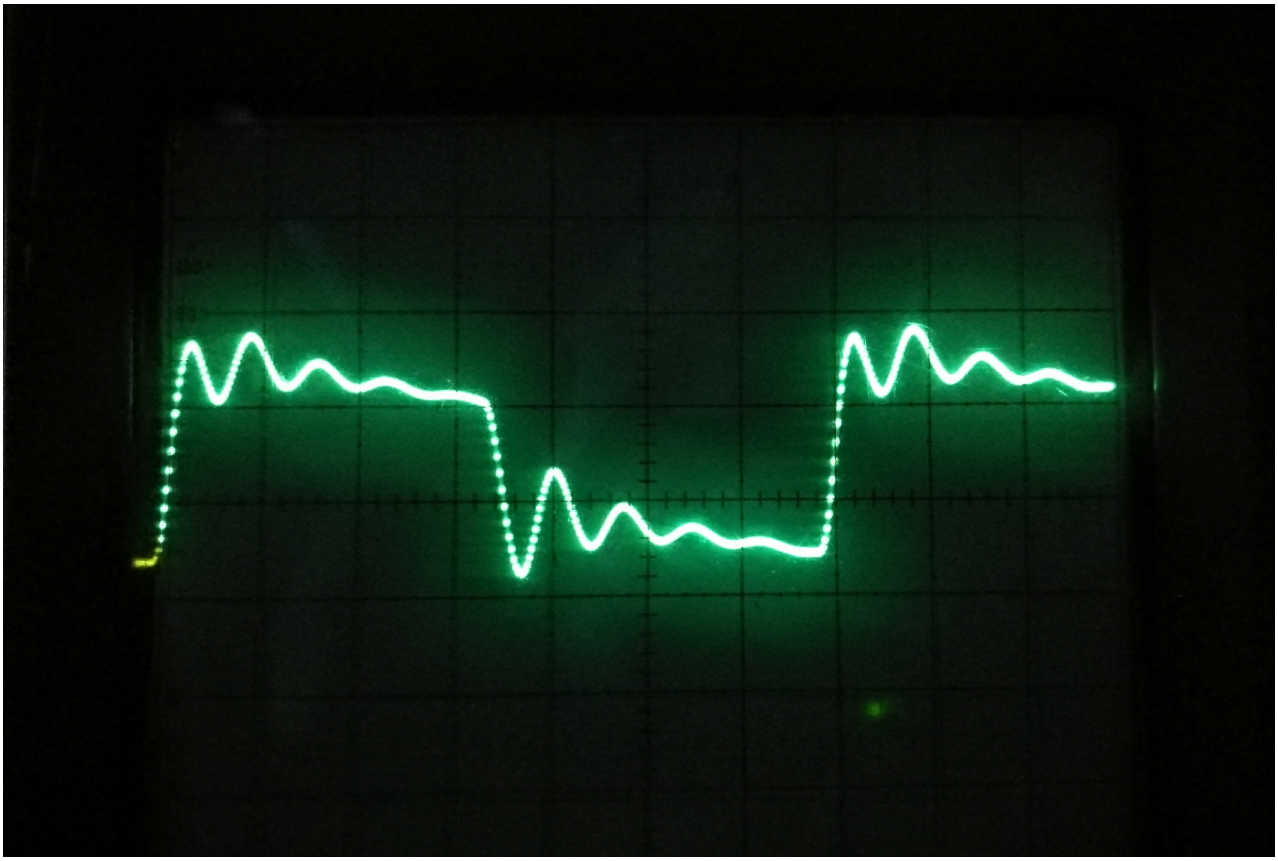
CHALMERS



Virtuell Analog Synt

Examensarbete i Datateknik

MIKAEL BERGSTRÖM
VICTOR LÓPEZ

Examinator: Peter Lundin
Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2017

EXAMENSARBETE

Virtuell Analog Synt

MIKAEL BERGSTRÖM
VICTOR LÓPEZ

Examinator: Peter Lundin
Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2017

Virtuell Analog Synt
MIKAEL BERGSTRÖM
VICTOR LÓPEZ

Examinator: Peter Lundin
Institutionen för data- och informationsteknik
Chalmers tekniska högskola
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

Omslag:
Oscilloskop som visar en ljudvåg genererad av synten.

Chalmers Reproservice
Göteborg, Sverige 2017

Virtuell Analog Synt

MIKAEL BERGSTRÖM
VICTOR LÓPEZ
Examinator: Peter Lundin
Institutionen för data- och informationsteknik, Chalmers tekniska högskola

Examensarbete

Sammanfattning

En traditionell synt använder sig av analoga komponenter för att producera ljud. En eller flera oscillatorer används för att producera en vågform som formas med hjälp av ett eller fler filter. Processen kallas subtraktiv ljudsyntes. Sedan mitten av 90-talet existerar så kallade VA-syntar (virtual analog syntesizer), som på digitalt vis emulerar den tidigare enbart analoga ljudsyntesen. Den här rapporten beskriver processen att från grunden designa och utveckla en synt med hjälp av en ARM-baserad mikrocontroller (STM32F407). Mjukvaran är skriven i C och inkluderar oscillatorer för de grundläggande vågformerna, lågpassfilter med variabel gränsfrekvens samt konturgeneratorer som kan modulera både gränsfrekvens och amplitud över tid.

Arbetet har resulterat i en mjukvaruprototyp samt design av en analogmodellerande synt, och har utförts på Chalmers Tekniska Högskola under 2016/2017 på initiativ av rapportförfattarna.

Nyckelord: Synt, DSP, Inbyggda system, Digitala filter, ARM, Musik

Abstract

Traditional synthesizers use analog components to generate sound. One or more oscillators are used to produce a waveform, which is in turn shaped by a low pass filter (and sometimes also a high pass filter). This process is called subtractive audio synthesis. Since the middle of the 90s there exists so-called virtual analog synthesizers that emulates analog audio synthesis by digital means.

This report documents the process of designing and assembling a synthesizer using an ARM-based microcontroller (STM32F407). The software is written in C and includes oscillators for the basic waveforms, a low pass filter with a variable cutoff frequency and envelope generators capable of modulating cutoff frequency and amplitude over time.

The work was conducted on Chalmers University of Technology during 2016/2017 at the initiative of the authors.

Förord

Som ett resultat av ett intresse för syntar och maskinnära programmering, och en vilja att kombinera dessa föddes idén till examensarbetet hos en av gruppens medlemmar. En fundering om vilken sorts synt det blir när man är med själv och bestämmer över hur den låter och över vilka funktioner den har. Går det att göra en synt som låter bra med en mikrokontroller? Dessa är frågor som ligger till grund för valet av ämne. Efter att författarna kommit i kontakt med varandra, visade det sig att de delade intresset för både ljudteknik och maskinnära programmering.

Arbetet är ett examensarbete omfattande 15 högskolepoäng vid Data- och informationsteknik vid Chalmers tekniska högskola.

Vi vill tacka vår handledare Dan Dolonius för hans för hans engagemang och stöd, Sven Knutsson och Roger Johansson på institutionen för deras hjälp med att använda MD407:an, våra vänner Dan Sihvonen och Sebastian Olsson, samt de hjälpsamma medlemmarna på 99musik.se.

Ordlista

Bandbredd

Beteckning för ett begränsat frekvensomfång.

Bodediagram

Bodediagrammet illustrerar ett systems frekvensberoende egenskaper. Det består av två delar, där den övre grafen beskriver amplituden i dB och den nedre beskriver fasförskjutningen. Frekvens representeras av en logaritmisk skala, vilket tydliggör brytpunkter och lutning. Bodediagrammet utvecklades 1938 av amerikanen Hendrik Wade Bode.

DMA

Direct Memory Access. Array i minnet som används för att snabbt komma åt olika data.

Filter

Används i en synt för att filtrera bort de frekvenser användaren inte vill ha i ljudet. Lågpassfilter finns på i stort sett alla syntar medan högpassfilter är mindre vanligt.

Klaviatur

Tangentuppsättning för musikinstrument. Återfinns på exempelvis piano, dragspel, keyboard och syntar. Dessa instrument kallas klaviaturinstrument.

Konturgenerator

Används för att styra olika parametrar över tid. Den engelska termen envelope är mer använd.

LFO

Lågfrekvent oscillator som används för att styra värden. Oftast gränsfrekvens eller stämning.

MD407

Mikrokontrollern synten är skriven för. Den är utvecklad av Chalmers baserad på STM32F407 och har en klockfrekvens på 168MHz.

MIDI

Står för Musical Instrument Digital Interface. MIDI är en industri-standard som föreslogs av synt-pionjären Dave Smith år 1981. Används till allt från att signalera vilka toner synten ska spela till ändring av parametrar.

Vikning

Fenomen inom digital signalbehandling vilket uppstår vid sampling av frekvenser ovan Nyquist-frekvensen. Nyquist-frekvensen är halva samplingsfrekvensen. Audiell vikning skapar missljud i Nyquist-frekvensen, medan vikning i film kan visa sig som att en propeller snurrar långsamt. Namnet Nyquist-frekvensen kommer från Nyquist-teoremet, vilket är uppkallat efter den svensk-amerikanske elektroingenjören Harry Nyquist.

Monofoni/Polyfoni

En monofonisk synt kan bara spela en ton i taget. Signalen kan dock bestå av en kombination av flera oscillatorer i olika frekvenser. En polyfonisk synt kan spela flera toner i taget.

Oscillator

Komponenten i synten som genererar ljud.

Resonans

Förstärkning av signalamplitud inom ett visst begränsat frekvensområde

Samplingsfrekvens

Vid vilken frekvens sampel tas vid sampling.

Synt/Synthesizer

Instrument som på analogt eller digitalt vis har förmågan att syntetisera ett stort antal ljud och klanger. Ej att förväxla med ett keyboard, vars ljud oftast baseras på inspelningar av verkliga instrument, och inte har samma förmåga att syntentisera ljud.

Ton

Västerländsk musik är uppbygd av den kromatiska skalan. Skalan innefattar tolv toner som alla ligger ett halvt tonsteg från varandra. Nästkommande tonsteg i skalan fås genom att multiplicera frekvensen med $2^{1/12}$, detta medför att en oktav en dubblering av frekvensen.

Innehåll

1 Inledning

1.1 Bakgrund

När de första syntarna dök upp på sextiotalet var de konstruerade av analoga komponenter och använde sig av en subtraktiv ljudsyntes, det vill säga att en eller flera oscillatorer skapar en vågform som sedan passerar genom ett lågpass- och ett eventuell högpassfilter, för att sist passera en förstärkarkrets. Syntarna har även en eller flera konturgeneratorer för att forma ljudets karaktär.

Den subtraktiva ljudsyntesen var väldigt populär fram till slutet av åttiotalet, då de hel-digitala syntarna med andra former av ljudsyntes dök upp. År 1995 tillverkade svenska Clavia den första virtual analog-synten, det vill säga en synt som på digitalt vis emulerar en subtraktiv ljudsyntes. VA-syntar blev väldigt populära och var standard fram till för några år sedan; då de analoga syntarna återigen blivit väldigt populära. De tidiga VA-syntarna hade många problem med svag hårdvara och andra barnsjukdomar. De var utrusta med lågupplösta AD-omvandlare vilket medförde relativt få olika inställningsmöjligheter på de olika parametrarna. Röster föll även bort ju mer avancerade ljud som synten generar, vikning vid högre frekvenser var inte alls ovanligt. Hårdvara som då var omöjlig att få i en synt kan vi nu använda för att bygga en egen kompetent VA-synt.

