Jori Hänninen

Sekvensseri modulaariselle syntetisaattorille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

21.5.2021

Tiivistelmä

Tekijä:	Jori Hänninen
Otsikko:	Sekvensseri modulaariselle syntetisaattorille
Sivumäärä:	27 sivua
Aika:	21.5.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Keijo Länsikunnas

Tätä insinöörityötä varten on rakennettu modulaarisille syntetisaattoreille tarkoitettu sekvensserimoduuli Arduino Mega -mikrokontrolleria käyttäen.

Insinöörityössä käydään läpi komponentit, jotka on valittu sekvensserin rakentamiseen, komponenttien kytkentäkaaviot sekä niiden haluttu toiminta.

Komponenttien ja kytkentäkaavioiden lisäksi insinöörityössä kuvataan ohjelmisto, joka on toteutettu Arduino-ohjelmointikielellä.

Insinöörityön päätteeksi on hieman pohdintaa siitä, miten hyvin Arduino Mega mikrokontrolleri soveltui projektiin.

Insinöörityön tuloksena on suunniteltu ja rakennettu CV-Signaalia lähettävä kahdeksankanavainen sekvensserimoduuli modulaarisille syntetisaattoreille.

Avainsanat: Arduino, syntetisaattori, sekvensseri, modulaarinen, mikrokontrolleri

Abstract

Author:	Jori Hänninen
Title:	Sequencer Module for Modular Synthesizer
Number of Pages:	27 pages
Date:	21 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Engineering
Supervisors:	Keijo Länsikunnas, Senior Lecturer

Within the study sequencer module for a modular synthesizer was built using the Arduino Mega microcontroller.

This thesis covers the components that were chosen for the sequencer, the circuit diagrams for the components and the desired function of the sequencer.

In addition to information on the components and circuit diagrams the thesis explains the software developed using the Arduino programming language. Furthermore, the thesis discusses the suitability of the Arduino Mega microcontroller to the project.

As a result of the study, a sequencer module that sends CV-signal for modular synthesizers was designed and built.

Keywords: Arduino, Sequencer, Modular, Synthesizer, Microcontroller

Sisällys

Lyhenteet

ORM: *Object-relational mapping*. Ohjelmiston oliomallin mukaisen esityksen kuvaaminen relaatiotietokantamallin mukaiseksi esitykseksi ja kääntäen.

CV: *Control Voltage.* Jännite, jota käytetään kontrolloimaan moduulien parametrejä.

VCO: *Voltage Controlled Oscillator.* Oskillaattori, jonka taajuutta voi kontrolloida jännitteen avulla.

VCF: *Voltage Controlled Filter.* Suodatin, jonka leikkauspistettä voi kontrolloida jännitteen avulla.

VCA: *Voltage Controlled Amplifier.* Vahvistin, jonka äänenvoimakkuutta voi muuttaa jännitteen avulla.

PWM: *Pulse-Width Modulation.* Aallon pulssisuhteen modulaatio.

ADSR: *Attack, Decay, Sustain, Release.* Moduuli, joka generoi verhokäyrän.

DAC: *Digital to Analog Converter.* Komponentti, jonka avulla voidaan muuttaa digitaalista signaalia analogiseksi.

1 Johdanto

Modulaariset syntetisaattorit ovat soittimia, jotka koostuvat erillisistä moduuleista, joilla on oma tehtävänsä. Näitä moduuleita yhdistetään toisiinsa eri menetelmillä. Nykyään yleisin standardi on saksalaisen Doepfer-yhtiön luoma Eurorack-standardi. Eurorack-standardin moduuleissa audiosignaali käyttää +/- 5 voltin jännitettä ja unipolaareissa CV-signaaleissa käytetään 0-8 V:n jännitettä. Jos kyseessä on bipolaarinen CV-Signaali, jännite on yleensä -2,5 V:n ja +2,5 V:n välillä. Lisäksi trigger-, clock- ja gate-signaalit käyttävät yleensä +5 V:n jännitettä. [1.]

Arduino-mikrokontrollerin lähettämä ja maksimaalinen vastaanotettava jännite on +5 V:n, joten se soveltuu erinomaisesti vastaanottamaan ja tuottamaan trigger-, clock- ja gate-signaaleja. Arduinon PWM:n ansiosta se soveltuu myös kohtalaisesti CV-signaalien tuottamiseen välillä 0 V ja +5 V.

Ensimmäisessä luvussa käydään läpi yleisesti modulaaristen syntetisaattoreiden historiaa ja toimintaa. Toisessa luvussa tarkastellaan hieman Arduino-mikrokontrolleria ja sen perustoimintoja. Kolmannessa luvussa määritellään sekvensseriin halutut toiminnot. Sen jälkeen esitellään tarvittavat komponentit ja niiden kytkentäkaaviot. Viidennessä luvussa käsitellään, kuinka sekvensserin ohjelmisto on toteutettu. Viimeisessä luvussa pohditaan, kuinka hyvin Arduino-mikrokontrolleri soveltui kyseiseen projektiin, mitä olisi voitu tehdä toisin ja miksi työssä päädyttiin niihin ratkaisuihin, joilla projekti toteutettiin.

2 Yleistä modulaarisista syntetisaattoreista

2.1 Historia

Saksalainen insinööri Harad Boden kehitti ensimmäisen modulaarisen syntetisaattorin jo 50- ja 60-lukujen taitteessa. 60-luvulla yhdysvaltalainen Robert Moog otti paljon vaikutteita Harad Bodenin kehittämästä soittimesta ja rakensi Moog-syntetisaattorin, mikä oli yksi ensimmäisistä kaupallisista modulaarisista syntetisaattoreista. Moog-syntetisaattori koostuu itsenäisistä moduuleista, joita kytkettiin toisiinsa käyttäen 6,3 mm:n audiokaapeleita. Myöhemmin 70-luvulla muutkin valmistajat alkoivat valmistaa omia modulaarisia syntetisaattoreita. Modulaariset syntetisaattorit menettivät kuitenkin huomattavasti suosiotaan 80-luvun aikana, kun markkinoille alkoi tulla paljon halvempia ja pienikokoisempia digitaalisia syntetisaattoreita. [2.]

90-luvulla saksalainen Dieter Doepfer kehittää pienempikokoisia Doepfer A-100 -modulaarisen syntetisaattorin. Doepfer A-100 -syntetisaattorin myötä syntyi Eurorack-standardi. Ennen Eurorack-standardia käytännössä jokaisella valmistajalla oli omat standardissa, joten eri syntetisaattoreiden moduuleita ei voitu yhdistää toisiinsa. Nykyään Eurorack-standardi on yleisin käytössä oleva standardi, ja monet eri valmistajat tekevät Eurorack-standardin periaatteella toimivia moduuleita, joten oman modulaarisen syntetisaattorin kokoaminen on nykyään huomattavasti halvempaa ja helpompaa, koska markkinoilla on paljon enemmän eri valmistajien moduuleita.

2.2 Modulaaristen syntetisaattoreiden toiminta

Modulaarisen syntetisaattorin äänilähteenä käytetään yleensä VCO-moduuleita. Tavallisia VCO-moduulien tuottamia aaltoja ovat siniaalto, saha-aalto, kolmioaalto ja kanttiaalto. Aallon taajuutta voidaan muokata CV-signaalia käyttäen. Yleensä VCO-moduulit tuottavat korkeamman taajuuden omaavaa altoa, mitä suurempi CV-signaalin jännite sille syötetään. Koska pelkkä tavallinen aaltofunktio ei välttämättä ole kovin mielenkiintoisen kuuloinen, sitä

yleensä suodatetaan käyttäen VCF-moduulia. VCF-moduuleilla voidaan suodattaa VCO-moduulin tuottamasta signaalista ei-toivottuja osia. VCF-moduulin jälkeen äänisignaali jatkaa matkaa VCA-moduuliin, millä voidaan säätää äänenvoimakkuutta. VCA-moduulia kontrolloidaan usein ADSR-moduuleilla.

Kun ADSR-moduulille lähetetään jännite, se tuottaa verhokäyrän, joka koostuu neljästä osasta. Attack määrittää ajan, kuinka kauan jännitteellä kestää nousta maksimiin. Decay määrittää ajan, kuinka kauan jännitteellä kestää laskeutua sustainin määrittämään arvoon. Jännitettä pidetään sustainin määrittämässä arvossa, kunnes ADSR-moduuliin ei lähetetä enää CV-signaalia. Release puolestaan määrittää ajan, kauanko jännitteellä kestää palautua takaisin arvoon 0 V, kun moduuliin ei enää lähetetä CV-signaalia.

