Tampereen yliopisto

Lassi Parkkila

MIKROKONTROLLERIOHJATUN METRONOMIN SUUNNITTELU JA TOTEUTUS

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Heinäkuu 2019

TIIVISTELMÄ

Lassi Parkkila: Mikrokontrolleriohjatun metronomin suunnittelu ja toteutus
Design and implementation of a microcontroller controlled metronome
Kandidaatintyö, 51 sivua, 2 liitettä
Tampereen yliopisto
Tieto- ja sähkötekniikka, TkK
Tarkastaja: Yliopistolehtori Erja Sipilä
Heinäkuu 2019

Tässä työssä suunnitellaan, rakennetaan ja testataan mukana kuljetettava ja riittävän pienikokoinen mikrokontrolleriohjattu metronomi. Metronomiin suunnitellaan niin kotelointi, kuin myös sähköinen kokonaisuuskin. Myös käyttöliittymä ja ohjelmakoodi toteutetaan tämän työn puitteissa. Elektroniikkakokonaisuus pyritään saamaan siististi piirilevylle. Metronomin ulkoasusta tehdään mukavan näköinen ja käyttöliittymästä helppokäyttöinen. Markkinoilla on paljon hyviä digitaalisia metronomeja, mutta niistä ei löydy mahdollisuutta tahdittaa additiivisia rytmejä eikä käyttäjän ole mahdollista ohjelmoida laitteita uudestaan, mikäli kokisi sen tarpeelliseksi. Tähän tarpeeseen rakennettava metronomi pyrkii vastaamaan. Tulevaisuudessa metronomilla voisi tahdittaa myös polyrytmisiä tahteja.

Metronomin suunnittelu ja toteutus onnistuivat pääsääntöisesti hyvin. Kuitenkin metronomia rakentaessa eteen tuli muutamia haasteita, jotka pyrittiin ratkaisemaan tai pienentämään haasteesta koituvaa haittaa. Esimerkiksi vahvistinta valittaessa piti kokeilla useita erilaisia komponentteja sopivan vahvistimen löytämiseksi. Myös testausvaiheessa tikahdusten aiheuttamat virtapiikit aiheuttivat ongelmatilanteita, jotka kuitenkin korjattiin asentamalla muutama ylimääräinen komponentti jälkikäteen.

Metronomin runko valmistettiin 3D-tulostamalla. Tulostustyö jouduttiin tekemään useaan kertaan ennen kuin lopputuloksen muoto oli haluttu ja laatu riittävä. Onneksi 3D-tulostaminen on hyvä tapa kokeilla ja valmistaa prototyyppejä. Kotelosta saatiin lopulta elegantin näköinen ja riittävän kestävä käyttötarkoitukseen. Ohjelmakoodi rakentui monien yritysten ja erehdysten kautta, mutta lopulta laite saatiin toimimaan halutulla tavalla.

Lopputuloksena saatiin lähes halutunlainen metronomi, joka on pienikokoinen ja helposti käytettävä. Muutamia laatuun negatiivisesti vaikuttavia asioita metronomiin vielä jäi, joita ei työn puitteissa saatu korjattua, yhtenä merkittävämpänä kokoaikainen pieni humina kaiuttimesta ja kuulokeliittimestä. Seuraava kehitysaskel on näiden epäkohtien korjaaminen, sillä metronomeja olisi tarkoitus valmistaa muutamia kappaleita käyttöön.

Avainsanat: Metronomi, mikrokontrolleri, musiikki, 3D-tulostus, kannettava laite

ALKUSANAT

Tämä kandidaatintyön idea sai alkunsa, kun muutama kaveri kaipasi metronomia, jollaista ei kaupasta saanut. Kyselin heiltä tarkempia tietoja siitä, mitä he kaipaavat metronomilta. Ajattelin tällaisen kokonaisen elektroniikkalaitteen toteuttamisen olevan mielenkiintoinen projekti ja kysyin ohjaajaltani Erja Sipilältä, voisinko tehdä kandidaatintyöni kyseisen metronomin rakentamisesta. Hän piti sitä hyvänä ideana.

Haluan kiittää muutamia ystäviäni, erityisesti Lauria ja Santeria, musiikin teorian opettamisesta minulle, sekä suorasta palautteesta koko prosessin ajan. Kiitän myös ohjaajaani Erja Sipilää kannustavasta ja rakentavasta palautteesta sekä kärsivällisyydestä hieman venyneen projektin osalta.

Tampereella, 2.7.2019

Päivittäjä: Lassi Parkkila

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ADC	engl. Analog to Digital Converter, analogi-digitaali muunnin
ALU	engl. Arithmetic Logic Unit, aritmeettislooginen yksikkö
AUX	engl. Auxiliary, yleinen kuulokeliitintyyppi
BPM	engl. Beats Per Minute, iskua minuutissa
C	Akun kapasiteeti
CC	engl. Constant Current, Vakiovirta
CV	engl. Constant Voltage, Vakiojännite
DAC	engl. Digital to Analog Converter, digitaali-analogia muunnin
EEPROM	engl. Electronically Erasable Programmable Read-Only Memory, sähköisesti tyhjennettävä ja kirjoitettava haihtumaton lukumuisti
FTDI	engl. Future Technology Devices International, mikropiirivalmistaja
HCI	engl. Human-Computer Interaction, tietokoneen ja ihmisen vuoro-vaikutus
I/O	engl. Input/Output, sisääntulo/ulostulo
IC	engl. Integrated Circuits, mikropiiri
IEEE	Institute of Electrical and Electronics Engineers
LCD	engl. Liquid Crystal Display, Nestekidenäyttö
LED	engl. Light-Emitting Diode, hohtodiodi
MOSFET	engl. Metal-Oxide-Semiconductor Field-Effect Transistor, Metalli-oksidi-puolijohdekanavatransistori
PC	engl. Personal Computer, henkilökohtainen tietokone
PLA	engl. Polylactic Acid, Polylaktidi, muovilaatu, jota käytetään mm. 3D-tulostamisessa
PWM	engl. Pulse-Width Modulation, Pulssinleveysmodulaatio
RAM	engl. Random Access Memory, keskusmuisti
RGB	engl Red, Green, Blue, punainen, vihreä, sininen
ROM	engl. Read Only Memory, lukumuisti
SD	engl. Secure Digital, muistikorttityyppi
SPI	engl. Serial Peripheral Interface, sarjaliikennerajapinta
SRAM	engl. Static Random Access Memory, staattinen lukumuisti
TFT	engl. Thin Film Transistor, ohutkalvotransistori
TNT	engl. Trinitrotoluene, yleinen räjähdysaine
UI	engl. User Interface, käyttöliittymä
USB	engl. Universal Serial Bus, sarjaväyläarkkitehtuuri

1. JOHDANTO

Musiikki on nykyään monipuolisempaa ja helpommin toteutettavaa kuin koskaan ennen. Tämän myötä myös musiikin monimuotoisuus ja monimutkaisuus ovat lisääntyneet. Perinteisesti yksi musiikkikappale on sisältänyt vain yhden tai muutaman tahtilajin, eikä tahtilajin muutoksia esiinny usein kappaleiden keskellä. Niissä ei myöskään ole ollut useita tahtilajeja päällekkäin.

Jotta soittajat pysyvät samassa ja halutussa tahdissa, on tapana ollut käyttää laitetta nimeltä metronomi. Laite tuottaa tahdin mukaista herätettä, usein tahdin mukaan naksahtelevaa ääntä. Koska tahtilajit ovat olleet staattisia, ovat myös metronomit olleet tasaisin väliajoin naksahtelevia laitteita.

Elektroniikan nopean kehityksen myötä nykyään on mahdollista luoda pieniä ja suorituskykyisiä laitteita, jotka helpottavat ihmisten jokapäiväistä elämää, harrastuksia ja työntekoa. Myös musiikin parissa työskentelevät käyttävät enenevissä määrin sähköisiä apuvälineitä.

Markkinoilta on kuitenkin puuttunut erilaisia tahtikuvioita mahdollistava metronomi, joka olisi helposti mukana kuljetettava, yksinkertainen käyttää sekä käyttäjän uudelleenohjelmoitavissa tarpeen mukaan. Tässä työssä suunnitellaan, rakennetaan ja testataan mikrokontrolleriohjattu metronomi. Mikrokontrollerina käytetään Microchipin valmistamaa AVR tuoteperheen AVR ATmega328p mikrokontrolleria.

Suunnittelun yhtenä lähtökohtana on saada kaikki järjestelmän toiminnalliset lohkot samalle piirilevylle. Laitteen koon tulisi myös olla sopiva, jotta sen mukana kuljettaminen olisi vaivatonta. Myös käytettävyyteen tullaan kiinnittämään huomiota, jotta laitteen käyttö olisi miellyttävää ja sen käyttö ei vaatisi tarpeettoman paljon opettelua.

Työn alussa esitetään tarkemmin, mikä on metronomi, sekä tässä työssä rakennettavaan metronomiin haluttavat ominaisuudet. Tämän jälkeen käsitellään laitteen ja käyttäjän välistä rajapintaa eli käyttöliittymää, sen ominaisuuksia ja mahdollisuuksia. Seuraavaksi käsitellään laitteen rakennetta, sekä sen sisäisten toiminnallisten lohkojen toimintaa. Lopussa ennen yhteenvetoa käydään läpi valmiin laitteen testaukseen liittyviä asioita sekä avataan ohjelmakoodin toimintaa.

2. METRONOMI

Metronomi on laite, jolla pystytään tuottamaan ennalta määrätty määrä ärsykkeitä, esimerkiksi naksahduksia tai kilahduksia, minuutissa siten, että musiikin parissa työskentelevä pysyy aina samassa ja halutussa tempossa. Metronomin esiaste oli 1700-luvun lopulla D. N. Winkelin kehittämä chronomètre, jossa käytetään pendulum-efektiä luomaan vakaa värähtelyaika. Kuitenkaan Winkel ei suojannut keksintöään patentilla ja vuonna 1816 Johann Nepomuk Mälzel patentoi Winkelin idean, johon oli lisätty asteikko tahdistuksen helpottamiseksi. Näin syntyi laite nimeltä ”Mälzel metronomi”. Kyseinen laitetyyppi on vielä nykyäänkin yleisesti käytössä oleva metronomimalli. [1]

Edellä mainittu metronomi on puinen, pyramidin muotoinen instrumentti, jonka edessä on terästanko, joka pääsee keinumaan pystysuunnassa. Tanko on noin 7 ½” pitkä ja ⅛” leveä. Tanko toimii kaksois-pendulum efektin avulla, eli tangon kummassakin päässä on paino. Toinen paino, yleensä ylempi, on liikutettava. Painon sijainnista riippuen metronomin oskillointitaajuus on säädettävissä, yleensä välillä 40 – 208 BPM *(engl. Beats Per Minute, iskua minuutissa)*. Joka heilahduksella tanko aiheuttaa pienen tikittävän äänen. Lisäksi laitteessa oleva kello voidaan määrätä kilahtamaan 2, 3 tai 4 isku välein. [1]

Kuva 1: *Perinteinen metronomi [2], sekä kaupallinen digitaalinen metronomi [3]*

Kuvassa 1 on esitettynä sekä perinteinen vedettävä metronomi, että uudenaikainen digitaalinen metronomi. Perinteisestä metronomista näkee hyvin metronomin tapin rakenteen, jossa on paino kummassakin päässä ja toinen paino on siirrettävä.

2.1 Rakennettavan metronomin ominaisuudet

Kaikkein monipuolisin metronomi saadaan, kun se toteutetaan digitaalisesti. Tällöin iskujen ajoitus voidaan tehdä hyvinkin tarkasti ja iskutaajuus on helposti muutettavissa numeerisesti. Tässä työssä toteutettavassa metronomissa käytetään kaiken laskennan sydämenä Microchipin valmistamaa AVR tuoteperheen AVR ATmega328p mikrokontrolleria. ATmega328p:n kellotaajuus on ulkoisella kiteellä 16 MHz [4], joka riittää metronomin kaiken toiminnallisuuden ohjaamiseen.

Koska metronomista toteutetaan mukana helposti kannettava, pitää siihen sisällyttää akku kaikkine oheispiireineen. Akun varaus on käyttäjän nähtävissä näytöltä. Akun ylilataus ja syväpurkusuojaukset on toteutettu suoraan piireinä, joten käyttäjä ei voi omilla toimillaan aiheuttaa akulle vahinkoa ylilataamalla tai käyttämällä akkua liian tyhjäksi.

Jotta metronomin käyttö olisi helppoa ja miellyttävää, on laitteessa seitsemän painonappia, joilla säädetään metronomin ominaisuuksia. Metronomissa on myös pieni näyttö, jonka kautta käyttäjä näkee metronomin tilan ja asetukset.

Metronomille päätettiin kehitellä myös nimi. Skandinaavinen mytologia sisältää paljon mielenkiintoista aineistoa ja toimii hyvänä inspiraation lähteenä. Nimeksi päädyttiin ottamaan skandinaavisen mytologian musiikin ja runollisuuden jumalan nimi, Bragi [5].

Kuva 2: *Metronomin nimi aloitusnäytössä, BRAGI-metronome.*

Kuvan 2 mukainen aloituskuva tulee metronomin näytölle aina käynnistettäessä. Aloituskuvan tehtävänä on antaa käyttäjälle aikaa valita asetusten, tehdasasetusten palautuksen ja normaalin käytön välillä.

2.2 Toiminta

Käynnistettäessä metronomi näytölle ilmestyy laitteen nimi, BRAGI-metronome. Mikäli käynnistettäessä ja BRAGI-tekstin aikana pidetään pohjassa näppäimiä ◀ ja ▶, siirrytään asetusvalikkoon, jossa voi säätää äänien taajuuksia, grafiikoiden värejä ja muita ulkoasuun liittyviä asetuksia. Näyttö on tällöin pystyasennossa ja metronomi pitää kääntää pystyyn siten, että kaiutin on alaspäin. Kuva asetusvalikosta on nähtävissä alla.

Kuva 3: *Metronomin asetusvalikko*

Muokattava kohde valitaan näppäimillä ja . Muokattavan kohteen ympärille muodostuu neliö, jonka väri määräytyy ”Edit box” asetuksen mukaan. Oletuksena tämä väri on sininen. Muokattavan kohteen arvoa vaihdetaan näppäimillä ja . Pitkään painettaessa lukuarvot kasvavat tai pienenevät nopeasti ja siten lukujen suuret muutokset ovat melko nopeita tehdä. Painamalla näppäintä pohjassa yli 3 sekuntia, asetukset tallentuvat metronomin muistiin ja siitä ilmoitetaan ledin vilkahtamisella ja pienellä äänimerkillä. Mikäli tallennusta ei tehdä ja metronomi käynnistetään uudelleen, on voimassa vanhat asetukset. Asetusvalikosta pois pääsemiseksi on metronomi käynnistettävä uudelleen kytkemällä metronomi virtakytkimestä ensin pois päältä ja sen jälkeen päälle.

Mikäli asetusten asettamisessa ei kaikki ole mennyt kuten on suunniteltu, esimerkiksi taustan ja tekstin väri on muutettu samaksi, tai käyttäjä haluaa palauttaa metronomin tehdasasetuksiin, pitää käynnistyksen ja BRAGI-tekstin aikana pitää pohjassa näppäimiä , , ja . Tällöin kaikki värit palautuvat alkuperäisiksi väreiksi ja kaikki ajoitukset, kuten tahdit ja tempot muuttuvat oletusarvoihinsa. Nämä asetukset pitää tallentaa painamalla näppäintä yli 3 sekuntia, tai muuten uudellenkäynnistyksen jälkeen voimassa ovat edelleen vanhat asetukset.

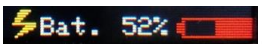
Normaaliin käyttöön päästään, kun käynnistyksen aikana ei pidetä mitään näppäimiä pohjassa. Alla on esitetty kuva normaalista käyttöliittymästä oletusväreillä.

Kuva 4: *Metronomin näyttö käytön aikana*

Kaikki seuraavien kappaleiden toiminnot viittaavat kuvan 4 käyttöliittymään. Käyttöliittymän yläreunassa on kahdeksan välilehteä, joiden välillä voi siirtyä näppäimillä ja . Aktiivinen välilehti näkyy hieman pidempänä kuin muut välilehdet. Muokattava kohde valitaan näppäimillä ja , ja kohdetta muokataan näppäimillä ja . Välilehdet soitetaan järjestyksessä vasemmalta oikealle ja vain ne välilehdet soitetaan, jotka ovat aktivoitu, eli vihreinä. Jokainen välilehti sisältää samat asetukset, joita voi muuttaa erikseen. Ensimmäinen muokattava kohde on välilehden aktivoiminen tai passivoiminen. Se onnistuu valitsemalla välilehti näppäimillä ja , ja siirtämällä muokkausvalitsin välilehden päälle näppäimillä ja . Välilehden tilaa voidaan vaihtaa näppäimillä ja .

Toinen muokattava kohde on tahtilaji, joka näytöllä näkyy kuvana . Tahtilajin osoittajaa ja nimittäjää voidaan säätää kumpaakin erikseen välillä 1-99. Kolmas muokattava kohde on , jolla määritetään, kuinka monta kertaa kyseinen välilehti soitetaan, ennen kuin siirrytään seuraavaan välilehteen. Neljäntenä muokkauskohtana on tempo, joka näkyy

näytössä kuvana  . Tempo määrittää, kuinka monta iskua minuutissa metronimi tuottaa tahtilajin nimittäjällä 4. Mikäli nimittäjä on esimerkiksi 8, on todellinen iskutiheys kaksi kertaa nopeampi. Viides muokkauskohta on kohta  , jolla määritetään, onko koko kierron ensimmäinen isku erillainen kuin muut iskut. kuudes muokkauskohta on  , jolla määritetään, onko jokaisen tahtilajin ensimmäinen isku erilainen kuin muut.

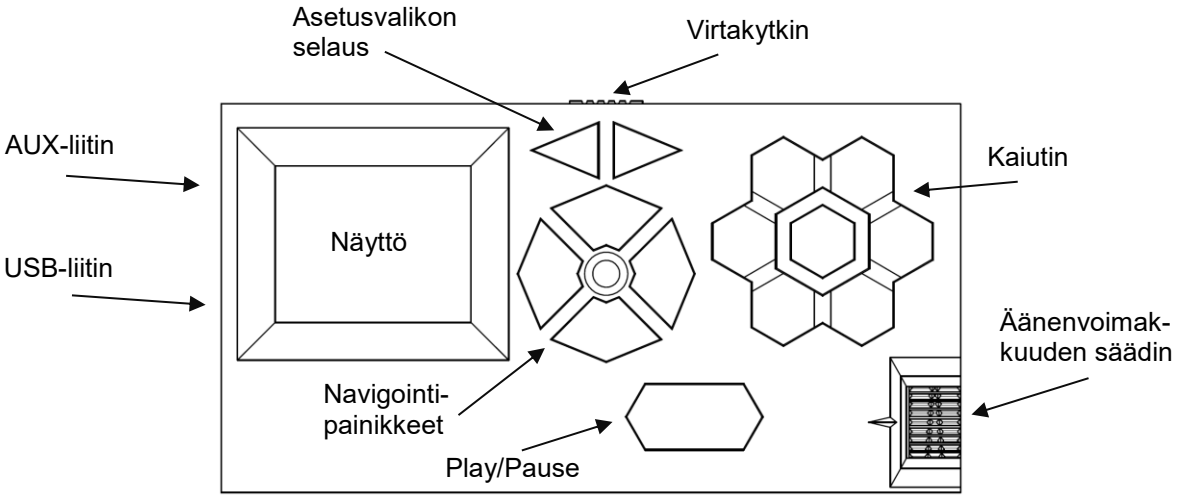
Näytöllä näkyy myös akun varauksen ilmoittava prosenttiluku sekä akkukuvake, joka näyttää visuaalisesti akun varauksen tilan  . Akkukuvakkeen edessä on salama, joka on väriltään keltainen latauksen aikana ja vihreä silloin, kun akku on täyteen ladattu, mutta kiinni laturissa. Salamaa ei näy silloin, kun metronomi ei ole kytketty lataukseen.

Metronomin tikitys voidaan kytkeä päälle ja pois napauttamalla näppäintä  . Mikäli asetuksissa on sallitu sisäisen ledin polttaminen, vilahtaa näppäinten keskellä kirkas LED-valo (*engl. Light-Emitting Diode, hohtodiodi*) samalla, kun tikahdus kuuluu kaiuttimesta. LED-valon väri ja kirkkaus on säädettävissä asetuksissa.

3. KÄYTTÄJÄN JA LAITTEEN RAJAPINTA

Käyttäjien ja tietokonejärjestelmien välisen kommunikoinnin tutkimusta kutsutaan nimellä HCI (*engl. Human-Computer Interaction, tietokoneen ja ihmisen vuorovaikutus*). HCI pitää sisällään muun muassa tietokoneen toiminnan, ihmisen psykologian, ergonomian, suunnittelun ja grafiikan tutkimisen. [6, s. 3 part 1]

Käyttöliittymä eli UI *(engl. User Interface)* on laitteen osa tai kokonaisuus, jonka kautta käyttäjä käyttää laitetta. Käyttöliittymän ominaisuudet vaikuttavat merkittävästi käyttökokemukseen ja käytön helppouteen [6]. Siihen kuuluvat kaikki toiminnallisuudet, joilla käyttäjä syöttää tietoa laitteeseen ja saa laitteelta tietoa itselleen. Esimerkiksi näyttö, näppäimet ja kaiutin ovat oleellisia osia metronomin käyttöliittymässä. Alla olevassa kuvassa 5 on esitetty kuva metronomin fyysisestä käyttöliittymästä.



Kuva 5: *Metronomin etupuoli*

Käyttöliittymän ansiosta käyttäjä voi käyttää laitetta tuntematta laitteen teknistä toimintaa ja toteutusta. Käyttöliittymä myös nopeuttaa laitteen käyttöä, sillä yhdellä käyttöliittymän toiminnolla voidaan toteuttaa useita laitteen toimintoja samanaikaisesti. [7] Alla esitetyissä kappaleissa on esitetty metronomin käyttöliittymän osat sekä selitetty niiden toiminnallisuutta.

3.1 Näyttö

Tärkein tietoa metronomin asetuksista ja tilasta käyttäjälle välittävä komponentti on näyttö. Sen kautta käyttäjä näkee metronomin parametrien arvot sekä sen, mitä parametriä muutetaan. Näyttönä tässä työssä tehtävässä metronomissa käytetään Adafruitin valmistamaa ST7735R 1,8”-näyttöä [8]. Kyseiseen näyttöön päädyttiin, sillä kyseessä on värinäyttö, jolla voidaan tehdä selkeä graafinen käyttöliittymä, sekä se on yhteensopiva Arduino-ympäristön kanssa.

Näytön ominaisuudet [8]:

- Koko: 1,8”
- Resoluutio: 128 x 160
- Näyttöteknologia: TFT, LCD *(engl. Thin Film Transistor, Ohutkalvotransistori, Liquid Crystal Display, nestekidenäyttö)*
- Väri: RGB, 18-bit *(engl. Red, Green, Blue, Punainen, Vihreä, Sininen)*
- Ajuripiiri: ST7735R
- Ohjaus: SPI *(engl. Serial Peripheral Interface, sarjamuotoinen oheislaiteväylä)*

Koska näyttö toimii 3,3 V jännitetasolla, tulee kaikki signaalilinjat muuttaa 3,3 V jänniteisiksi, sillä mikrokontrollerimme käyttää 5 V jännitetasoa. Tämä voidaan toteuttaa yksinkertaisella vastusten jännitteenjakokytkenällä, sillä näytöltä ei tule signaaleita mikrokontrollerille. Vastuksiksi valitaan 10 kΩ ja 20 kΩ vastukset, jolloin 5 V linjasta saadaan 3,33… V. Tämä on vielä alle näytön suurimman sallitun logiikkatason jännitteen, joka on V_{CC} +0,3 V [8]. Käyttöjännitteensä näyttö saa FTDI-piirin *(Future Technology Devices International)* 3,3 V ulostulosta.

Käytön aikana näytöstä on nähtävissä seuraavia asioita: tahtilaji, tempo, tahdin ensimmäisen iskun painotus, koko kierron ensimmäisen iskun painotus, välilehden toistokertojen lukumäärä, välilehdillä olevat muistipaikat sekä akun varaustila. Näytön grafiikkaa on nähtävillä alla kuvassa 6 ja valmis käyttöliittymä kuvassa 4.

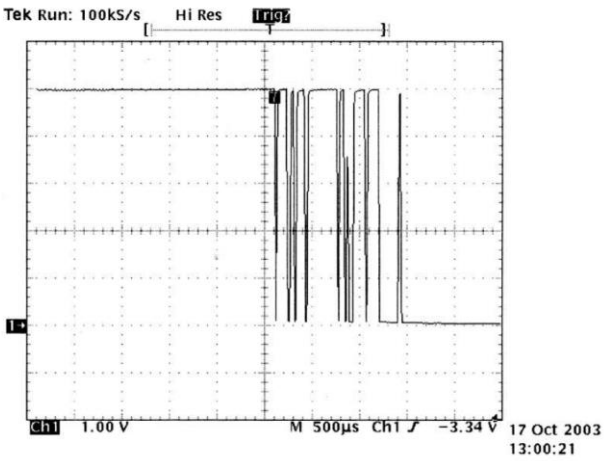
Kuva 6: *Näytön käyttöliittymä kehitysvaiheessa*

Näytön päivitykseen kuluva aika on melko suuri, joten kuvaa ei päivitetä silloin, kun metronomi soittaa ääntä. Päivitys myös saattaa vaikuttaa ajastimen tarkkuuteen. Jatkokehitettäessä ohjelmakoodia yhtenä tavoitteena on näytön suorituskyvyn parantaminen.

3.2 Näppäimet

Kaikki kommunikointi käyttäjältä metronomiin tapahtuu seitsemän näppäimen kautta. Näppäinasettelu on sellainen, että neljä napeista toimii navigoinnissa, kahdella napeista voidaan selata asetusvalikoita ja yksi nappi, jolla joko pysäytetään tai aloitetaan metronomin soitto. Painonappien sijoittelu on esitetty kuvassa 5. Koska painonapit valmistetaan 3D-tulostamalla, on niihin melko helppo sisällyttää yksinkertaiset merkit, jotka kertovat mitä kukin nappi tekee.

