentligen bromsade upp projektet nämnvärt.

4.3 Programmering

Även om den laboration som utgicks ifrån använde en potentiometer för att styra en pulsviddsmodulering skulle det visa sig knepigare att istället få den att styra frekvensen på en signal. Än svårare var att använda två olika potentiometrar för att styra två olika parametrar.

4.3.1 Användningen av avbrott

Den stora skillnaden mellan metoden som användes i laborationen och den metod som senare användes i ThereMini var användningen av avbrott. I laborationen användes istället en metod som helt enkelt väntade medan AD-omvandlingen skedde, men eftersom ThereMini behöver hämta värden från två

potentiometrar och samtidigt skicka ut värden till DA-omvandlaren blev denna metod alltför ineffektiv.

Inledningsvis användes metoden från laborationen, men det blev ganska snart klart att signalen som producerades var ojämn, och utsignalen hängde inte riktigt med när man vred på potentiometrarna. När insikten kring att avbrott kunde användas kom blev koden genast mer effektiv, och ThereMini blev (i någon mening) verklighet.

4.4 Ljudkvalitet

En riktig theremin har ett väldigt runt, behagligt ljud. ThereMini har, milt uttryckt, inte det. Detta beror främst på upplösningen på själva sinussignalen, samt vissa begränsningar i hårdvaran. De saker som hade kunnat förbättra ljudkvaliten är bland annat följande:

- Koppla en extern kristall på 16MHz till timern
- Ha fler värden i den vektor som representerade sinuskurvan
- Ha lite mer noggranna uträkningar för vilka frekvenser som önskades

Alla dessa saker hade gett en högre upplösning och större noggrannhet i hur ljudet producerades.

4.5 Potentiometrarnas känslighet

Potentiometrarna går från sina extremvärden genom att vridas ett varv. Det är en liten sträcka. Detta gjorde att det var svårt att hitta balansen mellan att kunna välja frekvens med precision, och samtidigt ha ett lagom spann av frekvenser att röra sig på. Med ett smalt spann gick det välja frekvens med precision, men man var istället begränsad av ett spann som sträckte sig cirka 100Hz. Önskades ett bredare spann, kanske 200-300Hz, blev istället känsligheten väldigt hög, och väldigt små rörelser gav stor förändring i frekvens.

Vid en eventuell ThereMini v2 skulle potentiometrarna bytas ut mot någonting med något lägre känslighet.

5 Resultat

Efter mycket krigande står slutligen en något undermålig prototyp till gruppens förfogande. Den är långt ifrån gruppen ursprungliga ambition, men kan åtminstone producera ljud.

5.1 Förenklad konstruktion

Inledningsvis skulle ThereMini ha två potentiometrar och en display. Displayen skulle visa aktuell frekvens som den just nu spelade upp. Displayen halkade ganska fort ner i prioritet eftersom det var nog med bekymmer att koppla in DA-omvandlaren och få den att fungera.

Det var nära att även en av potentiometrarna skrotades för att det var svårt att få dem att styra både frekvens och amplitud, men efter insikten att användningen av avbrottsrutiner kunde nyttjas lyckades gruppen med den ursprungliga ambitionen att styra både amplitud och frekvens med varsin potentiometer.

Amplitud-styrningen blev väldigt begränsad då tid inte fanns att finjustera värdena som användes, men principen som programmerades blev trots allt en styrning av amplituden.

5.2 Spelbarhet

I slutändan har gruppen lyckats bygga något som uppfyller den ursprungliga kravspecifikationen, vilket var att prodocuera ett ljud där amplitud och frekvens kan manipuleras med hjälp av två potentiometrar. Även om ljudkvaliten inte når upp till ens den mest okräsna av musiker skulle det fortfarande att gå, för den med fingertoppskänsla, att spela musikstycken på den.

6 Slutsats

Efter projektet har ett antal lärdomar nått gruppmedlemmarna, och i detta kapitel sammanfattas dessa kort.