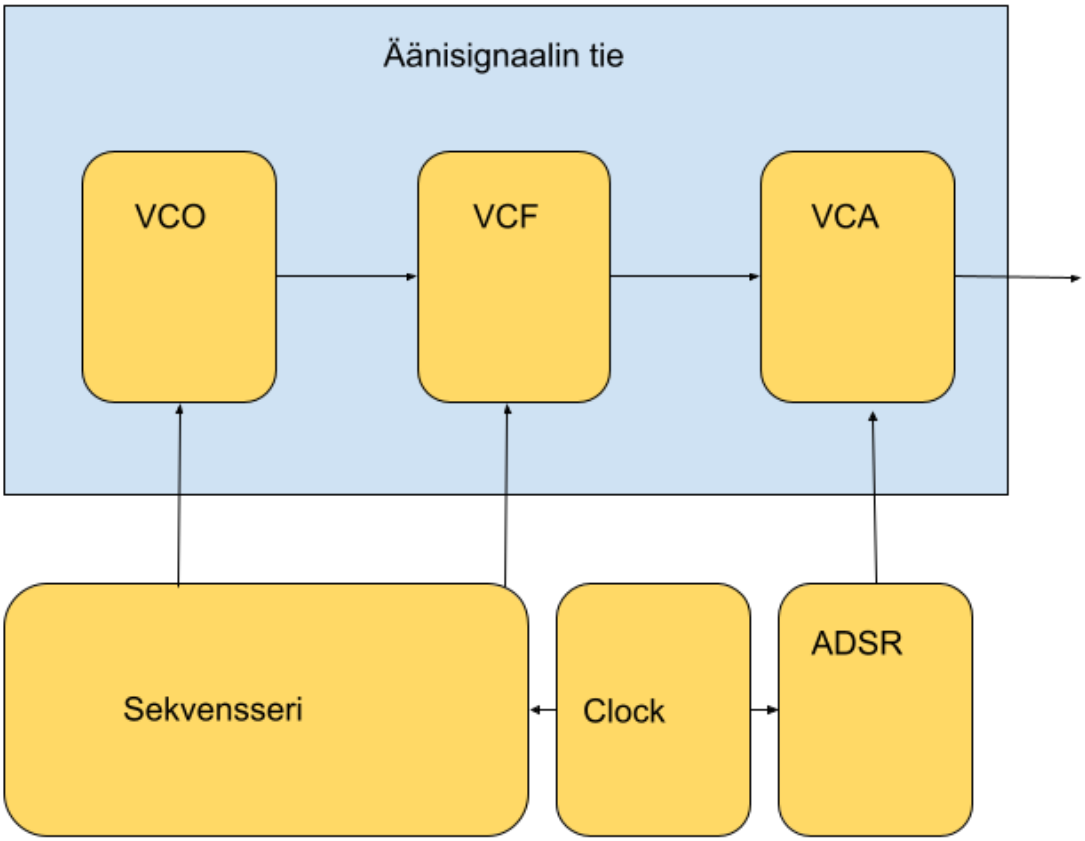
Nämä neljä moduulia muodostavat perustan sille, kuinka modulaarisilla syntetisaattoreilla tuotetaan ääntä. Lisäksi on lukuisia erilaisia moduuleita, joilla voidaan tuottaa ääntä, tuottaa CV-signaalia kontrolloimaan toisia moduuleja tai esimerkiksi muokata ääntä eri efektein.

Tässä insinöörityössä tehtyä sekvensseriä voidaan käyttää esimerkiksi melodioiden luomiseen. Kuvassa 1 on esimerkkikokoonpano, missä sekvensseriä voidaan käyttää VCO-moduulin ja VCF-moduulin modulaatioon. VCO-moduuli tuottaa äänisignaalin, mikä siirtyy VCF-moduulin kautta VCA-moduuliin ja sen jälkeen kuulokkeisiin tai kaiuttimiin. Clock-moduulista lähetetään jännitepulsseja sekvensserille ja ADSR-moduulille. Clock-moduulin kellotaajuudella määritellään tempo, millä sekvensserin tuottama melodia halutaan soittaa. Jännitepulssien saatuaan ADSR-moduuli tuottaa verhokäyrän ja lähettää sen VCA-moduulille, mikä vahvistaa äänisignaalia verhokäyrän mukaisesti.

Tässä esimerkissä yksi sekvensserin kanavista lähettää potentiometreiltä tallennettua CV-signaalia VCO-moduuliin. VCO-moduulin tuottaman äänenkorkeus määrittyy CV-signaalin jännitteen mukaan. Kun sekvensseri saa

jännitepulssin Clock-moduulilta, se siirtyy lähettämään seuraavaa muistista luettua arvoa, joilloin VCO-moduuli tuottaa uuden nuotin.

Esimerkkikokoonpanossa yhtä sekvensserin muista kanavista on käytetty ohjaamaan VCF-moduulin leikkauspistettä. Kun VCF-moduulin vastaanottama CV-signaali muuttuu, myös VCF-moduulin leikkauspiste muuttuu. Kun VCF-moduulin leikkauspistettä muutetaan, äänenväri muuttuu ja melodiasta saadaan mielenkiintoisempi.



Kuva 1. Esimerkkikokoonpano

Esimerkkikokoonpanossa esitelty systeemi toimii itsenäisesti siten, että se jatkaa määritellyn melodian toistamista ilman soittajan ja soittimen välistä vuorovaikutusta. Esimerkkikokoonpano siis toistaa määriteltyä melodiaa kunnes virta katkaistaan. Tällainen tyyli soittaa modulaarista syntetisaattoria on kehittynyt Yhdysvaltojen länsirannikolla ja sen takia sitä kutsutaan West Coast -

tyyliksi. West Coast -tyylistä poikkeava East Coast -tyyli on vastaavasti kehittynyt Yhdysvaltojen itärannikolla. East Coast -tyylissä melodioita generoivat moduulit, kuten sekvensseri, on yleensä korvattu koskettimilla. Tällaisen kokoonpanon soittaminen vaatii yleensä jatkuvaa soittajan ja soittimen välistä vuorovaikutusta ja on lähempänä “tavallisen” syntetisaattorin tai kosketinsoittimen soittamista. [3.]

3 Arduino-mikrokontrollerit

3.1 Yleistä tietoa

Arduino-mikrokontrollerit ovat avoimeen lähdekoodiin perustuvia yhden piirilevyn mikrokontrollereita. Samalla piirilevyllä on kaikki tarvittava mikrokontrollerin ohjelmointiin ja käyttöön. Piirilevyllä on myös valmiina liittimet, joiden avulla mikrokontrollerin pinneihin on helppo ja nopea tehdä kytkentöjä.

Arduino-mikrokontrollereita ohjelmoidaan käyttäen C- ja C++-ohjelmointikieliin pohjautuvaa Arduino-ohjelmointikieltä. Arduino-ohjelmointikielessä on määriteltynä eri funktioita, joiden avulla mikrokontrollerin ohjelmointi on vaivatonta ilman syvällisempää osaamista varsinaisesta mikrokontrollerin toiminnasta.

Tässä insinöörityössä on käytetty Arduino Mega 2560 -mikrokontrolleria, mikä perustuu ATmega2560-mikrokontrolleriin. Arduino Mega2560 -mikrokontrollerissa on 54 digitaalista I/O-pinniä, joista 15:tä voidaan käyttää PWM-ulostuloina: 16 analogista sisääntuloa ja neljä UART sarjaporttia.

3.2 Perustoiminnot

Arduinon digitaaliset I/O-pinnit voidaan määritellä joko sisään- tai ulostuloiksi. Sisääntuloksi määritellyltä pinniltä voidaan lukea arvot käyttäen *digitalRead()*-funktiota. Funktiolle annetaan parametrina Arduinon digitaalista pinniä vastaava arvo, jonka jälkeen funktio palauttaa arvoksi nollan, jos pinniin ei sillä hetkellä

ole kytketty jännitettä, tai ykkösen, jos pinniin on sillä hetkellä kytketty tarpeeksi suuri jännite. *DigitalWrite()*-funktiolla puolestaan voidaan lähettää pinniin joko +5 V:n tai 0 V:n jännite. Funktiolle annetaan parametrina pinnin arvo, jonka jännitettä halutaan muuttaa sekä arvo yksi tai nolla. Jos arvoksi annetaan yksi, pinniin lähetetään +5 V:n jännite. Jos arvoksi annetaan nolla, pinni kytkeytyy maahan ja jännitteen arvoksi tulee 0 V.