Ett projekt av den här karaktären kan vara mycket lärorikt inom såväl programmering som ljudteknik samt sannolikt en bra merit för framtida arbetssökande. Det är dessutom ett tillfälle att få arbeta med en kombination av hårdvara och mjukvara vilket tilltalar projektdeltagarna.

1.2 Syfte

Båda projektdeltagarna har sedan tidigare ett intresse för ljudteknik och maskinnära programmering. Dessutom är en av projektdeltagarna hobbymusiker med stort intresse för syntar, och intresserad av att komma in i branschen.

1.3 Mål

Inom ramen för projektet ska det utvecklas en analogmodellerande synt. Mjukvaran såväl som hårdvaran ska utvecklas till en fungerande prototyp. Det viktigaste i projektet är att få till en mjukvara som klarar av att generera ljud, och som har många av de funktioner som återfinns i en klassisk synt. Exempelvis oscillatorer, filter, och så kallade konturgeneratorer. Funktioner beskrivs mer utförligt i den tekniska bakgrunden i rapporten. Målet är att konstruera en prototyp till en synt, med nödvändig mjukvara och hårdvara. Fokus kommer att ligga på utveckling av mjukvaran.

1.4 Frågeställningar

Projektet ska försöka utreda och besvara följande frågeställningar:

- Hur en mjukvarubaserad oscillator och konturgenerator bäst implementeras?
- Hur ett mjukvarubaserat filter implementeras.
- Vilket sorts filter som lämpar sig till en synt.
- Huruvida de analoga komponenterna bör emuleras på sådant vis att även dess imperfekter återskapas?
- Krav på och val av lämpligt datorkort för en VA-synt.
- Hur implementeras MIDI-protokollet.
- Hur användarstyrda variabler styrs på bästa vis.

1.5 Avgränsningar

Vid ett eventuellt klaviatur kommer ett färdigt klaviatur att införskaffas, antingen används ett klaviatur från ett äldre MIDI-keyboard, eller köper ett. Moderna syntar har ofta inbyggda ljudeffekter, det kommer inte inte denna synt vara utrustad med.

2 Metod

Utvecklingsarbetet har varit uppdelat i följande olika delmoment:

- Utveckling av en kravspecifikation.
- Val av hårdvara, programspråk samt utvecklingsmiljö.
- Utveckling av mjukvara samt kontinuerlig testning. Utvecklingen delades upp i dessa moment:
  - Utveckling av oscillator.
  - Utveckling av filter.
  - Utveckling av konturgenerator
  - Drivrutiner till de periferienheter de olika komponenterna kräver.
- Sammansättning av hårdvaran.

Utvecklingen av den tänkta prototypen måste göras genom listning av de önskade egenskaper synten bör utrustas med. Dessa egenskaper gås sedan igenom för att fastställa vilka som är genomförbara, och vilken hårdvara dessa egenskaper kräver. Med hjälp av denna information beslutades val av programspråk samt mikrokontroller.

Mjukvarans utveckling skedde till största del med hjälp av parprogrammering. De olika komponenterna utvecklades var för sig gemensamt. Parprogrammeringen användes för att på bästa vis kunna ta tillvara på den gemensamma kunskap paret besitter. Detta tillvägagångssätt möjliggjorde även testning på ett smidigt sätt samt snabbare möjligheter till lösningar av problem. Vid behov av eventuella periferienheter skrevs drivrutiner till dessa allt eftersom behovet uppstod. Vid utvecklingen av komponenterna användes den helanaloga synten Korg MS-20 Mini för att notera hur dess komponenter beter sig med olika inställningar. För att tolka beteendet användes ett oscilloskop.

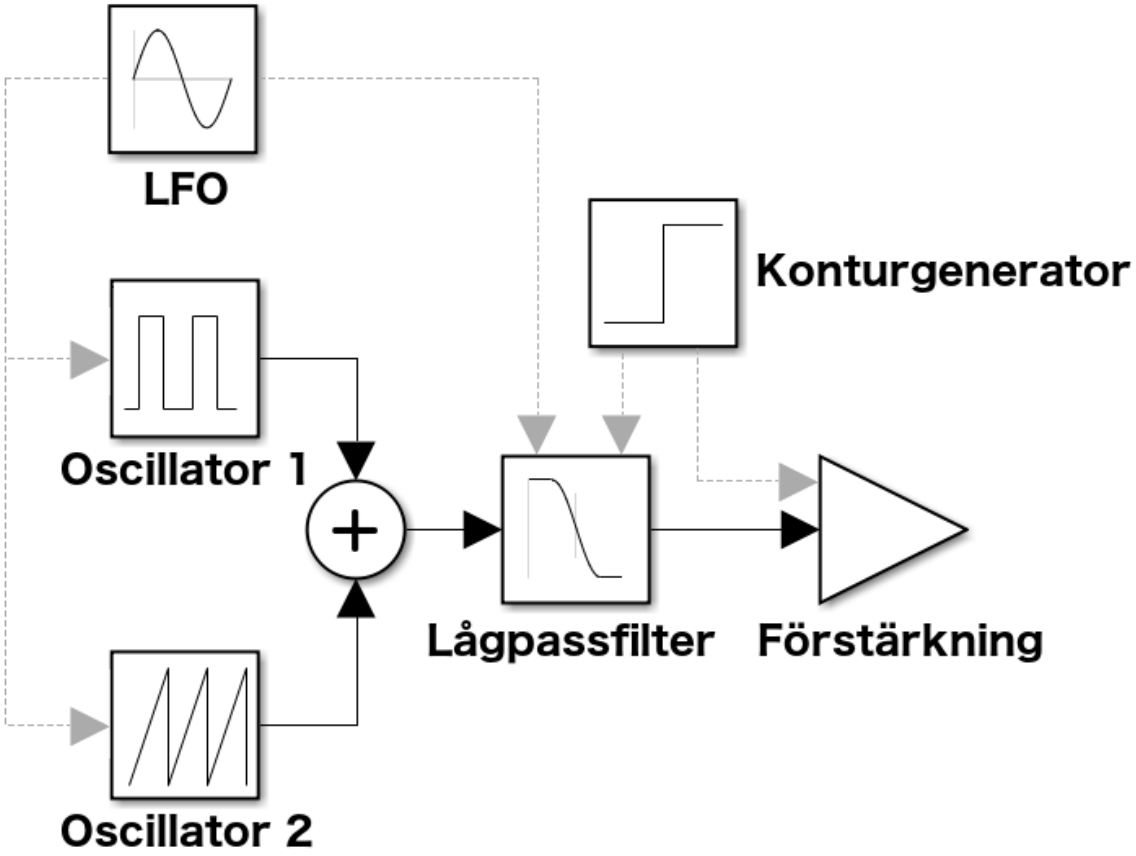
Testerna utfördes dels audiellt med hjälp av högtalare för att lyssna på det utmatande ljudet och subjektivt bedöma om det låter bra. Samt visuellt med hjälp av oscilloskop objektivt bedömma huruvida komponenten som testas uppför sig korrekt.

Mjukvaran för hårdvarustyrning utvecklades parallelt med respektive mjukvarubaserade komponenter. Lödlösa prototypkort användes för verifiering av de elektriska kretsar. Oscilloskop användes för att verifiera att signalen från de analoga reglagen lästes av på ett tillfredsställande vis. Den slutgiltiga monteringen hanns inte med i projektet.

3 Teknisk bakgrund

3.1 Signalkedjan

Signalkedjan på en synt representerar de olika steg som signalen passerar innan den matas ut. Dessa inkluderar oscillatorer, filter samt konturgeneratorer. De olika beståndsdelarna beskrivs i följande stycken.



Figur 3.1: Exempel på en vanlig signalkedja hos en synt med subtraktiv ljudsyntes

3.2 Oscillator

Oscillatorn är den första komponenten i syntens signalkedja och används för att generera en ton i form av en vågform med en specifik grundfrekvens. Vilken frekvens oscillatorn har styrs av klaviaturet. Vågformen används för att ändra karaktär på ljudet. Oscillatorer paras oftast ihop för att tillsammans kunna generera en sammanslagen signal, detta görs för att kunna generera en vågform med helt annat frekvensinnehåll än vad signalen från en enkel oscillator kan generera.

Oscillatorn i en traditionell synt har vanligtvis förmågan att generera ett antal olika vågformer. Sinus-, fyrkant-, triangel- eller sågtandsvågform hör till de mer vanliga vågformerna. En enkel sinusvåg består enbart av en grundfrekvens medan övriga består av en grundfrekvens samt ett antal övertoner som ger vågformen sin slutgiltiga form. En perfekt fyrkant-, triangel- eller sågtandsvåg existerar enbart i teorin eftersom den innehåller ett oändligt antal frekvenser.

Fyrkantsvågor går ofta att pulsbreddsmodulera, det vill säga att man varierar andelen av perioden då signalen då den är hög kontra då den är låg (se figur A.1,A.2, och A.3 i figurbilaga). Ett vanligt användningsområde för detta är att låta lågfrekvensoscillatorn (LFO) påverka detta för att göra ett ljud som passar bra till bas. Triangelvågen går även den ibland att justera fritt, man låter då användaren justera brytpunken, med vilket menas den punkt där triangeln övergår från en ökning till en minskning. Med hjälp av den inställning kan triangelvågen formas till allt från sågtand, till en inverterad sågtand. Alla med dessa olika vågformer har sen egna ljudbild. På syntar där triangelvågens brytpunkt inte går att justera, finns oftast sågformsvåg och triangelvåg med som separata vågformer istället.

3.3 Filter

Ett viktigt redskap för att kunna forma ljudet är användningen av ett eller flera filter. Filtret är hela fundamentet för den subtraktiva ljudsyntesen - filtret tar bort de frekvenser användaren inte vill ha i ljudet, och producerar en signal med ett reducerat frekvensomfång. Gränsfrekvensen är den frekvens vid vilken filtret börjar verka.

Lågpassfilter används för att släppa igenom de frekvenser som är lägre än gränsfrekvensen. Det återfinns på alla syntar med subtraktiv ljudsyntes.[1] Ett annat filter som ofta återfinns hos syntar är högpassfilter, som istället släpper igenom de frekvenser ovan gränsfrekvensen. Ett låg- och ett högpassfilter kan kombineras för att skapa ett så kallat bandpassfilter vilket antingen kan släppa igenom eller blockera frekvenserna innanför den centrala delen.

