Aalto-yliopisto
Sähkötekniikan korkeakoulu

ELEC-C5340
Sovellettu digitaalinen signaalinkäsittely

# Syntetisaattori - raportti

Arttu Pitkänen 597322
Juuso Korhonen 652377
Mikko Seppi 672768

6. joulukuuta 2020

# Syntetisaattori

Arttu Pitkänen
Juuso Korhonen
Mikko Seppi

**Tiivistelmä**

*Projektin tavoite on luoda soundiltaan ja teemaltaan 80-luvun syntetisaattori erillisenä ohjelmana. Valittu synteesityyppi on virtual analog -syntetisointi, jossa synteesi perustuu perustyyppisen oskillaattorin moduloimiseen ASDR-, LFO-, taajuus- ja amplitudimodulaatioilla. Ohjelma tukee liitettyä MIDI-kosketinta. Lopullinen toteutus tehtiin Matlab -ohjelmointiympäristössä, jossa toteutettiin sekä logiikka että käyttöliittymäkomponentit.*

## 1 Johdanto

Projektimme tarkoituksena on rakentaa äänisyntetisaattori, joka toimii liitettyjen MIDI-koskettimien kanssa. Syntetisaattorin äänisuunnittelun teemana on 80-luku.

Synteesityypiksi valitsimme virtual analog -synteesin, jolla on yleisesti emuloitu 70- ja 80-luvun musiikissa paljon käytettyä analogista synteesiä. Lähtökohtana synteesissä on jokin yksinkertainen oskillaattori, kuten sahalaita-aalto. Jotta äänestä saataisiin elävämmän kuuloinen, sen amplitudikäyttäytymistä lähdetään muokkaamaan aikatasossa ADSR-verhon mukaiseksi.

Efektien luominen virtuaalianalogisessa synteesissä tapahtuu LFO:n eli matalataajuisen oskillaattorin avulla. Tämän matalataajuisen oskillaattorin voidaan esimerkiksi sallia muokkaavan äänisignaalin amplitudia tai taajuussuodattimen rajataajuutta.

Mainittujen modulointien lisäksi, tässäkin synteesityypissä voidaan vielä tarpeen mukaan tehdä taajuussuodatusta ja amplitudivahvistusta.

## 2 Metodit

Projekti hyödyntää olennaisesti digitaalisen signaalinkäsittelyn kursseilla oppimiamme taitoja amplitudi- ja taajuusmodulaatiosta, joista valitsemamme synteesityyppi virtuaalianaloginen-synteesi pohjimmiltaan rakentuu.

### 2.1 Ohjelmointiympäristö

#### 2.1.1 Testitoteutus JUCE:lla

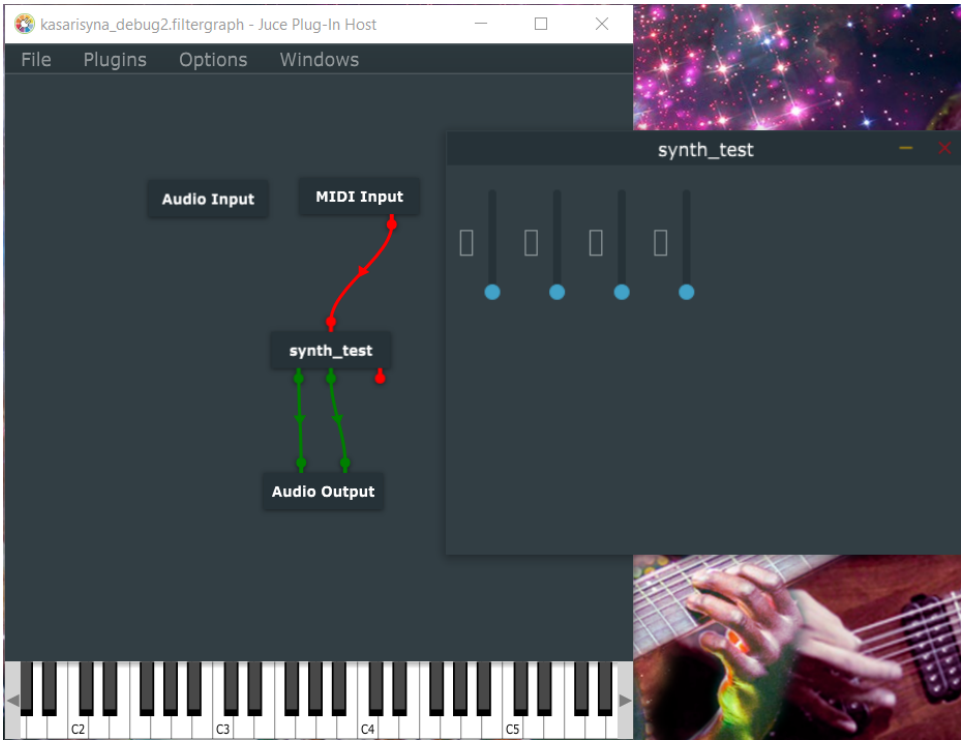
Alkuperäinen idea oli tehdä lopullinen syntetisaattori-toteutus JUCE:n avulla. Teimmekin JUCE:lla/Visual Studiolla yksinkertaisen syntetisaattorin VST3-