Arduinon PWM-ulostuloina käytettävien pinnien pulssisuhde voidaan määritellä *analogWrite()*-funktion avulla. Funktio ottaa vastaan parametreina pinnin arvon sekä kokonaislukuarvon nollan 255 väliltä, mikä määrittää pulssisuhteen arvon nollan ja 100 prosentin väliltä.

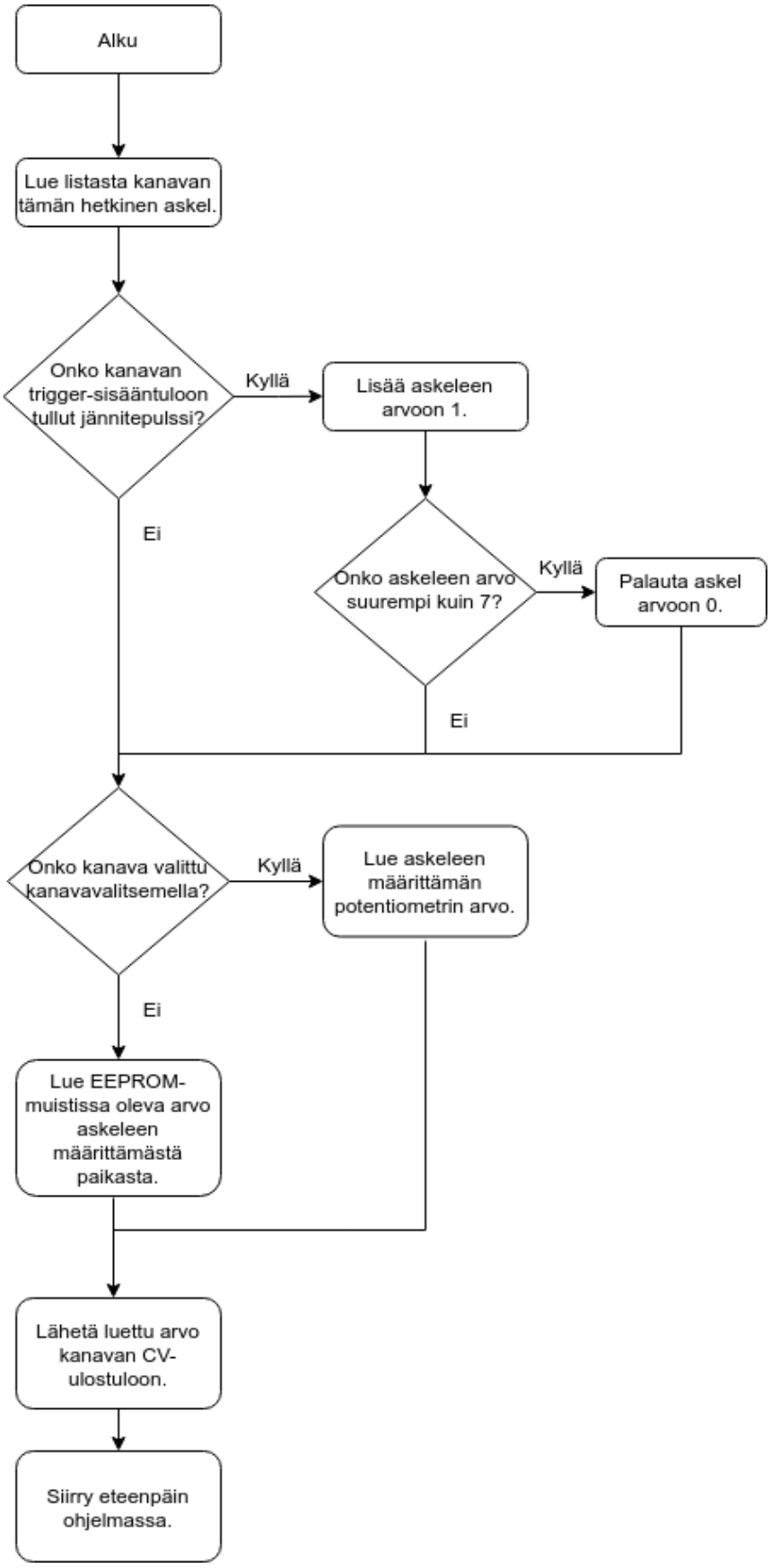
Esimerkiksi potentiometrien lukemiseen käytetään *analogRead()*-funktiota. Funktio vastaanottaa parametreina arvon pinnistä, jolta arvo halutaan lukea. Funktio palauttaa arvoja nollan ja 1023 väliltä.

4 Toiminta

Sekvensseri koostuu kahdeksasta itsenäisestä kanavasta, joiden jokaisen toiminta on identtinen toisiinsa nähden. Yhden kanavan toimintaa kuvataan vuokaavion avulla kuvassa 2. Jokaisella kanavalla on oma trigger-sisääntulo, CV-ulostulo ja matriisinäytöltä varattu rivi, mikä kuvaa, missä vaiheessa sekvenssiä kyseinen kanava on. Kanavaa voidaan vaihtaa kolmella vipukytkimellä, joista jokainen vastaa yhtä bittiä. Näin kolmella vipukytkimellä saadaan toteutettua binäärisesti kahdeksan eri lukua.

Aluksi sekvensseri lukee potentiometrin arvot. Kun sekvenssi on ensimmäisessä askeleessa, kanavavalitsimella valitun kanavan CV-ulostulo lähettää ensimmäiseltä potentiometriltä luetun arvon mukaan jännitettä nolla ja viiden voltin väliltä niin kauan, kunnes kyseisen kanavan trigger-sisääntuloon tulee uusi jännitepulssi. Tämän jännitepulssin seurauksena kanava siirtyy askeleen eteenpäin, ja valitun kanavan CV-ulostulo lähettää toisen

potentiometrin arvon mukaista jännitettä. Vain kanavavalitsimilla valittu kanava toistetaan potentiometrien sen hetkisten arvojen perusteella.