Nappien rakenteesta johtuen suljettaessa tai avattaessa kytkintä kytkimen kontaktipinnassa tapahtuu niin sanottua värähtelyä (*engl. debounce*). Ideaalisessa tilanteessa kytkin vaihtaisi tilaansa heti, mutta todellisessa maailmassa kun nappi painetaan pohjaan, kontaktori jää ”hyppimään” hetkeksi ja aiheuttaa päälle- ja poiskytkeytymistä kunnes kontakti stabiloituu. Todelinen kytkimen tilan vaihtuminen on esitetty alla. [9, s. 58]

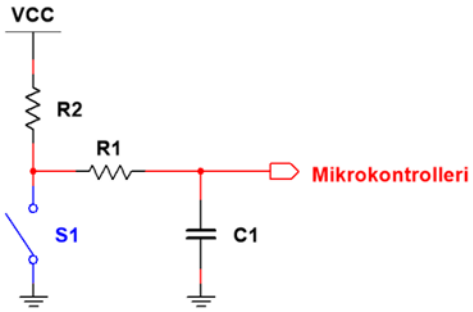


Kuva 7: *Kytkimen sulkemisesta aiheutuvaa värähtelyä [9, s. 58]*

Kuten kuvasta 7 nähdään, kytkimen tilan muuttuminen aiheuttaa värähtelyä. Mikäli kytkimen tila luettaisiin aina tilan muttuessa, saataisiin hyvin epävarma tulos ja useita painalluksia yhden painallksen sijasta. Kytkinvärähtelystä koituvaa haittaa voidaan vähentää helposti kahdella eri tavalla: ohjelmallisesti tai fyysisellä kytkennällä. [9, s. 58]

Ohjelmallisest toteutettuna kytkinvärähtely on helppo poistaa piirilevyn suunnittelun jälkeen eikä vaadi ulkoisia komponentteja. Toimintaperiaate on pääpiirteissään seuraavan kaltainen: Kun huomataan muuttuva kytkimen signaali, odotetaan hetken signaalin tasaantumista, jonka jälkeen luetaan kytkimen tila. Odotusaika voi sovellukesta riippuen olla muutamia millisekunteja [9, s. 60-62].

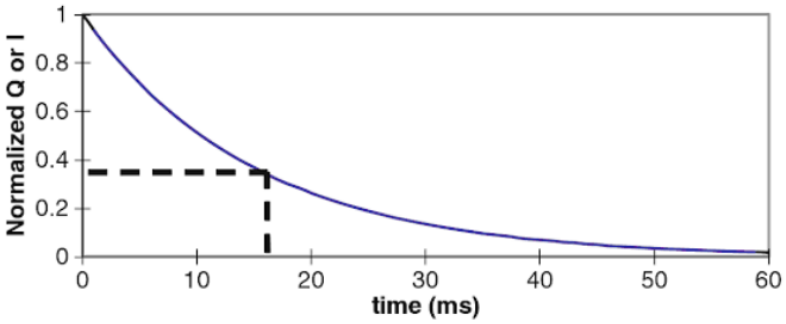
Toinen vaihtoehto värähtelyn vähentämiseen on alipäästösuodin. Alipäästösuotimen tarkoitus on suodattaa korkeat taajuudet pois signaalista ja päästää lävitseen vain matalat taajuudet, joita napin painalluskin edustaa. Yksinkertaisi alipäästösuodin koostuu vastuksesta ja kondensaattorista kuvan 8 tapaan. [10]



Kuva 8: *Kytkinvärähtelyä vähentävä kytkentä*

Kun kytkin S1 suljetaan, alkaa kondensaattori C1 purkautumaan vastuken R1 kautta. Kytkimen värähtely ei vaikuta paljoa kondensaattorin varaukseen, vaan kondensaattori tyhjenee muutamassa millisekunnissa komponentien arvoista riippuen.

Kondensaattorin purkautumisnopeus voidaan laskea, kun tiedetään alipäästösuotimen RC-aikavakio. Aikavakio saadaan kertomalla alipäästösuotimen resistanssin ja kapasitanssin arvot keskenään. Tulokseksi saadaan aika, jossa kondensaattori on joko latautunut tai purkautunut $1/e = 0{,}37$ alkuperäisestä arvostaan. Yleisesti voidaan olettaa, että kondensaattori on täysin latautunut tai purkautunut viiden RC-aikavakion aikana. [11, s. 413]



Kuva 9: *Normalisoitu RC-piirin kondensaattorin varaus. Katkoviivat merkitsevät yhtä RC-aikavakiota. [11, s. 413]*

Tässä työssä metronomin nappien kytkinvärähtelyn poisto toteutetaan ohjelmallisesti. Näin tarvitaan vähemmän komponentteja ja saavutetaan suurempi muokattavuus jälkikäteen.

3.3 Äänenvoimakkuuden säädin ja virtakytkin

Metronomin äänenvoimakkuus säädetään potentiometrillä, joka on kytketty mikrokontrollerin ulostulon ja vahvistimen väliin. Koska ihmisen kuuloalue on logaritminen, käytetään potentiometrinä logaritmista potentiometriä [12].

Metronomissa on nykytrendin vastaisesti vielä fyysinen kytkin, jolla laite kytketään pois päältä. Fyysiseen kytkimeen päädyttiin siksi, että se on varmatoimisempi sekä käyttäjän kannalta yksinkertaisempi käyttää. Fyysinen kytkin varmistaa myös, ettei ohjelmistovirheen sattuessa metronomi kytkeydy itsestään päälle esimerkiksi keskellä yötä ja aiheuta häiriötä.

3.4 Audio-out -liitin sekä kaiutin

Metronomia voidaan käyttää joko yksistään tai se voidaan kytkeä ulkoiseen vahvistimeen, kuten esimerkiksi kitaravahvistimeen tai stereovahvistimeen. Ulkoisen vahvistimen käyttö mahdollistaa suuremman äänenvoimakkuuden käyttämisen.

Nykytrendi kulutuselektroniikassa on liittimien vähentäminen ja sitä kautta AUX *(engl. Auxiliary)* -liittimen poisjättäminen esimerkiksi puhelimista [13]. Metronomiin on silti päätetty laittaa 3,5 mm naarasaudioliitin. Liittimeen voidaan liittää edellä mainittu vahvistin tai kuulokkeet hiljaiseen soittamiseen. Fyysiseen audioliittimeen päädyttiin, sillä mielestäni sen liitettävyys äänitekniikan ammattitason laitteisiin on vielä parempi kuin esimerkiksi bluetooth-laitteiden.

Metronomi sisältää myös sisäisen kaiuttimen, jolla saavutetaan laitteen käyttö ilman ulkoista vahvistinta langattomasti. Kaiutin on pieni, halkaisijaltaan 40 mm. Kaiuttimen nimellisteho on 3 W ja impedanssi 4 Ω.

Valinta ulkoisen AUX-lähdön ja sisäisen kaiuttimen välillä on automaattinen. Kun AUX-liittimeen ei ole kytketty mitään, sekä AUX-liittimeen että kaiuttimeen menevät vahvistetut signaalit. Koska AUX-liittimeen ei ole kytketty mitään, ei sen aktiivisena oleminen haittaa ja ääni kuuluu vain sisäisestä kaiuttimesta. Kun AUX-liittimeen liitetään 3,5 mm

plugi, avaa plugin työntäminen koskettimen AUX-liittimen sisältä ja näin mikrokontrolleri saa tiedon, että liittimeen on liitetty jotain. Tämän jälkeen mikrokontrolleri alasvetää sisäisen kaiuttimen ennen vahvistinta olevan signaalin ja näin ollen ääni kuuluu vain AUX-liittimen kautta.

3.5 USB

Metronomi sisältää microUSB-portin *(engl. Universal Serial Bus, universaali sarjaliikenneväylä)*. MicroUSB-portti on yleinen portti nykyelektroniikassa, jonka on kuitenkin syrjäyttämässä USB-C -tyyppinen portti, joka mahdollistaa suuremmat tiedonsiirtonopeudet sekä suuremman lataustehon [14]. MicroUSB on kuitenkin riittävä metronomin tarpeisiin.

Käyttäjän kannalta USB-portti on pääsääntöisesti latausta varten. Kuitenkin USB-portin datalinja on yhdistetty FTDI-piiriin, jolla mikrokontrolleri voidaan ohjelmoida uudelleen suoraan USB:n välityksellä.

4. TOIMINNALLISET LOHKOT

Metronomi sisältää useita erilaisia osakokonaisuuksia, joita tässä yhteydessä nimitetään toiminnallisiksi lohkoiksi. Kaikki lohkot ovat jotenkin yhteydessä toisiinsa ja luovat yhdessä toimivan metronomin. Seuraavissa luvuissa on esitetty tärkeimpien toiminnallisten lohkojen toimintaa ja suunnitteluun liittyviä seikkoja.

4.1 Mikrokontrolleri, ATmega328p

Koska metronomin toiminta on toteutettu digitaalisesti, tarvitsee se digitaalisen laskentayksikön. Laskentayksiköitä on yleisesti kahta erilaista, mikrokontrollereita ja mikroprosessoreita. Mikrokontrollerit ovat useissa tapauksissa yksinkertaisempia ja sisältävät enemmän oheislaitteita sisäänrakennettuna suoraan mikropiirille. Mikroprosessorit kykenevät laskemaan monimutkaisia laskuja nopeammin ja ovat joustavampia tekemään erilaisia laskutoimituksia kuin mikrokontrollerit. [15] Taulukossa 1 on esitetty suurimmat erot mikroprosessoreiden ja mikrokontrollereiden välillä.

Taulukko 1: *Mikroprosessorin ja mikrokontrollerin erot pääpiirteissään [15, s. 369]*

Mikroprosessori	Mikrokontrolleri
• Sisältää ALU:n *(engl. Arithmetic logic unit, Aritmeettislooginen yksikkö)*, yleiskäyttöisiä muistirekistereitä, muistipinoja ja osoittimia, ohjelmalaskureita, ajoituspiirejä ja keskeytyspiirejä • Monta tapaa siirtää dataa muistin ja CPU:n *(engl. Central processing unit, suoritin)* välillä • Muutama bitinkäsittelytapa • Vaatii enemmän ulkoisia komponenteja • Laitesuunnittelijan kannalta joustavampi toiminta • Yksi muisti koodille ja datalle • Harvempi komponentin pinneistä on monikäyttöisiä	• Voi sisältää sisäänrakennettuna useita oheislaitteita, kuten ROM *(engl. Read Only Memory, lukumuisti)*, RAM *(engl. Random Access Memory, Keskusmuisti)*, I/O-portteja *(engl, Input/Output, sisään/ulostulo)*, ajastimia, laskureita, ADC *(engl. Analog to Digital converter)* ja DAC *(engl. Digital to Analog Converter)* • Yksi tai kaksi tapaa siirtää dataa muistin ja CPU:n välillä • Monta erilaista bitin käsittelytapaa, kuten vähennys-, summaus- ja kertolaskuja, loogisia operaatioita ja liukulukuoperaatioita • Ulkoisten komponenttien tarve pienempi • Vähemmän muuntautuva ja erilaisiin toimiin erilaisia komponentteja • Erilliset muistit datalle ja koodille • Useampi komponentin pinneistä on monikäyttöisiä

Muunneltavuutensa, joustavuutensa ja nopeutensa vuoksi nykyisissä PC *(engl. Personal Computer, henkilökohtainen tietokone)* -tietokoneissa, matkapuhelimissa, kannettavissa tietokoneissa ja tableteissa käytetään mikroprosessoreita [16]. Nämä laitteet usein sisältävät jonkinlaisen käyttöjärjestelmän, jonka päällä ajetaan eri ohjelmia. Käyttöjärjestelmä ja laitteiston joustavuus ovat tarpeen, sillä laitevalmistaja ei voi tietää, millaisia ohjelmia käyttäjä aikoo suorittaa laitteella.

Mikroprosessoreita käytetään yleensä tietyn määritetyn tehtävän suorittamiseen. Määritetyllä tehtävällä tarkoitetaan tilannetta, jossa sisääntulojen ja ulostulojen suhde on jotenkin ennalta määritetty. Mikrokontrollereita käytetään paljon yksinkertaisissa laitteissa, kuten tietokoneen hiirissä ja näppäimistöissä, pesukoneissa, mikroaaltouuneissa, kauko-ohjaimissa ja muissa sulautetuissa järjestelmissä. [16]

Mikroprosessorit käyttävät yleensä paljon korkeampia kellotaajuuksia kuin mikrokontrollerit. Nykyisten mikroprosessoreiden kellotaajuudet ovat korkeimmillaan muutamia gigahertsejä, kun mikrokontrollerien kellotaajuudet rajoittuvat muutamiin satoihin megahertseihin. [16]

Rakennettavan metronomin laskentayksikkönä päädyttiin käyttämään Microchipin valmistamaa AVR tuoteperheen AVR ATmega328p mikrokontrolleria. Metronomi on riittävän yksinkertainen laite, ja siksi ei ole tarpeen käyttää kalliimpia mikroprosessoreita. Koska ulkoisia komponentteja tarvitaan vähemmän kuin mikroprosessoreissa, saadaan laitteesta fyysisesti pienempi.

ATmega328p valittiin käytettäväksi mikrokontrolleriksi, sillä sen ominaisuudet riittivät metronomin tarpeisiin. Tärkeimpinä valintakriteereinä pidettiin I/O-pinnien lukumäärää, muistien kokoa, digitaalisten kommunikaatiolohkojen sopivuutta näytön ohjaamiseen, saatavuutta ja hintaa. Yhteä hyvänä puolena ATmega328p -mikrokontrollerilla pidettiin myös sen yhteensopivuutta Arduino-ohjelmointiympäristön kanssa.

4.2 Akku

Koska metronomin on tarkoitus olla käytännöllinen ja kannettava, tarvitsee se myös langattoman virtalähteen. Laitteen virrankulutus on melko suurta, joten paristoja kuluisi merkittävä määrä metronomin käyttöiän aikana. Siksi laite sisältää uudelleenladattavan akun.

Akut tekevät laitteista osaltaan ympäristöystävällisempiä kuin paristot, sillä akullisissa laitteissa ei akkuja tarvitse fyysisesti vaihtaa kovinkaan useasti, vaan niistä purettu varaus voidaan palauttaa niihin johtamalla virtaa päinvastaiseen suuntaan akun purkuun nähden. Akkuteknologioita on monia erilaisia, muun muassa lyijyakku, nikkeli-rauta, nikkeli-cadmium, nikkeli-vety, litium-koboltti-oksidi, litium-ioni, litium-ioni-polymeeri, litium-ioni fosfaatti, litium rikki jne. Jokaisella tekniikalla on omat ominaisuutensa muun muassa jännitteen, energiatiheyden, itsepurkautuvuuden ja elinajan suhteen. [17]

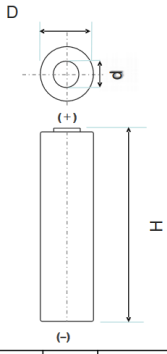
Metronomin akuksi valittiin melko yleinen litium-ioni –akku, muodoltaan ”18650”, joka on sylinterin muotoinen ja mitoiltaan nimensä mukaisesti sen halkaisija on 18 mm ja pituus 65 mm. Akun valintaan vaikuttivat muun muassa sen kilpailukykyinen hinta ja helppo saatavuus. Valittu akku on Panasonicin valmistama, mallinimeltään NCR18650GA ja kapasiteetiltaan 3 300 mAh. Käytettävä akku ei itsessään sisällä lataus- tai suojapiiriä, joten ne pitää toteuttaa itse laitteessa. [18]

Specifications

Rated capacity(1)		3300mAh
Capacity(2)	Minimum	3350mAh
	Typical	3450mAh
Nominal voltage		3.6V
Charging	Method	CC-CV
	Voltage	4.20V
	Current	Std. 1475mA
	Time	Std. 270 min.
Weight (max.)		48.0g
Temperature	Charge	10 to +45˚ C
	Discharge	-20 to +60˚ C
	Storage	-20 to +50˚ C
Energy density(3)	Volumetric	693 Wh/l
	Gravimetric	224 Wh/kg

(1) At 20˚ C (2) At 25˚ C
(3) Energy density is calculated using bare cell dimensions (without tube).

Dimensions



With tube	H	Max. 65.30mm
	D	Max. 18.50mm
	d	Max. 9.0mm

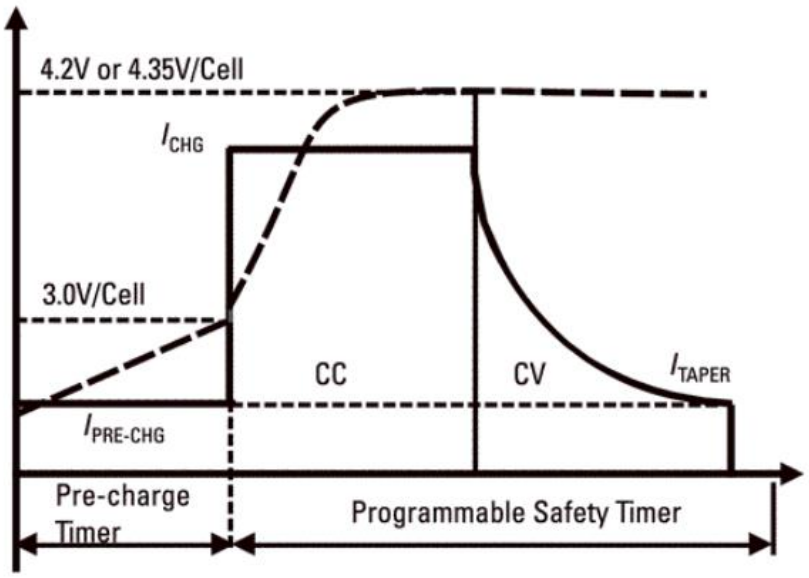
Kuva 10: *Panasonicin valmistaman NCR18650GA-akun tekniset tiedot [18]*

Kuvasta 10 nähdään tärkeimmät akun käyttöön liittyvät tiedot, muun muassa kapasiteetti, lataustapa, latausjännite ja -virta sekä käyttölämpötilat. Näitä tietoja voidaan pitää lähtökohtana valittaessa akulle sopivia latauksenhallinta- ja suojapiirejä.

4.2.1 Lataus

Laitesuunnittelussa yleistyvä trendi on, että porttien määrää pyritään vähentämään ja sitä kautta universaaleiden yleiskäyttöisten porttien, kuten USB–porttien, käyttö on lisääntynyt kuluttajalaitteissa [19]. Tämän takia myös rakennettavaan metronomiin lisätään vain yksi portti, MicroUSB, jonka kautta tapahtuu mahdollinen dataliikenne, ohjelmointi sekä lataus. Analoginen kaiutinulostulo päätettiin pitää omana liittimenään, joka mahdollistaa latauksen ja analogisen äänen ulosoton yhtä aikaa ilman erillisiä adaptereita.

Hyvin yleinen litium-ioni-akun lataustapa on CC-CV-lataus *(engl. Constant Current, Vakiovirta, Constant Voltage, Vakiojännite)* [20]. Tästä syystä myös metronomin latausteknikaksi valitaan kyseinen tekniikka. Kuvassa 11 on esitetty tyypillinen CC-CV –tekniikan mukainen latauskäyrä.



Kuva 11: *Esimerkki litium-ioni –akun latauskäyrästä CC-CV-tekniikalla [20, s. 50]*

Kuvan 11 pystyakselilla on esitetty akkukennon jännite sekä latausvirta ja vaaka-akselilla aika. Lataus voi alkaa mistä tahansa kohtaa käyrää, sillä akkua ei aina pureta loppuun tai ladata täyteen. Akun lataussyklin eteneminen esimerkkikuvassa 11 selitetään pääpiirteissään alla.

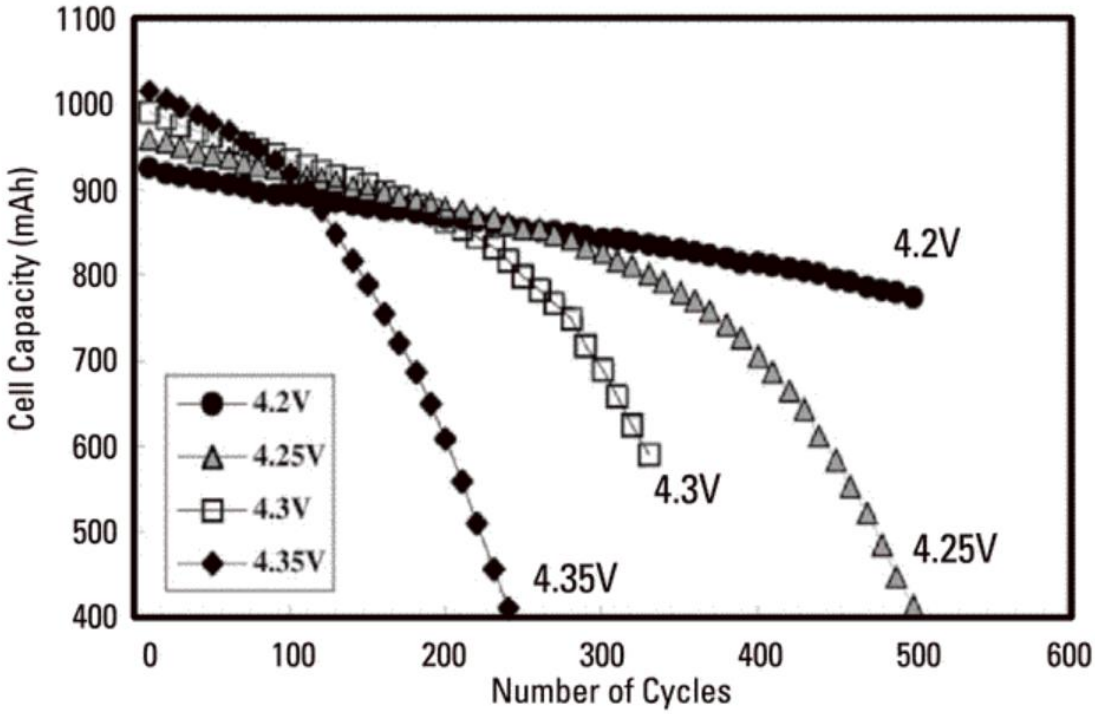
Mikäli akun jännite on päässyt purkautumaan alle tietyn pisteen, esimerkiksi 3 V, ladataan akkua hitaasti pienellä vakiovirralla, jotta tyhjää akkua ei rasitettaisi liikaa. Lataus nostaa akun jännitettä. Kun akun jännite on saatu nostettua riittävälle tasolle, nostetaan latausvirta ennalta määrätylle tasolle. Seuraavaksi siirrytään vaiheeseen CC, vakiovirta. Latausvirran pysyessä vakiona akkukennon jännite kasvaa latauksen edetessä. Kun akkukennon jännite on saavuttanut tietyn kennolle tyypillisen jännitteen, ei akkua enää ladata vakiovirralla vaan siirrytään lataamaan akkua vakiojännitteellä ja siirrytään vaiheeseen CV. Vakiojännitteellä ladattaessa melkein täyttä akkua latausvirta pienenee latauk-

sen lähentyessä loppuaan. Kun latausvirta on tippunut ennalta määriteltyyn arvoon, esimerkiksi kymmenesosaan vakiovirtalatausvirrasta, voidaan akun olettaa olevan täynnä ja lataus voidaan lopettaa. [20, s. 50-51]

4.2.2 Latauksen mitoitus

Mitoitettaessa akun latausta tulee tehdä kompromisseja akun eliniän ja latausnopeuden välillä. Jos halutaan akku, jonka elinikä lataussykleinä olisi mahdollisimman pitkä, pitäisi latausvirta mitoittaa riittävän pieneksi ja latausjännite oikeaksi valitulle akkukenotyypille. Jos taas halutaan nopeaa latausta, latausvirtaa pitää kasvattaa ja silloin akun eliniästä täytyy usein tinkiä. Akun eliniällä tässä yhteydessä viitataan lataussyklien määrään suhteessa akun kapasiteettiin. Mitä useampia lataussyklejä akku on kestänyt, sen vähemmän akku ottaa varausta ja sen kapasiteetti laskee. [20, s. 51-52]

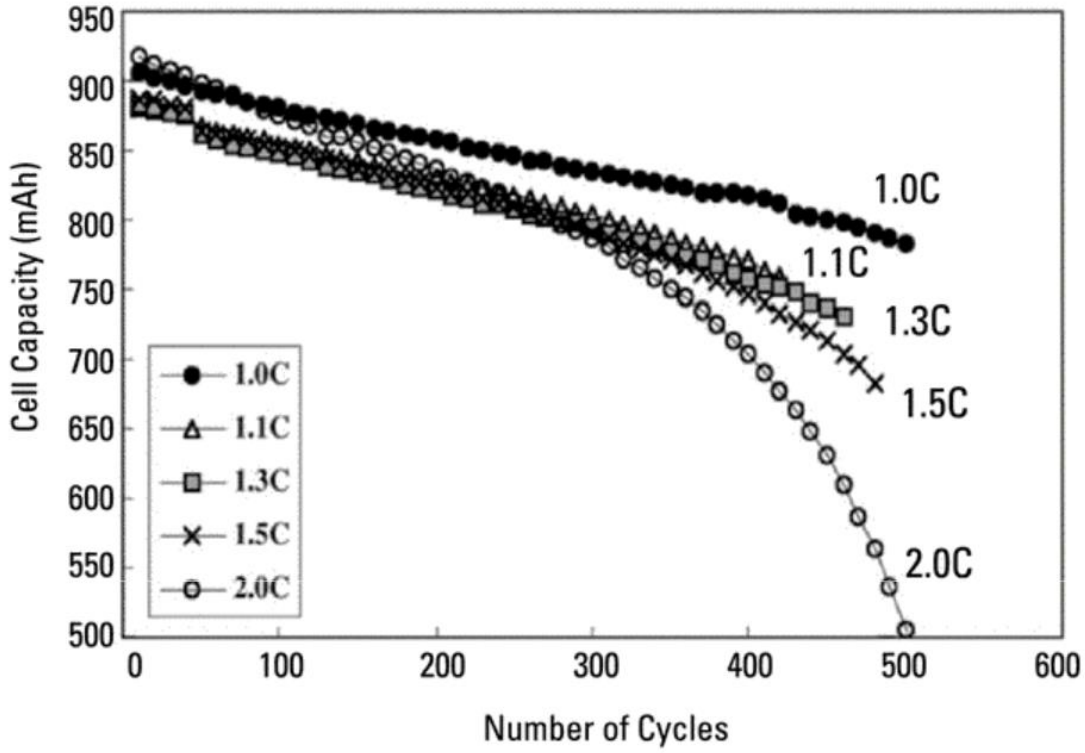
Yleensä mitä suurempi on akun kennojännite, myös kapasiteetti on suurempi. Kuitenkin suurempi latausjännite laskee akun elinikää. Akun katodimateriaali reagoi nopeammin elektrolyytin kanssa akun jännitteen noustessa. Nopean reagoinnin seurauksena akun kemikaalit reagoivat pidemmälle ja niiden kyky palautua estyy. Näin ollen energiaa varastoivien kemiallisten sidosten määrä vähenee ja kapasiteetti laskee. Akun latausjännitteen vaikutus akun kapasiteettiin on nähtävissä kuvassa 12. [20, s. 52]



Kuva 12: *Akun latausjännitteen ja lataussyklien määrän suhde akun kapasiteettiin Li-ioni akulle $LiCoO_2$ katodilla [20, s. 52]*

Akun latausjännitteen tarkkuus on hyvin tärkeää akun eliniän kannalta. Kuvasta 12 nähdään, että latausjännitteen kasvattaminen 4,25 V:stä 4,35 V:iin puolittaa akun eliniän. Kuitenkaan kapasiteettia ei alussa saada lisättyä kuin noin 10%. Latausjännitteen laskeminen 4,2 V:iin vähentää kapasiteettia vain vähän verrattuna hyötyyn, joka saadaan, kun akun elinikä pitenee merkittävästi. [20, s. 52]

Myös akun latausvirralla on suuri merkitys akun elinikään. Niin kuin latausjännitteen nosto, myös latausvirran nosto aiheuttaa akussa nopeammat kemialliset reaktiot. Nopeiden reaktioiden myötä anodista muodostuu metallista litiumia, joka ei enää reagoi akulle tyypillisellä tavalla vapauttaen sähköenergiaa käyttäjälle. Latausvirran vaikutus akun kapasiteettiin on havainnollistettu kuvassa 13. [20, s. 52-53]



Kuva 13: *Akun latausvirran ja lataussyklien määrän suhde akun kapasiteettiin Li-ioni akulle $LiCoO_2$ katodilla [20, s. 53]*

Kuvasta 13 nähdään, että latausvirran vaikutus akun elinikään ei ole niin merkittävä uudella akulla kuin latausjännitteen vaikutus akun kapasiteettiin. Kapasiteetti pysyy noin 10% sisällä riippumatta latausvirrasta ensimmäisten 250 lataussyklin aikana. Tämän jälkeen suuremmilla virroilla on havaittavissa merkittävää ja nopeaa kapasiteetin laskua.