liitännäismuodossa. Käytössä oli Projucerin tarjoama Audio plug-in -pohja. Toteutuksesta löytyy perus JUCE-kaura eli MIDI-toiminnalisuus sekä mahdollisuus liittää MIDI-kosketin. Apuna käytimme TheAudioProgrammerin (TheAudioProgrammer, 2020) ja feddenin (fedden, 2020) syntetisaattoridemoja

JUCE:lla toteutettu syntetisaattori soi kahden sahalaita-oskillaattorin avulla, jotka on erotettu hieman sointitaajuuksesta (detune). Siihen on toteutettu neljä slideria, jotka muuttavat muuttujien (Attack, decay, sustain, release) arvoja SynthVoice-luokassa. Toteuksessa ei kuitenkaan ole envelope generatoria, joten sliderit eivät vaikuta ääneen (testattu tulostamalla muuttujien arvoja).

Liitännäistä testattiin JUCE:n plug-in host ohjelmalla, ja lähdekoodi on riippuvainen JUCE:n kirjastoista. JUCE-koodi ja liitännäinen ovat liitetyssä lähdekoodissa JUCE_testitoteutus-kansiossa.

Kuvassa 1 on liitännäisen GUI.




**Kuva 1:** JUCE-liitännäinen JUCE:n testiympäristössä

Päädyimme kuitenkin tekemään lopullisen demon Matlabin avulla. Se on ennestään tuttu ja mielekkäämpi alusta signaalin käsittelyyn ja efektien testaamiseen. Seuraavat osiot käsittelevätkin varsinaista Matlab-toteutusta.

## 2.2 Virtuaalianalogisen synteesin rakennusosat

### 2.2.1 Oskillaattori

Synteesimme lähtee liikkeelle oskillaattorityypin valinnasta. Eniten huomiomme kiinnittyi sahalaita-aallon käyttämiseen oskillaattorina, sillä sahalaidalla on harmonisia taajuuksia ja se on suosittu audio-ohjelmaesimerkeissä.

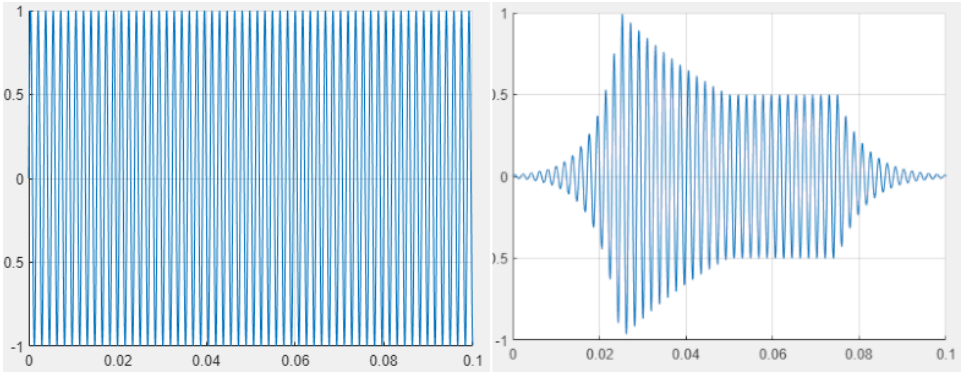
Sahalaita-aaltoa voidaan kuvailla vakioarvoisesti nousevana suorana, joka nollaantuu saavutettuaan maksimiarvonsa (skaalaamattoman sahalaita-aallon tapauksessa se on 1). Matemaattisesti se voidaan esittää ajan $t$ suhteen kaavan 1 mukaisesti:

$$y(t) = A\frac{f}{2\pi}t \mod A, \tag{1}$$

jossa $f$ on oskillaattorin taajuus ja $A$ on skaalauskerroin (maksimiarvo). Muita toteutettuja oskillaattorityyppejä ovat kolmio-, kantti-, sini-, pwm- ja vibraattosiniaalto (toteutus moduloimalla siniaallon vaihetta jatkuvasti toisella matalataajuisella ja pieniamplitudisella siniaallolla).

### 2.2.2 Amplitudiprofiili

Seuraavana vaiheena on amplitudiprofiilin valinta vakioprofiilin ja ADSR-verhon väliltä. ADSR-verhon toteutuksessa olennaista on eksponentiaalinen nousu ja lasku sen eri vaiheiden välillä. Profiilien kuvaajat siniaallon tapauksessa kuvassa 2.




**Kuva 2:** Siniaalto vakioamplitudiprofiililla (vas.) ja muokattuna ADSR-verhon mukaiseksi (oik.).

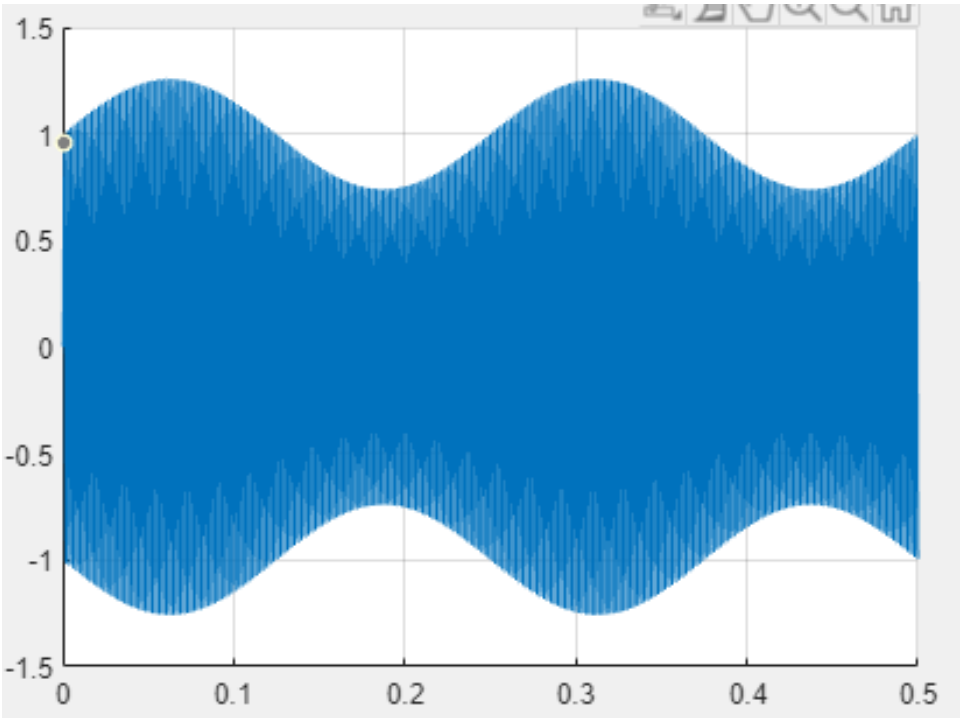
### 2.2.3 LFO

LFO-efektin lisääminen tapahtuu moduloimalla jotakin signaalin ominaisuutta matalataajuisella oskillaattorilla. Me toteutimme sekä tremolo-, että ripple-efektin. Tremolo-efektissä signaalin amplitudia moduloidaan kaavan 2 mukaisesti:

$$y(t) = x(t) \cdot (1 + a\sin(2\pi 4t)), \tag{2}$$

jossa *x(t)* on moduloimaton signaali, $a$ on amplitudivaihtelun skaalauskerroin, ja matalataajuisena oskillaattorina on käytetty siniaaltoa taajuudella 4 (hyvä vaihteluväli tälle taajuudelle on 1-20 Hz).