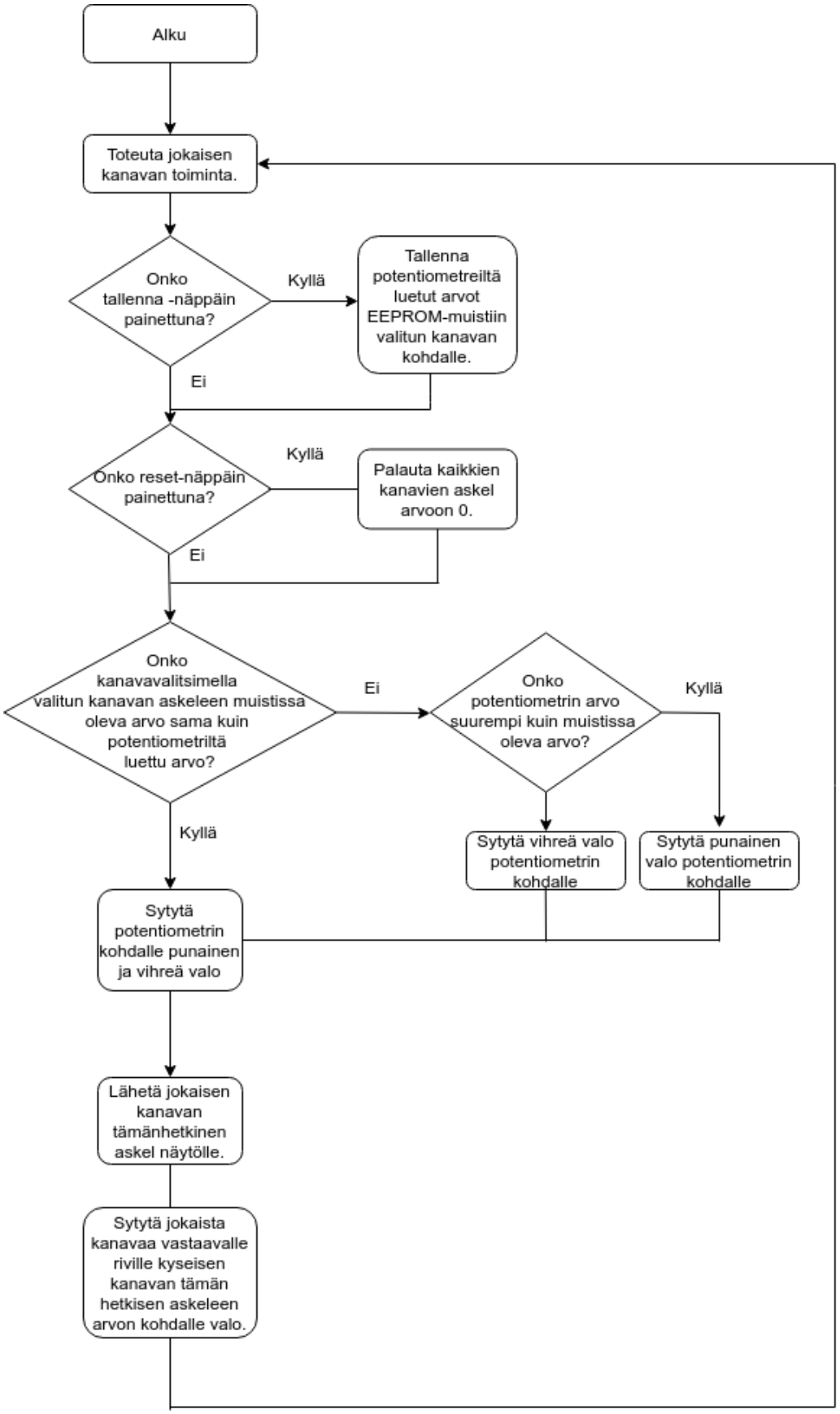


Kuva 2. Vuokaavio yhden kanavan toiminnasta.

Kun kanavalle on määritelty haluttu sekvenssi, se voidaan tallentaa tallennus-näppäintä painamalla. Tallennus tapahtuu Arduinon EEPROM-muistiin, jotta sekvenssi säilyy, vaikka Arduino sammutettaisiin. Tämän jälkeen voidaan valita toinen kanava, määritellä haluttu sekvenssi kyseiselle kanavalle ja tallentaa se muistiin. Sekvensseri toistaa muiden kuin valitun kanavan arvoja muistista.

Koska kaikilla kanavilla on yhteiset potentiometrit, joiden mukaan sekvensserin muistiin tallennetaan arvot, voi aikaisemmin tallennetun sekvenssin hienosäätö olla vaivalloista, koska tallennuksen jälkeen potentiometrien asentoa on muutettu. Tästä syystä jokaisella potentiometrillä on kaksi LED-valoa, joista toinen on potentiometrin oikealla ja toinen vasemmalla puolella. Nämä LED-valot kuvastavat sitä, mihin suuntaan potentiometrio on käännettävä, jotta potentiometrille saadaan asetettua sama arvo, mikä on jo muistissa.

Sekvensserissä on myös Reset-näppäin. Tätä näppäintä painamalla kaikki kanavat palautuvat ensimmäiseen askeleeseen.



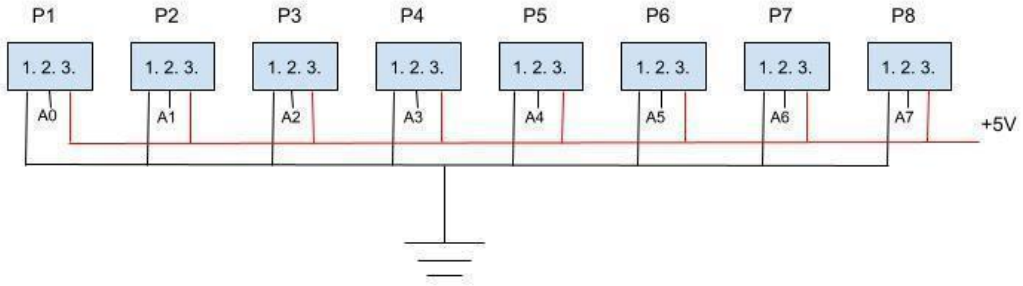
Kuva 3. Vuokaavio sekvensserin toiminnasta.

5 Komponentit ja kytkentäkaaviot

5.1 Potentiometrit

Tässä projektissa on käytetty 100 kΩ:n potentiometrejä. Potentiometrin resistanssilla ei ole juurikaan merkitystä sekvensserin toiminnan kannalta, koska resistanssista riippumatta, potentiometri palauttaa Arduinolle jännitteen 0 V ja +5 V väliltä.

Kuvassa 1 esitetään potentiometrien kytkentäkaavio. Kaikkien potentiometrien ensimmäiset terminaalit on kytketty yhteen ja siitä edelleen maahan. Myös kolmannet terminaalit ovat kytketty toisiinsa, mutta ne on kytketty arduinon +5 V:n ulostuloon. Potenttiometrien keskellä olevasta terminaalista saadaan potentiometrin asennosta riippuen 0 V - 5 V jännite. Jokaisen potentiometrin keskimmäinen terminaali on kytkettynä Arduinossa olevaan pinneihin (A0-A7), jotka lukevat analogisia arvoja.



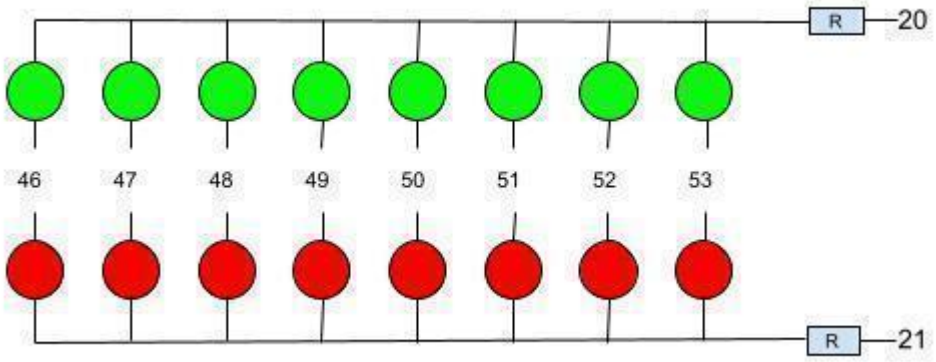
Kuva 2. Potentiometrien kytkentäkaavio

5.2 Potentiometrien valot

Potentiometrien säätämistä helpottavat valot on kytketty 2x8-matriisiin siten, että vihreiden LED:ien katodit on yhdistetty toisiinsa, siitä edelleen yhdistettynä

90 Ω:n vastukseen. Vastuksen toinen pää on kytkettynä Arduinon digitaaliseen pinniin 20. Punaiset LED:t on kytketty toisiinsa vastaavalla tavalla pinniin 21.

LED:ien katodit on kytketty puna-vihreä-pareittain toisiinsa. Jokainen pari on kytkettynä omaan digitaaliseen pinniin kuvan 3 mukaisella tavalla.



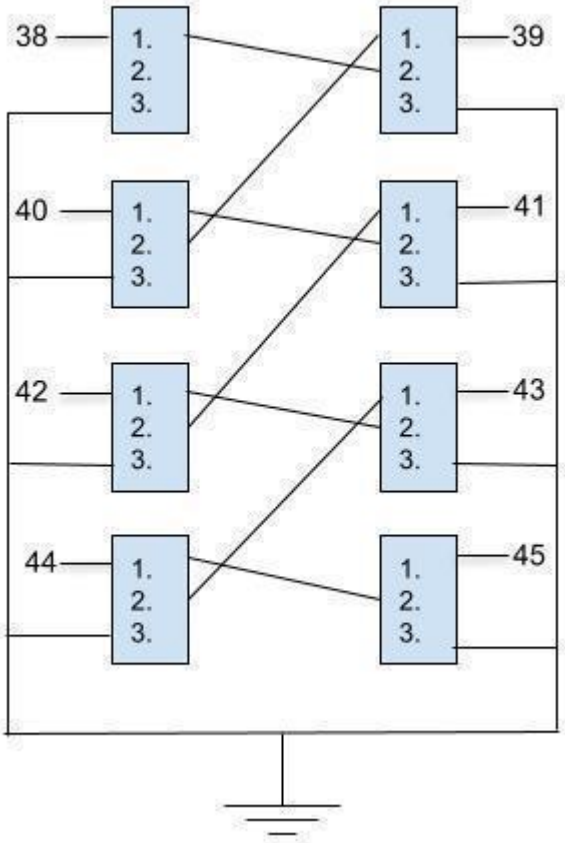
Kuva 3. Potentiometrien valojen kytkentäkaavio.