Latausvirran yleensä suositellaan olevan 0,5C ja 1,0C välissä [20, s. 53]. C tässä asiayhteydessä tarkoittaa akun nimelliskapasiteettia, joka meidän metronomiimme valitsemallamme akulla on 3 300 mAh [18]. 1,0C latausvirta tarkoittaisi akullamme 3,3 A virtaa.

Huomionarvoista on, että edellä mainitut tapaukset kuvissa 12 ja 13 ovat yksittäistapauksia ja pätevät vain tietylle kennotyypille tietyissä olosuhteissa. Kuitenkin tulokset ovat sovellettavissa muillekin litium-akku -tyypeille ja eri olosuhteille.

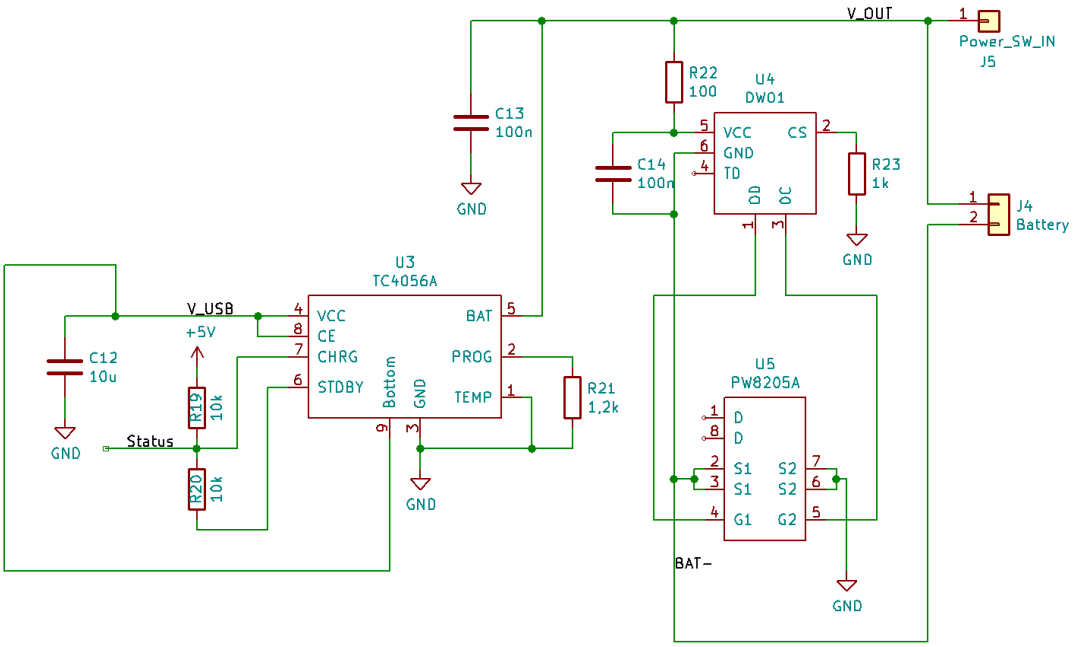
Metronomin latauksenhallinta toteutetaan valmiilla latauksenhallintapiirillä. Piiri on Nan-Jing Top Power ASIC Corporationin valmistama TP4056. Piiri on tarkoitettu CC-CV –lataukseen yhden akkukennon sovelluksissa. Kyseinen piiri valittiin, sillä sen CV –latausjännite on sama kuin valitun akun datalehdellä mainittu latausjännite, 4,20 V. Myös piirin

hinta on kilpailukykyinen, ulkopuolisten komponenttien tarve on vähäinen, sekä fyysinen koko on pieni. [18] [21]

TP4056 latausvirta on mahdollista säätää halutuksi ulkoisella vastuksella (R_{PROG}), kuitenkin suurimman latausvirran ollessa 1,2 A. Latausvirta voidaan laskea kaavalla: [21]

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG}} * 1200 \quad (V_{PROG} = 1\,V). \tag{1}$$

Latausvirran tulisi olla riittävän suuri, jotta lataamiseen ei kuluisi tarpeettoman paljon aikaa. Valitaan latausvirraksi 1 A. Tämän virran useimmat USB-laturit pystyvät antamaan, eikä latauksenhallintapiirimme joudu toimimaan äärirajoilla. Kaavalla 1 laskemalla saadaan $R_{PROG} = 1{,}2\ k\Omega$.



Kuva 14: *Metronomin latauskytkentä*

Kuvassa 14 on esitetty latauskytkentä kokonaisuudessaan. Linjaan V_USB tuodaan 5 V jännite suoraan USB-portista, liitin J4 on akun liitin ja liitin J5 on liitin päävirtakytkimelle. Latausvirran säätövastus R_{PROG} on vastus R21. Latauksen tilan edetessä TC4056A-piirin pinni 7 on sisäisesti vedetty maahan latauksen aikana ja pinni 6 on sisäisesti vedetty

maahan, kun akku on täyteen ladattu. Muissa tilanteissa pinnit ovat korkeaimpedanssisessa tilassa. Käyttäen hyväksi vastusten jännitteenjakokytkentää, vastuksilla R19 ja R20 voidaan mikrokontrollerille välittää tieto, onko akku latauksessa vai täyteen ladattu.

4.2.3 Suojaus

Koska laitteeseen valitussa akussa ei itsessään ole minkäänlaista suojapiiriä, pitää se toteuttaa akun ulkopuolella erillisenä lohkona. Markkinoilta on saatavilla useilta eri valmistajilta valmiita suojapiirejä. Muun muassa suurilta IC *(engl. Integrated Circuits, mikropiiri)* -valmistajilta, kuten Texas Instruments, Maxim, LTC, ON Semiconductor, Seiko, Renesas ja Fairchild, on saatavilla useita eri suojapiirejä eri akkutyypeille. [20]

Akun suojaus toteutetaan Fortune semiconductorin valmistamalla DW01-P-piirillä. Piiri suojaa akkua ylilataukselta, ylipurkamiselta sekä liikavirralta [22]. Piiriin päädyttiin, sillä sen sähköiset ominaisuudet ovat sopivat käyttökohteeseemme. Suojapiiri ei myöskään tarvitse paljoa ulkopuolisia komponentteja, joten sen viemä pinta-ala piirilevyltä on pieni.

4.2.4 Turvallisuus

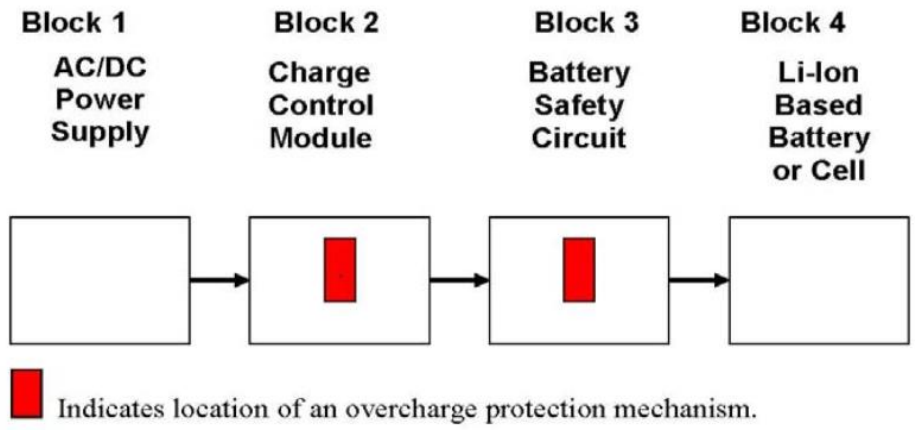
Nykyään käytettävien suuritehoisten akkujen, muun muassa litium-ioni akkujen, suojaaminen on erityisen tärkeää niiden sisältämän suuren energiamäärän ja herkästi reagoivien kemikaalien takia [23]. Sähköinen suojaus ylilataukselle ja muille sähköisille häiriöille on tärkeää, mutta myös fyysinen suojaus on otettava huomioon, sillä iskut akkuun saattavat aiheuttaa suuriakin vahinkoja.

Barsukov ja Qian vertailevat 18650 litium-ioni akun ja TNT *(engl. Trinitrotoluene, yleinen räjähdysaine)* -räjähteen energiasisältöjä kirjassaan ”Battery Power Management for Portable Devices” [20]. TNT vapauttaa energiaa noin 4,1 kJ/g ja litium-ioni akku noin 0,25 Wh/g, mikä vastaa noin 0,93 kJ/g. Tämä on noin neljäsosa TNT:n energiasta. Akku sisältää myös elektrolyytin sekä eristepolymeerejä, jotka voivat palaa ja reagoida onnettomuuden sattuessa. Tyypillisesti 18650-akussa on noin 10 g elektrolyyttiä ja 1,6 g poly-

meerieristeitä. Nämä tuottavat energiaa noin 280 kJ. 40 g painoiselle kennolle tämä tuottaa energiatiheydeksi noin 280/40 = 7 kJ/g. Kun tähän lisätään vielä sähköenergia, kennon energiatiheys, joka onnettomuudessa voi vapautua, on noin 8 kJ/g. Tämä on noin kaksinkertainen verrattuna TNT:hen. Esimerkkinä tarvitaan noin 5 kappaletta 18650-akkuja, jotta saadaan yhden käsikranaatin (U.S. M67, 180 g räjähdettä) verran räjähdysenergiaa. Tyypillisessä kannettavassa tietokoneessa on 3-6 kappaletta 18650 akkukennoja, joten sen räjähdysenergia voi olla suurempi kuin käsikranaatin. [20]

IEEE:n *(Institute of Electrical and Electronics Engineers)* standardit määrittelevät osittain, miten akun sisältävä laite tulee suunnitella [24]. Vaikkakaan tässä työssä ei ole tarkoitus suunnitella kaupallista tuotetta ja siksi valmistusmääräykset eivät ole tiukkoja, on silti hyvä ottaa huomioon joitain suunnittelusääntöjä ja standardeja, ainakin mikäli ne liittyvät laitteen ja käyttäjän turvallisuuteen.

Ylilataus voi aiheuttaa suurta vahinkoa laitteelle ja käyttäjälle. Tästä syystä olisi hyvä olla vähintään kaksinkertainen suoja ylilatausta vastaan. Mikäli toinen suojista vaurioituisi, toinen silti estäisi ylilatausta tapahtumasta. Alla kuvassa 15 on esitetty havainnekuva yhden virheen kestävästä ylilataussuojasta.



Kuva 15: *Yhden virheen kestävä ylilataussuoja [24, s. 64]*

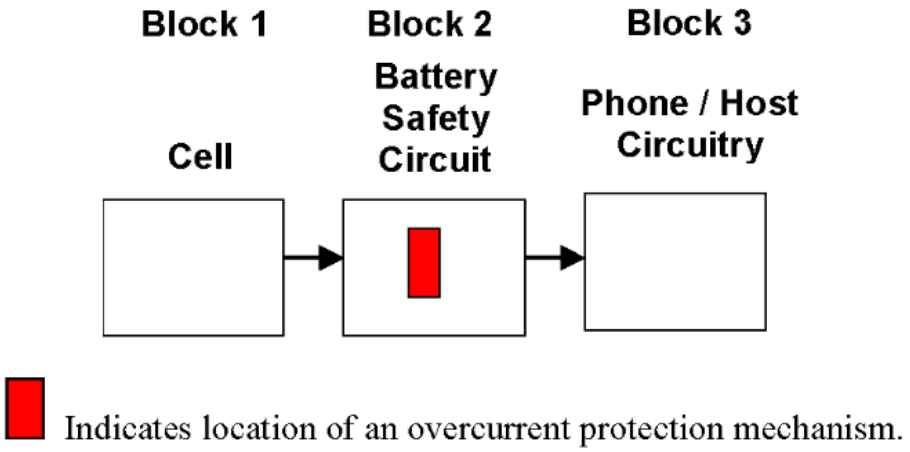
Mikäli virheitä tapahtuu molemmissa suojapiireissä, onnettomuus saattaa tapahtua. Näin voi käydä muun muassa suurien jännitepiikkien tai muun häiriön takia. Kuitenkaan laitesuunnittelussa ei voida ikinä ottaa kaikkea huomioon ja siksi laitteeseemme toteutetaan

kaksinkertainen ylilataussuoja. Latauspiiri TP4056 suojaa akkua ylilataukselta 4,2 V asti ja suojapiiri DW01-P katkaisee akun virran, mikäli akun jännite nousee yli 4,25 V.

Ylilatauksen seurauksena akun litium muuttuu metalliseksi litiumiksi ja saattaa riittävissä määrin aiheuttaa sisäisen oikosulun ja aiheuttaa lämpökarkaamisen (*engl. thermal runaway*). Metallinen litium voi myös reagoida elektrolyytin kanssa aiheuttaen lämpöä vapauttavan eksotermisen reaktion sekä lämpökarkaamisen. Lämpökarkaamisessa pieni lämpötilan nousu aiheuttaa suuremman lämpötilan nousun, joka nostaa lämpötilaa entisestään. Lopputuloksena on hallitsematon lämpötilan nousu sekä mahdollisesti räjähdys. [25, s. 59]

Alipurku ei ole akulle niin vaarallista kuin ylilataus, mutta silti se pitäisi ottaa laitesuunnittelussa huomioon. Jos alipurkua aiheutuu, silloin akkuun on varastoituneena vähemmän energiaa kuin ylilatauksen aikana, ja akku on siten vaarattomampi vahingoittuessaan. Alipurun aikana ei myöskään tapahdu lämpökarkaamista. Kuitenkin alipurku vahingoittaa pysyvästi elektrodeja, muun muassa liuottamalla kuparia. Mikäli syvän alipurun jälkeen akkua ladataan nopeasti uudestaan, saattaa akussa ilmetä samoja vaaratilanteita kuin ylilatauksen kohdalla aiheuttaen lämpökarkaamisen [25, s. 60-61].

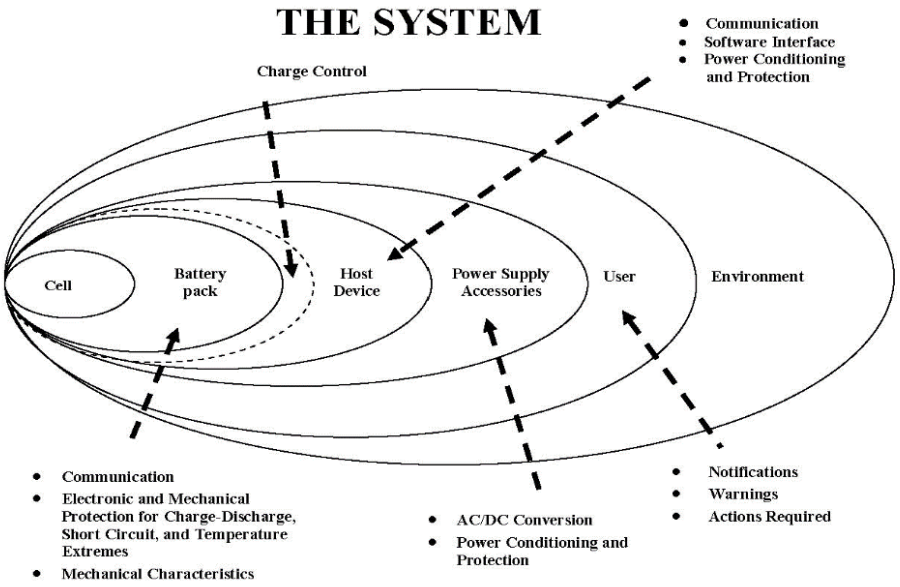
Suojapiiri DW01-P ohjaa kahta N-tyypin MOSFET:ä *(engl. Metal-Oxide-Semiconductor Field-Effect Transistor, Metallioksidi-puolijohdekanavatransistori)* ja ne katkaisevat akun käytön, mikäli akun jännite laskee alle 2,4 V. Seuraavaksi on esitetty IEEE:n havainnekuva akun syväpurkaus ja oikosulkusuojauksesta.



Kuva 16: *Ylipurku/oikosulkusuoja [24, s. 65]*

Piiri DW01-P suojaa akkua myös oikosululta, joka tapahtuu suojapiirin jälkeen. Mikäli akulta otetaan määritettyä enemmän virtaa, suojapiiri katkaisee virran syötön.

Yhteenvetona voidaan määrittää, että laite tulisi suojata niin, ettei normaalista käytöstä aiheudu vaaraa käyttäjälle, vaikka käytössä tapahtuisikin virheitä, kuten laite jätettäisiin lataukseen pitkäksikin aikaa. Laitteen pitää tässä tapauksessa osata katkaista lataus eikä käyttäjän tarvitse huolehtia siitä. Alla on esitetty kuva siitä, miten käyttäjä sijoittuu laitekokonaisuuteen nähden.



Kuva 17: *Havainnekuva laitteesta ja sen käyttäjästä [24, s. iv]*

Kuvasta 17 nähdään, että käyttäjä toimii omien tarpeidensa ja ympäristön sekä virransyötön välissä. Käyttäjä siis päättää koska haluaa laitettaan ladata. Virransyöttö puolestaan ohjaa laitetta, joka puolestaan ohjaa latauksenhallintapiiriä. Latauksenhallintapiiri lataa kennostoa, jossa on yksittäisiä kennoja. Käyttäjä ei siis suoraan hallitse latausta, vaan latausprosessi on jätetty automaation hoidettavaksi.

4.3 Vahvistin

Aluksi vahvistimena suunniteltiin käytettäväksi AB-luokan vahvistinta, joka olisi toteutettu suoraan valmistettavalle piirilevylle. Kuitenkaan suunnitellulla vahvistimella ei varhaisissa testeissä saavutettu riittävää vahvistusta suhteessa virrankulutukseen ja lämmöntuottoon. Tämän takia tutkittiin D-luokan vahvistimen käytön mahdollisuutta.

D-luokan vahvistin eroaa AB-luokan vahvistimesta siten, että ulostuleva vahvistettu signaali on ohjattu digitaalisesti PWM (*engl. Pulse-Width Modulation, Pulssinleveysmodulaatio)* -signaalilla, kun taas AB-luokan vahvistimen ulostuloa säädetään transistorien läpi kulkevaa virtaa säätämällä. Virran säätäminen aiheuttaa transistoreissa melko suuria tehohäviöitä ja AB-luokan vahvistimet eivät siksi ole optimaalisia kannettaviin laitteisiin. [26]

Pienen tutkimisen jälkeen löydettiin laitteeseen teknisesti sopiva valmis D-luokan vahvistinpiiri, OEP30Wx2. Kyseiselle piirille ei löydy kattavaa datalehteä, vaan tekniset tiedot on otettu Amazon-verkkokaupan tuotekuvauksesta. OEP30Wx2 on kaksikanavainen vahvistin, jonka vahvistukseksi voidaan valita joko 36 dB tai 26 dB [27]. Metronomin vahvistimen vahvistukseksi valittiin 26 dB, sillä sisäänmenona voidaan käyttää 5 V_{P-P} signaalia ja ulostulona voidaan suurimmillaan saada 24 V_{P-P}. Näin sisään menevää signaalia pitää vaimentaa vähemmän ja häiriösietoisuus on siten parempi.

Vahvistinta varten akulta saatava 4,2 V – 2,5 V jännite nostetaan step-up-muuntimella 12 V:iin, josta se reguloidaan lineaariregulaattorilla muulle järjestelmälle sopivaksi 5 V

jännitteeksi. Näin ollen järjestelmän käyttöjännitetasot pysyvät käytön aikana vakiona riippumatta akun varauksesta.

Vahvistin sisältää kaksi erillistä vahvistinyksikköä, jotka on tarkoitettu stereoäänelle, vasemmalle ja oikealle äänikanavalle. Metronomissa on kuitenkin vain monoääni eli yksikanavainen ääni, joten kahta erillistä vahvistinta voidaan käyttää ohjaamaan eri ulostuloja. Toinen vahvistin vahvistaa signaalin metronomin sisäiselle kaiuttimelle ja toinen vahvistin vahvistaa signaalin AUX-ulostuloon kuulokkeille tai ulkoiselle vahvistimelle.

Vahvistimen ulostulotaso mitattiin ja suurimmaksi ulostuloksi saatiin 23 $V_{P\text{-}P}$. Koska vahvistimen vahvistus on ennalta määritetty 26 dB, pitää sisään tulevan signaalin jännitettä rajoittaa, jotta signaalin leikkautumista ei tapahdu. Signaalin leikkautumisella tarkoitetaan ilmiötä, jossa sisäänmenosignaali on niin suuri, että vahvistin ei kykene vahvistamaan sitä vääristämättä, vaan jättää huippujännitteet ulostulonsa suurimpaan arvoon. Sisääntulosignaali rajoitetaan vastusten jännitteenjakokytkennällä, jossa toisena vastuksena toimii 50 kΩ logaritminen potentiometri. 26 dB vahvistus vastaa noin 20 kertaista vahvistusta, joten suurin sisääntulosignaali voi olla korkeintaan $V_{in_{P-P}} = \frac{23\,V_{P-P}}{20} = 1{,}15\,V_{P-P}$. Vastusten jännitteenjakosäännöllä laskemalla rajoitusvastuksn arvoksi saadaan noin 167 kΩ. Vastukseksi valitaan 180 kΩ vastus.

AUX-ulostulon vahvistimen vahvistusta määrittäessä pitää ottaa huomioon, että linjatasoinen äänisignaali on jännitetasoltaan paljon pienempi kuin sisäiselle kaiuttimelle menevä vahvistettu signaali. Useimmat ammattilaitteet on suunniteltu käyttämään sisään- ja ulostuloissaan +4 dBu jännitetasoa [28, s. 1299]. Jännitetaso voltteina voidaan laskea kaavalla

$$dBu = 20 \log \frac{V}{0{,}7745}, \tag{2}$$

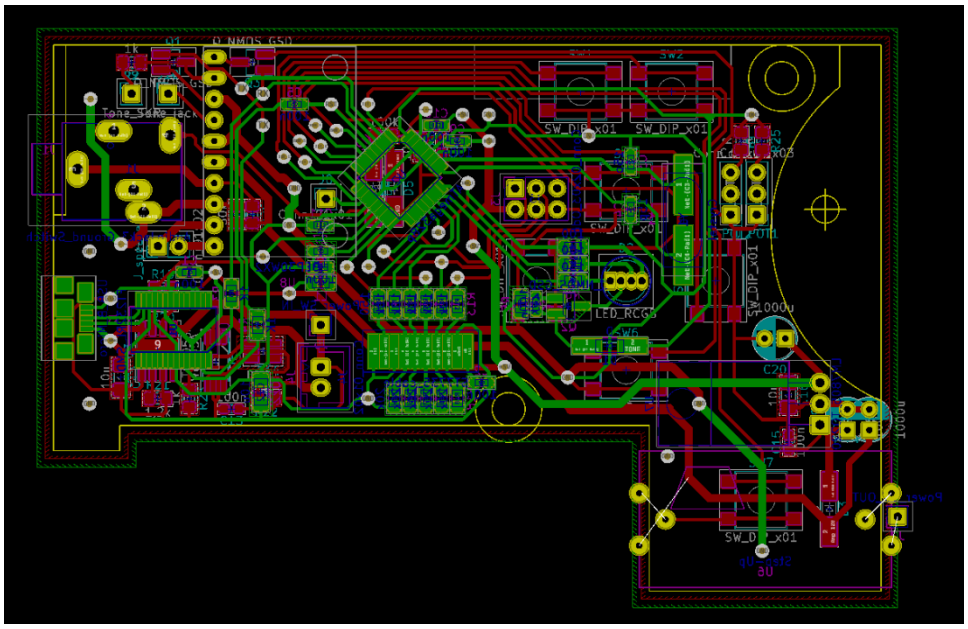
jossa *dBu* on jännitetason dBu-arvo, *V* on jännitetaso voltteina ja 0,7745 on referenssijännitetaso [28, s. 1299].