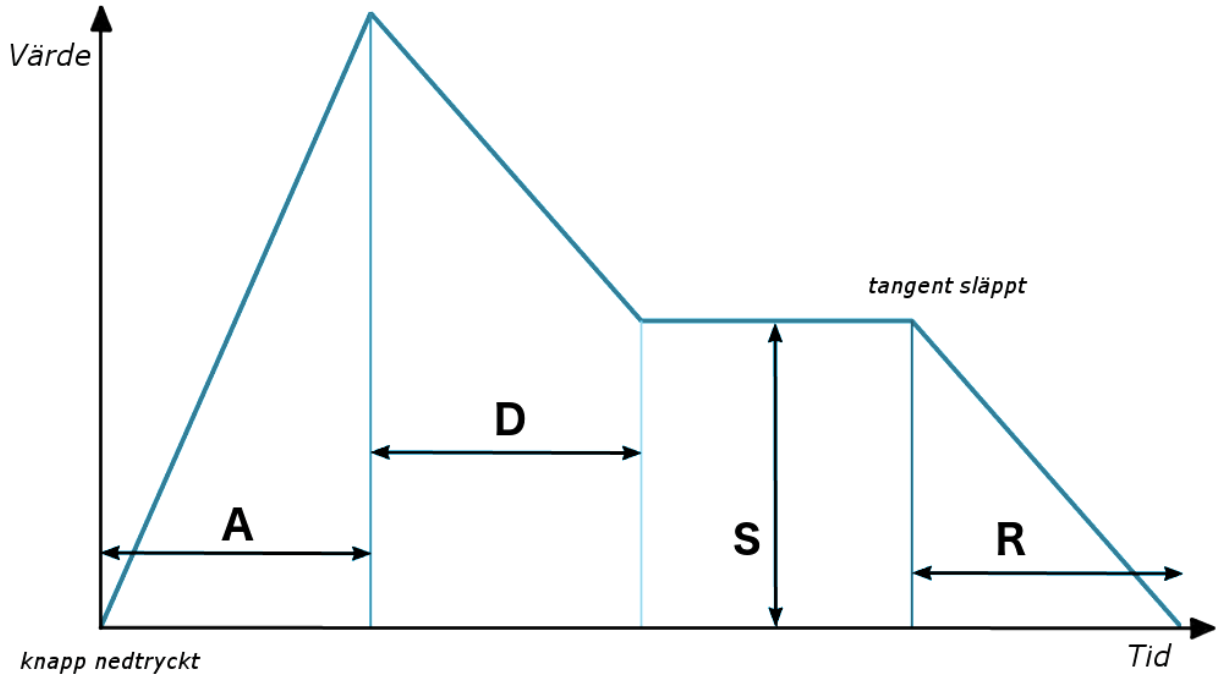
Dessa filter kan ha olika effektiv dämpning av frekvenserna utanför bandbredden, en egenskap som i musiksammanhang ofta mäts i dB per oktav. Ett högre värde indikerar en kraftigare dämpning. På analoga syntar är det vanligast med ett filter på 12 eller 24 dB/oktav, men ett digitalt filter kan dämpa signalen mycket mer effektivt än så. Ett filter på en synt bör även kunna betona en frekvens, vilket görs genom att att förstärka frekvensen vid filtrets gränsvärde, det kallas för det mesta resonans (se figur A.11 i appendix).

3.4 Konturgenerator

På en analog synt är konturgeneratorn är oftast kopplad till förstärkaren och används för att forma ljudets karaktär genom att styra förstärkningen. Konturgeneratorn kan även användas för att styra andra parametrar, exempelvis gränsfrekvensen för ett filter. Förmågan att styra ett parametervärde kan användas för att forma ett stort antal olika ljud. De effekter som styrs via en konturgenerator kan liknas vid det som ger olika instrument dess karaktär. En fiol som spelas med långa stråkdrag skulle återskapas på en synt med en lång attack, medan ett hamrande piano skulle få med en kort attack och lång release. Med hjälp av konturgeneratorn kan man även skapa helt annorlunda ljud som vanligtvis kanske inte förknippas med syntar, som virvel- och bastrumma. Utan en konturgenerator skulle syntar ha ett väldigt onaturligt och omusikaliskt ljud.

Konturgeneratorn består oftast av fyra parametrar (ADSR):

- Attack - styr hastigheten på parameterökningen.
- Decay - styr hur snabbt parametern sänks till jämn nivå.
- Sustain - bestämmer vilken nivå decay sänks till och hålls medan tangenten hålls.
- Release - tiden det tar för parametern att sänkas ned från och med att tangenten släpps.



Figur 3.2: Exempel på en konturgenerator då 'envelope amount' är satt till positivt värde

3.5 Röster

På en en synt med subtraktiv ljudsyntes består en röst av hela signalkedjan på synten. Det vill säga oscillator, filter, och utsignal. Varje röst har dessutom en eller flera konturgeneratorer för att ge varje röst en egen klang, på samma vis som på ett piano.

En monofonisk synt kännetecknas av att den har en röst, vilket innebär att synten kan spela en tangent i taget, ljudet går sedan genom filtret, och sedan ut till förstärkaren. På en polyfonisk synt är signalkedjan multiplicerad med hur många röster den har. En synt med sex rösters polyfoni och två oscillatorer per röst har således totalt sex stycken signalkedjor som kan spelas oberoende av varandra.

Antalet röster begränsar följaktligen hur många noter synten kan spela samtidigt. Även om synten då har sex stycken signalkedjor, så delar dessa samma inställningar. Figur 3.1 på sida 3 visar ett exempel på hur en vanlig signalkedja på en monofonisk synt med subraktiv ljudsyntes ser ut. Konturgeneratorn (envelope) kan styra filtervaribler och förstärkaren, medan LFOn kan styra filter och olika oscillatorvariabler. Det är på det här viset vi har designat våran synt.

3.6 MIDI

För att styra en synt på annat vis än med klaviatur utarbetades en industristandard under åttiotalet i form av ett kommunikationsprotokoll. Med det nya protokollet garanterades att syntar från olika tillverkare kunde kommunicera med varandra. Med hjälp av MIDI kan en synt med dator, trummaskiner, MIDI-keyboard, eller andra syntar styra bland annat vilken ton synten ska spela i. På många vis fungerar det som en digital motsvarighet till det självspelande pianot. MIDI skickar alltså inget ljud, utan bara information till instrumentet. De flesta syntar byggda efter 1984 har möjlighet att ta till sig information genom MIDI. Antingen genom en dedikerad MIDI-port eller genom en USB-port - något som blivit vanligare under senare år.

4 Genomförande

Kravspecifikationen togs fram genom att studera flera olika syntar och dess egenskaper. Analoga syntar som Korg MS-20, Moog Minitaur, samt DSI Prophet studerades samt analogmodelerande syntar som Clavia Nord Lead 2, och Meeblip Anode. Begränsande designval som återfinns hos de syntar vi kollade på uteslöts för att möjliggöra många olika ljudkombinationer.

4.1 Kravspecifikation

- Minst två oscillatorer per röst.
- Minst en röst.
- En lågfrekvent oscillator (LFO) för variabelmodulation.
- Möjlighet att stämma ur oscillatorerna från grundfrekvensen.
- Möjlighet att kombinera olika oscillatorinställningar helt separat från varandra.
- Oscillator med förmåga att generera sinus-, fyrkant-, och triangelvåg.
- Modulering av triangelvågens brytpunkt samt pulsbreddsmodulering av fyrkantsvågen.
- Oscillatorn ska kunna generera en frekvens med hög precision.
- Styrning av oscillator med hjälp av MIDI.
- Lågpassfilter med resonans och valbar gränsfrekvens.
- Syntens olika parametrar ska gå att styra på enkelt vis.
- Förmågan att generera toner i ett brett frekvensomfång.
- Konturgenerator som påverkar utsignalens styrka.
- Konturgenerator som påverkar filtrets gränsfrekvens.

4.2 Hårdvara

Synten drivs av Chalmers egenutvecklade MD407 med en klockfrekvens på 168 MHz. Tidigt i projektet diskuterades även Arduinos mikrokontroller, men då MD407 både är snabbare och är mer lättillgänglig via Chalmers valdes denna. Även att MD407 körs med vanlig C-kod - till skillnad från Arduino var till MD407:s fördel vid valet. MD407 bygger på mikrokontrollern STM32F407 vilken har en modern ARM-processor med flyttalsenhet. Mikrokontrollern har även inbyggda digital-analog-omvandlare (D/A-omvandlare), samt analog-digital-omvandlare(A/D-omvandlare). Dessa komponenter är en förutsättning för att synten ska kunna mata ut ljud och läsa av analoga reglage. De analoga reglagen används för att styra syntens olika användarstyrda variabler. Både A/D-omvandlaren och D/A-omvandlaren har en upplösning på 12 bitar. De analoga potentiometrarna som är kopplade till A/D-omvandlarna går att vrida 270°. Detta betyder att det är drygt 15 samplingspunkter för varje grad vilket medför att variabeländring upplevs som sömlöst.

4.3 Programmeringsspråk och utvecklingsmiljö

Programspråket C föreföll som ett naturligt val då programbiblioteken för mikrokontrollern är skrivna i språket. C passar även till vårt projekt då det är snabbt och maskinnära. Andra programspråk förkastades då de antingen ansågs vara besvärliga (Assembler), eller långsamma (Java). Mjukvaran skrevs med hjälp av utvecklingsmiljön CodeLite. Chalmers tillhandahåller instickskprogram till CodeLite för kommunikation med plattformen vilket gjorde utvecklingsmiljön till ett naturligt val. Vid utvecklingen av mjukvaran har bibliotek från microkontroller-tillverkaren (STMicroelectronics) samt hjälpbibliotek skrivna av Sven Knutsson på Data- & informationsteknik på Chalmers använts.

4.4 Mjukvarans struktur

Programmets grundstruktur är uppbyggt på följande vis: först initieras nödvändiga periferienheter och processorfunktioner (systemklocka, avbrott, GPIO-pinnar, DMA, samt D/A-omvandlare och A/D-omvandlare). För att initiera dessa komponenter används fördefinierade datastrukturer från datorkortets programbibliotek. Datastrukturen fylls med värden som styr hur enheten ska fungera.

Därefter träder programmet in i en slinga som växelvis matar ut en nivå till D/A-omvandlaren och växelvis beräknar nivån baserat på vilken signal vi vill mata ut. Skrivningarna till D/A-omvandlaren är synkroniserade med hjälp av systemklockan till 28 kHz.

4.4.1 Systemklockans konfigurering

Valet av utmatningsfrekvens bör vara tillräckligt hög för att synten ska kunna reproducera en signal som kan innehålla alla frekvenser inom det hörbara spektrat. Enligt Nyquistteoremet krävs en samplingsfrekvens som är mer än dubbelt så stor som de frekvenser man vill återge för att motverka fel i samplingen.[2] Det hörbara spektrat rymmer frekvenser mellan 20 Hz och 20 kHz[3], vid en samplingsfrekvens på 44,1 kHz skulle således synten inte ha några problem med återgivning av frekvenser i det spektrat. Samma resonemang fördes vid CD-skivans val av standardfrekvens.[4]

Tidsperioden mellan samplingarna är knappt 22,7 µs och under denna period genomförs de beräkningar som ligger till grund för utmatningsnivån från D/A-omvandlaren. De inkluderar oscillatorer, filter och konturgeneratorer. Vid en utmatningsfrekvens på 44,1 kHz märktes att synten inte hann med att beräkna hela signalkedjan. Detta yttrade sig i att synten inte skrev något värde till D/A-omvandlaren vilket resulterade i en tyst signal. För att motverka problemet valdes att sänka utmatningsfrekvensen Andra möjliga lösningar inkluderade att reducera funktionalitet till den grad att synten inte längre har en subtraktiv ljudsyntes. Dessa lösningar innefattade bland annat att inaktivera filtret, eller en oscillator. Att inaktivera filtret var aldrig aktuellt, då filtret är en kritisk komponent för den subtraktiva ljudsyntesen. Att inaktivera en oscillator skulle möjligtvis kunna lösa problemet, men det avfärdades då kravet att på två separata oscillatorer prioriterades högre än kravet på ett större frekvensomfång.