LED-valojen ohjaus on toteutettu multipleksauksen avulla. Multipleksauksen tarkoituksen on säästää rajallisia resursseja. Tässä kytkennässä multipleksausta on sovellettu siten, että LED-pareista voidaan valita haluttu väri lähettämällä 5 V:n vastajännite vastakkaisen LED-valon anodiin, jolloin virta kulkee vain halutun LED-valon läpi. Multipleksauksen myötä LED-valoja ei voida sytyttää samaan aikaan millä tahansa konfiguraatiolla, LED-valoja sytytetään vain yksi kerrallaan. Ihmissilmä havaitsee kuitenkin LED-valot jatkuvasti palavina, koska LED-valoja päivitetään riittävän usein. Mittauksissa potentiometrien LED-valojen ja matriisinäytön valojen päivitystaajuudeksi saatiin noin 1500 Hz.

5.3 Trigger-sisääntulot

Sisääntulojen liittiminä on käytettä PJ301M-12-tyyppisiä liittimiä. Näihin liittimiin voi kytkeä 3,5 mm:n monoaudiokaapelin. Kun kaapeli on kytketty liittimeen, kaapelin liittimen kärki, jossa varsinainen signaali kulkee, yhdistyy liittimen

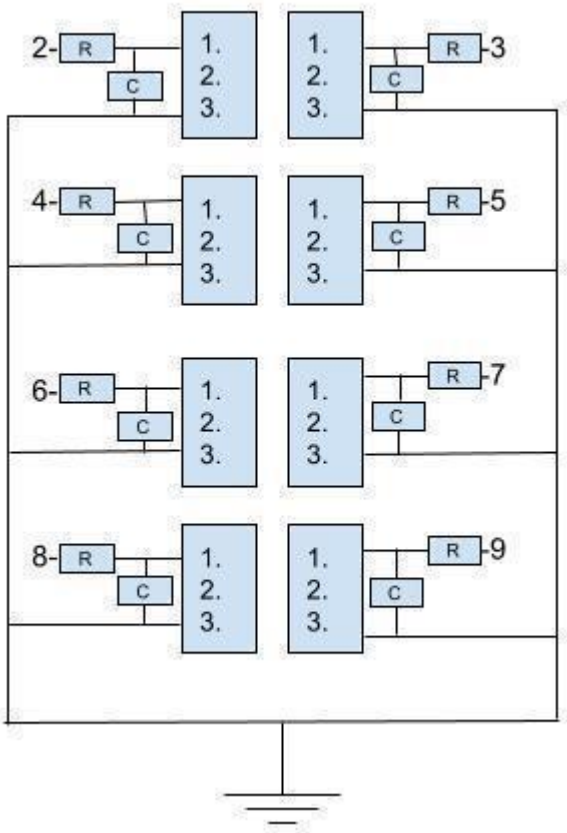
ensimmäiseen terminaaliin, ja kaapelin liittimen “hiha” yhdistyy liittimen kolmanteen terminaaliin. Kuvassa 3 kuvattu toinen terminaali on liittimessä oleva kytkin. Tämä kytkin yhdistyy ensimmäiseen terminaalin, mikäli liittimeen ei ole kytketty johtoa. Sisääntulot on linkitetty toisiinsa tämän kytkimen kautta siten, että ensimmäisen liittimen 1 terminaali on kytkettynä seuraavan liittimen kytkimeen, toisen liittimen 1 terminaali kytketty seuraavan liittimen kytkimeen ja niin edelleen. Tämän ansiosta jos ensimmäiseen liittimeen on kytketty kellosignaali, se yhdistyy myös seuraaviin liittimiin. Tämä ketju katkeaa siitä kohdasta, mihin johto kytketään. Tästä seuraaviin liittimiin kytkeytyy taas tämä signaali.



Kuva 4. Trigger-sisääntulojen kytkentäkaavio

5.4 CV-ulostulot

Ulostuloina on käytetty samoja PJ301M-12-liittimiä kuin trigger-sisääntuloissa. CV-ulostuloissa liittimien kytkintä ei tarvita ollenkaan. Kaikkien liittimien kolmannet terminaalit on kytketty toisiinsa ja siitä edelleen kytkettynä maahan, joka kytkeytyy Arduinon GND-pinniin. Ensimmäiseen terminaaliin on kytketty 910 Ω:n vastus, jota kuvataan kuvassa 5 komponentilla, jotka on nimetty R-kirjaimella. Vastuksen toinen pää on kytkettynä Arduinon digitaaliseen pinniin, joka pystyy tuottamaan PWM signaalia.



Kuva 5. CV-ulostulojen kytkentäkaavio

Terminaalien 1 ja 3 väliin on kytketty kondensaattori 100 nF:n kondensaattori, jota kuvataan kytkentäkaaviossa komponenteilla, jotka on merkitty kirjaimella C.

Näin kytkettynä kondensaattori ja vastus muodostavat yhdessä Low Pass RC -filtterin, joka pehmentää signaalia.

RC-filtterin kondensaattori saa aikaan sen, että Arduinon lähettämä signaali ei ole vain 0 V tai +5 V, vaan kondensaattorin varautumisen ja tyhjentymisen luoman viiveen myötä PWM:llä tuotettu digitaalinen signaali saadaan toimimaan lähes analogisesti, ja jännite voidaan säätää tasaisesti 0 V ja +5 V väliltä.

RC-filtterillä on myös ominainen rajataajuus. Rajataajuudella tarkoitetaan sitä taajuutta, mitä suurempia taajuuksia filtteri suodattaa pois. RC-filtterin rajataajuus määrittyy kondensaattorin kapasitanssin ja vastuksen resistiivisyyden välisestä suhteesta. Kyseinen rajataajuus voidaan laskea kaavan 1 mukaisella tavalla. Kaavassa R-kirjaimella merkitään vastuksen resistiivisyyttä ja C-kirjaimella kondensaattorin kapasitanssia. [4.]

$$RC = \frac{1}{2\pi \times fc} <=> fc = \frac{1}{2\pi \times RC}$$

(1)

Kun komponenttien arvot sijoitetaan kaavaan, RC-filtterin rajataajuus voidaan laskea ja arvoksi saadaan noin 1750 Hz (kaava 2).

$$fc = \frac{1}{2\pi \times 100nF \times 910\Omega} \approx 1750hz$$

(2)

RC-filtteri myös vaimentaa signaalia siten, että RC-filtteristä ulostuleva signaalin vaihteluväli on pienempi. Vaimennuksen myötä signaali muistuttaa

huomattavasti enemmän tasaista analogisignaali, kuin digitaalista PWM-signaalia. Tämä vaimennus on suurin syy sille, minkä takia RC-filtteri tarvitaan jokaiseen CV-ulostuloon. RC-filtterin tuottama vaimennus signaaliin voidaan laskea käyttämällä kaavaa 3. [4.]

$$_{Vaimennus\ (dB)} = -10log(1 + (2\pi \times f_{PWM} \times RC)^2)$$

(3)

Sekvensserin kolmas kanava käyttää Arduinon digitaalista pinniä 4. Arduinon digitaalinen pinni 4 käyttää Arduinon timeria 0, minkä PWM-taajuus on määritelty muista poiketen 62,5 kHz:iin. Muitten CV-ulostuloina käytettyjen pinnien PWM-taajuudeksi on määritetty 31372,55 Hz. Tämän vuoksi RC-filtterin tuottaman vaimennuksen arvoksi saadaan muista CV-ulostuloissa käytettyihin pinneihin verrattuna erisuuruinen vaimennus. Tällä erolla RC-filttereiden vaimennuksessa ei kuitenkaan ole merkittävää vaikutusta, koska molemmissa tapauksissa RC-filtteristä ulostuleva signaali on riittävän tasaista modulaaristen syntetisaattoreiden ohjaamiseen. Sekvensserin kolmannen kanavan RC-filtterin tuottama vaimennus on esitetty kaavassa 4 ja muiden kanavien RC-filtterien tuottama vaimennus kaavassa 5.