Jännitetasoksi *V* saadaan 1,227 V_P. Vahvistimelta ulostuleva signaali tulee tällöin olla 2,45 V_{P-P} ja täten vahvistimelle sisään menevän signaalitason suuruus voidaan laskea kuten sisäisen kaiuttimen tapauksessa. Sisäänmenosignaalin suuruudeksi saadaan 0,1227 V_{P-P}. Vastusten jännitteenjakosäännöllä voidaan myös tässä tapauksessa laskea rajoitusvastuksen suuruus, joksi saadaan 1,98 MΩ. Vastukseksi valitaan 2 MΩ vastus.

4.4 Piirilevy ja 3D-tulostettu runko

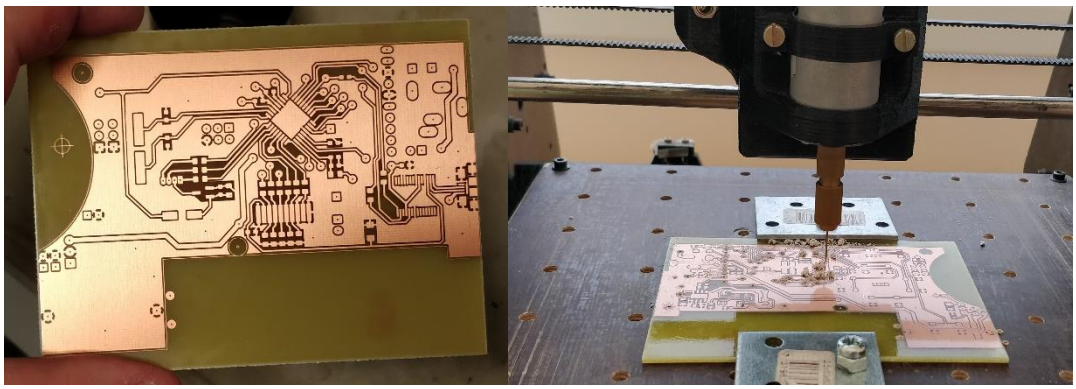
Suunnittelun yhtenä lähtökohtana oli saada kaikki komponentit sijoitettua samalle piirilevylle, jotta laitteen kokoaminen olisi yksinkertaista ja eliminoitaisiin liittimistä mahdollisesti aiheutuvat mahdolliset kontaktihäiriöt. Piirilevyä ja kotelointia suunniteltiin yhtä aikaa, jotta kaikki laitteen komponentit, kuten kaiutin ja akku, saadaan mahtumaan sopivasti piirilevyn kanssa.

Piirilevy suunniteltiin KiCAD-piirilevynsuunnitteluohjelmalla. Suunnittelussa käytettiin kaksipuoleista piirilevyä, sillä alussa arvioitiin kuparivetojen määrä niin suureksi, ettei yksipuoleinen piirilevy riittäisi. Kun piirilevyn ulkomitat oli saatu varmistettua, sijoitettiin ensin paikkakriittiset komponentit, kuten napit, AUX-jakki, microUSB-liitin, RGB-led ja näytön liitinrima. Tämän jälkeen sijoitettiin muut komponentit ja vedettiin kuparivedot komponenttien välille. Valmiin piirilevyn layout on nähtävillä kuvassa 18.



Kuva 18: *Valmis piirilevy KiCAD-ohjelmassa ilman kuparitäyttöjä. Punaiset ja vihreät kuparivedot ovat piirilevyn eri puolilla.*

Piirilevyyn pyrittiin tekemään mahdollisimman vähän läpivientejä, sillä reiät tuovat aina yhden heikon osan lisää piirilevyn toimintaan ja ovat työläitä porata ja juottaa käsin, mikäli niitä on monta. Piirilevyn reiät porattiin kuitenkin tietokoneohjatusti 3D-tulostimeen tehdyllä porauspäällä, joka on nähtävissä alla.



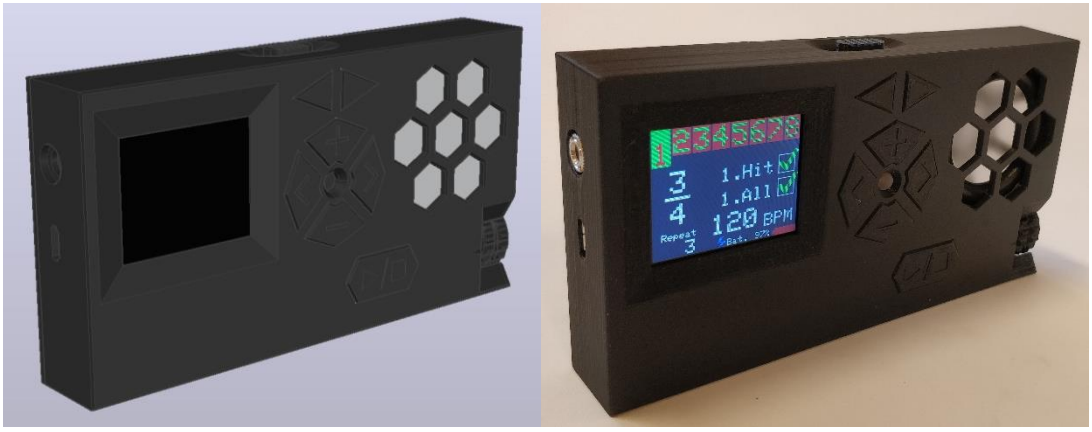
Kuva 19: *Piirilevy syövytyksen jälkeen (vasemmalla) ja reikien porausta (oikealla)*

Itse piirilevyn valmistus koostui valotusmaskien tulostamisesta, piirilevyn valottamisesta ja syövyttämisestä, reikien poraamisesta ja levyn äärimittojen leikkaamisesta. Piirilevyn valmistuksen jälkeen komponentit juotettiin paikoilleen ja kaikkien erillisten lohkojen toiminta varmistettiin pienillä ja yksinkertaisilla erillisillä ohjelmakoodeilla.

Metronomin kotelointi on merkittävin osa sitä kokonaisuutta, jonka käyttäjä näkee ja tuntee. Koteloinnin päätehtävä on kuitenkin suojata ja pitää paikoillaan laitteen sisäisiä komponentteja. Koteloa lähdettiin suunnittelemaan AutoCAD-suunnitteluohjelmalla. Kotelosta haluttiin tehdä jämäkkä mutta kuitenkin kevyt, joten 3D-tulostaminen koettiin tässä vaiheessa hyväksi vaihtoehdoksi kotelon fyysiselle toteutukselle. 3D-tulostimen käyttö mahdollistaa myös niin sanottujen negatiivisten muotojen tekemisen, sillä 3D-tulostamisessa ei tarvitse ottaa huomioon sitä, että lähteekö kappale pois muotista. Muotoja on havainnollistettu alla.

Kuva 20: *Yksinkertaiset havainnekuvat, kun muoto on negatiivinen (vasemmalla) ja positiivinen (oikealla)*

Koska 3D tulostamisessa ei käytetä muotteja, voidaan kappaleista tehdä monimutkaisia ilman että pitää huolehtia niiden pois saamisesta muotista. Kappaleen muoto on silloin negatiivinen, kun muotti sulkee sen kokonaan tai osittain sisäänsä ja kappaleen pois saaminen muotista on mahdotonta rikkomatta tai muovaamatta muottia. Perinteisessä valutekniikassa muottien tulee aina olla positiivisia, jotta kappale saadaan ehjänä pois. Mikäli halutaan kuitenkin tehdä negatiivisia muotoja, pitää muotti valmistaa useasta positiivisen muotoisesta muotista, jotka poistetaan eri aikaan. [29]



Kuva 21: *Kotelon suunnittelua AutaCAD-ohjelmassa ja valmis 3D tulostettu kotelo*

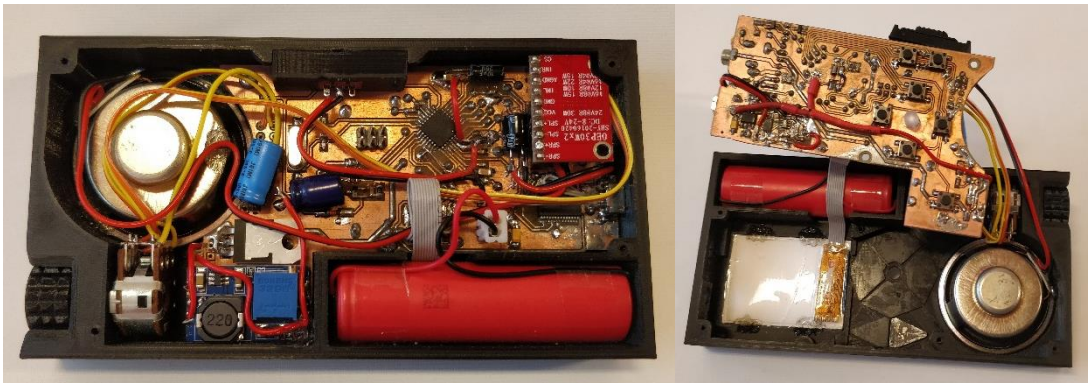
Varsinainen kotelo koostuu vain kahdesta osasta, itse rungosta ja takakannesta. Jotta metronomi ei näyttäisi vain laatikolta, haluttiin siihen tuoda pieniä koristekuvioita. Tästä johtuen takakanteen suunniteltiin etupuolen kaiutinritilää muistuttava kennorakennekuvio, joka on kuvattuna kuvassa 22. Kuviointi parantaa myös laitteeseen tarttumista, joka huomattiin vasta kotelon valmistuksen jälkeen.

Kuva 22: *Metronomin takakannen koristekuvio*

Kotelo 3D-tulostettiin Geeetech i3 pro B-tulostimella. Materiaalina käytettiin mustaa PLA *(engl. Polylactic Acid, polyaktidi)* -filamenttia eli muovilankaa, jonka paksuus oli 1,75 mm. Kotelon ensimmäinen versio tulostettiin tiedostaen, että se ei tulisi vielä olemaan lopullinen versio. Testiversio päätettiin tulostaa, sillä siihen olisi helppoa kokeilla kaikkien osien sopivuutta ja tehdä muutoksia esimerkiksi kaivertamalla ja siirtää nämä muutokset sen jälkeen 3D-malliin. Testiversio tulostettiin paljon karkeammilla tulostusasetuksilla kuin mitä lopullinen kotelo. Karkeilla asetuksilla saavutettiin paljon lyhyempi tulostusaika. Testiversion kerrosvahvuus oli 0,3 mm ja suuttimen halkaisija 0,4 mm. Lopullisen version

kerrospaksuus oli 0,1 mm ja suuttimen halkaisija 0,2 mm. Näin saatiin tarkka ja sileä lopputulos lopulliseen versioon.

Metronomiin tulostettiin myös painonapit, joihin suunnitteluvaiheessa tehtiin upottamalla kuviot. Kuviot kertovat käyttäjälle napin tarkoituksen. 3D-tulostaminen mahdollistaa erilaisten nappien tekemisen kotioloissa ja niitä voidaan tulostaa useita erilaisia ja päättää minkälaiset napit sopivat kyseiseen käyttötarkoitukseen.



Kuva 23: *Valmiin kootun metronomin sisältö. Piirilevy kuvattuna kummaltakin puolelta.*

Kuten kuvasta 23 nähdään, valmiin metronomin sisältö ei ole esteettisesti kovin kaunis ja hyppylankoja löytyy muutamia. Suurin osa johdoista on kuitenkin eri komponenttien välillä kulkevia johtoja, kuten äänenvoimakkuuden säätimelle, kaiuttimelle sekä akulle. Näitä johtoja voisi jatkossa lyhentää ja näin saataisiin hieman häiriönsietoisempi ja kauniimpi lopputulos.

5. TESTAUS

Metronomin rakentamisen ja ohjelmoinnin jälkeen valmis laite testattiin toimivuuden, määrittelyiden täyttämisen, akunkeston ja -latauksen, sekä mahdollisten suunnitteluvirheiden osalta. Joitain virheitä pyrittiin korjaamaan ja jotkin pienet virheet koettiin niin mitättömiksi, että niiden korjaaminen ei toisi laitteelle lisäarvoa siinä määrin mitä niiden korjaaminen kuluttaisi resursseja.

5.1 Huomiot ja tuotteen laatua heikentävät ominaisuudet

Metronomi saatiin toimimaan melko helposti heti komponenttien juottamisen jälkeen. Laite ladattiin ja lataus sujui ongelmitta. Laitteen käytössä ei alussa esiintynyt merkittäviä vikoja. Kuitenkin, kun laitetta oli käytetty jonkin aikaa, sammutti laite itsensä, jos äänenvoimakkuus oli suurella ja kaiuttimesta kuului ääntä. Akku ei vielä tässä vaiheessa ollut tyhjä, mikä todennettiin yleismittarilla mittaamalla akun jännite, joka oli 3,95 V. Vikaa tutkittiin ja syyksi selvisi akun suojapiirin oikosulkusuojan aktivoituminen. Kun kaiutin soi täydellä teholla, kulkee sen läpi teoriassa korkeintaan $I\_speaker = 12\,V/4\,\Omega = 3\,A$. Koska akku on nimellisjännitteeltään 3,6 V ja kaiuttimen vahvistimen käyttöjännite 12 V, ottaa step-up-muunnin toiminnastaan johtuen akulta teoriassa $12\,V/3{,}6\,V * 3A = 10\,A$. Tällainen piikkivirta aktivoi oikosulkusuojan ja sen saa vapautettua vasta sammuttamalla laitteen virtakytkimestä.

Vika korjattiin asentamalla suuri kondensaattori lähelle step-up -muuntimen sisääntuloa. Tämä vähensi sammuilua, mutta ei korjannut vikaa täysin. Seuraavaksi asennettiin pieni kondensaattori akun suojapiirin DW01-P virranmittauspinnin ja maatason välille ja pieni vastus virranmittauspinnin ja virranmittauskohdan välille. Koska virran muutos saa aikaan nopean jännitteenmuutoksen virranmittauspinnillä ja kondensaattori toimii RC-suotimena yhdessä vastuksen kanssa, eivät lyhyet virtapiikit ehdi laskemaan virranmittauspinnin jännitettä, jolloin oikosulkusuoja ei aktivoidu.

Akun suojapiireihin kajoaminen ei yleensä ole suotavaa. Kuitenkin laitteemme ottaa vain muutaman millisekunnin virtapiikin suhteellisen harvoin. Rajoittava tekijä virran suuruudessa on akun suojaustransistorilla, joka metronomissamme on Hottech Semiconductorin valmistama kaksikanavainen N-tyypin MOSFET mallia 8205A. Transistorin suurin piikkivita on datalehden mukaan 20 A, joten sen pitäisi kestää laitteemme piikkivirta tuhoutumatta [30]. RC-suotimen lisäämisen jälkeen metronomi ei enää sammunut käytön aikana. Suojapiiri toimi kuitenkin vielä oikosulkuja vastaan, mikä kokeiltiin oikosulkemalla käyttöjännite maatasoon.

Toinen laatuun vaikuttava asia on akun varauksen seuranta, joka ei toimi aivan halutulla tavalla. Koska metronomimme seuraa vain akun jännitettä, ei esimerkiksi latauksen CV-vaiheessa saada latauksesta kattavaa tietoa. Myös iskutiheys ja äänenvoimakkuus vaikuttavat akun jännitteeseen, sillä akulla on aina pieni sisäinen resistanssi. Kun virta akulta kasvaa, sen jännite laskee. Akun jäljellä oleva kapasiteetti ei myöskään ole suoraan verrannollinen akun jännitteeseen, joskin akun lepojännite antaa melko hyvän arvoin akun jäljellä olevasta kapasiteetista.

Akun varausta pitäisi seurata mittaamalla latausjännitettä ja -virtaa, sekä akun purkausjännitettä ja -virtaa. Kun tiedetään myös, koska akku on ladattu täyteen ja akku on kulunut loppuun, voitaisiin näistä tiedoista laskea tarkempi akun varaus. Näiden tietojen kerääminen mikrokontrollerilla kuluttaisi melko paljon laskenta-aikaa ja metronomin tapaisissa aikakriittisissä järjestelmissä tämä ei ole kovin suotavaa. Atmega328p:n rinnalle voitaisiin liittää jokin pienempi mikrokontrolleri, esimerkiksi Attiny85, joka hoitaisi akun tilan seuraamisen ja välittäisi tiedon päämikrokontrollerille pyydettäessä. Tällaista akun tilan seurantajärjestelmää ei kuitenkaan lähdetty toteuttamaan, vaan akun tilaa arvioidaan vain jännitteen perusteella.

Kolmas laatuun vaikuttava tekijä on vahvistimen aiheuttama melko suuri kohina AUX-ulostuloon. Tätä kohinaa ei huomaa suurilla äänenvoimakkuuksilla, mutta hiljaa soitettaessa kohina on helposti kuultavissa ja saattaa haitata käyttäjää. Sama kohina on kuultavissa myös sisäisestä kaiuttimesta, mutta ei yhtä voimakkaana.

Neljäntenä laatuun vaikuttavana asiana huomattiin näytön himmenemisen aiheuttavan häiriööääniä äänisignaalilinjaan. Häiriööääni syntyy, kun näyttöä himmennetään mikrokontrollerin PWM-ohjauksella. Häiriötä voitaisiin yrittää pienentää nostamalla PWM-taajuutta kuuloalueen ulkopuolelle. Ongelmaa ei yritetty korjata, sillä PWM-ohjaimen perustaajuuden muuttaminen saattaa vaikuttaa muihinkin taajuusriippuvaisiin asioihin, kuten äänen tuottamiseen käytettyyn ohjelmakoodin ”tone()” funktioon.

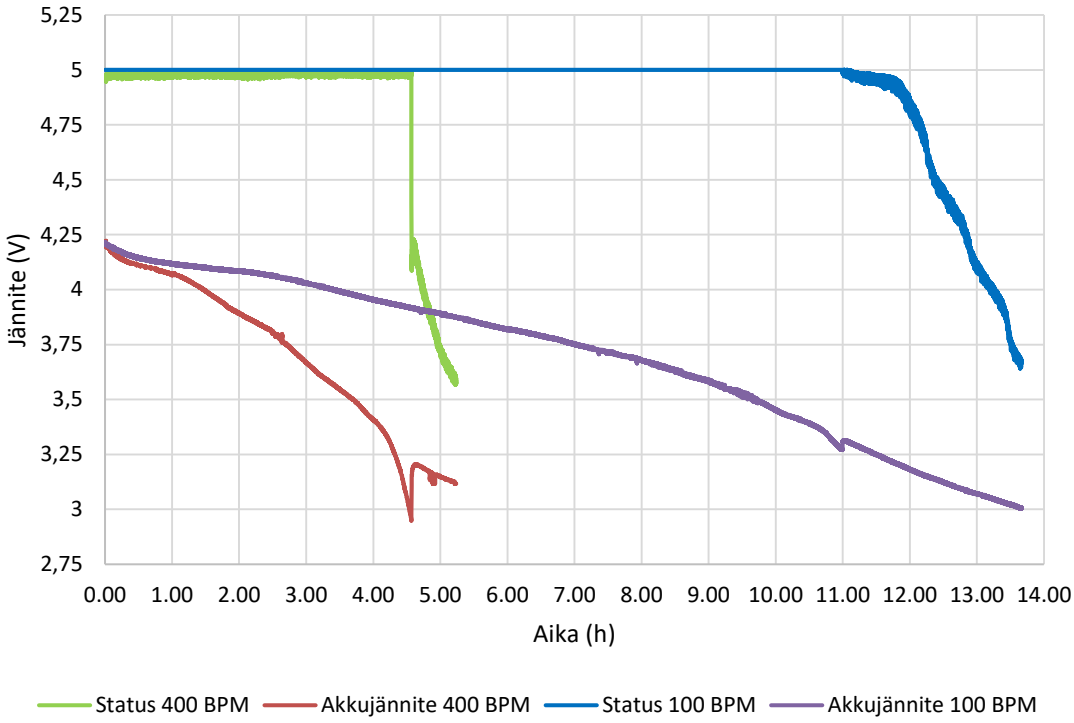
5.2 Lataus ja varauksen purku

Mittalaitteistona käytettiin Arduino MEGA-kehitysalustaa, johon oli liitetty reaaliaikakellopiiri, virranmittauspiiri sekä SD *(engl. Secure Digital)* -muistikortinlukija. Reaaliaikakellona käytettiin DS3231-mikropiiriä, virran mittaamiseen käytettiin ACS712-virranmittauspiiriä. Mitatut tiedot tallennettiin SD-kortille. Jännitteiden suuruudet mitattiin suoraan Arduino MEGA:n omilla analogi-digitaali -muuntimilla.

5.2.1 Purkauskoe

Ennen purkauskoetta akku ladattiin täyteen. Purkauskokeessa mitattiin akun jännitettä, sekä latauksen tilan ilmoittavan pinnin jännitettä. Pinnin jännitteen tulisi olla 0 V latauksen aikana, 2,5 V täyteen ladattuna, mutta laturiin kytkettynä ja 5 V akun purkamisen aikana. Purkauskoe suoritettiin kaksi kertaa, kumpikin eri asetuksilla. Ensimmäisellä purkauskokeella pyrittiin simuloimaan ääritilannetta, johon metronomi mahdollisesti joutuisi. Äänenvoimakkuus oli säädetty suurimmaksi mahdolliseksi ja iskutaajuudeksi 400 BPM. Toisella purkauskokeella pyrittiin simuloimaan enemmän normaalia käyttöä. Äänenvoi-

makkuus oli tässäkin kokeessa asetettu suurimmaksi mahdolliseksi, mutta iskutaajuudeksi oli asetettu 100 BPM. Jännitteet mitattiin ja kirjattiin 2 sekunnin välein. Purkauskokeen tuloksista koottiin alla nähtävä kuvaaja.



Kuva 24: *Purkauskokeen tulokset, kun metronomia käytetään suurimmalla äänenvoimakkuudella 400 BPM ja 100 BPM*

Purkauskokeiden aikana akun alijännitteen suojapiiri ei aktivoitunut kertaakaan ennen kuin metronomi lopetti äänen pitämisen. Äänen lopetuskohta on havaittavissa kuvasta 24 siitä kohdasta, josta status-pinnin jännite laskee alle 5 V. Koska status-pinni on purkauksen aikana ylösvedetty mikrokontrollerin käyttöjännitteeseen, voidaan status-pinnin jännitteen arvioida edustavan myös mikrokontrollerin ja koko muun 5 V järjestelmän jännitettä. Metronomin käyttöajan lähestyessä loppua akun jännite laskee niin alhaiseksi, ettei step-up -muunnin pysty antamaan ulos enää 12 V jännitettä vaan ulostulojännite romahtaa, jolloin myöskään 5 V regulaattori ei toimi halutulla tavalla ja alkaa leikkaamaan jännitettä enemmän.

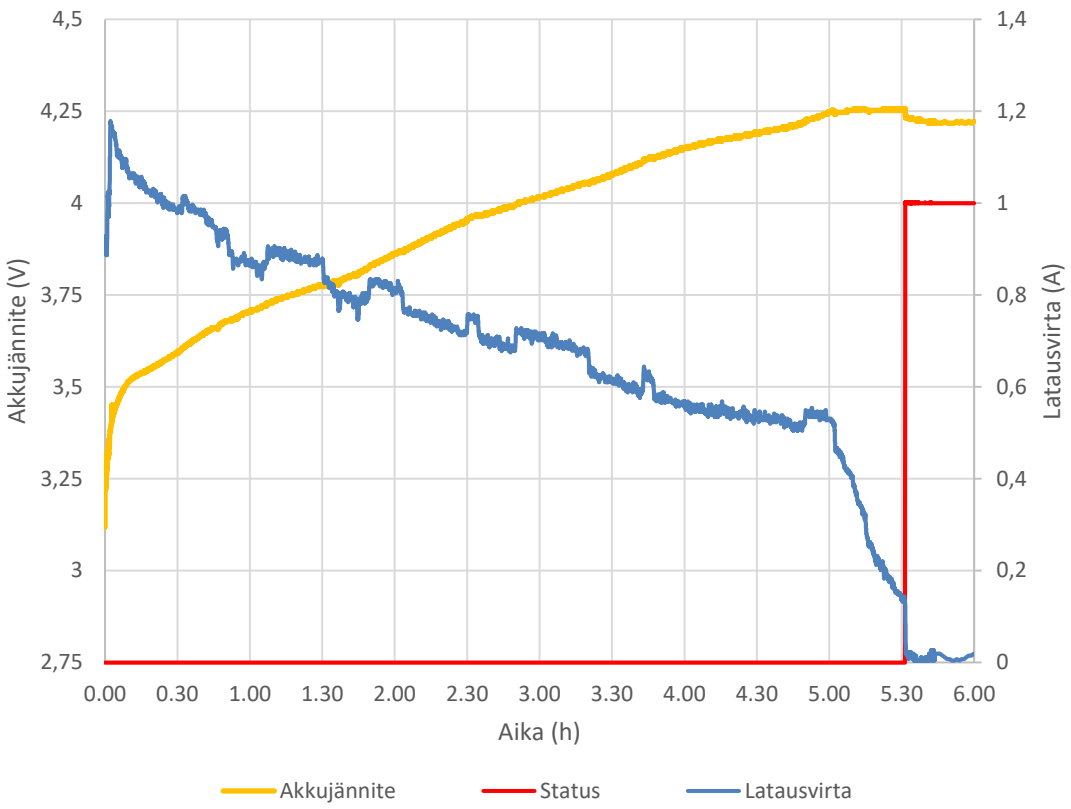
Kuvan 24 kuvaajat 100 ja 400 BPM iskutaajuuksilla eroavat muutoinkin kuin vain ajallisesti. 400 BPM status-kuvaaja laskee rajusti akkujännitteen laskettua alle tietyn pisteen,

kun taas 100 BPM kuvaaja laskee rauhallisemmin. Myös Akun jännite laskee matalammalle 400 BPM kuvaajassa. Arvelen tämän johtuvan akun sisäisistä resistansseista ja akun palautumisnopeudesta. Suurimmalla äänenvoimakkuudella yksi tikitys vie pulssimaisesti virtaa noin 10 A. Kun tällaisia pulsseja on tiheästi, ei akku ehdi palautua ja sen jännite laskee nopeammin, kuin harvoin virtaa otettaessa. Tämä ilmiö on nähtävissä myös 400 BPM kuvaajan akkujännitteessä, jossa tapahtuu suurempi akun palautuminen ja jännitteen nouseminen, kuin 100 BPM kuvaajassa, kun soitto loppuu.