4.4.2 Huvudslinga

Tidigt i projektet valdes en programstruktur som skulle möjliggöra polyfoni. Funktionen fick avfärdas på grund av prestandabegränsningar men strukturen möjliggör eventuell vidareutveckling till polyfoni, vid användning av en kraftigare processor. Möjligheten till polyfoni implementerades genom att låta varje röst representeras av en datastruktur innehållandes den information som krävs för att kunna generera signalen. Då en röst instansieras läggs nämnda datastruktur till en lista som innehåller samtliga aktiva röster, tillsammans med en räknare som representerar antalet röster. Då en röst tystnar raderas rösten ur fältet och räknaren minskar med ett. I programmets huvudslinga anropas en funktion som summerar rösternas utsignal. Denna funktion itererar genom listan av röster och beräknar deras momentana signalvärde baserat på variabler i röstens datastruktur.

Datastrukturen för en röst benämns *voice_parameters* och innehåller följande variabler:

- *key* - grundton
- *amplitude* - signalens amplitud i relation till konturgeneratorn
- *envelope_modifier* - ändring av gränsfrekvensen i relation till konturgeneratorn
- *amplitude_stage* - konturgeneratorns stadie i avseende till amplitud
- *filter_stage* - konturgeneratorns stadie i avseende till filtrets gränsfrekvens
- *filter_cache* - minne av de senaste samplen, för användning av filtret
- *filter_coefficients* - datastruktur innehållandes koefficienter för filtret
- *osc1* och *osc2* - datastrukturer innehållandes oscillatorvariabler

Vad dessa variabler gör mer specifikt kommer beskrivas mer ingående i implementationen av de specifika delarna. Slutligen gör en omvandling från flyttal till 12-bitars positivt heltal för utmatning till D/A-omvandlaren. Vid nästa systemavbrott skrivs detta värde till D/A-omvandlaren och huvudslingan återupprepas.

4.5 Användargränssnitt

Användargränssnittet är realiserat med hjälp av en uppsättning brytare och vridreglage (potentiometrar) för styrning samt lysdioder för återkoppling till användaren.

4.5.1 Styrning av periferienheter

Vid knapptryck för att ändra inställning på synten, och vid anslag på MIDI-klaviatur genereras ett avbrott. Beroende på vilken brytare som är nedtryckt anropas olika avbrottsrutiner. På MD407 finns det sex stycken olika avbrottshanterare med olika prioriteringsnivåer, vilket betyder att alla brytare kan ha en enskild metod. För att synten ska kännas snabb och responsiv att spela på har MIDI-anslag en helt egen avbrottshanterare, med högsta prioritet efter systemavbrottet. MIDI-avbrottet reagerar på när tangenten slås an, samt släpps.

Syntens vridreglage är potentiometrar kopplade till A/D-omvandlare på datorkortet. Till skillnad från brytarna genereras inget avbrott vid användning, istället sparas värdet till ett fält i minnet med hjälp av direct memory access (DMA). Fältet innehåller 12-bitars heltal och läses av regelbundet. Värderna i fältet multipliceras sedan med en lämplig konstant beroende på variabelns användning och önskat värdesomfång.

För att generera stabila värden ur potentiometrarna går det avlästa värdet genom en normaliseringsmetod. Normaliseringsmetoden tar en andel av föregående värde och adderar med en mindre andel av nya värdet. Metoden fungerar som ett enkelt lågpassfilter av första ordningen och resulterar i ett stabilt värde som inte fluktuerar.

4.5.2 Design av användargränsnitt

Gränssnittet är utformat dels för att konformera till traditionella syntars gränssnitt och dels för att tillhandahålla snabb åtkomst av viktiga funktioner. På grund av ett begränsat antal A/D-omvandlare fick antalet vridreglage begränsas. Detta gjordes genom att låta vissa vridreglage styra mer än än parametet. Följdaktligen är gränssnittet mindre lättanvänt än vad som hade varit möjligt med tillgång till fler A/D-omvandlare. Gränssnittet är uppdelat i 4 sektioner, där varje sektion styr en specifik del av signalkedjan. Med hjälp av dessa kriterier utarbetades en ritning på en prototyp fram (se figur 4.1).

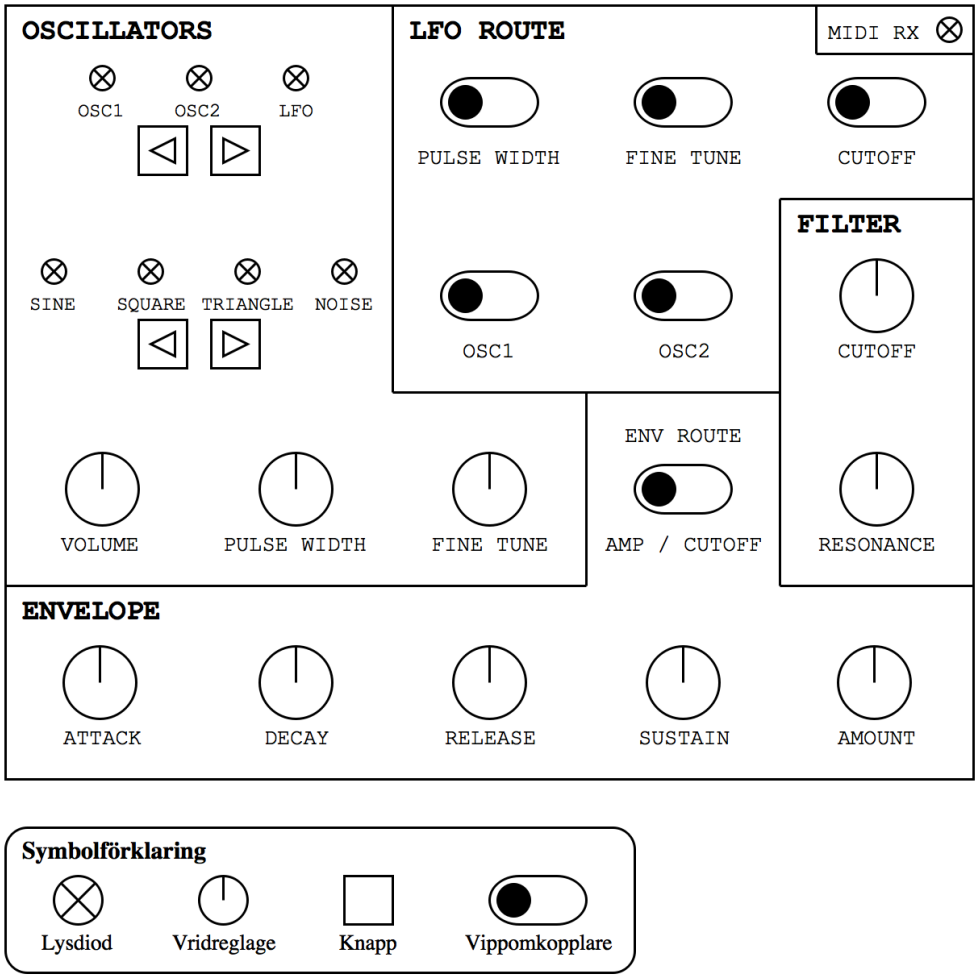
OSCILLATORS innehåller brytare och vridreglage för att styra syntens oscillatorer. I sektionens övre del finns två brytare som används för att välja vilken oscillator som skall styras. Lysdioder indikerar valet av oscillator. I den mellersta delen kan vald oscillators vågform väljas på liknande sätt. Sektionens nedre del innehåller vridreglage för att justera parametrarna hos den valda oscillatorn.

LFO ROUTE innehåller ett antal vippomkopplare som styr vilka parametrar LFO:n skall styra, samt vilka oscillatorer som skall beröras.

ENVELOPE innehåller ett antal vridreglage för att justera konturgeneratorernas parametrar, samt en vippomkopplare för att välja huruvida det är amplitudens eller filtrets konturgenerator som berörs.

FILTER innehåller två vridreglage för att justera lågpassfiltrets parametrar.

Vidare finns en lysdiod i panelens övre högra del som indikerar att synten tar emot MIDI-meddelanden.



Figur 4.1: Ritning över det planerade gränssnittet

4.6 Implementation av oscillator och LFO

Följande kapitel beskriver hur oscillatorns olika våggeneratorer är implementerade. Efter en inledande generell text följer en djupare beskrivning av hur de olika vågformerna generas. Motsvarighet till en analog synts oscillator är egentligen bara en metod i mjukvaran. Metoden läser av vilken vågform som den sprecifika oscillatorn är inställd på. Metoden ropar sedan på den angivna vågformens metod.

Oscillatorn kan generera fyra olika vågformer: sinus-, triangel- och fyrkantsvåg, samt vitt brus. Synten har två ljudalstrande oscillatorer vilket möjliggör exempelvis en kombination av fyrkant- och triangelvåg, vilket kan skapa komplexa vågformer (figur A.7 i appendix). Oscillatorns användarinställda värden sparas i en anpassad datastruktur som innehåller respektive rattars värden, medan en annan datastruktur sparar de dolda värdena som oscillatorn behöver för att kunna räkna ut samplingen som ska spelas.

Datastrukturen med de användarvariablerna benämns *global_oscillator* och innehåller följande variabler:

- *shape* - variabeln anger vilken vågform som ska genereras
- *multiplier* - anger var brytpunkten placeras (0-1)
- $finetune$ - variabel som styr hur mycket oscillatorn ska frångå grundfrekvensen
- *route* - flaggor som används för att bestämma vad LFOn ska påverka.
- *amplitude* - dikterar högsta signalstyrka. För LFOn påverkar variabeln hur mycket den påverkar annan variabel.

Datastrukturen för de dolda värdena benämns *local_oscillator* och består av dessa variabler:

- t - tidsvariabel, används för att generera rätt sampel.
- *period* - antal samplingar per period sparas i variabeln.
- *angular_velo* - vinkelfrekvensen. Används vid generering av sinusvågen
- $actual\_freq$ - vågformens egentliga frekvens
- *triangle_positive* - lutningen på signalökningen vid genering av triangel-våg
- *triangle_negative* - lutningen på signalminskningen vid genering av triangel-våg

Triangel- och fyrkantsvågen har en brytpunk som kan varieras. Brytpunkt är den punkt i perioden där en fyrkantsvåg går från hög till låg. För en triangelvåg motsvarar brytpunkten den tidpunkt i perioden då vågformen har sitt maximala värde. Triangelvågens brytpunkt realiseras på sådant vis att signalen utgår från minimivån för att sedan nå maxnivån där brytpunkten placeras. När maxnivån är nådd minskas signalen mot miniminivå som signalen når vid periodens slut. Triangelvågens implementation möjliggör på detta vis en mängd olika vågformer med egen karaktär (figur A.4, A.5, och A.6 i bilaga).