Tremolo-efekti luo vaikutelman ikään kuin ääni kuuluisi veden alla. Kuvassa 3 on kuvaaja tremolo-efektin vaikutuksesta aikatasossa.



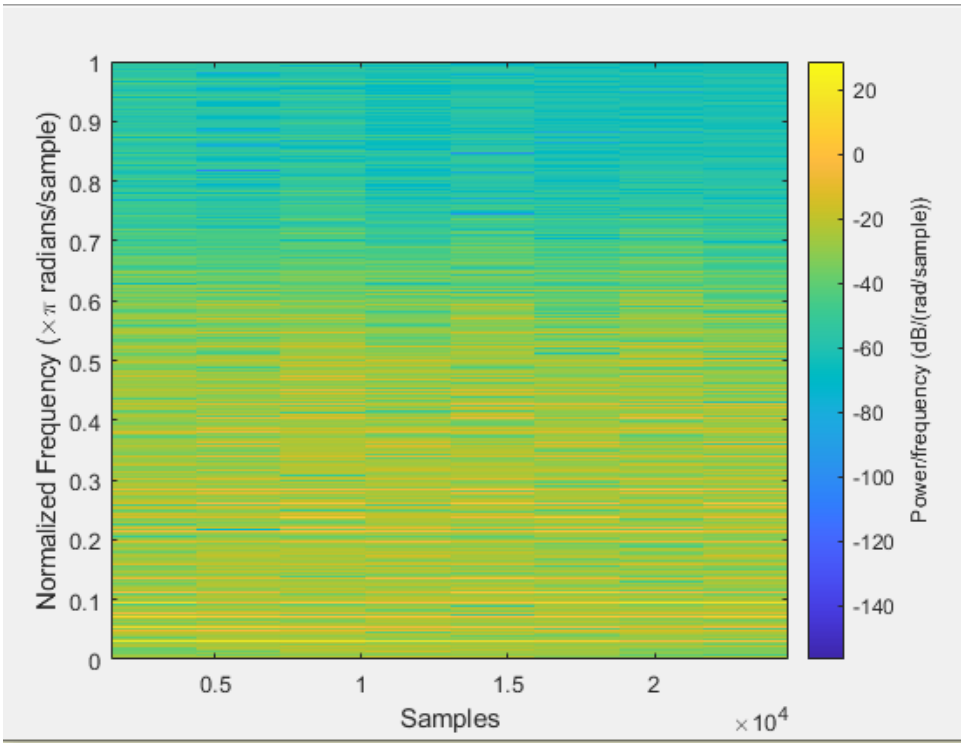

**Kuva 3:** Tremolo-efekti siniaallolle aikatasossa

Ripple-efektin toteutus tapahtui muokkaamalla signaalia suodattavan butterworth-suodattimen rajataajuutta tietyin näytevälein matalataajuisen oskillaattorin mukaisesti. Meillä näyteväliksi valikoitui 100 näytettä. Rajataajuutta kerrottiin sen mukaan mikä kaavan 3 kuvaillun matalataajuisen oskillaattorin *l(t)* arvo oli kyseisellä näytteellä (meillä joka sadas näyte):

$$l(t) = 1 + a\sin(2\pi f t),\ \ t = 0, N, 2N, \ldots, \tag{3}$$

jossa $N$ on valittu näyteväli ja $a$ on skaalauskerroin rajataajuuden vaihtelulle säädetyn rajataajuuden ympärillä.

Pelkästään näin tehtynä, ääneen saattoi ilmestyä rosoisuutta johtuen jyrkästä amplitudikäyttäytymisestä näytevälillä, mutta tätä ei-haluttua efektiä saatiin vähennettyä ikkunoimalla näytevälit Hamming-ikkunalla. Kuvassa 4 ripple-efektin vaikutus taajuustasossa, josta havainnollistuu hyvin korkeampien taajuuksien ajoittainen häviäminen ja ilmestyminen takaisin (näytevälein).