Purkauskokeen tuloksena nähdään, että normaalikäytössä metronomin käyttöaika on noin 11 tuntia ja suurimmassa rasituksessakin noin 4 tuntia 30 minuuttia. Tämän jälkeen äänen tuottaminen ei enää onnistu, mutta kaikki muu toiminnallisuus, kuten asetusten säätö, onnistuu.

5.2.2 Latauskoe

Kun akku oli purettu, suoritettiin latauskoe. Latauskokeessa mitattiin latausvirtaa, akun jännitettä sekä latauksen tilan ilmoittavan pinnin jännitettä. Lataus suoritettiin USB-liittimen kautta verkkovirtamuuntajalla, tutummin puhelimen laturilla, jonka suurin ulostulovirta on tyyppikilven mukaan 2 A ja jännite 5 V. Latauskokeen tulokset on koottu kuvaan 25.



Kuva 25: *Latauskokeen tulokset*

Latausvirtakuvaajasta on nähtävillä vakiovirta (CC) ja vakiojännite (CV) latausalueet. Kuitenkin vakiovirta-alue ei näyttäisi olevan vakio, jonka arvelen johtuvan latauspiirin TP4056 lämpenemisestä. Vakiovirta-alue on noin 0 – 5 h ja vakiojännitealue on noin 5 h – 5 h 30 min. Vakiojännitealue toimi odotetulla tavalla pitäen akun jännitteen vakiona ja pienentäen latausvirtaa latauksen edetessä. Lataus päättyi noin 5h 30 min latauksen aloittamisesta, joka on nähtävissä latauksen tilan ilmoittavan pinnin jännitteen nousemisena ja latausvirran laskemisena lähelle nollaa. Kuvassa 25 latauksen tilan ilmoittava ”status” pinnin jännitearvo on skaalattu siten, että se mahtuisi jännite- ja virtakuvaajien kanssa samaan kuvaajaan. Oikealta akselilta luettaessa latauksen tilan ilmoittava ”status” pinnin arvo 1 vastaa jännitearvoa 2,5 V ja arvo 0 vastaa arvoa 0 V.

Tyhjän metronomin lataamiseen kuluu noin 5 tuntia 30 minuuttia, joka on nykyään melko pitkä aika verrattuna nykyisen kulutuselektroniikan latausaikoihin, joissa on samaa kokoluokkaa oleva akku. Nykyisten älypuhelimien akkukapasiteetti on 3000 – 4000 mAh ja

latausajat vaihtelevat valmistajasta riippuen puolestatoista tunnista kahteen tuntiin [31]. Toisaalta, kuten kohdassa 4.2.2 aihetta käsiteltiin, hidas lataus ei kuluta akun elinikää yhtä paljon kuin nopea lataus, joten hitaampi lataus on akun kannalta parempi.

6. OHJELMAKOODI

Metronomin ohjelmakoodi on hyvin laaja kokonaisuus eikä sen toiminnallisuuden käsitteleminen kokonaisuudessaan ole mielekästä tässä työssä. Tässä luvussa käsitellään vain ohjelmakoodin keskeisimpiä ja tärkeimpiä kohtia, jotka vaikuttavat merkittävästi laitteen toimintaan.

Ohjelmakoodi päätettiin toteuttaa Arduino ohjelmointiympäristössä, sillä Arduinon ohjelmoiminen on melko helppoa ja käyttöön löytyy paljon valmiita kirjastoja. Arduinon ohjelmointikieli perustuu C ja C++ ohjelmointikieliin ja on yhteensopiva näiden kielien kanssa. Itse ohjelman kirjoittaminen suoritettiin Microsoftin Visual Studiolla, johon oli asennettu VisualMicro-lisäosa Arduino yhteensopivuuden saamiseksi.

Jotta metronomin ohjelmoiminen onnistuisi jatkossa pelkän USB-portin kautta, pitää ATmega328p-mikrokontrolleriin polttaa käynnistyksenlataaja *(engl. bootloader)*, joka on lyhyt ohjelmakoodi. Käynnistyksenlataaja mahdollistaa ohjelmakoodin ohjelmoimisen laitteelle ilman ohjelmointilaitetta suoraan FTDI piirin ja USB:n kautta. [32] Arduinon käynnistyksenlataaja poltettiin metronomiin ArduinoIDE ohjelmalla ja AVRISP mkII-ohjelmointilaitteella. Koska metronomissamme ja Arduino NANO:ssa on sama mikrokontrolleri [33], poltettiin metronomiin Arduino NANO:n käynnistyksenlataaja. Käynnistyksenlataajan kanssa metronomi toimii ohjelmointitilanteessa kuin täysiverinen Arduino NANO.

Metronomin osien toimintaa testattiin Arduinon omien esimerkkien sekä käytettyjen kirjastojen esimerkkien avulla. Kun oli varmistuttu metronomin laitteiston toiminnasta, voitiin kirjoittaa varsinainen metronomin ohjelmakoodi.

Koska metronomin ohjelmakoodi on melko pitkä, ei sen uudelleenkirjoittaminen tähän työhön ole lukijan kannalta mielekästä. Ohjelmakoodi on kokonaisuutena tämän työn liitteenä (Liite B: Ohjelmakoodi) ja kyseiseen liitteeseen viitataan tässä luvussa useasti.

Funktio ”void button(int button_number)” käsittelee yhden napin painamiseen liittyvän toiminnallisuuden, riippuen funktiolle parametrinä annetusta napin numerosta. Button-funktiota pollataan koko ajan, eli sitä kutsutaan jokaisella ohjelman suorituskierroksella. Näitä suorituskierroksia on useita tuhansia sekunnissa. Jokaisen napin toiminnan pääperiaate on sama. Kun nappi painetaan pohjaan, alkaa mikrokontrolleri laskemaan aikaa. Kun nappi vapautetaan, pysäytetään laskuri ja laskurin lukua verrataan ennalta määrättyihin arvoihin. Laskurin arvosta riippuen saadaan tieto, että oliko nappia näpäytetty, pidetty hetki pohjassa vai oliko nappia pidetty pitkään pohjassa. Kaikille kolmelle eri painotyypille voidaan määritellä eri toiminnot, esimerkiksi kun halutaan suurentaa jotain lukua. Yhdellä näpäytyksellä luku kasvaa yhdellä, jos nappia pidetään pohjassa, se alkaa kasvamaan ensin viiden numeron välein, jonka jälkeen tietyn ajan päästä luku alkaa kasvaa kymmenen numeron välein.

Funktio ”void save_settings()” tallentaa kaikki metronomin asetukset mikrokontrollerin sisäiseen EEPROM-muistiin (*engl. Electronically Erasable Programmable Read-Only Memory, sähköisesti tyhjennettävä ja kirjoitettava haihtumaton lukumuisti*), joka on haihtumatonta muistia, eli muisti säilyy, vaikka laitteesta katkaistaisiin virta. Tallentaminen tapahtuu, kun ”play/pause” -nappia on pidetty pohjassa yli kolme sekuntia. Koska EEPROM-muistin yhden muistilohkon koko on vain 8-bittiä [34], pitää joitain suuria arvoja käsitellä ja jakaa pienempiin korkeintaan 8-bitin lukuihin. Lukujen jakaminen suoritettiin yksinkertaisesti jakamalla luku esimerkiksi neliään pienempään lukuun, jotka luettaessa summataan yhteen. Esimerkiksi metronomissa tempon suurin arvo on 999 BPM, jolloin se pitää tallettaa neljään eri muistipaikkaan, 255+255+255+234=999. Tämä tapa on yksinkertainen, mutta kuluttaa paljon muistia. Parempi tapa olisi esimerkiksi 16 bittisen luvun kohdalla tallentaa 8 ensimmäistä bittiä ja 8 jälkimmäistä bittiä omiin muisteihinsa. Näin saadaan jopa luku 65 536 mahtumaan vain kahteen 8-bittiseen muistipaikkaan, kun muistipaikkoja edellä mainitulla tavalla tarvittaisiin 258 kappaletta. Funktiolla ”void read_settings()” luetaan tallennetut arvot EEPROM-muistista, ja tallennetaan ne

mikrokontrollerin SRAM-muistiin (*engl. Static Random Access Memory, Staattinen RAM-muisti*) ohjelman suorittamista varten. EEPROM luetaan aina laitteen käynnistyessä, jotta käyttäjän tallentamat asetukset tulisivat voimaan.

Näytön kanssa käytetään useita erilaisia funktioita sekä Adafruitin valmiita ohjelmakirjastoja näytöllä näkyville merkeille ja näytön ohjaamiseen. Annettaessa käsky näytölle määritetään ensin väri ja koordinaatti, jonka jälkeen määritetään haluttu muoto tai merkki. Koko näyttöä ei tarvitse päivittää kerralla, sillä ainoastaan päivittyvät pikselit päivitetään. Tämä nopeuttaa näytön kuvan päivittämistä. näyttöä ei myöskään voida päivittää soiton aikana, sillä näytön päivittäminen kulutta liian paljon mikrokontrollerin laskenta-aikaa ja äänen ajoitus voisi näin mennä väärin.

Varsinainen äänen tuottamiseen kuuluva ohjelmakoodin osa on melko lyhyt. Kun laite käynnistetään tai asetuksia muutetaan, lasketaan iskujen väliset ajat uudestaan ja tallennetaan ne. Soitettaessa metronomia tikitysten väliset ajat tarkistetaan muistista, jolloin niitä ei tarvitse laskea uudestaan ja säästetään laskenta-aikaa.

```
// Metronomin toiminnallisuus
if (play == true) {
   if (millis() - last_hit_time >= delay_between_hits_millis[current_playing_tab]) {

      last_hit_time = millis();

      if (hit_num >= tahti_1[current_playing_tab] && repeat_count != 1) {
         first_beat = false;
         repeat_count -= 1;
         hit_num = 0;
      }

      else if (hit_num >= tahti_1[current_playing_tab] && repeat_count == 1) {
         current_playing_tab += 1;
         if (current_playing_tab >= 8) {
            current_playing_tab = 0;
            first_beat = true;
         }
      repeat_count = repeat[current_playing_tab];
      hit_num = 0;
      }

      if (on_off_array[current_playing_tab] == true) {

        if (first_hit_array[current_playing_tab] == true && hit_num == 0 && current_playing_tab
              == 0 && first_hit == true && first_beat == true) {
            tone(tone_pin, very_high_note_frequency, note_duration); // very high note
         }
         else if (first_hit_array[current_playing_tab] == true && hit_num == 0) {
            tone(tone_pin, high_note_frequency, note_duration); // high note
         }
         else {
            tone(tone_pin, low_note_frequency, note_duration); // low note
         }
      }
```

```
        hit_num += 1;
37    }
  }
39 else {
    // reset
41    unsigned long last_hit_time = 0;
    int current_playing_tab = 0;
43    int hit_num = 0;
    repeat_count = repeat[current_playing_tab];
45    }
  }
```

Ohjelma 1: *Metronomin toiminnallisuuden ohjelmakoodi*

Metronomin äänensoittoa pollataan eli tarkistetaan monta tuhatta kertaa sekunnissa. Kun tarkistetaan, onko aika soittaa tikahdus, verrataan mikrokontrollerin sisäisen kellon ja edellisen tikahduksen aikaa. Mikäli tämä aika on suurempi kuin haluttu tauko, soitetaan tikahdus. Tikahduksen taajuuden ja keston määrittää juuri kyseiselle tahdille määritetyt parametrit. Ohjelman 1 rivit 7-21 liittyvät tahdin toistokertoihin (repeat), sekä välilehtien soittamisen siirtymiseen lopusta alkuun. Varsinaisen äänen tuottamiseen käytetään Arduinon omaa funktoita ” tone(pin, frequency, duration)”, jonka kutsuminen saa määrätyn pinnin oskilloimaan määrätyllä taajuudella määrätyn aikaa [35]. Riippuen kohdasta, jota ollaan soittamassa, voidaan soittaa joko erittäin korkea tikahdus, korkea tikahdus tai normaali tikahdus.

7. YHTEENVETO

Työssä suunniteltiin, rakennettiin ja testattiin mikrokontrolleriohjattu metronomi. Metronomista pyrittiin tekemään hyvä kokonaisuus niin ulkonäöllisesti, käytettävyydeltään kuin myös toiminnaltaan. Hyvin yleinen ongelma suunnittelussa on, että työn edetessä laitteeseen haluaisi lisätä lisää ominaisuuksia, joka pitkittävät ja hankaloittavat suunnitteluprosessia. Näitä pyrittiin eliminoimaan ideoimalla alussa kaikki tuotteelle haluttavat ominaisuudet ja pitämällä kiinni suunnitelluista toiminnoista. Kesken projektin tulleita lisäideoita pyrittiin välttämään, joskin ne kirjattiin ylös myöhempää tarvetta ja seuraavaa versiota varten.

Yksittäisenä asiana ohjelmakoodin kirjoittamiseen kului eniten aikaa. Vaikka mikrokontrollerin ohjelmoiminen on melko yksinkertaista, on toimivan ja laajan kokonaisuuden aikaansaaminen työlästä. Itse rakentamiseen kului melko vähän aikaa. Toisaalta suunnitteluun sitäkin enemmän aikaa, sillä haluttiin varmistua, ettei tuotteessa olisi valmistuksen jälkeen suuria virheitä, jotka pitäisi korjata hyppylangoilla tai muilla kyseenalaisilla metodeilla.

Lopputuloksena saatiin toimiva metronomi, joka on testikäyttäjien testikäytön osalta osoittautunut toimivaksi ja tarpeelliseksi. Käyttöliittymä on myös melko helposti opittava ulkopuoleisen testaajan mielestä. Kuva 26 esittää valmiin metronomin ja myös sen koon suhteessa perinteiseen tulitikkuaskiin.

Kuva 26: *Valmis metronomi ja koon määrittämisen avuksi tulitikkuaski*

Laitteen toteutuksen tekeminen suunnittelusta testaukseen antoi hyvän kuvan elektroniikkalaitteen suunnitteluprosessista. Työn aikana esiintyi myös ongelmia, joita ei osattu odottaa ja jotka olisi voitu ottaa huomioon jo suunnitteluvaiheessa. Työn aikana sain paljon kantapään kautta opittua tietoa ja taitoa elektroniikkalaitteen toteutuksesta.

Tulevaisuudessa työn aikana havaittuja ongelmia pyritään vähentämään ja parantelemaan piirilevyä sekä korvaamaan joitain johtoja lattakaapeleilla ja liittimillä. Parantelujen jälkeen, kun ollaan riittävän tyytyväisiä metronomin rautapuoleen, olisi tarkoitus tilata valmiit piirilevyt ilman komponentteja ulkopuoliselta taholta. Metronomeja tullaan valmistamaan näillä näkymin muutamia ja siksi ammattimaisesti valmistettu piirilevy on helpompi ratkaisu.

Myös ohjelmakoodia olisi tarkoitus jatkossa parannella ja lisätä ominaisuuksia. Suurimpana lisäominaisuutena mainittakoon mahdollinen mahdollisuus soittaa polyrytmisiä rytmejä. Tämä kuitenkin vaatii hieman enemmän optimointia koodilta.

Jatkossa saatetaan myös kartoittaa mahdollisuutta lisätä erillinen latauksenseurantapiiri, joka välittäisi akun tilan päämikrokontrollerille, sekä myös erillinen näytön ohjaamiseen erikoistunut piiri, jolla saavutettaisiin reaaliaikaisempaa toimintaa ja näytön päivitystä yhtä aikaa.

LÄHTEET

[1] M. Kennedy, J. B. Kennedy: The Concise Oxford Dictionary of Music (5. ed.), Oxford University Press, 2007. 839 s. Saatavissa: http://www.oxfordreference.com/view/10.1093/acref/9780199203833.001.0001/acref-9780199203833-e-6020

[2] Gear4music: Wittner Taktell Piccolo Metronome, Black, [Verkkosivu]. Saatavissa [Viitattu 31. 5. 2019]: https://www.gear4music.com/Woodwind-Brass-Strings/Wittner-Taktell-Piccolo-Metronome-Black/22WN

[3] RussoMusic: Boss DB-90 Digital Metronome, [Verkkosivu]. Saatavissa [Viitattu 31. 5. 2019]: https://www.russomusic.com/products/boss-db-90-digital-metronome.

[4] Microchip: ATmega328p datasheet, [Datalehti]. Saatavissa [Viitattu 25. 2. 2019]: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[5] Norse-mythology.net: Bragi the God of Poetry, [Verkkosivu]. Saatavissa [Viitattu 31. 5. 2019]: https://norse-mythology.net/bragi-the-god-of-poetry-and-music-in-norse-mythology/

[6] D. Stone, C. Jarrett, M. Woodroffe ja S. Minocha: User Interface Design and Evaluation, Morgan Kaufmann Publishers Inc, 2005, 1314 s. Saatavissa: https://books.google.fi/books?id=VvSoyqPBPbMC&printsec=frontcover&hl=fi#v=onepage&q&f=false

[7] S. Lauesen: User Interface Design: A Software Engineering Perspective, Pearson Education UK, 2004, 623 s. Saatavissa: https://books.google.fi/books?id=8-LhbEfLSGsC&printsec=frontcover&hl=fi#v=onepage&q&f=false

[8] Adafruit: 1,8" Color TFT LCD display, [Verkkosivu]. Saatavissa [Viitattu 25. 2. 2019]: https://www.adafruit.com/product/358.

[9] J. R. Smith: Programming the PIC Microcontroller with MBasic, NEWNES, 2005. 783 s. Saatavissa: http://www.sciencedirect.com/science/article/pii/B9780750679466500018

[10] S. F. Barrett: Embedded System Design with the Atmel AVR Microcontroller, Part 2, Morgan & Claypool Publishers, 2010, 176 s. https://books.google.fi/books?id=3qSSAwAAQBAJ&printsec=frontcover&hl=fi#v=onepage&q&f=false

[11] J. Newman: Physics of the Life Sciences, Springer Science + Business Media LLC, 2008, 718 s. https://books.google.fi/books?id=vY5rXC4xlMgC&printsec=frontcover&hl=fi#v=onepage&q&f=false

[12] Learn About Electronics: Logarithmic and Linear Controls, [Verkkosivu]. Saatavissa [Viitattu 22. 4. 2019]: http://www.learnabout-electronics.org/Resistors/resistors_09a.php.

[13] Soundguys: Was ditching the headphone jack a good idea? [Verkkosivu]. Saatavissa [Viitattu 1. 7. 2019]: https://www.soundguys.com/was-ditching-the-headphone-jack-a-good-idea-13825/.

[14] usb.org: Universal Serial Bus: Power Delivery Specification. [Tekninen määritys] Saatavissa [Viitattu 1. 7. 2019]: https://www.usb.org/document-library/usb-power-delivery

[15] S. K. Venkata Ram: Advanced Microprocessor & Microcontrollers, New Delhi, Firewall Media, 2004. 499 s. Saatavissa: https://books.google.fi/books?id=MUI1ioZrnzcC&printsec=frontcover&hl=fi#v=onepage&q&f=false

[16] H. Choudhary: Engineers Garage, [Verkkosivu]. Saatavissa [Viitattu 18. 5. 2019]: https://www.engineersgarage.com/tutorials/difference-between-microprocessor-and-microcontroller.

[17] E. Ayoub ja N. Karami: Review on the charging techniques of a Li-Ion battery, 2015 Third International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), Beirut, 2015 [Konferenssi] Saatavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7113599

[18] Panasonic: NCR18650GA-battery datasheet, [Datalehti]. Saatavissa [Viitattu 2. 3. 2019]: https://www.orbtronic.com/content/Datasheet-specs-Sanyo-Panasonic-NCR18650GA-3500mah.pdf.

[19] A. Cunningham: A brief history of USB, what it replaced, and what has failed to replace it, [Verkkosivu]. Saatavissa [Viitattu 23. 6. 2019]: https://arstechnica.com/gadgets/2014/08/a-brief-history-of-usb-what-it-replaced-and-what-has-failed-to-replace-it/.

[20] Y. Barsukov ja J. Qian: Battery Power Management for Portable Devices, London: Artech House, 2013. 241 s. Saatavissa: https://app.knovel.com/web/toc.v/cid:kpBPMPD002/viewerType:toc//root_slug:battery-power-management/url_slug:li-ion-li-polymer-charge?=undefined&issue_id=kt011LSS62

[21] Micros: TP4056A datasheet, [Datalehti]. Saatavissa [Viitattu 2. 3. 2019]: http://www.image.micros.com.pl/_dane_techniczne_auto/uitp4056.pdf

[22] Sparkfun: DW01-P datasheet, [Datalehti]. Saatavissa [Viitattu 2. 3. 2019]: https://cdn.sparkfun.com/assets/learn_tutorials/2/5/1/DW01-P_DataSheet_V10.pdf

[23] A. Kwade ja J. Diekmann: Recycling of Lithium-Ion Batteries: The LithoRec Way, Springer, 2018, 312 s. Saatavissa: https://books.google.fi/books?id=7dpCDwAAQBAJ&printsec=frontcover&hl=fi#v=onepage&q&f=false

[24] IEEE: IEEE 1725-2011 Standard for Rechargeable Batteries for Cellular Telephones, 2011. [Standardi].
Saatavissa [Viitattu 31. 3. 2019]:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5930304

[25] C. Mikolajczak, M. Kahn, K. White ja R. T. Long: Lithium-Ion Batteries Hazard and Use Assessment, Springer-Verlag, 2011, 115 s. Saatavissa:
https://books.google.fi/books?id=V4IVCvgv558C&printsec=frontcover&hl=fi#v=onepage&q&f=false

[26] D. Self: Audio Power Amplifier Design Handbook, (5. ed.) Lontoo: Focal Press, 2013. 624 s. Saatavissa:
https://books.google.fi/books?id=TLt0DO6LAokC&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[27] Amazon-verkkokauppa: AIMELIAE OEP30Wx2 Module DC 12V-24V OEP 30W x2 36dB Class D Digital Power Amplifier Board, [Verkkosivu].
Saatavissa [Viitattu 19. 5. 2019]:
https://www.amazon.com/AIMELIAE-OEP30Wx2-12V-24V-Digital-Amplifier/dp/B074TC9N8B

[28] G. M. Ballou: Handbook for sound engineers, (4. ed.) Oxford: Focal Press, 2008. 1778 s. Saatavissa:
https://nikospapachristou.files.wordpress.com/2013/02/handbook-for-sound-engineers.pdf

[29] F. Glast: Mold Construction Guide, [Verkkosivu].
Saatavissa [Viitattu 23. 6. 2019]:
https://www.fibreglast.com/product/mold-construction/Learning_Center.

[30] Hottech Semiconductor: 8205A datasheet, [Datalehti].
Saatavissa [Viitattu 2. 3. 2019]:
https://www.maritex.com.pl/product/attachment/91261/8205A.pdf.