Oscillatorerna går även att ”stämma ur” en hel oktav både upp och ned. Med detta menas att oscillatorns faktiska frekvens är en annan än den av grundtonen givna frekvensen. Funktionen implementeras genom en variabel med ett värde som kan justeras från 0,5 till 2. Variabeln multipliceras sedan med grundtonens frekvens. Oscillatorn spelar således upp tonen i grundtonens frekvens när variabeln är ställd till 1. I följade text kommer variabeln att benämnas *fine tune*. För djupare förståelse finns pseudokod för samtliga oscillatorer bland bilagorna. (B.1.2, B.1.1, B.1.4, och B.1.3 i bilagan)

4.6.1 Triangelvåg

Vid generering av triangelvåg räknas först den faktiska frekvensen ut, med hjälp av tonens grundfrekvens samt värdet på värdet på *fine tune*. Vågformens periodtid i sampel räknas sedan ut med hjälp av den faktiska frekvensen dividerat med samplingsfrekvensen. Detta tal multipliceras sedan med 100 för att motverka problem som uppkommer med tidsdiskreta variblar. Med hjälp av multipliceringen flyttas det ursprungliga decimaltecknet bak vilket möjliggör en återgivning av frekvenser med högre noggranhet. Även tidsvariablen adderas i steg av 100 istället för 1. Triangelvågens brytpunkt realiseras på sådant vis att variabeln antar ett värde mellan 0 och 1, vilket multipliceras med antal samplingar perioden består av. Detta delar upp perioden i två skeden - ett där signalen ska öka från lägsta till högsta nivå, samt ett där nivån går från högsta till lägsta nivå. Lutningen i dessa skeden räknas ut genom att multiplicera högsta nivån med 2, för att sedan dividera med antal samplingar det specifika skedet består av.

4.6.2 Fyrkantsvåg

Implementeringen av fyrkantsvågen följer samma mall. Det som skiljer mellan dessa metoder är att i första skedet genererar fyrkantsvågen en hög signal och sedan i det efterföljande skedet en låg signal.

4.6.3 Sinusvåg

Oscillatorn generar sinusvågen med hjälp av en funktion från ett programbibliotek. Programbiblioteket innehåller en tabell för en sinuskurva med 512 samplingar. Vid funktionsanrop bestäms ett värde utifrån tabellen med hjälp av linjär interpolering. Vid vågformens beräkning anropas sinusfunktionen med en konstant multiplicerad med oscillatorns tidsvariabel. Denna konstant är lika med 2π dividerat med syntens utmatningsfrekvens. Konstanten beräknas redan innan huvudslingan för att spara beräkningskraft.

4.6.4 Vitt brus

Vitt brus består av en jämn fördelning av samtliga frekvenser. Detta genereras med hjälp av datorkortets inbyggda slumptalsgenerator.

4.6.5 LFO

LFO:n är implementerad nästan precis som de ljudalstrande oscillatorerna. För att på en enkelt vis kunna styra LFOn:s frekvens med en potentiometer valdes den att konstant spelas i F0 (21,83 Hz). Grundfrekvensen multipliceras sedan med värdet från en användarstyrd variabel som går mellan 0-1. Variabeln fungerar på samma vis som finjusteringen av frekvensen på de ljudalstrande oscillatorerna - men då variabeln har ett annat värdesomfång förverkligas det på annat vis. Det gör att LFOn får ett frekvensomfång på 0-21,83 Hz. LFOn kan väljas att styra pulsbredd respektive brytpunk, finstämning, samt filtrets gränsfrekvens. Vid modulation av oscillatorvärden kan LFOn väljas att påverka båda oscillatorerna eller bara en av dem med hjälp av vippomkopplare på frontpanelen.

4.7 Implementation av filter

En viktig komponent av mjukvaran är ett så kallat lågpassfilter. Vad gäller digitala filter så finns det flera olika sorter, som kan implementeras på en mängd olika vis. Efter fördjupning i ämnet föll valet på ett filter av typen *infinite impulse response* (IIR) av andra ordningen. Som filtrets namn antyder har det ett impulssvar som i teorin är nollskillt oändligt länge. Detta beror på det faktum att filtret bygger på återkoppling. Återkopplingen möjliggör inte bara instabilitet men också en viss självsvängning, även kallat resonans. Resonansen yttrar sig i en förstärkning av signalen vid och omkring filtrets gränsfrekvens. Filtrets ordningstal representerar dess fördröjning räknat i sampel och är på samma gång ett mått på dess beräkningskomplexitet. Filtrets effektiva dämpning är dessutom avhängigt av dess ordningstal.

Ett alternativ hade varit att implementera ett filter av type *finite impulse response* (FIR). Till skillnad från IIR-filtret finns ingen återkoppling, och därmed ingen möjlighet till resonans. Dessutom krävs FIR-filtret mer beräkningskraft för att åstadkomma samma dämpning som ett IIR-filter med lägre beräkningsintensitet. Med hänvisning till kravspecifikationen där resonans förekommer och IIR-filtrets lägre beräkningsintensitet föreföll det sig som ett självklart val.

Filtret som implementerats ger en dämpning på ca 12 dB/oktav och kan kedjekopplas med sig själv för en dubbelt så hög dämpningseffekt (24 dB/oktav). För att använda så lite beräkningskraft som möjligt har synten inte utrustasts med möjligheten att välja dämpningen 24 dB/oktav på filtret.

Filtret realiseras med differensekvation 4.1 där x representerar ofiltrerad signal, y representerar filtrerad signal och t representerar en tidsvariabel. Koefficienterna α, β och γ ges av ekvationerna 4.2, 4.3, och 4.4 där variablerna θ_c representerar filtrets gränsfrekvens och d dess dämpningsfaktor.[5] Dämpningsfaktor ska ej förväxlas med dämpning, då det senare är ett mått på filtrets dämpning av signalen, medan föregående begrepp beskriver dämpningen av filtrets återkoppling. Dämpningsfaktorn kan varieras för att styra resonansens intensitet och är omvänt proportionell mot densamma. För filtrets frekvenssvar se figur 4.2. För ett komplett bodediagram inkl. fasförskjutning, se figur A.11.

$$y[n] = 2(\alpha x[n] + 2\alpha x[n-1] + \alpha x[n-2] + \gamma y[n-1] - \beta y[n-2]) \tag{4.1}$$

$$\beta = \frac{1}{2} \cdot \frac{1 - (d/2)\sin\theta_c}{1 + (d/2)\sin\theta_c} \tag{4.2}$$

$$\alpha = \frac{(1/2) + \beta - \gamma}{4} \tag{4.3}$$

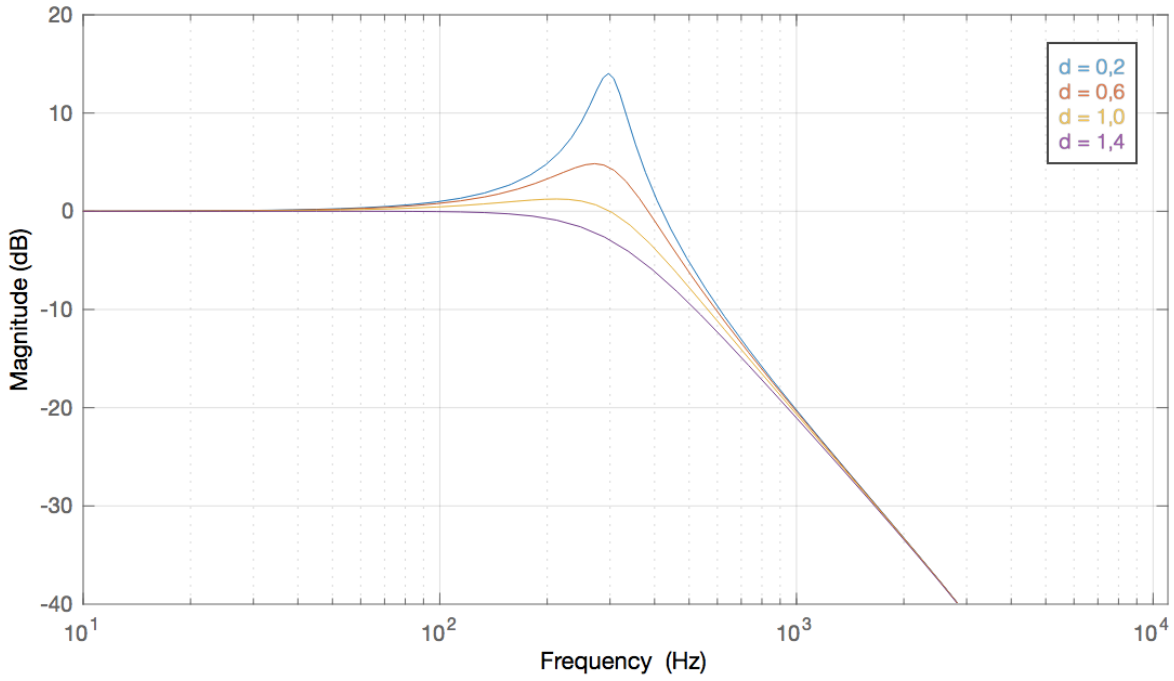
$$\gamma = (\frac{1}{2} + \beta)\cos\theta_c \tag{4.4}$$

För att lagra nödvändig data för filtrets beräkningar definierades två sorters datastrukturer: *coefficients* innehållandes ovan nämnda koefficienter samt *voice_cache* innehållandes följande:

- x - fält innehållandes signalens tre senaste ofiltrerade värdena
- y - fält innehållandes signalens tre senaste filtrerade värdena
- n - index för senaste värdet

Vid varje förändring av filtrets gränsfrekvens måste koefficienterna beräknas på nytt. Eftersom såväl konturgenerator som LFO modulerar gränsfrekvensen resulterar detta i att koefficienterna räknas om för varje sampel som ska filtreras. Filtreringen utförs i följande steg för varje sampel:

1. Beräkning av koefficienter (4.2, 4.3, samt 4.4) samt lagring av dessa i röstens *coefficients*-variabel
2. Lagring av ofiltrerat värde i röstens *voice_cache*-variabel
3. Beräkning av filtrerat värde (4.1) utifrån röstens *voice_cache*-variabel och *coefficients*-variabel
4. Lagring av filtrerat värde i röstens *voice_cache*-variabel
5. Inkrementering av index i röstens *voice_cache*-variabel (mod 3)



Figur 4.2: Filtrets frekvenssvar för olika d då $\theta_c = 300$ Hz.

4.8 Implementation av konturgenerator

Synten är uttrustad med två separata konturgeneratorer. En konturgenerator styr röstens amplitud, medan den andra styr filtrets gränsfrekvens. Dessa två körs helt separat och har egna värden. Synten utrustades med två konturgeneratorer då det ansågs viktigt att både kunna påverka styrkan på utsignalen samt filtrets gränsfrekvens över tid. Konturgeneratorerna på synten är programmerade så att rösten har ett par av konturgeneratorer. Vid en eventuell vidareutveckling till en synt med fler röster är det enkelt att implementera dessa.

En rösts konturgenerator kan i en given tidpunkt befinna sig i ett av de fyra stadierna (se bilaga B.2). Vilket stadie konturgeneratorerna befinner sig i sparas i röstens datastruktur. Konturgeneratorerna är uppbyggda med hjälp av en switch-sats. För båda konturgeneratorerna gäller att när en röst initieras, så befinner sig båda i attack.

Inför implementationen av konturgeneratorn diskuterades att låta konturgeneratorns stadie vara enbart beroende av tid. Detta avfärdades dock efter insikten om att konturgeneratorn då skulle kunna hoppa till ett tidigare stadie om dess parametrar förändrades under röstens livstid. Istället valdes att beräkna en förändringstakt baserad på de inställda tidsvärdena och låta konturgeneratorn avancera till nästa stadie då ett visst villkor är uppfyllt. En uppräkningstyp (enum) håller reda på varje rösts stadie.