**Kuva 4:** Spektrogrammi-kuvaaja ripple-efektin vaikutuksesta sahalaita-aaltoon

#### 2.2.4 Suodatus ja vahvistus

Lisäsimme synteesiimme vielä lopuksi signaalin suodatuksen butterworth suodattimella ja amplitudivahvistuksen käyttäjän säätämillä parametreilla (suodattimen rajataajuus ja vahvistustaso).

# 3 Toteutus

## 3.1 Synteesin toteutus Matlabissa

Synteesin kannalta olennainen toteutus tapahtuu koodissamme CreateSamples() -funktion sisällä, jolle näkyy globaalina muuttujana struct-rakenne nimeltä Synth. Synth:in sisäisten muuttujien perusteella CreateSamples() saa parametrinsa eri synteesivaiheisiin. Kyseistä funktiota kutsutaan heti sovelluksen käynnistyttyä Synth:in default-arvoilla ja aina kun käyttäjä tekee muutoksia Synth:in sisäisiin muuttujiin käyttöliittymän kautta.

CreateSamples() -funktion toteutus rakentuu olennaisesti switch-case ehtolauserakenteen varaan. Valinnat oskillaattorista, amplitudiprofiilista ja LFO-efektistä tulevat Synth:in muuttujien arvojen perusteella, joita käyttäjä voi siis muokata käyttöliittymän kautta. Tästä rakenteesta esimerkkinä LFO-efektin valinta funktion sisällä pseudokoodina:

```
% s : ADSR-moduloitu oskillaattori
% a: oskillaattorin vaihe
% f_LFO: Moduloivan matalataajuisen oskillaattorin taajuus (1-20 Hz)
```

```
% f_cut:  butterworth suodattimen rajataajuus
% Synth.gain: Synth-structuren sisäinen muuttuja, määrää vahvistuksen
% Synth.Effect: Synth-structuren sisäinen muuttuja, määrää efektin tyypin
% LFO_A: matalataajuisen oskillaattorin amplitudi, määrää efektin voimakkuuden
switch Synth.Effect
        case 1  % tremolo
                s = virtualSynth(s, a, f_LFO , f_cut, Synth.gain, 1, LFO_A);
        case 2          % ripple
             s = virtualSynth(s, a, f_LFO , f_cut, Synth.gain, 2, LFO_A);
        case _  % ei efektiä
end

function virtualSynth(osc, a, f_LFO , f_cut, gain, LFO_type)
switch LFO_type
        case 1  % tremolo
                s = butterworth(osc, f_cut);
                s = s * (1 + LFO_A * sin(f_LFO * a))
                return s
        case 2          % ripple
                lfo = (1 + LFO_A * sin(f_LFO * a))
                s = zeros(length(a))
windowsize = 100
while i < 0
        windowsize = 100 if length(a) - i > 0 else  length(a) - i
        s(i:i+windowsize) = butterworth(osc, fcut * lfo(i))
        i = i + windowsize
      return s
case _
      return osc
```

## 3.2 Käyttöliittymä

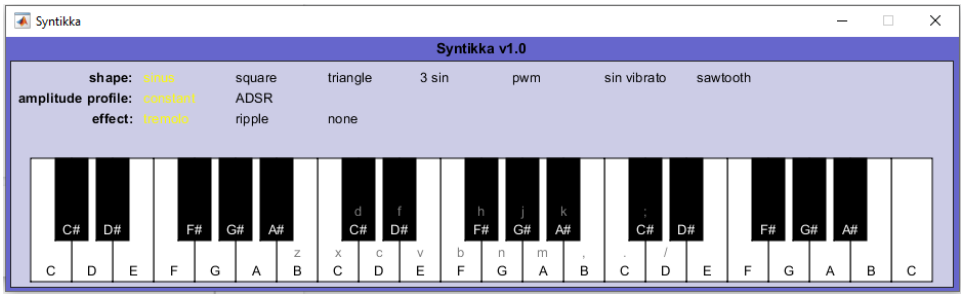
### 3.2.1 Komponentit

Käyttöliittymän toteutimme Matlabin erinäisin käyttölittymäkomponentein. Käytimme mm. figure-, uifigure-, uislider-, text- ja uiswitch-komponentteja, joilla saimme luotua ikkunan, ja säätimiä ja kuvaajia siihen.