[31] PhoneArena.com: Samsung vs Apple, Huawei and OnePlus: whose phones charge the fastest?, [Verkkosivu].
Saatavissa [Viitattu 30. 5. 2019]:
https://www.phonearena.com/news/Fastest-charging-phones-mAh-battery-minute-Samsung-Apple-LG-Huawei_id107862

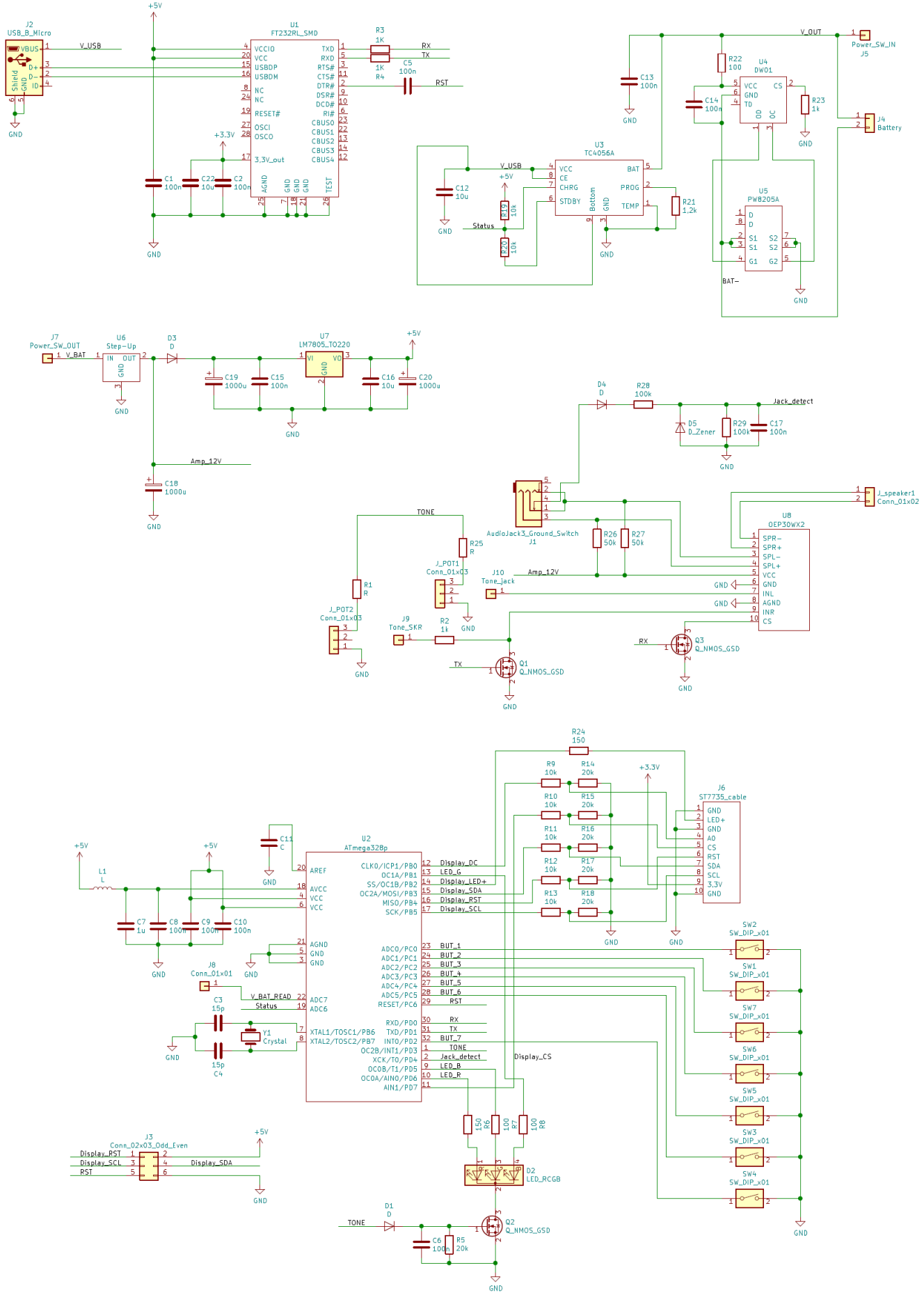
[32] Arduino: Bootloader Development, [Verkkosivu].
Saatavissa [Viitattu 31. 5. 2019]:
https://www.arduino.cc/en/Hacking/Bootloader?from=Tutorial.Bootloader

[33] Arduino: Arduino Nano, [Verkkosivu].
Saatavissa [Viitattu 31. 5. 2019]:
https://store.arduino.cc/arduino-nano

[34] Arduino: EEPROM_Write, [Verkkosivu].
Saatavissa [Viitattu 31. 5. 2019]:
https://www.arduino.cc/en/Reference/EEPROMWrite

[35] Arduino: tone(), [Verkkosivu].
Saatavissa [Viitattu 31. 5. 2019]:
https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