De tidsbaserade stadiumen i konturgeneratorn har alla en separat formel som används för att räkna ut förändringstakten varje sampel. Där variabeln som reglaget modifierar representerar tiden i sekunder. Alla formler använder sig av makrot $OUTPUT\_FREQ$ vilket är syntens samplingsfrekvens. *Sustain*-reglaget anger istället en nivå. Sätts tiden till 0 i något stadie används inte formeln, detta för att förhindra division med 0. Stadiet med tidsvariabeln 0 utelämnas då från konturgeneratorn. Ställs *release*-reglaget till 0 tas rösten bort direkt då tangenten släpps.

Gemensamt för både amplitudens och filtrets konturgeneratorer är att de returnerar ett flyttal mellan 0,0 och 1,0 som representerar konturgeneratorns utveckling. Vad gäller amplituden multipliceras detta värde med röstens utsignal för att realisera en amplitudutveckling.

I filtrets fall är förfarandet något mer komplicerat. Konturgeneratorn förflyttar filtrets faktiska gränsfrekvens (*actual_cutoff*) från det inställda grundvärdet (*cutoff*) till ett värde som anges av inställningen *envelope amount*, och sedan tillbaka igen. Mer specifikt multipliceras värdet som konturgeneratorn returnerar *envelope modifier* med differensen mellan den av användaren inställda gränsfrekvensen (*cutoff*) och målfrekvensen som ges av *envelope_amount*. *envelope_amount* omfattar samtliga värden filtrets gränsfrekvens kan anta och indikerar *attack*-stadiets målnivå. Detta kan således resultera i en konturgenerator med en negativ utveckling då $envelope\_amount < cutoff$. Beräkningen tydliggörs av formel 4.5.

$$actual\_cutoff = cutoff + envelope\_modifier \cdot (envelope\_amount - cutoff) \tag{4.5}$$

4.8.1 *Attack*-stadiet

Förändringstakten räknas ut genom formel 4.6 där *attack_time* representerar tiden som konturgeneratorn ska befinna sig i stadiet.

$$attack\_slope = \frac{1}{attack\_time \cdot OUTPUT\_FREQ} \quad 0 < attack\_time \leqslant 20 \tag{4.6}$$

Attack-stadiets villkor är uppfyllt då returvärdet nått 1. Konturgeneratorn går då över till nästföljande stadie; *decay*.

4.8.2 *Decay*-stadiet

Variabeln *decay_time* indikerar längden på tiden det tar från att värdet går från attack-sektionens slutvärde till en nivå angiven av *sustain_level*-variabeln. När värdet på variabeln nått det angivna *sustain_level*-värdet ändras konturgeneratorn stadie till *sustain*. Formeln 4.7 beskriver med vilket värde varibeln ändras varje sampel.

$$decay\_slope = \frac{sustain\_level - 1}{decay\_time \cdot OUTPUT\_FREQ} \tag{4.7}$$

$$0 < decay\_time \leqslant 20 \qquad 0 \leqslant sustain\_level \leqslant 1$$

4.8.3 *Sustain*-stadiet

I *sustain*-stadiet ändras ingen variabel, utan värdes hålls till det satta värdet. När ett MIDI-avbrott generas, läses det tänkta MIDI-meddelandet av. Är meddelandet ett *note off*-meddelande äntrar rösten *release*-stadiet.

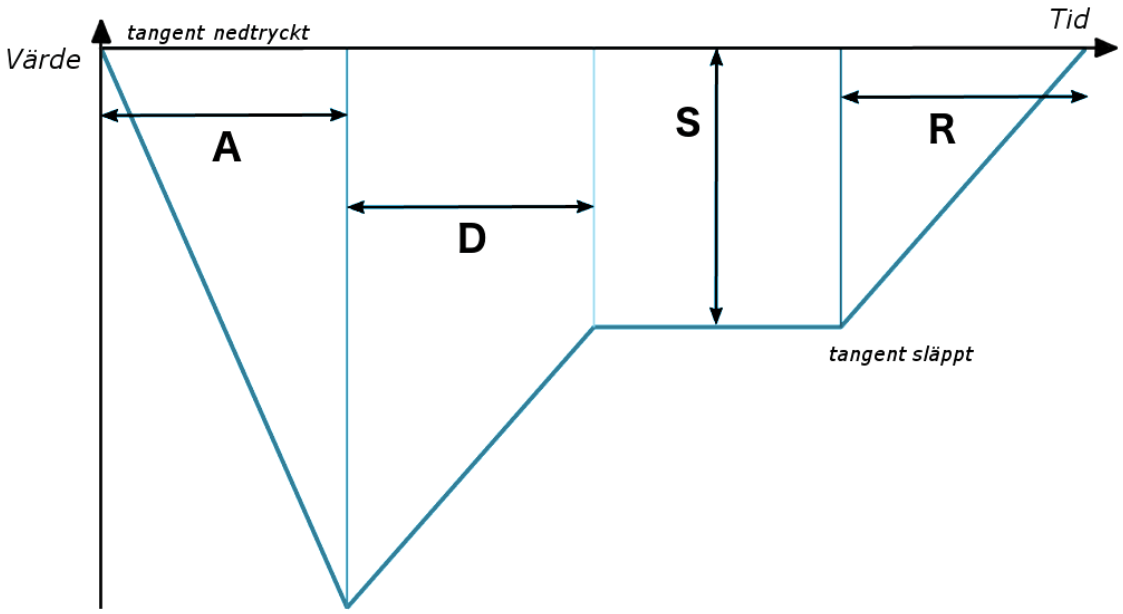
4.8.4 *Release*-stadiet

Lutningen på variabeländringen anges av formeln 4.8.

$$release\_slope = \frac{sustain\_level}{release\_time \cdot OUTPUT\_FREQ} \tag{4.8}$$

$$0 < release\_time \leqslant 20 \quad 0 \leqslant sustain\_level \leqslant 1$$

Konturgeneratorn som påverkar amplitudet tar bort rösten när variabeln nått ursprungsnivån. Konturgeneratorn vilken påverkar filtrets gränsfrekvens avslutar minskningen av variabeln och förblir på ursprungsvärdet.



Figur 4.3: Exempel på en konturgenerator då *envelope_amount* är satt till negativt värde

5 Resultat

I projektet har vi utvecklat mjukvaran samt designat en prototyp av en analogmodellerande synt samt besvarat följande frågeställningar:

- Hur en mjukvarubaserad oscillator implementeras.
- Hur ett filter implementeras samt vilket filter som bäst lämpas sig för en synt.
- Hur en digital konturgenerator implementeras.

Vi har inte besvarat på fullgott vis vad som krävs av datorkortet. Utan snarare har vi besvarat vad som *inte* orkar med att driva en synt med de eganskaper vi hade i åtanke. Den slutgiltiga monteringen hanns inte med. Mjukvara finns för att hantera de användargränssnitt som synten använder sig av, men lödningen av komponter till ett chassi slutfördes ej. Inte heller har hanteringen av MIDI-meddelande hunnits med.

6 Diskussion och slutsats

Projektet visade sig mer omfattande än vad vi hade trott. De flesta moment var väldigt tidskrävande. Saker som vi trodde skulle vara enkla visade sig vara komplicerade. Det var inte självklart hur en given funktion skulle implementeras, och vi fick utforska och utvärdera olika lösningar. Att konfigurera periferienheter och få dem att bete sig på önskat vis var även det mycket tidskrävande. Något som däremot visade sig vara mycket enklare än beräknat var implementationen av konturgenerator och filter.

6.1 Bedöming av resultat

I arbetet har oscillatorer utvecklats vilka generar vågformer väldigt lika de från analog källa. Figur A.9 visar en triangelvåg som är genererad av en Korg MS-20 mini, vilken har en helt analog signalkedja. Vågformen skiljer sig till viss del från vår synts triangelvåg (figur A.5) med rundade hörn till skillnad från skarpa. Anledningen till detta är inte att en analog synt är inkapabel till att generera en vågform med skarpa ändringar, utan är snarare ett designval från dem. När respektive syntars fyrkantsvåg jämförs är skillnaden mindre (se figur A.2 och A.8). Figur A.10 visar MS-20 Minis sågtand, även den skiljer sig från vår sågtand A.4, men då dessa utgår från triangeln är det inte särskilt oväntat.

Oscillatorn har inga direkt emulerade felaktigheter - skavanker som kan påstås “väcka liv” i analoga syntar har utelämnats helt. En analog oscillator kan ibland uppträda på ett oväntat vis, där de mjukvarubaserade oscillatorerna alltid generar samma frekvens. Dessa felaktigheter kan istället emuleras med hjälp av LFOn.

Filterdelen uppfyller de krav som ställdes. Vid högre resonans fås en mer digital karaktär på ljudet än vad ett analogt filter producerar. Vi tror att det är väldigt svårt att skapa ett mjukvarubaserat filter som låter som ett analogt. Många av de syntar med en digital signalgenerering är utrustade med analoga filter, vilket vi tror kan ha att göra med att det krävs mycket beräkningskraft samt bra skrivna algoritmer för att utveckla ett motsvarande digitalt filter. Filtret låter dock bra i de mer sansade inställningarna.

Den utvecklade konturgeneratorn uppträder på samma vis som på analoga syntar. I många fall har syntar med en analog signalkedja en digital konturgenerator, då de uppträder på ett väldigt snarlikt sätt som den analoga motsvarigheten. De första digitala konturgeneratorerna hade låg uppdateringsfrekvens, vilket omöjliggjorde en snabb övergång mellan stadierna i jämförelse med dess analoga motsvarighet[6]. Konturgeneratorn utvecklad i detta projekt är dock tillräckligt snabb för att några sådana problem inte ska infinna sig.

Det visade sig att MD407:ans processorkraft inte räckte till för att göra alla de beräkningar vi ville göra och till följd av detta fick vi göra vissa eftergifter. Bland annat valdes att sänka samplingsfrekvensen vilket begränsar frekvensomvånget synten kan återskapa. Det slutgiltiga samplingsfrekvensen sattes till 28 kHz, vilket medför att synten inte kan generera frekvenser högre än nyquistfrekvensen på 14 000 Hz. Synten får även problem med vikning när ljudvågor spelas som innehåller frekvenser över 14 000 Hz, det vill säga de olika

övertoner som återfinns i ljudet.[7] Vikningen uppenbarar sig i form av högfrekventa missljud.

Hur MIDI-styrning implementeras besvarades inte på fullvärdigt vis.

6.2 Diskussion

För att motverka de nämnda missljuden skulle synten kunna utrustas med ett passivt analogt filter efter signalutmatningen. Detta filter skulle konstrueras för att ha en sådan gränsfrekvens att missljuden kraftigt dämpas. En fördel med ett analogt filter är att det inte ökar belastningen på processorn.