Käytimme käyttöliittymän pohjana valmista Matlab-koodia (Kalarus, 2020), josta saimme yksinkertaisen kosketinsoitin-komponentin, jolla oli hyvä testata ohjelmaa ilman liitettyä MIDI-kosketinsoitinta. Lisäsimme tähän ikkunaan myös päävalinnat oskillaattorista, amplitudiprofiilista ja efektistä.

Pohjakoodissa oli myös valmiiksi toteutettuna reagointi kosketinten klikkailuun hiirellä, jota myös onnistuimme hyödyntämään fyysisen MIDI-laitteen syötteen käsittelyssä.

Kuvassa 5 on kuvattuna tämä virtuaalisen kosketinsoittimen sisältävä ikkuna, jonka voisi ajatella olevan ohjelmamme pääikkuna.

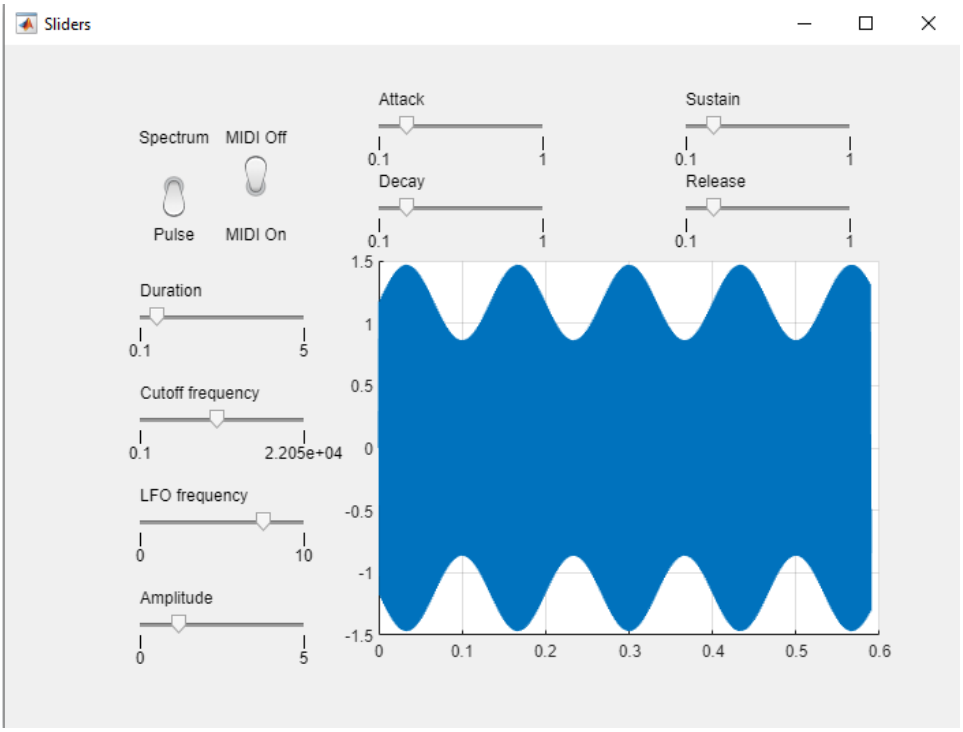



**Kuva 5:** Virtuaalinen kosketinsoitin ja pääasetukset (oskillaattorityyppi, amplitudiprofiili ja efekti)

Meidän itse alusta loppuun toteuttamamme säätöikkuna (kuva 6) avautuu demossa käynnistyessä:

- Spectrum-pulse vaihtimesta säätää kuvaajan tyyppiä aikatasosta taajuustasoon
- MIDI-vaihtimen On-asento lähtee etsimään liitettyä MIDI-laitetta (ja löytyessä lukee sen MIDI-viestejä)
- Attack-, Decay-, Sustain- ja Release-säätimet määrittävät näiden eri vaiheiden pituudet ADSR-verhossa
- Duration-säädin arvo toimii kertoimena koko signaalin kestolle (joka siis määräytyy ADSR-verhon kestosta)
- Cutoff frequency-säädin määrittää butterworth-suodattimen rajataajuuden
- LFO frequency-säädin säätää efektien matalataajuisen oskillaattorin taajuuden
- Amplitude-säädin määrittää vahvistuksen

Kehitysideana säädinikkunan voisi laittaa avautumaan esimerkiksi erillisestä valikosta.