LIITE A: KYTKENTÄKAAVIO



LIITE B: OHJELMAKOODI

Tässä liitteessä on esitetty kaksi ohjelmakooditiedostoa. Ensimmäisessä tiedostossa on varsinainen ohjelmakoodi ja metronomin toiminnallisuus (metronomi_v5.ino). Toisessa tiedostossa on kolme erilaista kuvaa heksaluvuilla piirrettynä (images.h). Kuvat ovat aloitusnäytön BRAGI-metronome-teksti, ensimmäiset iskut ilmoittavan ruudun check-merkki, sekä latauksen tilan ilmoittava salama-kuvake

```
// BRAGI
// Tekijä: Lassi Parkkila
// Kandidaatintyö, Metronomi

#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library
#include <SPI.h>
#include <EEPROM.h>
#include "Images.h"

// Värit
byte wallpaper_color = 8;
byte text_color = 7;
byte tab_on_color = 3;
byte tab_on_number_color = 1;
byte tab_off_color = 1;
byte tab_off_number_color = 3;
byte edit_box_color = 5;
byte LED_color = 1;

int colorpalette[] = { ST7735_RED, ST7735_YELLOW, ST7735_GREEN, ST7735_CYAN, ST7735_BLUE, ST7735_MAGENTA, ST7735_WHITE, ST7735_BLACK };
int led_colorpalette[][3] = {
{ 1,0,0 },  // red
{ 1,1,0 },  // yellow
{ 0,1,0 },  // green
{ 0,1,1 },  // cyan
{ 0,0,1 },  // blue
{ 1,0,1 },  // magenta
{ 1,1,1 },  // white
{ 0,0,0 }   // black
};

// !!!! Valmista koodia n�yt�lle
#define TFT_CS     7
#define TFT_RST    12
#define TFT_DC     8
int width = 160;
int height = 128;

#define TFT_SCLK 13   // set these to be whatever pins you like!
#define TFT_MOSI 11   // set these to be whatever pins you like!
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK, TFT_RST);
// !!!! Valmista koodia n�yt�lle loppuu

int x = 0;
int y = 0;

int tempo[8] = { 240 , 240 , 240 , 300 , 360 , 420 , 480 , 540 };   // Tempo
byte tahti_1[8] = { 3 , 2 , 3 , 3 , 3 , 3 , 3 , 3 };            // Tahtilajin osoittaja
byte tahti_2[8] = { 5 , 4 , 4 , 4 , 4 , 4 , 4 , 4 };            // Tahtilajin nimitt�j�
byte active_tab = 1;                                            // Aktiivinen v�lilehti (n�kyy n�yt�ss�)
byte on_off_array[8] = { true , true , false , false , false , false , false , false }; // Soitettavat v�lilehdet
byte first_hit_array[8] = { true , true , true , true , true , true , true , true };    // Painotus ensimmäiselle iskulle
bool first_hit = true;
byte edit_box = 8;      // Editoitava kohde
int repeat[8] = { 3 , 2 , 5 , 6 , 4 , 7 , 2 , 1 };  // repeat

int repeat_count = 1;
bool first_beat = true;

byte buttonpins[7] = { 14 , 15 , 18 , 2 , 19 , 17 , 16 };                               // Painonappien pinnit
boolean buttonState[7] = { HIGH , HIGH , HIGH , HIGH , HIGH , HIGH , HIGH };
unsigned long lastDebounceTime[7] = { 0 , 0 , 0 , 0 , 0 , 0 , 0 };
unsigned long debounceDelay = 50;
boolean lastState[7] = { HIGH , HIGH , HIGH , HIGH , HIGH , HIGH , HIGH };
int button_timer_counter[7] = { 0 , 0 , 0 , 0 , 0 , 0 , 0 };
bool but_enable[7] = {true, true, true, true, true, true, true};
byte interrupt_button = 0;

unsigned long delay_between_hits_millis[8] = { 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 };     // Iskujen v�lit

unsigned long last_hit_time = 0;
int current_playing_tab = 0;
int hit_num = 0;

#define backlight_pin   9
int  backlight_on_time = 30;
#define backlight_fade_time_1   10
#define backlight_dim_time      5000
#define backlight_fade_time_2   100
byte backlight_on_brightness = 255;
#define backlight_dim_brightness    10
byte current_brightness = 255;
unsigned long last_brightness_update = 0;
unsigned long last_press = 0;

#define tone_pin        3
int high_note_frequency = 1500;
int low_note_frequency = 1000;
int very_high_note_frequency = 2000;
int note_duration = 10;
```

```
92
93  #define led_pin_red     6
94  #define led_pin_green   5
95  #define led_pin_blue    10
96  byte led_brightness = 10;
97
98  int settings_edit_box = 0;
99
100 int settings_counter = 0;
101 int settings_counter_limit = 150;
102 int reset_counter = 0;
103 int reset_counter_limit = 150;
104
105 #define battery_voltage_pin A7
106 #define charge_status_pin A6
107
108 int battery_voltage_reference[11] = { 512 , 651 , 686 , 704 , 722 , 740 , 758 , 776 , 794 , 809 , 860 }; // 0%, 10% ... 90%, 1000%
109 int battery_percent = 0;
110 int last_battery_percent = 255;
111 int raw_battery_voltage = 0;
112 int charge_status = 0;
113 int last_charge_status = 3;
114 unsigned long last_battery_update = 0;
115
116 const int voltage_readings_count = 10;
117 int voltage_readings[voltage_readings_count];
118 int voltage_read_index = 0;
119 int total_sum_voltage = 0;
120
121 bool play = false;
122 int amp_CS_pin = 0;
123 int speaker_mute_pin = 1;
124 int headphone_detect_pin = 4;
125
126 void setup(void) {
127     // N♦pp♦inten pinMode:t
128     for (int i = 0; i < sizeof buttonpins; i++) {
129         pinMode(buttonpins[i], INPUT_PULLUP);
130     }
131
132     pinMode(amp_CS_pin, OUTPUT);
133     pinMode(speaker_mute_pin, OUTPUT);
134     pinMode(headphone_detect_pin, INPUT);
135     pinMode(tone_pin, OUTPUT);
136     pinMode(led_pin_red, OUTPUT);
137     pinMode(led_pin_green, OUTPUT);
138     pinMode(led_pin_blue, OUTPUT);
139
140     digitalWrite(amp_CS_pin, HIGH);     // amp off
141
142     tft.initR(0x3);   // initialize a ST7735S chip, black tab
143     tft.invertDisplay(true);
144     tft.setRotation(1);
145
146     tft.fillScreen(colorpalette[wallpaper_color - 1]); // Tyhjennä näyttö
147
148     tft.drawBitmap(0, 0, bragi_160_128, 160, 128, ST7735_WHITE); // Piirtää Logon
149     update_brightness(); // Taustavalo päälle
150
151     // Jos nappeja 1 ja 2 pidetty pohjassa --> settings, jos 3,4,5,6 --> reset
152     for (int i = 0; i < 300; i++) {
153         if (digitalRead(buttonpins[0]) == LOW && digitalRead(buttonpins[1]) == LOW) {
154             settings_counter += 1;
155         }
156
157         if (digitalRead(buttonpins[2]) == LOW && digitalRead(buttonpins[3]) == LOW && digitalRead(buttonpins[4]) == LOW && digitalRead
              (buttonpins[5]) == LOW) {
158             reset_counter += 1;
159         }
160
161         delay(10);
162     }
163
164     // Jos painettu riittävän kauan reset-nappeja, ei lueta EEPROMista asetuksia vaan oletusasetukset, ja tallennetaan ne
165     if (reset_counter < reset_counter_limit) {
166         read_settings();
167     }
168     else {
169         save_settings();
170     }
171
172     tft.fillScreen(colorpalette[wallpaper_color - 1]); // Tyhjennä näyttö
173
174     // Jos asetusnappeja painettu riittävän kauan, mennään asetuksiin, muuten "normaali"
175     if (settings_counter >= settings_counter_limit) {
176         tft.setRotation(0);
177
178         print_settings_text();
179
180         height = 160;
181         width = 128;
```

```
182
183         for (int a = 1; a < 16; a++) {
184             settings_edit_box = a;
185             print_settings();
186         }
187     }
188
189     else {
190         // Pirt�� v�lilehdet
191         for (int i = 0; i < 8; i++) {
192             print_tab(i + 1, on_off_array[i], active_tab);
193         }
194
195         print_screen();         // Piirt�� n�ytt��n numerot yms.
196         calculate_delays();     // Laskee iskujen v�lit
197         repeat_count = repeat[0];
198     }
199
200     print_LED(LED_color);
201     print_battery();
202
203     // asettaa akun lähtöjännitekeskiarvovektorin
204     int battery_voltage = analogRead(battery_voltage_pin);
205     for (int thisReading = 0; thisReading < voltage_readings_count; thisReading++) {
206         voltage_readings[thisReading] = battery_voltage;
207     }
208     total_sum_voltage = voltage_readings_count * battery_voltage;
209 }
210
211 void loop() {
212
213     detect_headphone_jack();
214
215     update_battery();
216
217     update_brightness(); // päivittää kirkkautta
218
219     if (settings_counter >= settings_counter_limit) {
220
221         for (int i = 0; i < sizeof buttonpins; i++) {
222             button_settings(i);
223         }
224
225     }
226
227     else {
228
229         // Tarkistaa kaikkien nappien tilan
230         for (int i = 0; i < sizeof buttonpins; i++) {
231             button(i);
232         }
233
234         // Metronomin toiminnallisuus
235         if (play == true) {
236             if (millis() - last_hit_time >= delay_between_hits_millis[current_playing_tab]) {
237
238                 last_hit_time = millis();
239
240                 if (hit_num >= tahti_1[current_playing_tab] && repeat_count != 1) {
241                     first_beat = false;
242                     repeat_count -= 1;
243                     hit_num = 0;
244                 }
245
246                 else if (hit_num >= tahti_1[current_playing_tab] && repeat_count == 1) {
247                     current_playing_tab += 1;
248                     if (current_playing_tab >= 8) {
249                         current_playing_tab = 0;
250                         first_beat = true;
251
252                     }
253                     repeat_count = repeat[current_playing_tab];
254                     hit_num = 0;
255                 }
256
257                 if (on_off_array[current_playing_tab] == true) { // aktiivinen v�lilehti
258
259                     if (first_hit_array[current_playing_tab] == true && hit_num == 0 && current_playing_tab == 0 && first_hit == true &&
                          first_beat == true) {
260                         tone(tone_pin, very_high_note_frequency, note_duration); // very high note
261                     }
262                     else if (first_hit_array[current_playing_tab] == true && hit_num == 0) {
263                         tone(tone_pin, high_note_frequency, note_duration); // high note
264                     }
265                     else {
266                         tone(tone_pin, low_note_frequency, note_duration); // low note
267                     }
268                 }
269
270                 hit_num += 1;
271             }
```

```
272           }
273           else {
274               // reset
275               unsigned long last_hit_time = 0;
276               int current_playing_tab = 0;
277               int hit_num = 0;
278               repeat_count = repeat[current_playing_tab];
279           }
280       }
281   }
282
283   // Laskee iskujen v♠lit
284   void calculate_delays() {
285       for (int i = 0; i < 8; i++) {
286           if (on_off_array[i] == true) {
287               delay_between_hits_millis[i] = (unsigned long)((1.0 / (((float)tempo[i] / (4.0 / (float)tahti_2[i])) / 60.0)) * 1000.0);
288           }
289           else {
290               delay_between_hits_millis[i] = 0;
291
292           }
293       }
294   }
295
296   // Tarkastaa n♠pp♠imen ja suorittaa toiminnon
297   void button(int button_number) { //boolean buttonState, int pin, unsigned long lastDebounceTime, boolean lastState, int
        button_timer_counter) {
298       buttonState[button_number] = digitalRead(buttonpins[button_number]);
299       if (millis() - lastDebounceTime[button_number] > debounceDelay) {
300           lastDebounceTime[button_number] = millis();
301           if (buttonState[button_number] == LOW && lastState[button_number] == HIGH) { // 1. time after press
302               button_timer_counter[button_number] = 0;
303               lastState[button_number] = LOW;
304               lastDebounceTime[button_number] = millis();
305               last_press = millis();
306               if (current_brightness < backlight_on_brightness) {
307                   but_enable[button_number] = false;
308               }
309           }
310           else if (buttonState[button_number] == LOW && lastState[button_number] == LOW) { // during press
311               button_timer_counter[button_number] += 1;
312               lastState[button_number] = LOW;
313               last_press = millis();
314           }
315
316           else if (buttonState[button_number] == HIGH && lastState[button_number] == LOW && button_timer_counter[button_number] < 20) { //
              normal press
317               lastState[button_number] = HIGH;
318
319               if (but_enable[button_number] == true) {
320
321                   update_editbox_delete();
322                   switch (button_number)
323                   {
324                   case 0: // tab +
325                       active_tab += 1;
326                       if (active_tab > 8) {
327                           active_tab = 1;
328                       }
329                       update_tabs();
330                       update_tahti();
331                       update_BPM();
332                       update_hit();
333                       update_repeat();
334                       break;
335
336                   case 1: // tab -
337                       active_tab -= 1;
338                       if (active_tab < 1) {
339                           active_tab = 8;
340                       }
341                       update_tabs();
342                       update_tahti();
343                       update_BPM();
344                       update_hit();
345                       update_repeat();
346                       break;
347
348                   case 2: // editbox +
349                       edit_box += 1;
350                       if (edit_box > 8) {
351                           edit_box = 1;
352                       }
353                       break;
354
355                   case 3: // editbox -
356                       edit_box -= 1;
357                       if (edit_box < 1) {
358                           edit_box = 8;
359                       }
360                       break;
```

```
361
362                 case 4: // button up
363                     switch (edit_box)
364                     {
365                     case 8:
366                         // no box
367                         break;
368                     case 1: // Tab
369                         on_off_array[active_tab - 1] = !on_off_array[active_tab - 1];
370                         update_tabs();
371                         break;
372                     case 2: // tahti 1
373                         tahti_1[active_tab - 1] += 1;
374                         if (tahti_1[active_tab - 1] > 99) {
375                             tahti_1[active_tab - 1] = 1;
376                         }
377                         update_tahti();
378                         break;
379                     case 3: // tahti 2
380                         tahti_2[active_tab - 1] += 1;
381                         if (tahti_2[active_tab - 1] > 99) {
382                             tahti_2[active_tab - 1] = 1;
383                         }
384                         update_tahti();
385                         break;
386                     case 4: // repeat
387                         repeat[active_tab - 1] += 1;
388                         if (repeat[active_tab - 1] > 999) {
389                             repeat[active_tab - 1] = 1;
390                         }
391                         update_repeat();
392                         break;
393                     case 5: // tempo
394                         tempo[active_tab - 1] += 1;
395                         if (tempo[active_tab - 1] > 999) {
396                             tempo[active_tab - 1] = 1;
397                         }
398                         update_BPM();
399                         break;
400                     case 6: // 1. all
401                         first_hit = !first_hit;
402                         update_hit();
403                         break;
404                     case 7: // 1. hit
405                         first_hit_array[active_tab - 1] = !first_hit_array[active_tab - 1];
406                         update_hit();
407                         break;
408                     default:
409                         break;
410                     }
411                     break;
412
413                 case 5: // button down
414                     switch (edit_box)
415                     {
416                     case 8:
417                         // no box
418                         break;
419                     case 1: // tab
420                         on_off_array[active_tab - 1] = !on_off_array[active_tab - 1];
421                         update_tabs();
422
423                         break;
424                     case 2: // tahti 1
425                         tahti_1[active_tab - 1] -= 1;
426                         if (tahti_1[active_tab - 1] < 1) {
427                             tahti_1[active_tab - 1] = 99;
428                         }
429                         update_tahti();
430                         break;
431                     case 3: // tahti 2
432                         tahti_2[active_tab - 1] -= 1;
433                         if (tahti_2[active_tab - 1] < 1) {
434                             tahti_2[active_tab - 1] = 99;
435                         }
436                         update_tahti();
437                         break;
438                     case 4: // repeat
439                         repeat[active_tab - 1] -= 1;
440                         if (repeat[active_tab - 1] < 1) {
441                             repeat[active_tab - 1] = 999;
442                         }
443                         update_repeat();
444                         break;
445                     case 5: // tempo
446                         tempo[active_tab - 1] -= 1;
447                         if (tempo[active_tab - 1] < 1) {
448                             tempo[active_tab - 1] = 999;
449                         }
450                         update_BPM();
451                         break;
```

```
452                     case 6: // 1. all
453                         first_hit = !first_hit;
454                         update_hit();
455                         break;
456                     case 7: // 1. hit
457                         first_hit_array[active_tab - 1] = !first_hit_array[active_tab - 1];
458                         update_hit();
459                         break;
460                     default:
461                         break;
462                     }
463                     break;
464                 case 6:
465                     play = !play;
466                     if (play == true) {
467                         reset_play();
468                     }
469                     soundcard();
470                     break;
471                 default:
472                     break;
473                 }
474                 update_editbox();
475                 calculate_delays();     // Laskee iskujen v♦lit
476             }
477         }
478         if (button_timer_counter[button_number] > 40 && buttonState[button_number] == LOW) { // very long press
479             switch (button_number)
480             {
481             case 4: // up
482                 switch (edit_box)
483                 {
484                 case 4: // repeat
485                     repeat[active_tab - 1] += 5;
486                     if (repeat[active_tab - 1] > 999) {
487                         repeat[active_tab - 1] = 1;
488                     }
489                     update_repeat();
490                     break;
491                 case 5: // tempo
492                     tempo[active_tab - 1] += 5;
493                     if (tempo[active_tab - 1] > 999) {
494                         tempo[active_tab - 1] = 1;
495                     }
496                     update_BPM();
497                     break;
498                 default:
499                     break;
500                 }
501                 break;
502             case 5: // down
503                 switch (edit_box)
504                 {
505                 case 4: // repeat
506                     repeat[active_tab - 1] -= 5;
507                     if (repeat[active_tab - 1] < 1) {
508                         repeat[active_tab - 1] = 999;
509                     }
510                     update_repeat();
511                     break;
512                 case 5: // tempo
513                     tempo[active_tab - 1] -= 5;
514                     if (tempo[active_tab - 1] < 1) {
515                         tempo[active_tab - 1] = 999;
516                     }
517                     update_BPM();
518                     break;
519                 default:
520                     break;
521                 }
522                 break;
523             case 6: //save settings
524                 save_settings();
525                 noTone(tone_pin);
526                 digitalWrite(amp_CS_pin, LOW);      // vahvistin päälle
527                 print_LED(1);
528                 tone(tone_pin, 2000, 200);
529                 delay(200);
530                 print_LED(3);
531                 tone(tone_pin, 1000, 200);
532                 while (digitalRead(buttonpins[6]) == LOW) {}
533                 delay(200);
534                 print_LED(LED_color);
535                 break;
536             default:
537                 break;
538             }
539             update_editbox();
540             calculate_delays();     // Laskee iskujen v♦lit
541         }
542         else if (button_timer_counter[button_number] > 20 && buttonState[button_number] == LOW) { // long press
```

```
543             switch (button_number)
544             {
545             case 4: // up
546                 switch (edit_box)
547                 {
548                 case 4: // repeat
549                     repeat[active_tab - 1] += 2;
550                     if (repeat[active_tab - 1] > 999) {
551                         repeat[active_tab - 1] = 1;
552                     }
553                     update_repeat();
554                     break;
555                 case 5: // tempo
556                     tempo[active_tab - 1] += 2;
557                     if (tempo[active_tab - 1] > 999) {
558                         tempo[active_tab - 1] = 1;
559                     }
560                     update_BPM();
561                     break;
562                 default:
563                     break;
564                 }
565                 break;
566             case 5: // down
567                 switch (edit_box)
568                 {
569                 case 4: // repeat
570                     repeat[active_tab - 1] -= 2;
571                     if (repeat[active_tab - 1] < 1) {
572                         repeat[active_tab - 1] = 999;
573                     }
574                     update_repeat();
575                     break;
576                 case 5: // tempo
577                     tempo[active_tab - 1] -= 2;
578                     if (tempo[active_tab - 1] < 1) {
579                         tempo[active_tab - 1] = 999;
580                     }
581                     update_BPM();
582                     break;
583                 default:
584                     break;
585                 }
586                 break;
587             default:
588                 break;
589             }
590             update_editbox();
591             calculate_delays();     // Laskee iskujen v♦lit
592         }
593         if (buttonState[button_number] == HIGH) { // no press
594             button_timer_counter[button_number] = 0;
595             lastState[button_number] = HIGH;
596             but_enable[button_number] = true;
597         }
598     }
599 }
600
601 void save_settings() {
602     int address = 0;
603
604     // tempo
605     for (int i = 0; i < 8; i++) {
606
607         byte bpm_1 = 0;
608         byte bpm_2 = 0;
609         byte bpm_3 = 0;
610         byte bpm_4 = 0;
611
612         if (tempo[i] <= 255) {
613             bpm_1 = tempo[i];
614             bpm_2 = 0;
615             bpm_3 = 0;
616             bpm_4 = 0;
617         }
618         else if (tempo[i] > 255 && tempo[i] <= 510) {
619             bpm_1 = 255;
620             bpm_2 = tempo[i] - 255;
621             bpm_3 = 0;
622             bpm_4 = 0;
623         }
624         else if (tempo[i] > 510 && tempo[i] <= 765) {
625             bpm_1 = 255;
626             bpm_2 = 255;
627             bpm_3 = tempo[i] - 255;
628             bpm_4 = 0;
629         }
630         else if (tempo[i] > 765 && tempo[i] <= 1020) {
631             bpm_1 = 255;
632             bpm_2 = 255;
633             bpm_3 = 255;
```

```
634             bpm_4 = tempo[i] - 255;
635         }
636
637         EEPROM.update(address, bpm_1);
638         address += 1;
639         EEPROM.update(address, bpm_2);
640         address += 1;
641         EEPROM.update(address, bpm_3);
642         address += 1;
643         EEPROM.update(address, bpm_4);
644         address += 1;
645     }
646
647     // tahti_1
648     for (int i = 0; i < 8; i++) {
649         EEPROM.update(address, tahti_1[i]);
650         address += 1;
651     }
652
653     // tahti_2
654     for (int i = 0; i < 8; i++) {
655         EEPROM.update(address, tahti_2[i]);
656         address += 1;
657     }
658
659     // active
660     EEPROM.update(address, active_tab);
661     address += 1;
662
663     // on_off
664     for (int i = 0; i < 8; i++) {
665         EEPROM.update(address, on_off_array[i]);
666         address += 1;
667     }
668
669     // first hit
670     for (int i = 0; i < 8; i++) {
671         EEPROM.update(address, first_hit_array[i]);
672         address += 1;
673     }
674
675     // 1. All
676     EEPROM.update(address, first_hit);
677     address += 1;
678
679     // repeat
680     for (int i = 0; i < 8; i++) {
681
682         byte rep_1 = 0;
683         byte rep_2 = 0;
684         byte rep_3 = 0;
685         byte rep_4 = 0;
686
687         if (repeat[i] <= 255) {
688             rep_1 = repeat[i];
689             rep_2 = 0;
690             rep_3 = 0;
691             rep_4 = 0;
692         }
693         else if (repeat[i] > 255 && repeat[i] <= 510) {
694             rep_1 = 255;
695             rep_2 = repeat[i] - 255;
696             rep_3 = 0;
697             rep_4 = 0;
698         }
699         else if (repeat[i] > 510 && repeat[i] <= 765) {
700             rep_1 = 255;
701             rep_2 = 255;
702             rep_3 = repeat[i] - 255;
703             rep_4 = 0;
704         }
705         else if (repeat[i] > 765 && repeat[i] <= 1020) {
706             rep_1 = 255;
707             rep_2 = 255;
708             rep_3 = 255;
709             rep_4 = repeat[i] - 255;
710         }
711
712         EEPROM.update(address, rep_1);
713         address += 1;
714         EEPROM.update(address, rep_2);
715         address += 1;
716         EEPROM.update(address, rep_3);
717         address += 1;
718         EEPROM.update(address, rep_4);
719         address += 1;
720     }
721
722     //***************** SETTINGS ****************
723
724     byte high_bit = 0;
```

```
725       byte low_bit = 0;
726
727       // Very high note freg
728       high_bit = very_high_note_frequency / 100;
729       low_bit = very_high_note_frequency - (high_bit * 100);
730       EEPROM.update(address, high_bit);
731       address += 1;
732       EEPROM.update(address, low_bit);
733       address += 1;
734
735       // High note freg
736       high_bit = high_note_frequency / 100;
737       low_bit = high_note_frequency - (high_bit * 100);
738       EEPROM.update(address, high_bit);
739       address += 1;
740       EEPROM.update(address, low_bit);
741       address += 1;
742
743       // Low note freg
744       high_bit = low_note_frequency / 100;
745       low_bit = low_note_frequency - (high_bit * 100);
746       EEPROM.update(address, high_bit);
747       address += 1;
748       EEPROM.update(address, low_bit);
749       address += 1;
750
751       // Hit duration
752       high_bit = note_duration / 100;
753       low_bit = note_duration - (high_bit * 100);
754       EEPROM.update(address, high_bit);
755       address += 1;
756       EEPROM.update(address, low_bit);
757       address += 1;
758
759       // backlight on time
760       high_bit = backlight_on_time / 100;
761       low_bit = backlight_on_time - (high_bit * 100);
762       EEPROM.update(address, high_bit);
763       address += 1;
764       EEPROM.update(address, low_bit);
765       address += 1;
766
767       // On brightness
768       EEPROM.update(address, backlight_on_brightness);
769       address += 1;
770
771       // LED brightness
772       EEPROM.update(address, led_brightness);
773       address += 1;
774
775       // Wallpaper
776       EEPROM.update(address, wallpaper_color);
777       address += 1;
778
779       // Text
780       EEPROM.update(address, text_color);
781       address += 1;
782
783       // Tab on
784       EEPROM.update(address, tab_on_color);
785       address += 1;
786
787       // Tab on number
788       EEPROM.update(address, tab_on_number_color);
789       address += 1;
790
791       // Tab off
792       EEPROM.update(address, tab_off_color);
793       address += 1;
794
795       // Tab off number
796       EEPROM.update(address, tab_off_number_color);
797       address += 1;
798
799       // Editbox
800       EEPROM.update(address, edit_box_color);
801       address += 1;
802
803       // Hit LED
804       EEPROM.update(address, LED_color);
805       address += 1;
806   }
807
808   void read_settings() {
809       int address = 0;
810
811       // tempo
812       for (int i = 0; i < 8; i++) {
813
814           byte bpm_1 = EEPROM.read(address);
815           address += 1;
```

```
816           byte bpm_2 = EEPROM.read(address);
817           address += 1;
818           byte bpm_3 = EEPROM.read(address);
819           address += 1;
820           byte bpm_4 = EEPROM.read(address);
821           address += 1;
822
823           tempo[i] = bpm_1 + bpm_2 + bpm_3 + bpm_4;
824       }
825
826       // tahti_1
827       for (int i = 0; i < 8; i++) {
828           tahti_1[i] = EEPROM.read(address);
829           address += 1;
830       }
831
832       // tahti_2
833       for (int i = 0; i < 8; i++) {
834           tahti_2[i] = EEPROM.read(address);
835           address += 1;
836       }
837
838       // active
839       active_tab = EEPROM.read(address);
840       address += 1;
841
842       // on_off
843       for (int i = 0; i < 8; i++) {
844           on_off_array[i] = EEPROM.read(address);
845           address += 1;
846       }
847
848       // first_hit
849       for (int i = 0; i < 8; i++) {
850           first_hit_array[i] = EEPROM.read(address);
851           address += 1;
852       }
853
854       // 1.All
855       first_hit = EEPROM.read(address);
856       address += 1;
857
858       // repeat
859       for (int i = 0; i < 8; i++) {
860
861           byte rep_1 = EEPROM.read(address);
862           address += 1;
863           byte rep_2 = EEPROM.read(address);
864           address += 1;
865           byte rep_3 = EEPROM.read(address);
866           address += 1;
867           byte rep_4 = EEPROM.read(address);
868           address += 1;
869
870           repeat[i] = rep_1 + rep_2 + rep_3 + rep_4;
871       }
872
873       //********************** SETTINGS ******************
874
875       byte high_bit = 0;
876       byte low_bit = 0;
877
878       // Very high note freg
879       high_bit = EEPROM.read(address);
880       address += 1;
881       low_bit = EEPROM.read(address);
882       address += 1;
883       very_high_note_frequency = (high_bit * 100) + low_bit;
884
885       // High note freg
886       high_bit = EEPROM.read(address);
887       address += 1;
888       low_bit = EEPROM.read(address);
889       address += 1;
890       high_note_frequency = (high_bit * 100) + low_bit;
891
892       // Low note freg
893       high_bit = EEPROM.read(address);
894       address += 1;
895       low_bit = EEPROM.read(address);
896       address += 1;
897       low_note_frequency = (high_bit * 100) + low_bit;
898
899       // Hit duration
900       high_bit = EEPROM.read(address);
901       address += 1;
902       low_bit = EEPROM.read(address);
903       address += 1;
904       note_duration = (high_bit * 100) + low_bit;
905
906       // Backlight on time
```

```
907       high_bit = EEPROM.read(address);
908       address += 1;
909       low_bit = EEPROM.read(address);
910       address += 1;
911       backlight_on_time = (high_bit * 100) + low_bit;
912
913       // On brightness
914       backlight_on_brightness = EEPROM.read(address);
915       address += 1;
916
917       // LED brightness
918       led_brightness = EEPROM.read(address);
919       address += 1;
920
921       // Wallpaper
922       wallpaper_color = EEPROM.read(address);
923       address += 1;
924
925       // Text
926       text_color = EEPROM.read(address);
927       address += 1;
928
929       // Tab on
930       tab_on_color = EEPROM.read(address);
931       address += 1;
932
933       // Tab on number
934       tab_on_number_color = EEPROM.read(address);
935       address += 1;
936
937       // Tab off
938       tab_off_color = EEPROM.read(address);
939       address += 1;
940
941       // Tab off number
942       tab_off_number_color = EEPROM.read(address);
943       address += 1;
944
945       // Editbox
946       edit_box_color = EEPROM.read(address);
947       address += 1;
948
949       // Hit LED
950       LED_color = EEPROM.read(address);
951       address += 1;
952   }
953
954   void print_screen() {
955       tft.fillRect(0, 35, 160, 95, colorpalette[wallpaper_color - 1]);
956       tft.setTextColor(colorpalette[text_color - 1]);
957
958       // Tempo
959       tft.setCursor(120, 101);
960       tft.setTextSize(2);
961       tft.print("BPM");
962       // 1. hit
963       tft.setCursor(70, 45);
964       tft.setTextSize(2);
965       tft.print("1.Hit");
966
967       tft.setCursor(70, 70);
968       tft.setTextSize(2);
969       tft.print("1.All");
970
971       // Repeat
972       tft.setCursor(9, 102);
973       tft.setTextSize(1);
974       tft.print("Repeat");
975
976       update_BPM();
977       update_hit();
978       update_tahti();
979       update_repeat();
980       update_repeat();
981   }
982
983   void update_tahti() {
984       tft.fillRect(5, 40, 42, 55, colorpalette[wallpaper_color - 1]);
985
986       tft.setTextSize(3);
987       tft.setTextColor(colorpalette[text_color - 1]);
988
989       // V◆liviiva
990       if (tahti_1[active_tab - 1] >= 10 || tahti_2[active_tab - 1] >= 10) {
991           tft.fillRect(6, 66, 40, 3, colorpalette[text_color - 1]);
992       }
993       else {
994           tft.fillRect(13, 66, 25, 3, colorpalette[text_color - 1]);
995       }
996
997       // Osottaja
```

```
998       if (tahti_1[active_tab - 1] >= 10) {
999           tft.setCursor(10, 41);
1000          tft.print(tahti_1[active_tab - 1]);
1001      }
1002      else {
1003          tft.setCursor(18, 41);
1004          tft.print(tahti_1[active_tab - 1]);
1005      }
1006
1007      // Nimitt♦j♦
1008      if (tahti_2[active_tab - 1] >= 10) {
1009          tft.setCursor(10, 73);
1010          tft.print(tahti_2[active_tab - 1]);
1011      }
1012      else {
1013          tft.setCursor(18, 73);
1014          tft.print(tahti_2[active_tab - 1]);
1015      }
1016  }
1017
1018  void update_BPM() {
1019      tft.fillRect(59, 91, 53, 23, colorpalette[wallpaper_color - 1]);
1020
1021      tft.setTextSize(3);
1022      tft.setTextColor(colorpalette[text_color - 1]);
1023
1024      tft.setCursor(60, 92);
1025
1026      if (tempo[active_tab - 1] < 10) {
1027          tft.print("  ");
1028          tft.print(tempo[active_tab - 1]);
1029      }
1030
1031      if (tempo[active_tab - 1] >= 10 && tempo[active_tab - 1] < 100) {
1032          tft.print(" ");
1033          tft.print(tempo[active_tab - 1]);
1034      }
1035
1036      if (tempo[active_tab - 1] >= 100) {
1037          tft.print(tempo[active_tab - 1]);
1038      }
1039  }
1040
1041  void update_hit() {
1042      tft.fillRect(134, 35, 26, 52, colorpalette[wallpaper_color - 1]);
1043
1044      tft.setTextSize(3);
1045      tft.setTextColor(colorpalette[text_color - 1]);
1046
1047      tft.drawRect(135, 40, 20, 20, colorpalette[text_color - 1]);
1048      tft.drawRect(135, 65, 20, 20, colorpalette[text_color - 1]);
1049
1050      if (first_hit_array[active_tab - 1] == true)
1051          tft.drawBitmap(135, 35, logo16_glcd_bmp, 24, 24, ST7735_GREEN);
1052      if (first_hit == true) {
1053          tft.drawBitmap(135, 60, logo16_glcd_bmp, 24, 24, ST7735_GREEN);
1054      }
1055  }
1056
1057  void print_tab(byte tab, bool on_off, byte active) {
1058      int tab_height = 25;
1059      int tab_width = 20;
1060      int tab_back_color = colorpalette[tab_on_color - 1];
1061      int tab_number_color = colorpalette[tab_on_number_color - 1];
1062      int cursor = 2;
1063
1064      if (active == tab) {
1065          tab_height = 35;
1066          cursor = 12;
1067      }
1068
1069      else { // back color Rect
1070          tft.fillRect(tab_width * (tab - 1), tab_height, tab_width, 10, colorpalette[wallpaper_color - 1]);
1071      }
1072
1073      if (on_off == false) {
1074          tab_back_color = colorpalette[tab_off_color - 1];
1075          tab_number_color = colorpalette[tab_off_number_color - 1];
1076      }
1077
1078      tft.fillRect(tab_width * (tab - 1) + 1, 0, tab_width - 2, tab_height, tab_back_color);
1079
1080      tft.setCursor((tab_width * (tab - 1)) + 2, cursor);
1081      tft.setTextSize(3);
1082      tft.setTextColor(tab_number_color);
1083      tft.print(tab);
1084  }
1085
1086  void update_tabs() {
1087      if (active_tab == 1) {
1088          print_tab(8, on_off_array[7], active_tab);
```

```
1089            print_tab(1, on_off_array[0], active_tab);
1090            print_tab(2, on_off_array[1], active_tab);
1091        }
1092        else if (active_tab == 8) {
1093            print_tab(7, on_off_array[6], active_tab);
1094            print_tab(8, on_off_array[7], active_tab);
1095            print_tab(1, on_off_array[0], active_tab);
1096        }
1097        else {
1098            print_tab(active_tab - 1, on_off_array[active_tab - 2], active_tab);
1099            print_tab(active_tab, on_off_array[active_tab - 1], active_tab);
1100            print_tab(active_tab + 1, on_off_array[active_tab], active_tab);
1101        }
1102    }
1103
1104    void update_editbox() {
1105        switch (edit_box)
1106        {
1107        case 0:
1108            break;
1109        case 1: // tab
1110            tft.drawLine((20 * (active_tab - 1)), 0, (20 * (active_tab - 1)), 34, colorpalette[edit_box_color - 1]);
1111            tft.drawLine((20 * (active_tab - 1)) + 19, 0, (20 * (active_tab - 1)) + 19, 34, colorpalette[edit_box_color - 1]);
1112            tft.fillRect((20 * (active_tab - 1)), 0, 20, 5, colorpalette[edit_box_color - 1]);
1113            break;
1114        case 2: // tahti_1
1115            tft.drawRect(5, 38, 42, 26, colorpalette[edit_box_color - 1]);
1116            break;
1117        case 3: // tahti_2
1118            tft.drawRect(5, 70, 42, 26, colorpalette[edit_box_color - 1]);
1119            break;
1120        case 4: // repeat
1121            tft.drawRect(7, 110, 38, 18, colorpalette[edit_box_color - 1]);
1122            break;
1123        case 5: // bpm
1124            tft.drawRect(57, 89, 57, 26, colorpalette[edit_box_color - 1]);
1125            break;
1126        case 6: // all
1127            tft.drawRect(133, 63, 24, 24, colorpalette[edit_box_color - 1]);
1128            break;
1129        case 7: // 1.hit
1130            tft.drawRect(133, 38, 24, 24, colorpalette[edit_box_color - 1]);
1131            break;
1132
1133        default:
1134            break;
1135        }
1136    }
1137
1138    void update_editbox_delete() {
1139        switch (edit_box)
1140        {
1141        case 0:
1142            break;
1143        case 1: // tabs
1144            tft.drawLine((20 * (active_tab - 1)), 0, (20 * (active_tab - 1)), 34, colorpalette[wallpaper_color - 1]);
1145            tft.drawLine((20 * (active_tab - 1)) + 19, 0, (20 * (active_tab - 1)) + 19, 34, colorpalette[wallpaper_color - 1]);
1146            tft.fillRect((20 * (active_tab - 1)), 0, 20, 5, colorpalette[wallpaper_color - 1]);
1147            update_tabs();
1148            break;
1149        case 2: // tahti_1
1150            tft.drawRect(5, 38, 42, 26, colorpalette[wallpaper_color - 1]);
1151            break;
1152        case 3: // tahti_2
1153            tft.drawRect(5, 70, 42, 26, colorpalette[wallpaper_color - 1]);
1154            break;
1155        case 4: // repeat
1156            tft.drawRect(7, 110, 38, 18, colorpalette[wallpaper_color - 1]);
1157            break;
1158        case 5: // bpm
1159            tft.drawRect(57, 89, 57, 26, colorpalette[wallpaper_color - 1]);
1160            break;
1161        case 6: // all
1162            tft.drawRect(133, 63, 24, 24, colorpalette[wallpaper_color - 1]);
1163            update_hit();
1164            break;
1165        case 7: // 1.hit
1166            tft.drawRect(133, 38, 24, 24, colorpalette[wallpaper_color - 1]);
1167            update_hit();
1168            break;
1169
1170        default:
1171            break;
1172        }
1173    }
1174
1175    void update_repeat() {
1176        repeat_count = repeat[current_playing_tab];
1177
1178        tft.fillRect(8, 111, 36, 16, colorpalette[wallpaper_color - 1]);
1179
```

```
1180        tft.setTextSize(2);
1181        tft.setTextColor(colorpalette[text_color - 1]);
1182
1183        tft.setCursor(9, 112);
1184
1185        if (repeat[active_tab - 1] < 10) {
1186            tft.print("  ");
1187            tft.print(repeat[active_tab - 1]);
1188        }
1189
1190        if (repeat[active_tab - 1] >= 10 && repeat[active_tab - 1] < 100) {
1191            tft.print(" ");
1192            tft.print(repeat[active_tab - 1]);
1193        }
1194
1195        if (repeat[active_tab - 1] >= 100) {
1196            tft.print(repeat[active_tab - 1]);
1197        }
1198    }
1199
1200    void update_brightness() {
1201        if (millis() - last_press < ((unsigned long)backlight_on_time * 1000) || backlight_on_time == 0) {