Problemet skulle helt kunna lösas genom att använda en snabbare mikrokontroller, eller flera för att på så vis dela upp arbetsbördan. En mikrokontroller som tar hand om oscillatorerna medan en annan sköter filter och andra beräkningar skulle antagligen lösa alla våra problem. Uppskalning med fler funktioner och röster skulle då inte heller vara allt för besvärligt. Funderingar fanns om dessa skulle kunna kopplas ihop med hjälp av CAN-bussen; men efter en uppskattning av mängden data som behöver skickas kom vi fram till att CAN-protokollet är för långsamt. Andra överföringsprotokoll skulle behöva användas, där USB skulle kunna vara ett möjligt alternativ. En kombination av digital och analog teknik skulle också kunna vara ett alternativ för att minska beräkningsbördan på datorkortet. Då går dock lite av frågeställningen bort.

Det är också mycket möjligt att andra algoritmer hade gett ett bättre resultat. Eftersom det är första gången vi tar oss an ett projekt av den här omfattningen kan vi inte utesluta att det går att göra på ett mer effektivt sätt. Filterkoefficienterna skulle möjligtvis kunna placeras i en så kallad *look-up-table* för koefficienterna vid de vanligaste frekvenserna. Koefficienterna för de frekvenser vilka inte förekommer i tabellen kan sedan genom interpolering tas fram. Att spara färdigräknade filterkoefficienter på det viset skulle antagligen effektivisera syntens filterdelen.

Oförmågan att besvara hur MIDI-protokollet på bäst vis implementerades har att göra med underskattning av problemet från vårat håll. Det antogs att MIDI bara enkelt kan kopplas in och sedan läsa av meddelandena, vilket var fel. En hel del olika elektriska kretsar krävdes; bland annat en opto-kopplare vilken används för att motverka jordfel. Kretsarna som krävdes visade sig vara besvärliga att anskaffa. Problemet resulterade i att vi fick använda kretsar som inte var angivna i MIDI-specifikationerna. När vi sedan hade problem att läsa av meddelandet skapades en osäkerhet om kretsen fungerade korrekt, vilket gjorde att vi avbröt utvecklingen och koncentrerade oss på andra problem. MIDI-protokollet hade mycket väl kunnat lösats om vi tidigare satt oss in i problemet.

Då ett syfte med att göra arbetet var att uttöka vår förståelse för maskinnära programmering, får det anses uppfyllt. Båda projektdeltagarna navigerar med större säkerhet genom olika programbibliotek. Det som tidigare uppfattades som något relativt krångligt kunde i slutet av projektet göras med ett större självförtroende. I projektet har vi utvecklat metoder som skiljer sig mycket från våra tidigare projekt. Det har varit väldigt lärorikt att dels behöva komma fram till vilka algoritmer bäst lämpar sig för att uppnå det vi vill, samt att sedan implementera dessa. Den begränsade beräkningskapaciteten hos MD407 har även tvingat oss att skriva effektiv kod och tänka i nya banor för att hushålla med beräkningskraft.

6.3 Miljö och etik

Projektet har diskuteras utifrån miljö- och etiksynpunkt men några särskilda för- eller nackdelar har inte funnits.

Referenser

[1] I. Synthesis. (). Basic synthesis: Part 2 – filters, URL: http://www.innovativesynthesis.com/basic-synthesis-part-2-%E2%80%93-filters/ (hämtad 2017-02-08).

[2] H. Nyquist, "Certain topics in telegraph transmission theory", tekn. rapport, 1928. URL: http://www.physics.oregonstate.edu/~hetheriw/whiki/ph415_s15/tasks/dsp/files/nyquist/Nyquist.pdf.

[3] T. Rossing, *Springer Handbook of Acoustics*. Springer, 2007, ISBN: 978-0-387-30446-5.

[4] J. Watkinson, *The Art of Digital Audio*. Butterworth-Heinemann, 1993, ISBN: 0240513207.

[5] J. Lane och G. Hillman, *Implementing iir/fir filters, With motorola's dsp56000/sps/dsp56001 digital signal processors*, version rev 2, Motorola Inc., 1993.

[6] (). Jx-8p, URL: http://www.antilles-music.com/synthesizers/jx8p (hämtad 2017-02-21).

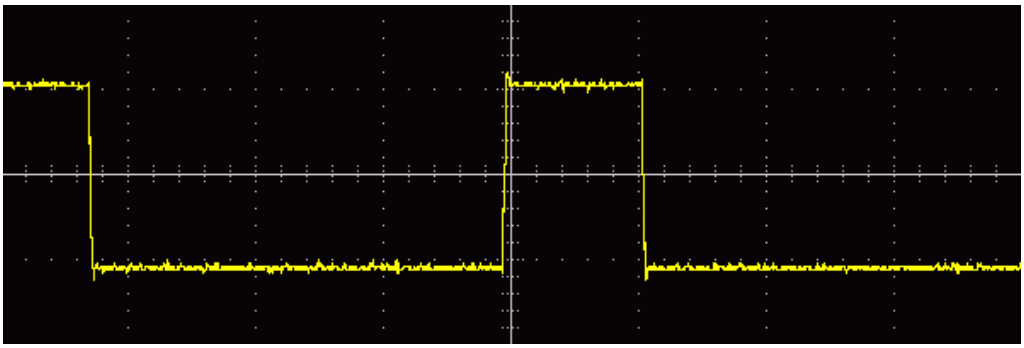
[7] S. R. Lab. (). Sampling rate, URL: http://www.asel.udel.edu/speech/tutorials/instrument/sam_rat.html (hämtad 2017-02-08).

[8] J. Yiu, *The definite guide to arm cortex-m3 and m4 processors*, ARM Ltd.

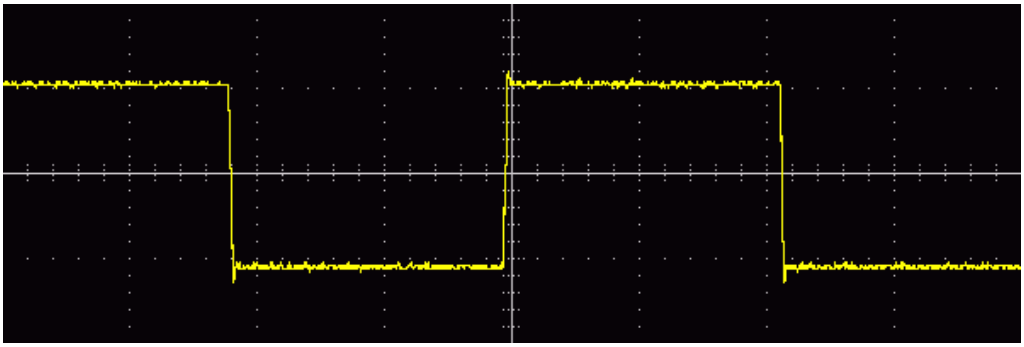
Bilagor

A Figurer

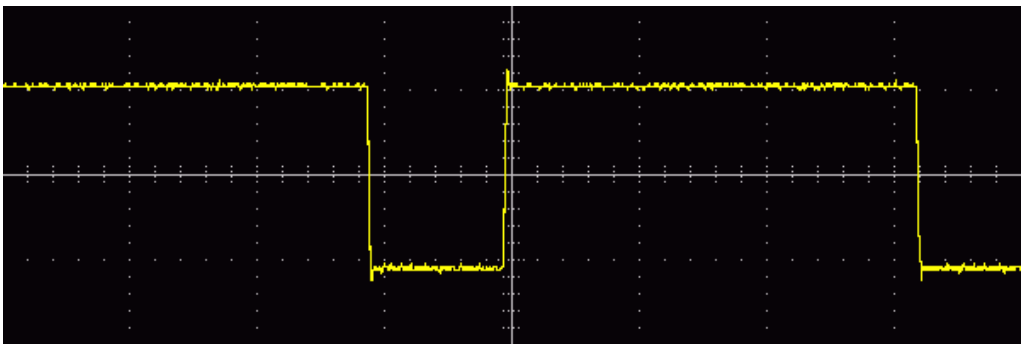
A.1 Fyrkantsvåg



Figur A.1: Fyrkantsvåg med 25% pulsbredd

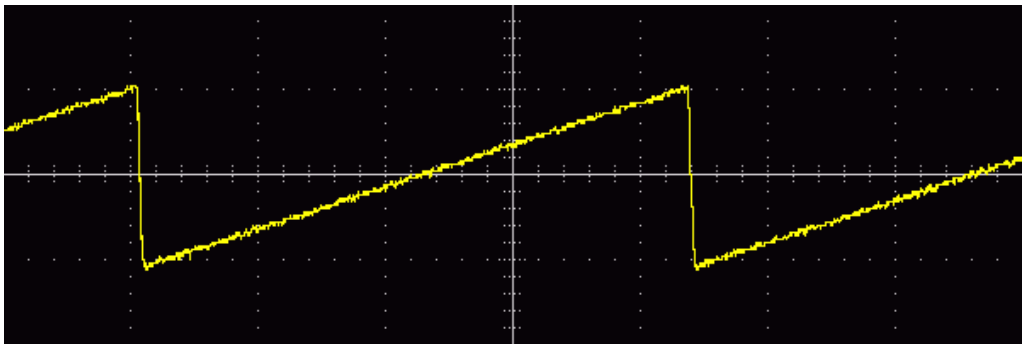


Figur A.2: Fyrkantsvåg med 50% pulsbredd

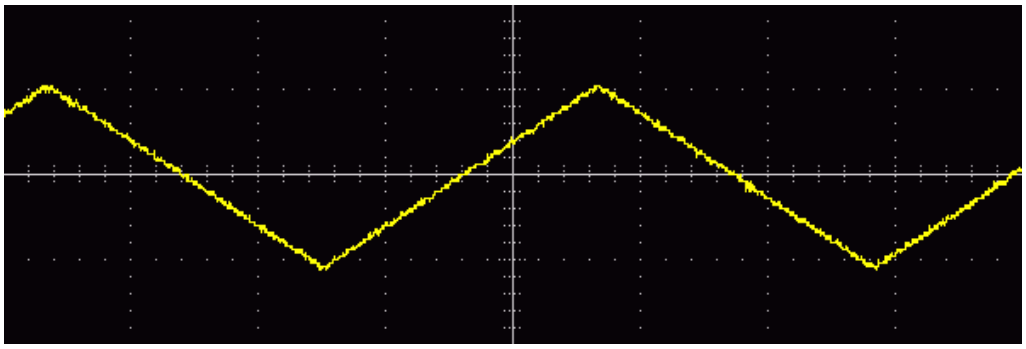


Figur A.3: Fyrkantsvåg med 75% pulsbredd

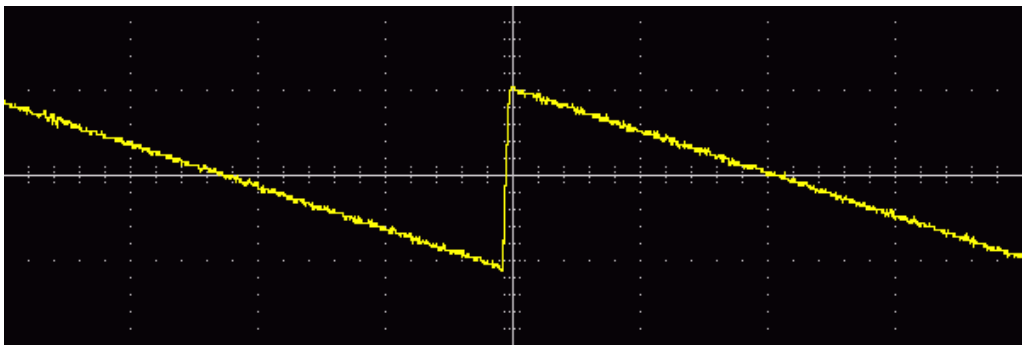
A.2 Triangel- och sågtandsvåg



Figur A.4: Triangelvåg med brytpunkt vid 50% (Sågtandsvåg)