**Kuva 6:** Säädinikkuna

### 3.2.2 Käyttäjän syötteeseen reagointi

Toteutimme käyttäjän syötteeseen reagoinnin säädinten osalta yksinkertaisesti määrittämällä arvon muuttamisesta aktivoituvan funktion säädintä luodessa. Tarkempi toiminnan määrittely Matlabin sisäisestä käyttöliittymäkomponenttien dokumentoinnista. Näiden aktivoituvien funktioiden toiminta voidaan kuvailla seuraavasti: kun arvo muuttuu, tallenna arvo Synth-struct-rakenteeseen ja kutsu näytteet uudelleen luovaa CreateSamples()-funktiota.

Monimutkaisempaa toteutusta vaati liitetyn MIDI-laitteen syötteen lukeminen, sillä sen tuli tapahtua asynkronisesti, niin että käyttäjä pystyi muuttamaan asetuksia samalla. Toteutimme tämän lopulta Matlabista löytyvän timer-toiminnallisuuden kautta. Timer-rakenteelle on mahdollista määrittää funktio, jota se kutsuu tietyin välein. Me toteutimme saapuneet MIDI-viestit käsittelevän MidiData()-funktion tätä varten.

# 4 Tulokset

## 4.1 Käyttöliittymä

Käyttöliittymä tarjoaa hyvin ne ominaisuudet, jotka syntetisaattoriin haluammekin. Säädinikkunalla saa sujuvasti muokattua ääntä, sekä näkee nykyisen äänen kuvaajan aika- ja taajuustasossa. Ikkuna, jossa ovat pääasetukset ja virtuaaliset koskettimet toimivat myös sujuvasti ja ovat suhteellisen responsiivisia. Pääasetukset vaihtuvat sujuvasti niitä klikkaamalla. Ainoa poikkeus on ripple efektin klikkaaminen, joka aiheuttaa hieman hitautta ohjelmalle. Tästä kerrotaan lisää efektit-osiossa. Virtuaaliset koskettimet toimivat myös

sujuvasti ja halutulla tavalla.

Hitautta käyttöliittymälle myös aiheuttaa hieman midi-koskettimien käyttöönotto. Tällöin ohjelma toimii hieman hitaammin, sillä sen pitää tietyin väliajoin kutsua MIDI-viestit lukevaa funktiota, ja tämä saattaa mennä funktiokutsuissa kuvaajan päivittämisen ja sliderien arvojen säätämisen edelle. Tällöin käyttöliittymän käyttöön voi tulla satunnaisesti pientä viivettä, mutta ei kuitenkaan havaittavasti.

## 4.2 Efektit

### 4.2.1 Tremolo

Tremolo efekti onnistui kokonaisuudessaan hyvin. Saimme sen avulla luotua hyvän kuuloisia ääniä. LFO:n taajuuden säätämisellä on selkeä vaikutus ääneen. LFO toi ääneen mukavaa aaltoisuutta, jota toivoimmekin tremolo efektiltä.

### 4.2.2 Ripple

Ripple efektin toteutus onnistui ihan hyvin. Sen toteutus matlabissa on vain hieman hidas ja se näkyy myös syntetisaattorin toiminnassa. Tämä johtuu siitä, kun ohjelma tekee butter- ja filter-komennot useaan kertaan, sillä haluamme moduloida suodattimen rajataajuutta LFO:lla tietyin näytevälein eri taajuuksille. Efekti kuitenkin luo hyvillä säädöillä mukavan kuuloista ääntä ja efektin vaikutus kuuluu hyvin äänessä. Taajuustason kuvaajasta efektin vaikutus näkyy myös hyvin kun taajuudet ilmestyvät ja katoavat tietyn näytteiden välein.