1202            current_brightness = backlight_on_brightness;
1203        }
1204
1205        else if (current_brightness != 0) {
1206            if (millis() - last_brightness_update >= backlight_fade_time_1 && current_brightness > backlight_dim_brightness) {
1207                last_brightness_update = millis();
1208                current_brightness -= 1;
1209            }
1210
1211            else if (millis() - last_brightness_update >= backlight_dim_time) {
1212                last_brightness_update = millis();
1213                current_brightness -= 1;
1214            }
1215
1216            else if (millis() - last_brightness_update >= backlight_fade_time_2 && current_brightness < backlight_dim_brightness) {
1217                last_brightness_update = millis();
1218                current_brightness -= 1;
1219            }
1220
1221        }
1222        analogWrite(backlight_pin, current_brightness);
1223    }
1224
1225    void print_battery() {
1226
1227        tft.setTextSize(1);
1228        x = width - 82;
1229        y = height - 8;
1230        tft.setCursor(x, y);
1231        tft.print("Bat.");
1232        x = width - 38;
1233        tft.setCursor(x, y);
1234        tft.print("%");
1235
1236        x = width - 27;
1237        tft.fillRect(x, y, 27, 8, ST7735_RED);
1238        x = width - 29;
1239        y = height - 6;
1240        tft.drawRect(x, y, 2, 4, ST7735_RED);
1241    }
1242
1243    int read_battery_percent() {
1244
1245        total_sum_voltage = total_sum_voltage - voltage_readings[voltage_read_index];
1246        voltage_readings[voltage_read_index] = analogRead(battery_voltage_pin);
1247        total_sum_voltage = total_sum_voltage + voltage_readings[voltage_read_index];
1248
1249        voltage_read_index = voltage_read_index + 1;
1250
1251        if (voltage_read_index >= voltage_readings_count) {
1252            voltage_read_index = 0;
1253        }
1254
1255        raw_battery_voltage = total_sum_voltage / voltage_readings_count;
1256
1257
1258        if (raw_battery_voltage < battery_voltage_reference[0]) {
1259            return 0;
1260        }
1261
1262        if (raw_battery_voltage > battery_voltage_reference[10]) {
1263            return 100;
1264        }
1265
1266        for (int i = 0; i < sizeof(battery_voltage_reference) - 1; i++) {
1267            if (raw_battery_voltage < battery_voltage_reference[i]) {
1268                continue;
1269            }
1270            else if (raw_battery_voltage >= battery_voltage_reference[i] && raw_battery_voltage <= battery_voltage_reference[i + 1]) {
```

```
1271                return map(raw_battery_voltage, battery_voltage_reference[i], battery_voltage_reference[i + 1], i * 10, (i + 1) * 10);
1272            }
1273        }
1274    }
1275
1276    int read_charge_status() {
1277        int raw_status = analogRead(charge_status_pin);
1278        // 510 = full
1279        // 0 = charge
1280        // 1023 = unblugged
1281
1282        if (raw_status <= 250) {
1283            return 1;       // Charge
1284        }
1285        else if (raw_status > 250 && raw_status <= 750) {
1286            return 2;       // Full
1287        }
1288        else if (raw_status > 750) {
1289            return 0;       // Unblugged
1290        }
1291    }
1292
1293    void update_battery() {
1294
1295        if (millis() - last_battery_update > 1000) {
1296            last_battery_update = millis();
1297            battery_percent = read_battery_percent(); // read_battery_percent();
1298            charge_status = read_charge_status();
1299            x = width - 93;
1300            y = height - 11;
1301            if (charge_status == 1 && last_charge_status != 1) {
1302                tft.fillRect(x, y, 9, 11, colorpalette[wallpaper_color - 1]); // lataus
1303                tft.drawBitmap(x, y, charge_9_11, 9, 11, ST7735_YELLOW);
1304            }
1305            if (charge_status == 2 && last_charge_status != 2) {
1306                tft.fillRect(x, y, 9, 11, colorpalette[wallpaper_color - 1]); // täynnä
1307                tft.drawBitmap(x, y, charge_9_11, 9, 11, ST7735_GREEN);
1308            }
1309            if (charge_status == 0 && last_charge_status != 0) {
1310                tft.fillRect(x, y, 9, 11, colorpalette[wallpaper_color - 1]); // tyhjennys
1311                // tft.drawBitmap(x, y, charge_9_11, 9, 11, colorpalette[wallpaper_color - 1]);
1312            }
1313            if (last_battery_percent != battery_percent || charge_status != last_charge_status) {
1314                last_battery_percent = battery_percent;
1315
1316                x = width - 56;
1317                y = height - 8;
1318                tft.fillRect(x, y, 17, 8, colorpalette[wallpaper_color - 1]); // tyhjennys
1319
1320                tft.setTextSize(1);
1321
1322                tft.setCursor(x, y);
1323                tft.setTextColor(colorpalette[text_color - 1]);
1324
1325
1326                if (charge_status >= 2) {
1327                    tft.print(100);
1328                }
1329
1330                else if (battery_percent >= 100 && charge_status == 1) {
1331                    tft.print(" ");
1332                    tft.print(99);
1333                }
1334
1335                else if (battery_percent >= 100 && charge_status == 0) {
1336                    tft.print(100);
1337                }
1338
1339                else if (battery_percent >= 10) {
1340                    tft.print(" ");
1341                    tft.print(battery_percent);
1342                }
1343                else {
1344                    tft.print("  ");
1345                    tft.print(battery_percent);
1346                }
1347
1348                x = width - 26;
1349                y = height - 7;
1350                tft.fillRect(x, y, 25 - (battery_percent / 4), 6, colorpalette[wallpaper_color - 1]);
1351                x = ((width - 26) + (25 - (battery_percent / 4)));
1352                tft.fillRect(x, y, (battery_percent / 4), 6, ST7735_RED);
1353            }
1354            last_charge_status = charge_status;
1355        }
1356    }
1357
1358    void print_LED(int color) {
1359        analogWrite(led_pin_red, (led_colorpalette[color - 1][0]) * led_brightness);
1360        analogWrite(led_pin_green, (led_colorpalette[color - 1][1]) * led_brightness);
1361        analogWrite(led_pin_blue, (led_colorpalette[color - 1][2]) * led_brightness);
```

```
1362  }
1363
1364  void detect_headphone_jack() {
1365      if (digitalRead(headphone_detect_pin) == HIGH) {
1366          digitalWrite(speaker_mute_pin, LOW);
1367      }
1368      else {
1369          digitalWrite(speaker_mute_pin, HIGH);
1370      }
1371  }
1372
1373  void soundcard() {
1374      if (play == true) {
1375          digitalWrite(amp_CS_pin, LOW);
1376          delay(50);
1377      }
1378      else {
1379          digitalWrite(amp_CS_pin, HIGH);
1380      }
1381  }
1382
1383  void reset_play() {
1384      last_hit_time = 0;
1385      current_playing_tab = 0;
1386      repeat_count = repeat[current_playing_tab];
1387      hit_num = 0;
1388      first_beat = true;
1389  }
1390
1391  //******** SETTINGS *****************************************************************************************************
1392
1393  void button_settings(int button_number) { //boolean buttonState, int pin, unsigned long lastDebounceTime, boolean lastState, int
        button_timer_counter) {
1394      buttonState[button_number] = digitalRead(buttonpins[button_number]);
1395      if (millis() - lastDebounceTime[button_number] > debounceDelay) {
1396          lastDebounceTime[button_number] = millis();
1397
1398          if (buttonState[button_number] == LOW && lastState[button_number] == HIGH) { // 1. time aftre press
1399
1400              button_timer_counter[button_number] = 0;
1401              lastState[button_number] = LOW;
1402              lastDebounceTime[button_number] = millis();
1403              last_press = millis();
1404              //return 0
1405          }
1406
1407          else if (buttonState[button_number] == LOW && lastState[button_number] == LOW) { // during press
1408              button_timer_counter[button_number] += 1;
1409              lastState[button_number] = LOW;
1410              last_press = millis();
1411          }
1412
1413          else if (buttonState[button_number] == HIGH && lastState[button_number] == LOW && button_timer_counter[button_number] < 20) { //
              normal press
1414              lastState[button_number] = HIGH;
1415              delete_settings_edit_box();
1416              switch (button_number)
1417              {
1418              case 0: // tab +
1419
1420                  break;
1421
1422              case 1: // tab -
1423
1424                  break;
1425
1426              case 2: // editbox +
1427                  settings_edit_box += 1;
1428                  if (settings_edit_box > 15) {
1429                      settings_edit_box = 1;
1430                  }
1431                  break;
1432
1433              case 3: // editbox -
1434                  settings_edit_box -= 1;
1435                  if (settings_edit_box < 1) {
1436                      settings_edit_box = 15;
1437                  }
1438                  break;
1439
1440              case 4: // button up
1441                  switch (settings_edit_box)
1442                  {
1443                  case 0:
1444                      // no box
1445                      break;
1446                  case 1:
1447                      very_high_note_frequency += 1;
1448                      if (very_high_note_frequency > 20000) {
1449                          very_high_note_frequency = 100;
1450                      }
```

```
1451                    break;
1452                case 2:
1453                    high_note_frequency += 1;
1454                    if (high_note_frequency > 20000) {
1455                        high_note_frequency = 100;
1456                    }
1457                    break;
1458                case 3:
1459                    low_note_frequency += 1;
1460                    if (low_note_frequency > 20000) {
1461                        low_note_frequency = 100;
1462                    }
1463                    break;
1464                case 4:
1465                    note_duration += 1;
1466                    if (note_duration > 1000) {
1467                        note_duration = 1;
1468                    }
1469                    break;
1470                case 5:
1471                    backlight_on_time += 1;
1472                    if (backlight_on_time > 600) {
1473                        backlight_on_time = 0;
1474                    }
1475                    break;
1476                case 6:
1477                    backlight_on_brightness += 1;
1478                    if (backlight_on_brightness == 0) {
1479                        backlight_on_brightness = 1;
1480                    }
1481                    break;
1482                case 7:
1483                    led_brightness += 1;
1484                    break;
1485                case 8:
1486                    wallpaper_color += 1;
1487                    if (wallpaper_color > 8) {
1488                        wallpaper_color = 1;
1489                    }
1490                    break;
1491                case 9:
1492                    text_color += 1;
1493                    if (text_color > 8) {
1494                        text_color = 1;
1495                    }
1496                    break;
1497                case 10:
1498                    tab_on_color += 1;
1499                    if (tab_on_color > 8) {
1500                        tab_on_color = 1;
1501                    }
1502                    break;
1503                case 11:
1504                    tab_on_number_color += 1;
1505                    if (tab_on_number_color > 8) {
1506                        tab_on_number_color = 1;
1507                    }
1508                    break;
1509                case 12:
1510                    tab_off_color += 1;
1511                    if (tab_off_color > 8) {
1512                        tab_off_color = 1;
1513                    }
1514                    break;
1515                case 13:
1516                    tab_off_number_color += 1;
1517                    if (tab_off_number_color > 8) {
1518                        tab_off_number_color = 1;
1519                    }
1520                    break;
1521                case 14:
1522                    edit_box_color += 1;
1523                    if (edit_box_color > 8) {
1524                        edit_box_color = 1;
1525                    }
1526                    break;
1527                case 15:
1528                    LED_color += 1;
1529                    if (LED_color > 8) {
1530                        LED_color = 1;
1531                    }
1532                    break;
1533                default:
1534                    break;
1535                }
1536                break;
1537
1538            case 5: // button down
1539                switch (settings_edit_box)
1540                {
1541                case 0:
```

```
1542                        // no box
1543                        break;
1544                    case 1:
1545                        very_high_note_frequency -= 1;
1546                        if (very_high_note_frequency < 100) {
1547                            very_high_note_frequency = 20000;
1548                        }
1549                        break;
1550                    case 2:
1551                        high_note_frequency -= 1;
1552                        if (high_note_frequency < 100) {
1553                            high_note_frequency = 20000;
1554                        }
1555                        break;
1556                    case 3:
1557                        low_note_frequency -= 1;
1558                        if (low_note_frequency < 100) {
1559                            low_note_frequency = 20000;
1560                        }
1561                        break;
1562                    case 4:
1563                        note_duration -= 1;
1564                        if (note_duration < 1) {
1565                            note_duration = 1000;
1566                        }
1567                        break;
1568                    case 5:
1569                        backlight_on_time -= 1;
1570                        if (backlight_on_time < 0) {
1571                            backlight_on_time = 600;
1572                        }
1573                        break;
1574                    case 6:
1575                        backlight_on_brightness -= 1;
1576                        if (backlight_on_brightness == 0) {
1577                            backlight_on_brightness = 255;
1578                        }
1579                        break;
1580                    case 7:
1581                        led_brightness -= 1;
1582                        break;
1583                    case 8:
1584                        wallpaper_color -= 1;
1585                        if (wallpaper_color < 1) {
1586                            wallpaper_color = 8;
1587                        }
1588                        break;
1589                    case 9:
1590                        text_color -= 1;
1591                        if (text_color < 1) {
1592                            text_color = 8;
1593                        }
1594                        break;
1595                    case 10:
1596                        tab_on_color -= 1;
1597                        if (tab_on_color < 1) {
1598                            tab_on_color = 8;
1599                        }
1600                        break;
1601                    case 11:
1602                        tab_on_number_color -= 1;
1603                        if (tab_on_number_color < 1) {
1604                            tab_on_number_color = 8;
1605                        }
1606                        break;
1607                    case 12:
1608                        tab_off_color -= 1;
1609                        if (tab_off_color < 1) {
1610                            tab_off_color = 8;
1611                        }
1612                        break;
1613                    case 13:
1614                        tab_off_number_color -= 1;
1615                        if (tab_off_number_color < 1) {
1616                            tab_off_number_color = 8;
1617                        }
1618                        break;
1619                    case 14:
1620                        edit_box_color -= 1;
1621                        if (edit_box_color < 1) {
1622                            edit_box_color = 8;
1623                        }
1624                        break;
1625                    case 15:
1626                        LED_color -= 1;
1627                        if (LED_color < 1) {
1628                            LED_color = 8;
1629                        }
1630                        break;
1631                    default:
1632                        break;
```

```
                }
                break;
            default:
                break;
            }
            print_settings_edit_box();
            print_settings();
        }

        if (button_timer_counter[button_number] > 40 && buttonState[button_number] == LOW) { // very long press
            switch (button_number)
            {
            case 4: // button up
                switch (settings_edit_box)
                {
                case 0:
                    // no box
                    break;
                case 1:
                    very_high_note_frequency += 50;
                    if (very_high_note_frequency > 20000) {
                        very_high_note_frequency = 100;
                    }
                    break;
                case 2:
                    high_note_frequency += 50;
                    if (high_note_frequency > 20000) {
                        high_note_frequency = 100;
                    }
                    break;
                case 3:
                    low_note_frequency += 50;
                    if (low_note_frequency > 20000) {
                        low_note_frequency = 100;
                    }
                    break;
                case 4:
                    note_duration += 10;
                    if (note_duration > 1000) {
                        note_duration = 1;
                    }
                    break;
                case 5:
                    backlight_on_time += 10;
                    if (backlight_on_time > 600) {
                        backlight_on_time = 0;
                    }
                    break;
                case 6:
                    backlight_on_brightness += 5;
                    if (backlight_on_brightness == 0) {
                        backlight_on_brightness = 1;
                    }
                    break;
                case 7:
                    led_brightness += 5;
                    break;
                default:
                    break;
                }
                break;

            case 5: // button down
                switch (settings_edit_box)
                {
                case 0:
                    // no box
                    break;
                case 1:
                    very_high_note_frequency -= 50;
                    if (very_high_note_frequency < 100) {
                        very_high_note_frequency = 20000;
                    }
                    break;
                case 2:
                    high_note_frequency -= 50;
                    if (high_note_frequency < 100) {
                        high_note_frequency = 20000;
                    }
                    break;
                case 3:
                    low_note_frequency -= 50;
                    if (low_note_frequency < 100) {
                        low_note_frequency = 20000;
                    }
                    break;
                case 4:
                    note_duration -= 10;
                    if (note_duration < 1) {
                        note_duration = 1000;
                    }
```

```
                    break;
                case 5:
                    backlight_on_time -= 10;
                    if (backlight_on_time < 0) {
                        backlight_on_time = 600;
                    }
                    break;
                case 6:
                    backlight_on_brightness -= 5;
                    if (backlight_on_brightness == 0) {
                        backlight_on_brightness = 255;
                    }
                    break;
                case 7:
                    led_brightness -= 5;
                    break;
                default:
                    break;
                }
                break;

            case 6:
                save_settings();
                noTone(tone_pin);
                digitalWrite(amp_CS_pin, LOW);      // vahvistin päälle
                print_LED(1);
                tone(tone_pin, 2000, 200);
                delay(200);
                print_LED(3);
                tone(tone_pin, 1000, 200);
                while (digitalRead(buttonpins[6]) == LOW) {}
                delay(200);
                print_LED(LED_color);
                digitalWrite(amp_CS_pin, HIGH);     // vahvistin pois
                break;
            default:
                break;
            }
            print_settings();
        }

        else if (button_timer_counter[button_number] > 20 && buttonState[button_number] == LOW) { // long press
            switch (button_number)
            {
            case 4: // button up
                switch (settings_edit_box)
                {
                case 0:
                    // no box
                    break;
                case 1:
                    very_high_note_frequency += 25;
                    if (very_high_note_frequency > 20000) {
                        very_high_note_frequency = 100;
                    }
                    break;
                case 2:
                    high_note_frequency += 25;
                    if (high_note_frequency > 20000) {
                        high_note_frequency = 100;
                    }
                    break;
                case 3:
                    low_note_frequency += 25;
                    if (low_note_frequency > 20000) {
                        low_note_frequency = 100;
                    }
                    break;
                case 4:
                    note_duration += 5;
                    if (note_duration > 1000) {
                        note_duration = 1;
                    }
                    break;
                case 5:
                    backlight_on_time += 5;
                    if (backlight_on_time > 600) {
                        backlight_on_time = 0;
                    }
                    break;
                case 6:
                    backlight_on_brightness += 5;
                    if (backlight_on_brightness == 0) {
                        backlight_on_brightness = 1;
                    }
                    break;
                case 7:
                    led_brightness += 5;
                    break;
                default:
                    break;
```

```
                }
                break;

            case 5: // button down
                switch (settings_edit_box)
                {
                case 0:
                    // no box
                    break;
                case 1:
                    very_high_note_frequency -= 25;
                    if (very_high_note_frequency < 100) {
                        very_high_note_frequency = 20000;
                    }
                    break;
                case 2:
                    high_note_frequency -= 25;
                    if (high_note_frequency < 100) {
                        high_note_frequency = 20000;
                    }
                    break;
                case 3:
                    low_note_frequency -= 25;
                    if (low_note_frequency < 100) {
                        low_note_frequency = 20000;
                    }
                    break;
                case 4:
                    note_duration -= 5;
                    if (note_duration < 1) {
                        note_duration = 1000;
                    }
                    break;
                case 5:
                    backlight_on_time -= 5;
                    if (backlight_on_time < 0) {
                        backlight_on_time = 600;
                    }
                    break;
                case 6:
                    backlight_on_brightness -= 5;
                    if (backlight_on_brightness == 0) {
                        backlight_on_brightness = 255;
                    }
                    break;
                case 7:
                    led_brightness -= 5;
                    break;
                default:
                    break;
                }
                break;

            default:
                break;
            }
            print_settings();
        }

        if (buttonState[button_number] == HIGH) { // no press
            button_timer_counter[button_number] = 0;
            lastState[button_number] = HIGH;
        }
    }
}

void print_settings_text() {
    int b = 0;
    int c = 116;
    tft.setTextSize(1);
    tft.setTextColor(colorpalette[text_color - 1]);

    tft.setCursor(b, 1);
    tft.print("1.All freg. --");
    tft.setCursor(c, 1);
    tft.print("Hz");

    tft.setCursor(b, 9);
    tft.print("1.Hit freg. --");
    tft.setCursor(c, 9);
    tft.print("Hz");

    tft.setCursor(b, 17);
    tft.print("Beat freg. ---");
    tft.setCursor(c, 17);
    tft.print("Hz");

    tft.setCursor(b, 25);
    tft.print("Hit duration -");
    tft.setCursor(c, 25);
    tft.print("ms");
```

```
1906
1907        tft.setCursor(b, 33);
1908        tft.print("Light on time");
1909        tft.setCursor(c, 33);
1910        tft.print("s");
1911
1912        tft.setCursor(b, 41);
1913        tft.print("Brightness ---");
1914
1915        tft.setCursor(b, 49);
1916        tft.print("LED brightness");
1917
1918        tft.setCursor(b, 57);
1919        tft.print("Wallpaper ----");
1920
1921        tft.setCursor(b, 65);
1922        tft.print("Text ---------");
1923
1924        tft.setCursor(b, 73);
1925        tft.print("Tab on -------");
1926
1927        tft.setCursor(b, 81);
1928        tft.print("Tab on number ");
1929
1930        tft.setCursor(b, 89);
1931        tft.print("Tab off ------");
1932
1933        tft.setCursor(b, 97);
1934        tft.print("Tab off number");
1935
1936        tft.setCursor(b, 105);
1937        tft.print("Edit box -----");
1938
1939        tft.setCursor(b, 113);
1940        tft.print("Hit LED ------");
1941    }
1942
1943    void print_settings() {
1944
1945        int a = 85; // leveys/aloitus
1946        tft.setTextSize(1);
1947        tft.setTextColor(colorpalette[text_color - 1]);
1948
1949        switch (settings_edit_box)
1950        {
1951        case 1:
1952            tft.fillRect(a, 1, 29, 7, colorpalette[wallpaper_color - 1]);
1953            tft.setCursor(a, 1);
1954            tft.print(very_high_note_frequency);
1955            break;
1956        case 2:
1957            tft.fillRect(a, 9, 29, 7, colorpalette[wallpaper_color - 1]);
1958            tft.setCursor(a, 9);
1959            tft.print(high_note_frequency);
1960            break;
1961        case 3:
1962            tft.fillRect(a, 17, 29, 7, colorpalette[wallpaper_color - 1]);
1963            tft.setCursor(a, 17);
1964            tft.print(low_note_frequency);
1965            break;
1966        case 4:
1967            tft.fillRect(a, 25, 29, 7, colorpalette[wallpaper_color - 1]);
1968            tft.setCursor(a, 25);
1969            tft.print(note_duration);
1970            break;
1971        case 5:
1972            tft.fillRect(a, 33, 29, 7, colorpalette[wallpaper_color - 1]);
1973            tft.setCursor(a, 33);
1974            tft.print(backlight_on_time);
1975            break;
1976        case 6:
1977            tft.fillRect(a, 41, 29, 7, colorpalette[wallpaper_color - 1]);
1978            tft.setCursor(a, 41);
1979            tft.print(backlight_on_brightness);
1980            break;
1981        case 7:
1982            tft.fillRect(a, 49, 29, 7, colorpalette[wallpaper_color - 1]);
1983            tft.setCursor(a, 49);
1984            tft.print(led_brightness);
1985            break;
1986        case 8:
1987            tft.fillRect(a, 57, 29, 7, colorpalette[wallpaper_color - 1]);
1988            break;
1989        case 9:
1990            tft.fillRect(a, 65, 29, 7, colorpalette[text_color - 1]);
1991            break;
1992        case 10:
1993            tft.fillRect(a, 73, 29, 7, colorpalette[tab_on_color - 1]);
1994            break;
1995        case 11:
1996            tft.fillRect(a, 81, 29, 7, colorpalette[tab_on_number_color - 1]);
```

```
            break;
        case 12:
            tft.fillRect(a, 89, 29, 7, colorpalette[tab_off_color - 1]);
            break;
        case 13:
            tft.fillRect(a, 97, 29, 7, colorpalette[tab_off_number_color - 1]);
            break;
        case 14:
            tft.fillRect(a, 105, 29, 7, colorpalette[edit_box_color - 1]);
            break;
        case 15:
            tft.fillRect(a, 113, 29, 7, colorpalette[LED_color - 1]);
            break;
        default:
            break;
        }
    }

    void print_settings_edit_box() {
        int a = 84;

        tft.drawRect(a, (settings_edit_box - 1) * 8, 31, 9, colorpalette[edit_box_color - 1]);
    }

    void delete_settings_edit_box() {
        int a = 84;

        tft.drawRect(a, (settings_edit_box - 1) * 8, 31, 9, colorpalette[wallpaper_color - 1]);
    }
```

```
// Images.h
// Tekijä: Lassi Parkkila

#ifndef _IMAGES_h
#define _IMAGES_h

#if defined(ARDUINO) && ARDUINO >= 100
    #include "arduino.h"
#else
    #include "WProgram.h"
#endif

// Bragi
static const uint8_t PROGMEM bragi_160_128[] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x01, 0xff, 0x00, 0x1e, 0x01, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x07, 0xff, 0xff, 0xfe, 0x0f, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x0f, 0xff, 0xff, 0xfc, 0x3f, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x1f, 0xff, 0xff, 0xf1, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8,
    0x1f, 0xff, 0xff, 0xc7, 0xf3, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0,
    0x1e, 0xff, 0xff, 0x8f, 0xc1, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0,
    0x38, 0x00, 0x1e, 0x1f, 0x80, 0x7f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0,
    0x38, 0x00, 0x7c, 0x3f, 0x00, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xc0,
    0x30, 0x00, 0xfc, 0x7e, 0x00, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xc0,
    0x00, 0x01, 0xf8, 0x7e, 0x00, 0x1f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80,
    0x00, 0x03, 0xf0, 0xfe, 0x00, 0x1f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x03, 0xf0, 0xfe, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x07, 0xf0, 0xfe, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x07, 0xf0, 0xfe, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x07, 0xf0, 0xfe, 0x60, 0x3f, 0x00, 0x07, 0x00, 0x60, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x0e, 0x00,
    0x00, 0x07, 0xe0, 0xfe, 0x60, 0x3e, 0x00, 0x0f, 0x80, 0xf0, 0x00, 0x1f, 0xc0, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x1f, 0x80,
    0x00, 0x0f, 0xe0, 0xfe, 0x60, 0x7c, 0x00, 0x1f, 0xc3, 0xf8, 0x00, 0xff, 0xfd, 0xc0, 0x00, 0xff, 0xe0, 0x18, 0x3f, 0xc0,
    0x00, 0x0f, 0xe0, 0xfe, 0x60, 0x78, 0x00, 0x3f, 0xe7, 0xfc, 0x0f, 0xff, 0xff, 0xc0, 0x07, 0xff, 0xf8, 0x78, 0x7f, 0xe0,
    0x00, 0x0f, 0xe1, 0xfe, 0x60, 0xf0, 0x00, 0x7f, 0xef, 0xfe, 0x07, 0xff, 0xff, 0x80, 0x3f, 0xff, 0xff, 0xf8, 0xff, 0xf0,
    0x03, 0xff, 0xe1, 0xfe, 0x61, 0xc0, 0x00, 0xff, 0xfd, 0xfe, 0x06, 0x7f, 0xff, 0x80, 0x7f, 0xff, 0xff, 0xf8, 0x7f, 0xe0,
    0x0f, 0xff, 0xe1, 0xfe, 0x63, 0xfe, 0x00, 0xcf, 0xf9, 0xfe, 0x07, 0x0f, 0xff, 0x00, 0x7f, 0x1f, 0xff, 0xf8, 0x7f, 0xc0,
    0x1f, 0xff, 0xe1, 0xfe, 0x6f, 0xff, 0x80, 0x0f, 0xf0, 0xf8, 0x03, 0x01, 0xff, 0x00, 0x7f, 0x03, 0xff, 0xf8, 0x3f, 0x80,
    0x1f, 0xff, 0xe1, 0xfe, 0x7c, 0x7f, 0xc0, 0x0f, 0xe0, 0x70, 0x03, 0x80, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0xf8, 0x3f, 0x80,
    0x1e, 0x7f, 0xe1, 0xfe, 0x78, 0x1f, 0xc0, 0x0f, 0xe0, 0x60, 0x01, 0x80, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x38, 0x1f, 0xe1, 0xfe, 0x60, 0x1f, 0xe0, 0x0f, 0xe0, 0x00, 0x01, 0xc0, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x38, 0x0f, 0xe1, 0xfe, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x00, 0xc0, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x0f, 0xe1, 0xfe, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x01, 0xe0, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x07, 0xe1, 0xfe, 0x7c, 0x7f, 0xe0, 0x0f, 0xe0, 0x00, 0x01, 0xf8, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x07, 0xe1, 0xfe, 0x7f, 0xff, 0xe0, 0x0f, 0xe0, 0x00, 0x03, 0xfe, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x07, 0xe1, 0xfc, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x07, 0x8f, 0xff, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0xff, 0xe1, 0xfc, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x07, 0x81, 0xff, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x01, 0xff, 0xe1, 0xfc, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x0f, 0x00, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x03, 0xff, 0xe1, 0xfc, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x1f, 0x00, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x03, 0xff, 0xe1, 0xf8, 0x7f, 0xff, 0xe0, 0x0f, 0xe0, 0x00, 0x1f, 0x00, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x03, 0xdf, 0xe3, 0xf0, 0x7c, 0x7f, 0xe0, 0x0f, 0xe0, 0x00, 0x3f, 0x00, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x07, 0x0f, 0xe3, 0xe0, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x3f, 0x00, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x07, 0x07, 0xc7, 0xc0, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x3f, 0x00, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x07, 0xcf, 0x00, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x3f, 0x80, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x07, 0xbc, 0x00, 0x60, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x3f, 0x80, 0x7f, 0x00, 0x7f, 0x00, 0xfe, 0x00, 0x3f, 0x80,
    0x00, 0x0f, 0xff, 0xff, 0xe0, 0x0f, 0xe0, 0x0f, 0xe0, 0x00, 0x3f, 0xc0, 0xff, 0x00, 0x7f, 0x01, 0xff, 0x00, 0x3f, 0x80,
    0x00, 0x1f, 0xff, 0xff, 0xe0, 0x1f, 0xe0, 0x0f, 0xf0, 0x00, 0x3f, 0xc1, 0xff, 0x00, 0x7f, 0x03, 0xff, 0x00, 0x3f, 0xc0,
    0x00, 0x3f, 0xff, 0xff, 0xf8, 0xff, 0xc0, 0x1f, 0xfc, 0xe0, 0x3f, 0xe7, 0xff, 0x30, 0x7f, 0xc7, 0xff, 0x00, 0x3f, 0xcc,
    0x00, 0x3f, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x3f, 0xff, 0xe0, 0x3f, 0xff, 0x7f, 0xf0, 0x7f, 0xee, 0x7f, 0x80, 0x3f, 0xfc,
    0x00, 0x7f, 0xff, 0xff, 0xff, 0x80, 0x00, 0x3f, 0xff, 0x80, 0x1f, 0xfc, 0x7f, 0xe0, 0x7f, 0xfe, 0x7f, 0x80, 0x1f, 0xf8,
    0x00, 0xff, 0x0f, 0xff, 0xfe, 0x00, 0x00, 0x1f, 0xff, 0x00, 0x1f, 0xf8, 0x7f, 0xc0, 0x1f, 0xfc, 0x3f, 0xc0, 0x1f, 0xf0,
    0x01, 0xf0, 0x00, 0x3f, 0xf8, 0x00, 0x00, 0x03, 0xfe, 0x00, 0x0f, 0xf0, 0x3f, 0x80, 0x0f, 0xf8, 0x3f, 0xe0, 0x1f, 0xc0,
    0x03, 0xc0, 0x00, 0x0f, 0xe0, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x07, 0xc0, 0x3f, 0x00, 0x03, 0xf0, 0x1f, 0xf0, 0x0f, 0x80,
    0x07, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x00, 0x70, 0x00, 0x03, 0x80, 0x0c, 0x00, 0x01, 0xe0, 0x0f, 0xf0, 0x07, 0x00,
    0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x0f, 0xf8, 0x00, 0x00,
    0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xc0, 0x07, 0xf8, 0x00, 0x00,
    0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xf3, 0xf8, 0x00, 0x00,
    0x0f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xfd, 0xf0, 0x00, 0x00,
    0x0f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xff, 0xf0, 0x00, 0x00,
    0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xe0, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0x80, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xc7, 0xff, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x00, 0x7c, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
92      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
93      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
94      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
95      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
96      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
97      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
98      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
99      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
100     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
101     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
102     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
103     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
104     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
105     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
106     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
107     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
108     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
109     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
110     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
111     0x00, 0x00, 0x1c, 0x70, 0xe0, 0x1e, 0x1e, 0x06, 0x30, 0x3c, 0x07, 0x1c, 0x03, 0x80, 0xe3, 0x87, 0x00, 0xf0, 0x00, 0x00,
112     0x00, 0x00, 0x3e, 0xfb, 0xfc, 0xfe, 0x1f, 0xcf, 0x79, 0xff, 0x8f, 0x7f, 0x9f, 0xf1, 0xf7, 0xdf, 0xe7, 0xf0, 0x00, 0x00,
113     0x00, 0x00, 0x7f, 0xff, 0xf8, 0xff, 0x3f, 0xdf, 0xf9, 0xff, 0x9f, 0xff, 0x1f, 0xfb, 0xff, 0xff, 0xc7, 0xf8, 0x00, 0x00,
114     0x00, 0x00, 0x1f, 0x3c, 0xf0, 0xff, 0x9e, 0x3f, 0xf9, 0xef, 0x9f, 0x9e, 0x1c, 0xf8, 0xf9, 0xe7, 0x87, 0xfc, 0x00, 0x00,
115     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xf7, 0xde, 0x0f, 0x31, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0xbe, 0x00, 0x00,
116     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xf3, 0xde, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0x9e, 0x00, 0x00,
117     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xf3, 0x9e, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0x9c, 0x00, 0x00,
118     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xfe, 0x1e, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0xf0, 0x00, 0x00,
119     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xfc, 0x1e, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0xe0, 0x00, 0x00,
120     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xf8, 0x1e, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0xc0, 0x00, 0x00,
121     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xf0, 0x1e, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0x80, 0x00, 0x00,
122     0x00, 0x00, 0x1e, 0x3c, 0xf0, 0xf0, 0x1e, 0x0f, 0x01, 0xe7, 0x87, 0x1e, 0x1c, 0x78, 0xf1, 0xe7, 0x87, 0x80, 0x00, 0x00,
123     0x00, 0x00, 0x1f, 0x7c, 0xf1, 0xf1, 0x9e, 0x0f, 0x03, 0xf7, 0x8f, 0x9e, 0x3f, 0x78, 0xfb, 0xe7, 0x8f, 0x8c, 0x00, 0x00,
124     0x00, 0x00, 0x3f, 0x7e, 0xfd, 0xff, 0xbf, 0xdf, 0xe3, 0xff, 0x8f, 0x9f, 0xbf, 0xf9, 0xfb, 0xf7, 0xef, 0xfc, 0x00, 0x00,
125     0x00, 0x00, 0x1e, 0x3c, 0x78, 0x7f, 0x1f, 0xcf, 0xe0, 0xff, 0x8f, 0x8f, 0x1f, 0xf8, 0xf1, 0xe3, 0xc3, 0xf8, 0x00, 0x00,
126     0x00, 0x00, 0x0c, 0x18, 0x70, 0x1c, 0x07, 0x03, 0x80, 0x3c, 0x07, 0x0e, 0x03, 0xc0, 0x60, 0xc3, 0x80, 0xe0, 0x00, 0x00,
127     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
128     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
129     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
130     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
131     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
132     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
133     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
134     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
135     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
136     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
137     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
138     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
139     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
140     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
141     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
142     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
143 };
144
145 // Check-merkki
146 static const uint8_t PROGMEM logo16_glcd_bmp[] = {
147     0x00, 0x00, 0x00,
148     0x00, 0x00, 0x1c,
149     0x00, 0x00, 0x1f,
150     0x00, 0x00, 0x3f,
151     0x00, 0x00, 0x7e,
152     0x00, 0x00, 0x7c,
153     0x00, 0x00, 0xfc,
154     0x00, 0x01, 0xf8,
155     0x00, 0x01, 0xf8,
156     0x00, 0x03, 0xf0,
157     0x00, 0x07, 0xe0,
158     0x00, 0x07, 0xc0,
159     0x38, 0x0f, 0xc0,
160     0x7c, 0x1f, 0x80,
161     0xfe, 0x1f, 0x00,
162     0x7f, 0x3f, 0x00,
163     0x3f, 0xfe, 0x00,
164     0x1f, 0xfc, 0x00,
165     0x0f, 0xfc, 0x00,
166     0x07, 0xf8, 0x00,
167     0x03, 0xf0, 0x00,
168     0x01, 0xf0, 0x00,
169     0x00, 0xe0, 0x00,
170     0x00, 0x00, 0x00
171 };
172
173 // Salama
174 static const uint8_t PROGMEM charge_9_11[] = {
175     0x1e, 0x00,
176     0x3c, 0x00,
177     0x38, 0x00,
178     0x70, 0x00,
179     0x60, 0x00,
180     0xff, 0x80,
181     0x07, 0x80,
182     0x0f, 0x00,
```

```
183       0x1c, 0x00,
184       0x30, 0x00,
185       0x40, 0x00
186   };
187
188   #endif
189
```