6.1 Avbrott är av godo

Att använda, och framförallt förstå hur avbrott fungerar, är en grundläggande del i denna kurs. Det förklarades för oss under föreläsningarna att det är att jämföra med ifall man konstant går till dörren för att kolla om någon är där, eller vänta på att klockan ringer innan man går dit. Den förståelsen känns mer befäst och införstådd nu. All tid som man ödslar på att låta en processor göra ingenting är tid som kunde användas till något bättre. I fallet med ThereMini handlade det framförallt om inhämtningen av data från AD-omvandlaren (mer i avsnitt 3.2.1).

6.2 Trägen (och debug) vinner

Möjligheten att använda en debugger även när fysiska kretsar programmeras är ovärdeligt. Att kunna följa en variabel för att se om den får det värde som förväntas, och om inte, lista ut varför, är själva kärnan i den här sortens programmering. Det kan ta tid, och det kan kännas jobbigt, men plötsligt lossnar det, och så känns all tid som lagts ner värt det.

7 Referenser

Andersson A., Jens och Kihl, Maria (2020). *Datakommunikation och nätverk*. 2. uppl. Lund: Studentlitteratur.

Brandt, Christian, Liljencrants, Johan och Sundell, Björn (u. å.). *Nationalencyklopedin. CD:* URL: `https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/cd`. Hämtad 2023-05-12.

Lai, Edmund (2004). *Practical Digital Signal Processing For Engineers and Technicians*. Oxford: Newnes.

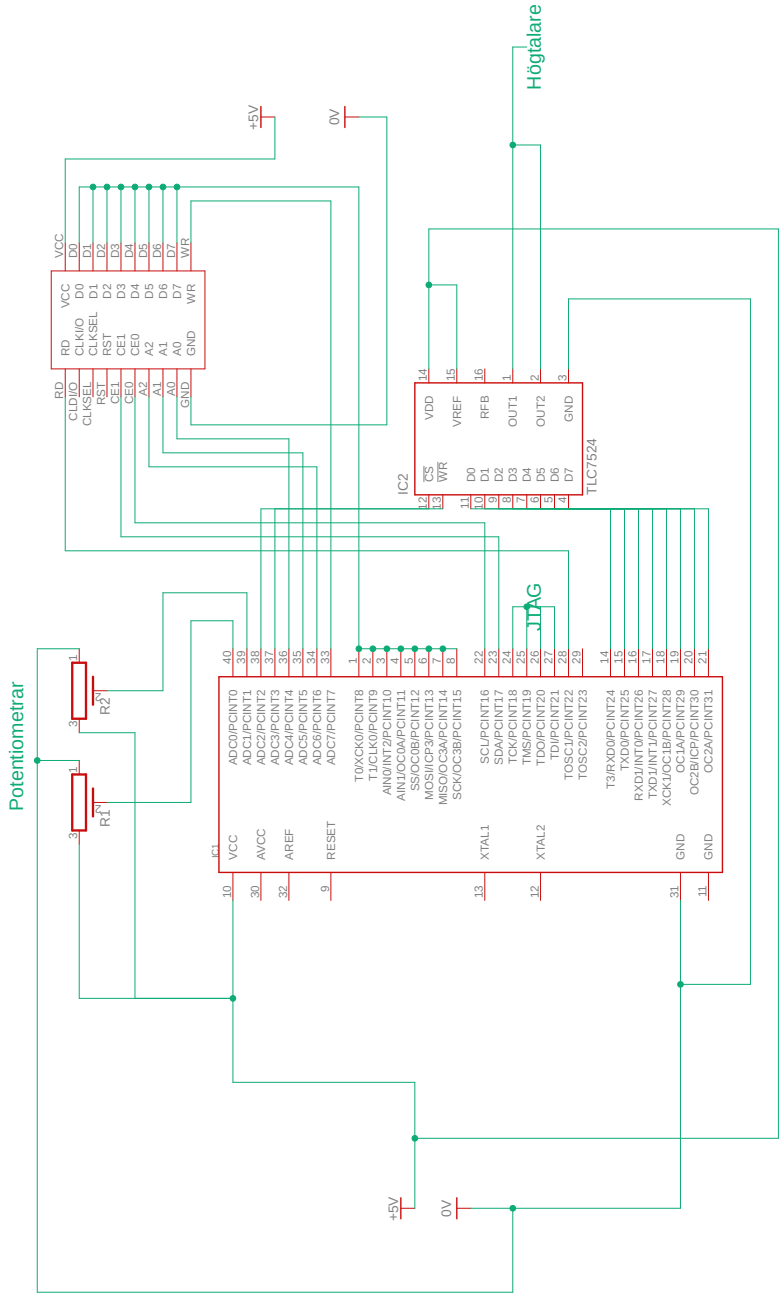
Datablad för:

- ATMega1284
- DA-omvandlare TLC7524
- Display PD2437

Labbhandledning för Laboration 2 i Datorteknikdelen av EITA15 Digitala System

8 Bilagor

8.1 Kretsschema



2023-03-30 15:58 C:\Users\ludwi\AppData\Local\Temp\Neutron\ElectronFileOutput\9192\sch-80b2ed38-b15e-4eb3-84d9-6e9987460b08\Untitled.sch (Sheet: 1/1)

Figur 6: Detaljerat kretsschema över Theremini

8.2 Källkod

```
#include <avr/io.h>
#include <stdint.h>
#include <avr/interrupt.h>
#include <stdbool.h>

//Setup of the sinus wave, the size of the array as well as an index counter
    .
int sineWaveSampleRate = 30;
int currentIndex = 0;
int sineWaveValues[30] = {62, 61, 59, 56, 52, 47, 41, 34, 28, 21, 16, 10, 6,
    3, 1, 0, 1, 3, 6, 10, 16, 21, 28, 34, 41, 47, 52, 56, 59, 61};

//Boolean value to keep track of whether frequency or volume should be read
   next by the ADC
bool readFrequencyNext = true;

//Global variables to set the frequency and the volume (amplitude) of the
   sin-wave
uint8_t frequency = 0;
uint8_t volume = 0;

//Declaration of initialization-methods
void adc_init();
void timer0_init();
void adc_read_frequency();

int main(void)
{
  //Run initialization-methods
  timer0_init();
  adc_init();

  //Enable interrupts
  sei();

  //Start the first AD-conversion.
  adc_read_frequency();

  while (1)
  {
    //Empty while-loop
  }

  return 0;
}

//ADC initialization-method
```

```
void adc_init() {
  //Enables the ADC, including enabling the interrupt and
  //setting it to work at 1/128 of the processors clock frequency
  ADCSRA = 0b10001111;
  ADMUX |= (1 << ADLAR); //Turning ADLAR to get 8-bit values from the ADC
   instead of 10-bit values
  DDRA = 0b00001100; //Sets data direction for the I/O pins on PORT A
  DDRD = 0xff; //Sets the data direction for the I/O pins on PORT D
}

//Timer initialization-method
void timer0_init() {
  TCCR0A = 0x00; //Sets the counter value to zero
  TCCR0B = 0b00000010; //Sets the prescaler to a division factor of 8
  TIMSK0 = 0b00000010; //Enables Compare Match interrupt
}

void adc_read_frequency()
{
  //Set so that the frequency is the first thing to be read
  readFrequencyNext = true;
  //Changes the ADMUX register to read from the correct potentiometer
  ADMUX |= (0 << MUX0);
  //Starts the conversion process and waits for interrupt.
  ADCSRA |= (1 << ADSC);
}

//Interrupt routine for the ADC
ISR (ADC_vect) {
  //If the frequency is to be read, set the frequency, then change so that
   the volume is read the next time
  if (readFrequencyNext) {
    frequency = ADCH;
    readFrequencyNext = false;
  }
  else
  {
    volume = ADCH/64;
    readFrequencyNext = true;
  }

  //Toggle the channel from which to read and start the next conversion
  ADMUX ^= (1<<MUX0); //Switch channel
  ADCSRA |= (1<<ADSC); //Start conversion
}

//Interrupt routine for the counter which decides the frequency of the
   output signal
```

```
ISR (TIMER0_COMPA_vect)
{
  //Set the next value to which the counter will count before triggering the
    next interrupt
  OCR0A = frequency;

  //Sets PORT D to the value in the current index of the sineWaveValues
   array
  PORTD = volume*sineWaveValues[currentIndex];

  //Go to the next value in the sin-wave-array
  currentIndex = (currentIndex + 1) % (sineWaveSampleRate);

  TCNT0 = 0x00; //Resets the timer
}
```