## 4.3 MIDI-toiminnalisuus

Saimme liitettyä syntetisaattoriimme MIDI-koskettimet. Koskettimilla sai soitettua syntetisaattoria suhteellisen hyvin, mutta sen toteutus ei kuitenkaan ollut paras mahdollinen, MIDI-toiminnallisuus lukee vain NoteOn-viestejä, jolloin äänen pituus päätetään duration-sliderilla. Tämä tekee toteutuksesta musikaalisuuden ja responsiivisuuden kannalta kankean. Äänen analysoinnin ja tarkkaan säätämiseen slider-säätö ovat toisaalta kätevä.

Lisäksi kuvaajien piirtäminen ja muu prosessointi, muun muassa MIDI-viestejä kuunteleva timer, tekee syntetisaattorista aika raskaan ajaa, mutta nämä antavat lisää käyttö- ja informaatiotoiminnallisuutta. Yksi kehitysmahdollisuus olisi optimoida raskaimmat toiminnot vähemmän prosessointitehoa vieviksi. Toisaalta, vaikka MIDI-toiminnallisuudessa olisi kehittämisen varaa, oli se meidän syntetisaattoritoteutukseen tarpeeksi hyvä ja sillä sai soiteltua syntetisaattoria.

# 5 Johtopäätökset

Kokonaisuudessaan syntetisaattori onnistui hyvin ja suunnilleen suunnitelmien mukaisesti. Ainoa muutos oli, että emme tehneet lopullista toteutusta JUCE:lla vaan Matlabilla, sillä saimme Matlabilla vaatimuksia nähden paremman toteutuksen tehtyä. Kaikki muut suunnitelmassa olleet vaatimukset saatiin toteutettua.

Syntetisaattori tuotti hyvin ääntä ja oli suhteellisen responsiivinen. Säätimet ja pääasetuksien säätäminen toimivat hyvin ja tarjoavat äänelle hyvän säädettävyyden. Kuvaaja myös havainnollisti hyvin kuinka ääni muuttuu sitä säädettäessä. Taajuustason kuvaajasta olisi ehkä voinut tehdä pikemminkin jatkuvan lyhyemmillä ikkunoilla, jolloin esimerkiksi ripple-efekti olisi havainnollistettu paremmin.

Vaikka syntetisaattori toimii suunnitelmien mukaisesti, kehitettävää jäi. Ripple efektin toteutus tulisi tehdä tehokkaammin, sillä nykyinen toteutus hidastaa ohjelman toimintaa. Myös MIDI-koskettimien liittämisessä on kehitettävää, sillä tällä hetkellä äänien kesto ei määräydy siitä, kuinka kauan kosketinta painetaan, vaan se riippuu ADSR ja Duration säädöistä. Kuitenkin kokonaisuudessaan saimme tehtyä toimivan syntetisaattorin johon olemme tyytyväisiä.

Opimme projektin aikana syntetisaattorin rakenteesta, muokkausmahdollisuuksista, äänisynteesistä sekä eri audio-efektien tuottamisesta. Käyttöliittymäohjelmointi ja käyttäjän syötteen hallinta (sekä JUCE:ssa että Matlabissa) oli myös osa projektia. Syntetisaattorin kanssa pääsimme onnistuneesti kasarifiilikseen, joka oli projektin yksi päätavoite.

# 6 Lähteet


Kalarus M., (2020), *Synthesizer*,
MATLAB Central File Exchange, katsottu 24.11.2020,
URL:https://se.mathworks.com/matlabcentral/fileexchange/69051-synthesizer

fedden, (2020),
*Simple Juce & Maximilian VSTs*, Github, katsottu 22.11.2020,
URL:https://github.com/fedden/juceSynths

TheAudioProgrammer, (2020).
*juceSynthFramework*, Github, katsottu 22.11.2020,
URL:https://github.com/TheAudioProgrammer/juceSynthFramework

# 7 Liitteet

Palautuskansio *syntetisaattori_palautus* sisältää:

*Virtuaalianaloginensyntetisaattori*: lopullisen demon lähdekoodi

*JUCE_testitoteutus*: testitoteutuksen lähdekoodi ja liitännäinen

*syntetisaattori_group1_raportti.pdf*: projektiraportti

*syntetisaattori_demoslides.pdf*: esityksen diat