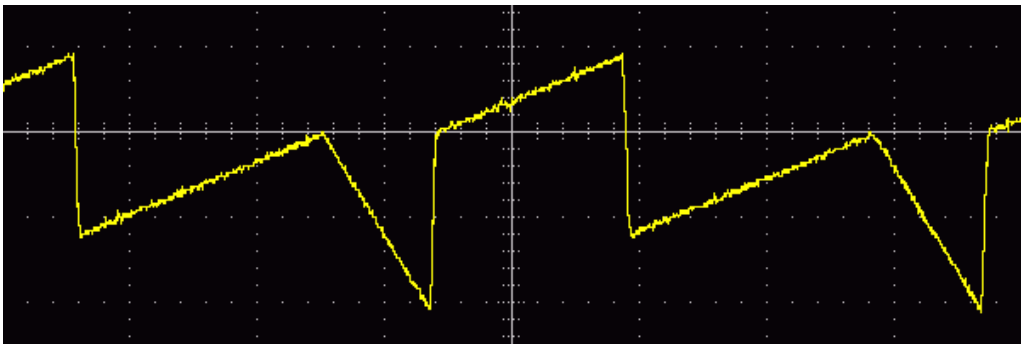


Figur A.5: Triangelvåg med brytpunkt vid 50%



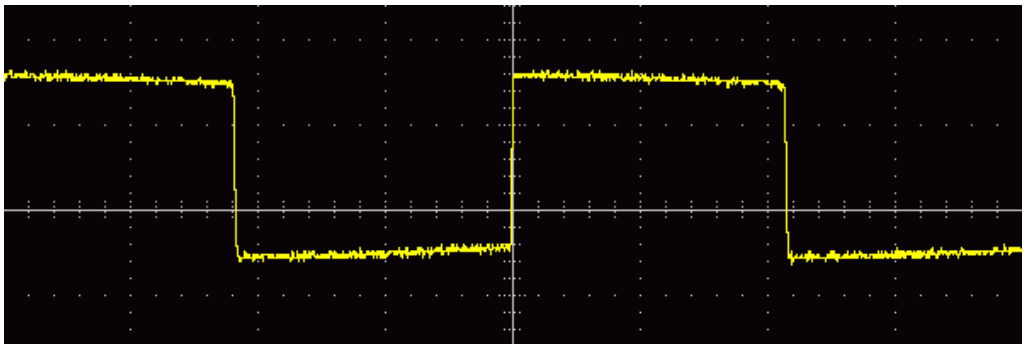
Figur A.6: Triangelvåg med brytpunkt vid 0% (Inverterad sågtandsvåg)

A.3 Kombinerad signal

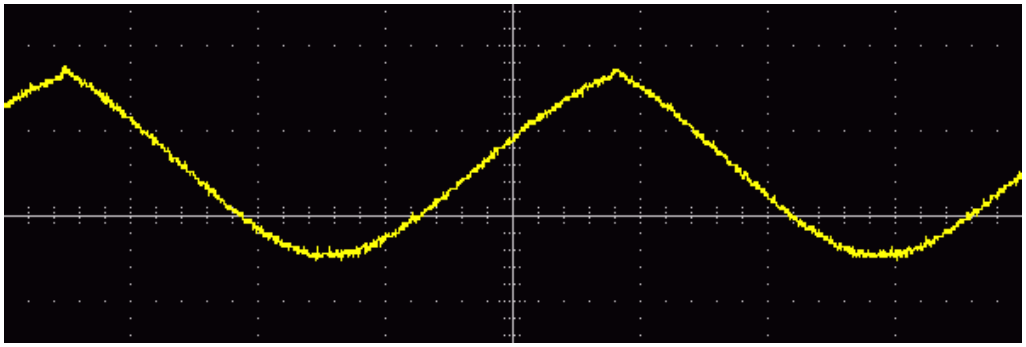


Figur A.7: Fyrkantsvåg med 35% pulsbredd i kombination med triangelvåg med brytpunkt vid 80%

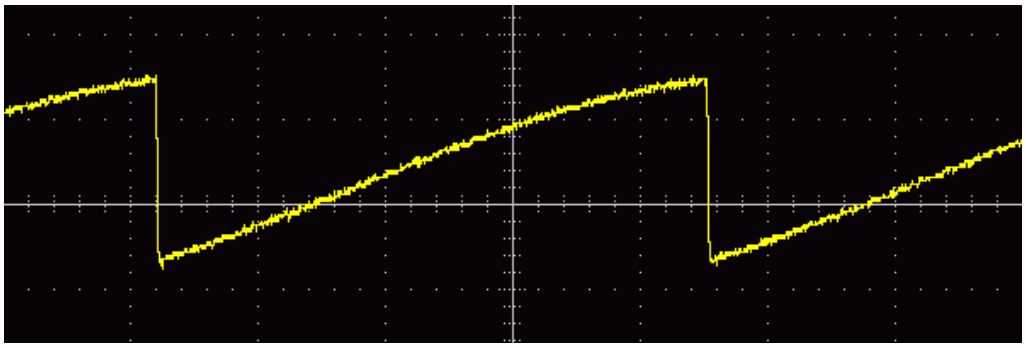
A.4 Vågformer från analog källa



Figur A.8: Fyrkantsvåg med 50%pulsbredd

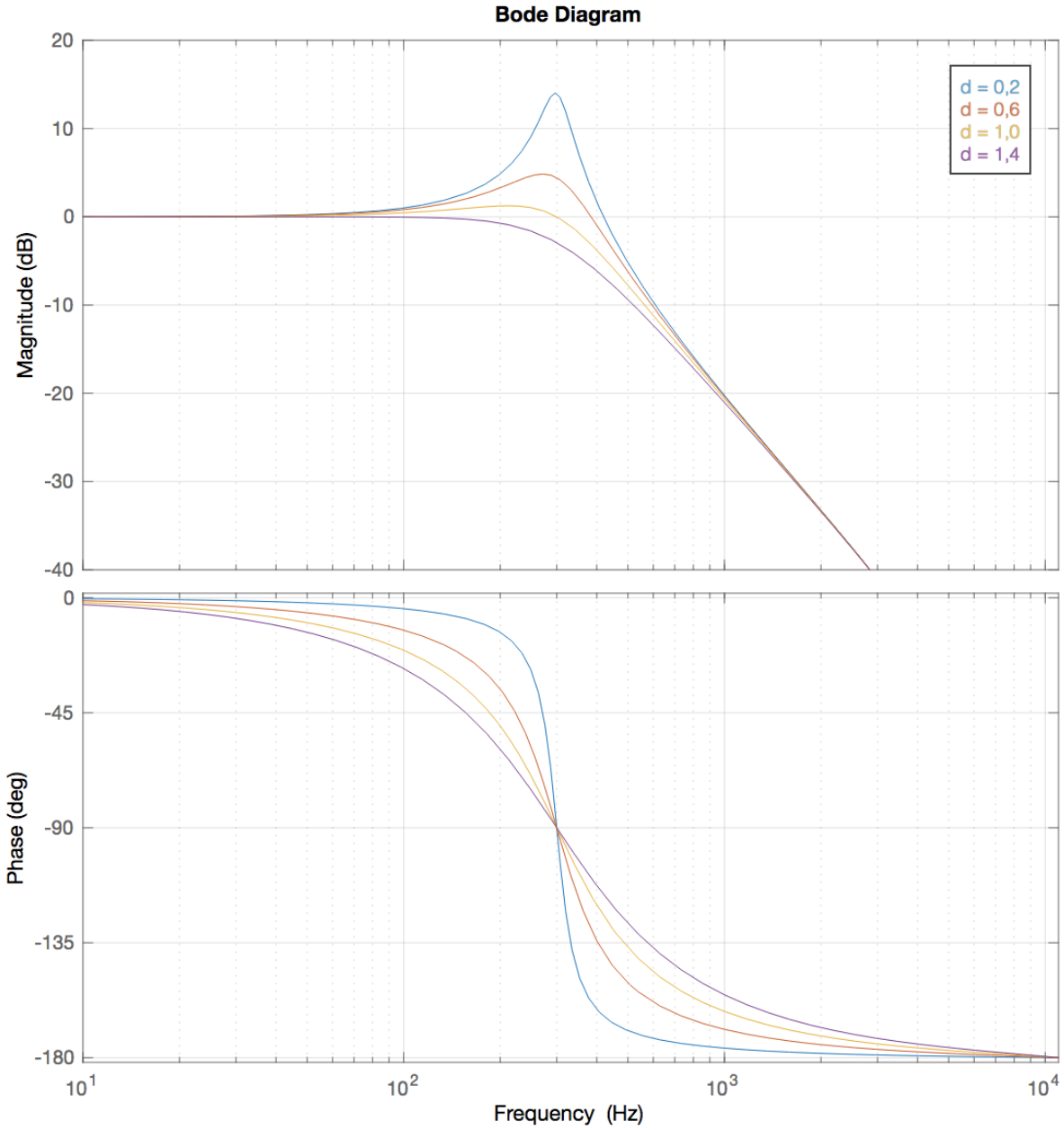


Figur A.9: Triangelvåg med brytpunkt vid 50%



Figur A.10: Sågtandsvåg

A.5 Filter



Figur A.11: Filtrets bodediagram för olika d då $\theta c = 300$ Hz

B Pseudokod

B.1 Oscillatorer

B.1.1 Triangelvåg

```
do each sample
    compute actual freq with fine tune and key
    compute samples in period
    compute where the breakpoint is
    compute increase and decrease
    if (t is in first period)
        increment output
    else decrement output
    make sure t doesnt overflow
    increase t
    return output
```

B.1.2 Fyrkantsvåg

```
do each sample
    compute actual freq with fine tune and key
    compute samples in period
    compute samples in first period with pwm-multiplier
    if (t is in the first period)
        positive output
    else negative output
    make sure t doesnt overflow
    increase t
    return output
```

B.1.3 Sinusvåg

```
w := 2*pi/OUTPUT_FREQ
do each sample
    compute actual freq with fine tune and key
    compute angular velocity by multiplying w with the actual freq
    compute sin with t multiplied with the angular velocity as parameter
    make sure t doesnt overflow
    increase t
    return output
```

B.1.4 Vitt brus

```
do each sample
    return random value
```

B.2 Konturgenerator

```
/*
* The change each sample is decided by the potentiometers.
* The amp envelope will delete the voice when release is
* done.
*/
envelope_output := 0
filter_stage := ATTACK
do each sample
    switch (filter_stage)
            case ATTACK
                    envelope_output += attack_slope
                    if (envelope = max_value)
                            change filter stage to decay
            case DECAY
                    envelope_output -= decay_slope
                    if (envelope_output = sustain_level)
                            change filter stage to sustain
            case SUSTAIN
                    if (key is released)
                            Change filter stage to release
                    else
                        do nothing
            case RELEASE
                    if (envelope_output = start_pos)
                            switch (envelope_type)
                                case amp:
                                    delete voice
                                case filter:
                                    do nothing
                    else
                        envelope_output -= release_slope
```