$$_{Vaimennus\ (dB)} = -10log(1 + (2\pi \times 62500hz \times 910\Omega \times 100nF)^2)$$
$$Vaimennus\ (dB) = -31.07dB$$

(4)

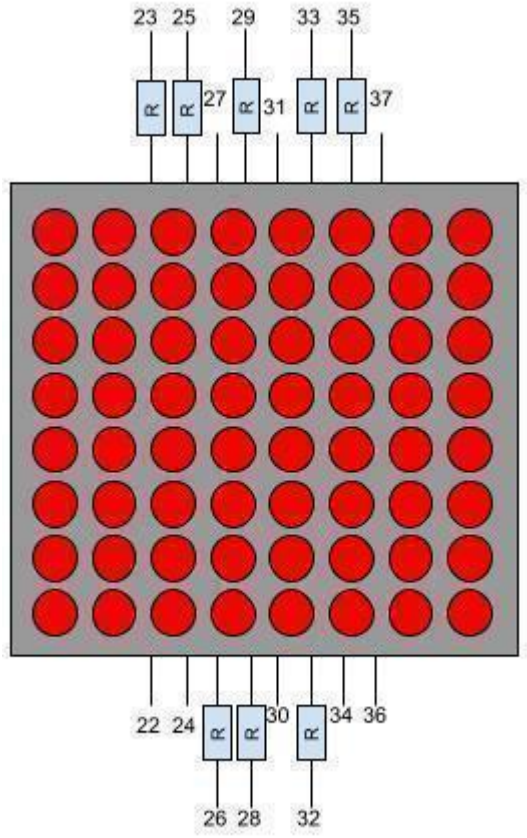
$$_{Vaimennus\ (dB)} = -10log(1 + (2\pi \times 31372{,}55hz \times 910\Omega \times 100nF)^2)$$
$$Vaimennus\ (dB) = -25.09dB$$

(5)

5.5 Matriisinäyttö

Matriisinäytön kytkennöissä on käytetty apuna Verkkokaupasta ostetun Arduino Atmega Mega2560 Starter Kit -aloituspakkaukseen sisältyvien ohjeiden esimerkkiä 4.20. LED-matriisin katodeihin on kytketty 91 Ω:n vastukset suojaamaan ja himmentämään ledejä. Vastusten toinen pää on yhdistettynä Arduinon digitaalisiin pinneihin. Anodit myös on yhdistetty omiin pinneihinsä.

Matriisinäytön toteutuksessa on käytetty avuksi multipleksausta, kuten potentiometrien LED-valojen kytkennöissä. Matriisinäytöllä palaa kerrallaan vain yksi LED-valo. Matriisinäyttöä päivitetään kuitenkin noin 1500 kertaa sekunnissa, joten ihmissilmä havaitsee vilkkuvat LED-valot jatkuvasti palavina.



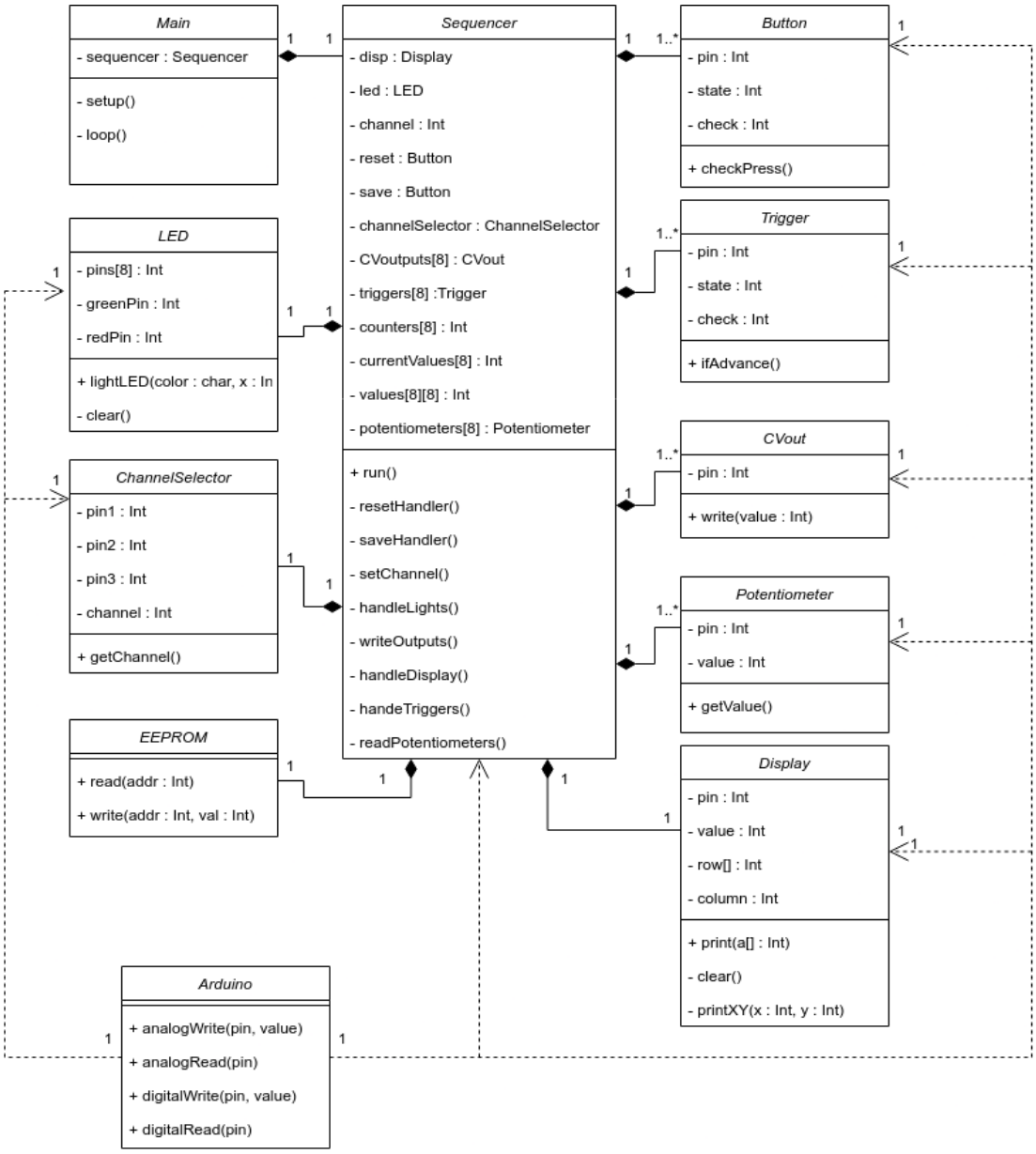
Kuva 6. 8x8-LED-matriisinäytön kytkentäkaavio

5.6 Kytkimet ja napit

Tässä sekvensserissä on käytössä kolme vipukytkintä kanavanvaihtoa varten ja kaksi painonappia, joista toista painamalla saadaan tallennettua arvoja Arduinon EEPROM-muistiin ja toista painamalla saadaan palautettua jokainen kanava ensimmäiseen askeleeseen. Vipukytkimet ja painonapit on kaikki liitetty Arduinoon samalla periaatteella. Kytkimien ja nappien ensimmäinen terminaali on kytketty Arduinon GND-pinniin ja toinen terminaali on kytketty digitaaliseen pinniin. Vipukytkimet on yhdistetty pinneihin 10, 11 ja 12. Reset -nappi on kytketty pinniin 15 ja tallennus -nappi pinniin 14.

6 Ohjelmisto

Arduinon pääohjelma koostuu kahdesta osasta, jotka ovat *setup* ja *loop. Setup*-funktiossa määritellään halutut alustukset, jotka suoritetaan, kun Arduino-mikrokontrolleriin kytketään virrat. *Loop*-funktio on käytännössä ikuinen silmukka, jota suoritetaan niin kauan, kun Arduino on käynnissä.



Kuva 7. *Luokkakaavio*

Tässä ohjelmistossa *setup*-funktiossa määritellään Arduinon PWM-kellotaajuus korkeammalle, kuin se normaalisti on, koska korkeamman kellotaajuuden myötä CV-signaaleista saadaan tasaisempia. Arduinon digitaalinen pinni 4 ja 13 käyttää arduinon timeria 0, mikä eroaa Arduinon muista timereista siten, että sen PWM-kello taajuus määrittyy noin kaksinkertaiseksi muihin verrattuna, kun timerit määritellään toimiaan mahdollisimman suurella taajuudella. Tällä erolla ei kuitenkaan ole sekvensserin toiminnan kannalta suurta merkitystä, koska sen

eroa ei käytössä huomaa. *Setup*- ja *loop*-funktioiden ulkopuolella luodaan *Sequencer*-olio. *Loop*-funktiossa kutsutaan vain *Sequencer*-olion run-metodia.

```
#include "Sequencer.h"
Sequencer sequencer;
void setup() {
    //PWM for pins D2, D3, D5
    TCCR3B = TCCR3B & B11111000 | B00000001;    // set timer 3 divisor to     1 for PWM frequency of 31372.55 Hz
    //PWM for pins D4, D13
    TCCR0B = TCCR0B & B11111000 | B00000001;    // set timer 0 divisor to     1 for PWM frequency of 62500.00 Hz
    //PWM for pins D6, D7, D8
    TCCR4B = TCCR4B & B11111000 | B00000001;    // set timer 4 divisor to     1 for PWM frequency of 31372.55 Hz
    //PWM for pins D9, D10
    TCCR2B = TCCR2B & B11111000 | B00000001;    // set timer 2 divisor to     1 for PWM frequency of 31372.55 Hz
    //PWM for pins D11, D12
    TCCR1B = TCCR1B & B11111000 | B00000001;    // set timer 1 divisor to     1 for PWM frequency of 31372.55 Hz
}
void loop() {
    sequencer.run();
}
```

Kuva 8. *Setup*- ja *loop*-metodit

6.1 Potentiometrit

Potentiometer-oliota luotaessa lähetetään konstruktorille parametrinä pinnin arvo, josta potentiometrin arvoa halutaan lukea. Potentiometrillä on yksi public-jäsenfunktio, joka lukee *analogRead*-metodia käyttämällä potentiometrin arvon. Tämän jälkeen funktio jakaa saadun arvon neljällä ja palauttaa sen. Arvo on mielekästä jakaa neljällä ennen kuin se palautetaan, koska EEPROM-tallennuksessa käytettävä *EEPROM.write*-metodi ja CV-ulostuloihin käytettävä *analogWrite*-metodi käyttää kokonaislukuarvoja 0:n ja 255:n väliltä.

```
#include "Potentiometer.h"
Potentiometer::Potentiometer(int p){
    pin = p;
    pinMode(p, INPUT);
    value = 0;
}
int Potentiometer::getValue(){
    analogRead(pin); value = analogRead(pin);
    return value/4;
}
```

Kuva 9. *Potentiometer*-luokka.

6.2 Potentiometrien LED-valot

Potentiometrien LED-valoja varten on luotu luokka, jonka avulla voidaan sytyttää haluttu LED-valo palamaan. LED-luokan konstruktorissa määritellään LED-valojen käyttämät pinnit ja pinnien tiloista vastaavat *clear*- ja *lightLED*-metodit. *Clear*-metodi asettaa kaikki LED-valoihin kytketyt pinnit lähettämään 0 V:n jännitettä, jottei yksikään ledi palaisi. *LightLED*-metodi ottaa vastaan parametreina tiedon, halutaanko vihreä vai punainen LED-valo syttymään, sekä numeroarvon siitä, minkä potentiometrin kohdalla sytytettävä valo on. Lopuksi *LightLED*-metodi kutsuu *clear*-metodia, jotta LED-valot sammuvat. Vaikka LED-valot sytytetään yksi kerrallaan palamaan, ihmissilmä havaitsee LED-valot jatkuvasti palavina, koska LED-valoja sytytetään ja sammutetaan noin 1500 kertaa sekunnissa. LED-valojen päälläoloaika pidetään erittäin lyhyenä, koska sekvensserin toimintaan ei ole haluttu luoda ylimääräistä viivettä, sillä se vaikuttaisi koko sekvensserin toiminta nopeuteen. Musiikin tekemissä tällaiset ylimääräiset viiveet halutaan minimoida.

```
LED::LED(){
    greenPin = 20;
    redPin = 21;
    for(int i = 0; i < 8; i++){
        pinMode(pins[i],OUTPUT);
    }
    pinMode(greenPin,OUTPUT);
    pinMode(redPin,OUTPUT);
}
void LED::clear(){
    digitalWrite(greenPin, LOW);
    digitalWrite(redPin, LOW);
    for(int i = 0; i < 8; i++){
        digitalWrite(pins[i], LOW);

    }
}
void LED::lightLED(char color, int x){
    if (color == 'g'){
        digitalWrite(redPin,HIGH);
    }
    else if (color =='r'){
        digitalWrite(greenPin,HIGH);
    }
    digitalWrite(pins[x], HIGH);
    clear();
}
```

Kuva 10. *LED*-luokan metodit.

6.3 Trigger-sisääntulot

Jokaisesta trigger-sisääntulosta luodaan Trigger-olio. Olio luodaan konstruktoria käyttäen, joka vastaanottaa pinnin arvon, johon kyseinen trigger-sisääntulo on kytkettynä. Konstruktorissa määritellään myös käytetty pinni *OUTPUT*-tilaan. Trigger-oliolla on privaatti muuttujina state ja check. State-muuttujaan talletetaan trigger-sisääntulon pinniltä luettu arvo *digitalRead*-metodia käyttäen ja check-muuttujaa käytetään tarkistamaan, että state-muuttujan arvo on käynyt arvossa 0. Tämä tarkistus on tärkeä tehdä, koska muuten sekvensseri siirtyisi seuraavaan askeleeseen joka kerta, kun pinniltä luettaisiin arvo, joka on 1. Jos pinniin ei ole kytketty jännitettä samaan aikaan kuin arvo luetaan, muuttujan state arvoksi asetetaan 0 ja check saa arvoksi 1. Jos molemmat muuttujat state ja check ovat arvolla 1, check-arvo muutetaan 0:ksi, ja funktio palauttaa arvon 1. Muissa tapauksissa funktio palauttaa arvon 0.

```
17 #include "Trigger.h"
16 Trigger::Trigger(int p){
15     pin = p;
14     //Set Pin as input with a pullup resistor
13     pinMode(p, INPUT_PULLUP);
12     state = 0;
11     check = 0;
10 }
 9 int Trigger::ifAdvance()
 8     state = digitalRead(pin);
 7     if (state == 0 ) check= 1;
 6     if (state == 1 && check == 1){
 5         check=0;
 4         return 1;
 3     }
 2     else
 1         return 0;
18
```

Kuva 11. *Trigger*-luokka

6.4 CV-ulostulot

CV-ulostuloja kuvataan *CVout*-oliolla. Oliota luodessa konstruktorille lähetetään parametrin arvo, joka määrittää, mitä pinniä halutaan käyttää ulostulona. *CVout*-oliolla on yksi public-funktio, *write*, joka toimii rajapintana Arduino-

ohjelmointikielen omalle *analogWrite*-metodille. *Write*-funtkiolle lähetetään parametrina arvo 0:n ja 255:n väliltä, ja se kirjoitetaan *analogWrite*-metodia, kun käytetään olion konstruktoriin lähetetyn parametrin mukaiselle pinnille.

```
#include "CVout.h"
CVout::CVout(int p){
    pin = p;
    pinMode(p, OUTPUT);
}
void CVout::write(int value){
    analogWrite(pin, value);
}
```

Kuva 12. *CVout*-luokka.

6.5 Matriisinäyttö

Matriisinäytölle piirtoa hallitse *Display*-luokka. *Display*-luokan konstruktorissa määritellään Arduinon käyttämät pinnit kahteen listaan. Listassa *rows* on määritelty pinnit, jotka hallitsevat x-akselieita ja *columns*-listassa on vastaavasti pinnit, jotka hallitsevat y-akseleita. X-akselilla olevat pinnit kytkeytyvät matriisinäytön anodeihin ja y-akselilla olevat pinnit kytkeytyvät katodeihin. Kun esimerkiksi vain ensimmäisen x-akselin pinni määritellään lähettämään jännitettä ja kaikki muut y-akselin pinnit paitsi ensimmäinen määritellään lähettämään jännitettä, vain vasemmassa yläreunassa oleva LED-valo syttyy.

Display-luokalla on kolme metodia: *clear*, *printXY* ja *printList. clear*-metodi alustaa kaikki pinnit lähettämään 0 V jännitettä. Näin näyttö tyhjenee, eikä yksikään valo pala. *PrintXY*-metodi ottaa parametreina x- ja y-koordinaatit, joiden mukaan Arduinon pinneihin lähetetään jännite, jotta halutussa koordinaatissa oleva LED-valo syttyy. *PrintList*-metodi puolestaan ottaa vastaan kokonaisluku listan, mikä sisältää tiedot siitä, missä askeleessa sekvensserin kanavat ovat kyseisellä hetkellä. Metodissa kutsutaan *printXY*-metodia for-silmukassa siten, että x-koordinaatiksi lähetetään arvo, johon listan indeksi viittaa, ja y-koordinaatiksi indeksin arvo.

```
1 void Display::printList(int a[])
2 {
3     for(int i = 0; i < 8; i++){
4         printXY(a[i], i);
5     }
6 }
7 void Display::printXY(int x, int y)
8 {
9     clear();
10    digitalWrite(row[y], HIGH);
11    for(int i = 0; i < 8; i++){
12        if (i != y)
13            digitalWrite(column[x], HIGH);
14    }
15    clear();
16 }
17 void Display::clear()
18 {
19    for (int i =0;i<8;i++){
20        digitalWrite(row[i], LOW);
21    }
22    for (int i =0;i<8;i++){
23        digitalWrite(column[i], LOW);
24    }
25 }
```

Kuva 13. *Display*-luokan metodit.

6.6 Kanavanvaihtokytkimet

Kanavanvaihtokytkimet määritellään *ChannelSelector*-luokassa. Luokan konstruktori saa parametrina kolme arvoa, jotka vastaavat pinnejä, joihin kyseiset kytkimet on yhdistetty. Nämä kolme pinniä määritellään *INPUT_PULLUP*-tilaan, jolloin pinneihin on yhdistetty ylösvetovastus. ChannelSelector-luokalla on *getChannel*-metodi, joka lukee konstruktorille lähetetyt pinnit järjestyksessä. Jos ensimmäiseltä pinniltä luettu arvo on yksi, palautusarvoon lisätään 4. Jos toiselta pinniltä luettu arvo on yksi, palautusarvoon lisätään 2, ja jos viimeiseltä pinniltä luettu arvo on yksi, palautusarvoon lisätään yksi. Näin palautusarvoksi saadaan lukuja nollan ja seitsemän väliltä, mikä on sama määrä, kuin tässä sekvensserissä on kanavia.

6.7 Sequencer

Sequencer-luokka yhdistää edellä mainitut luokat loogiseksi kokonaisuudeksi. *Sequencer*-luokan konstruktorissa alustetaan muut oliot ja tarvittavat taulukot. *Potenttiometri-*, *Trigger-, ja CVout*-oliot on luotu omiin taulukoihin, jotta niiden läpikäyminen on huomattavasti helpompaa. Sekvensserillä on lisäksi kolme muuta taulukkoa. *Counters*-taulukko on kahdeksansarakkeinen integer-taulukko, jossa pidetään tallessa, millä askeleella sekvensserin kanavat ovat. *CurrentValues*-taulukkoon talletetaan potentiometreilta viimeksi luetut arvot. Kanaville tallennettuja arvoja säilytetään *values*-taulukossa. *Values*-taulukko on kaksiulotteinen taulukko, missä on kahdeksan riviä ja kahdeksan saraketta. Jokainen rivi vastaa yhtä kanavaa, ja sarakkeille on tallennettu kanavan askelten arvot.

Sequencer-luokalla on metodit, joiden avulla edellä mainittuja listoja käyttäen luodaan logiikka, jotta sekvensseri toimii halutulla tavalla. *ReadPotentiometers*-funktiolla kutsutaan potentiometrien *getValue*-metodia ja talletetaan saadut arvot *currentValues*-listaan. *HandleTriggers*-funktiolla tarkastellaan *trigger*-olioiden tilaa ja päivitetään tiedot *counters*-listaan, mikäli se on tarpeen. CV-ulostuloja hallinnoidaan *writeOutputs*-metodissa, mikä kutsuu *CVout*-olioiden *write*-metodia. Jos CV-ulostulo on määritelty kanavalle, mikä on tällä hetkellä valittuna, *write*-metodille lähetetään parametrina potentiometreiltä luetut arvot listana. Muussa tapauksessa *write*-metodille lähetetään arvot, jotka on tallennettu *values*-taulukkoon. *Sequencer*-luokalla on myös *run*-metodi, mikä kutsuu edellämainittuja metodeja oikeassa järjestyksessä.

7 Pohdintaa

Arduino Mega soveltuu tällaiseen projektiin yllättävän hyvin lukuisten pinnien ansiosta. Etenkin loogiikan ohjelmointiin ja digitaalisten komponenttien käyttöön Arduino sopii hyvin. Suurin heikkous Arduino Megassa syntetisaattorimoduulien rakentamiseen on matalaresoluutioinen PWM-toteutus. Koska sekvensserin lähettämä jännite voidaan valita nollan ja viiden voltin välillä vain 255 eri tavalla,

se ei sovellu täysin musikaalisten melodioiden tuottamiseen, koska CV-signaalia ei saa viritetty tarpeeksi tarkasti, jotta oskillaattori tuottaisi vireessä olevan nuotin. Tämän voi toki kiertää liittämällä CV-ulostulo ensin vaimenninmoduuliin ja vaimentaa Arduinon lähettämä CV-signaali esimerkiksi nollan ja kahden voltin välille, jolloin PWM-resoluutio kasvaa näennäisesti.

Toinen mahdollisuus parantaa arduinon PWM-resoluutiota olisi kytkeä kaksi PWM-pinniä yhteen jännitteenjakajan avulla. Näin kaksi pinniä yhdistämällä kahdesta 8-bittisestä PWM-pinnistä voitaisiin luoda yksi 16-bittinen ulostulo. Tätä ei kuitenkaan tässä projektissa voitu käyttää, koska jokainen ulostulo olisi tarvinnut kaksi pinniä käyttöön eikä pinnejä olisi riittänyt kaikille kahdeksalle kanavalle.

Sekvensserin CV-ulostulot olisi voitu myös toteuttaa erillisillä DAC-komponenteilla, joiden avulla olisi voitu tuottaa aidosti analogista signaalia. Tätä toteutusta ei tässä projektissa kuitenkaan käytetty, koska kyseiset komponentit olisivat tarvinneet useamman kuin yhden pinnin käyttöön Arduino-mikrokontrollerista.

DAC-piiri olisi voitu myös toteuuttaa esimerkiksi 4 x 16 bit DAC SPI-liitännällä. SPI-liitäntä vaatii Arduinolta kolme pinniä, sekä yhden pinnin jokaista piiriä kohti. Tällaisella toteuksella saataisiin kahta DAC-piiriä käyttämällä kahdeksan analogikanavaa vain viittä pinniä käyttämällä. 16 bittisten DAC-piirien resoluutio olisi huomattavasti parempi sekvensserin toiminnan kannalta.

Koska sekvensserin potentiometrien lukemiseen käytetään Arduino-mikrokontrollerin analogia-digitaalimuuntimia, joiden resoluutio on vain 10 bittiä, voitaisiin ulostulon arvo määrittää silti vain 1024:llä eri tavalla. Arduinon analogia-digitaalimuuntimen pienemmän resoluution takia DAC-piirien resoluutiosta voitaisiin hyödyntää vain murto-osa. Tämän takia sekvensserin rakentamisessa päädyttiin käyttämään RC-filttereitä ja Arduinon PWM-ominaisuutta.

Lähteet

1 2021, Eurorack, Verkkodokumentti <https://en.wikipedia.org/wiki/Eurorack>, Luettu 5.5.2021.

2 2021, Modular synthesizer, Verkkodokumentti <https://en.wikipedia.org/wiki/Modular_synthesizer>, Luettu 5.5.2021.

3 2014, Robert Fantinatto, I Dream of Wires, Dokumenttielokuva

4 Using PWM to Generate an Analog Output, Verkkodokumentti <http://ww1.microchip.com/downloads/en/Appnotes/90003250A.pdf> Luettu 21.5.2021.

5 Joy-it Mega2560 Microcontroller Leraning Kit, Verkkodokumentti <https://cdn-a.verkkokauppa.com/35/16_12489.pdf> Luettu 3.5.2021.