Tampereen yliopisto

Aapo Manni

ARDUINO-ALUSTAN HYÖDYNTÄMINEN ELEKTRONIIKKALAITTEEN SUUNNITTELUSSA

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta (ITC)
Tarkastaja: Katja Laine
toukokuu 2025

TIIVISTELMÄ

Aapo Manni: Arduino-alustan hyödyntäminen elektroniikkalaitteen suunnittelussa (Utilizing the Arduino platform in desinging of an electronic device)
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Toukokuu 2025

Arduinojen suosio elektroniikkalaitteiden rakentamisessa on saavuttanut suurta suosiota viimeisen kymmenen vuoden aikana. Syynä tälle suosiolle on niiden halpa hinta ja aloittelijaystävällisyys. Suosion kasvua on kiihdyttänyt myös niiden avoimen lähdekoodin saavutettavuus, jonka ansiosta Arduinon piirikaaviot, sekä muut resurssit ovat kaikkien saatavilla.

Tässä kandidaatintyössä suunnitellaan, simuloidaan ja rakennetaan Arduino-pohjainen FM-radio, joka hyödyntää mikrokontrollerinaan Arduinosta tuttua Atmega328p:tä. Samalla Arduinoa käytetään laitteen testaamiseen koekytkentälevyillä, sekä lopullisen kokoonpanon ohjelmointiin. Tässä työssä Arduinoa siis käytetään melkein jokaisessa suunnittelun vaiheessa.

Tämä työ pyrkii vastaamaan kysymykseen siitä, voiko Arduinolla itse rakentaa toimivan FM-radion. Samalla perehdytään siihen, miten Arduinoa voidaan hyödyntää elektroniikkalaitteen suunnittelun eri vaiheessa sekä siihen, että voiko itse tehdyllä radiolla rahallisesti kilpailla kaupallista tuotetta vastaan.

Rakennettu laite osoittautuu toimivaksi, mutta samalla sen toimintaa parantavia kehityskohteita on yhä havaittavissa. Lopulta päädytään tulokseen, että kaupallinen radio on käytännöllisempi ja kustannustehokkaampi valinta, mikäli käyttäjä tarvitsee yksinkertaisen vaihtoehdon radiotaajuuksien kuunteluun. FM-radiota itse suunniteltaessa on kuitenkin mahdollista vaikuttaa laajemmin luodun laitteen ominaisuuksiin. Lopuksi todetaan, että Arduino-alusta on tehokas ja edullinen tapa suunnitella elektroniikkalaitteita, koska sitä on mahdollista käyttää lähes jokaisessa suunnittelun vaiheessa aina prototyypin rakentamisesta mikrokontrollerin ohjelmointiin asti.

Avainsanat: Arduino, FM-radio, elektroniikkalaitteen suunnittelu, sulautettu järjestelmä, mikrokontrolleri, C++-ohjelmointi

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

☐ Ei
☒ Kyllä

Ilmoitukseni mukaan olen käyttänyt opinnäytteessäni tutkielmaprosessin aikana seuraavia tekoälysovelluksia: ChatGPT, Andor Tekoälyavustaja

Tekoälysovellusten nimet ja versiot: GPT-4o, GPT-4, GPT-4o mini

Käyttötarkoitus: Osittainen käsitteiden suomennos, lauserakenteiden parannus, ohjelmakoodin syntaksin tarkistus sekä avustaminen lyhenneluettelon muodostamisessa. Lisäksi tiedonhaussa on käytetty Andorin Tekoälyavustajaa lähteiden etsintään.

Osiot, joissa tekoälyä on käytetty: Lyhenneluettelo, luvut 2 ja 3, lähteet ja liite D

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien osat, joissa on hyödynnetty tekoälyä, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

3D	engl. Three-dimensional, kolmiulotteinen
AC	engl. Alternating Current, vaihtovirta
ACK	engl. Acknowledgement, tiedonsiirron vahvistus
ADC	engl. Analog to Digital Conversion, analogisen signaalin muuntaminen digitaaliseksi
AVR	engl. Alf and Vegard's RISC processor, Atmelin mikrokontrolleriarkkitehtuuri
BOD	engl. Brown-Out Detector, jännitteen alaraja, jolla mikrokontrolleri resetoituu
C++	Ohjelmointikieli
DC	engl. Direct Current, tasavirta
DIP28	engl. Dual In-line Package, 28-pinninen kotelointi
FM	engl. Frequency Modulation, taajuusmodulaatio
GPIO	engl. General Purpose Input/Output, yleiskäyttöiset tulo-/lähtöpinnit
I2C	engl. Inter-Integrated Circuit, sarjamuotoinen tietoliikenneväylä mikropiirien välillä
IC	engl. Integrated Circuit, integroitu piiri
IDE	engl. Integrated Development Environment, integroitu kehitysympäristö
ISP	engl. In-System Programming, mikrokontrollerin ohjelmointi sen ollessa osa valmista piiriä
LCD	engl. Liquid Crystal Display, nestekidenäyttö
MPX	engl. Stereo Multiplexing, stereon ja monon yhdistäminen FM-signaalissa
PCB	engl. Printed Circuit Board, piirilevy
ppm	engl. Parts Per Million, miljoonasosa
RC	engl. Resistor-Capacitor, vastuksesta ja kondensaattorista koostuva oskillaattori
RDS	engl. Radio Data System, radiotietojärjestelmä
RST	engl. Reset, resetointipinni
R/W	engl. Read/Write, lukeminen/kirjoittaminen tiedonsiirrossa
SCL	engl. Serial Clock, kellosignaali
SDA	engl. Serial Data, sarjadata
SEN	engl. Serial Enable, kolmijohtoyhteyden käyttömahdollisuus

C	kapasitanssi
dB	desibeli
f_{-3dB}	-3 dB-rajataajuus, taajuus, jossa amplitudi on laskenut kolme desibeliä
Hz	hertsi, taajuuden yksikkö
I_G	maajohtimen virta
I_{in}	tulovirta
I_L	kuormavirta

kHz	kilohertsi, taajuuden yksikkö
kΩ	kilo-ohmi, resistanssin yksikkö
mA	milliampeeri, virran yksikkö
MHz	megahertsi, taajuuden yksikkö
mil	piirilevysuunnittelussa käytetty mittayksikkö, 0,0254 mm
mm	millimetri
nH	nanohenry, induktanssin yksikkö
P_D	regulaattorin tehohäviö
R	resistanssi
V_{in}	tulojännite
V_{out}	lähtöjännite
W	watti, tehon yksikkö
µF	mikrofaradi, kapasitanssin yksikkö

1. JOHDANTO

Ostettuani Arduino-aloituspakkauksen elektroniikkaharrastustani varten, havahduin pohtimaan, mitä kaikkea kyseisellä pakkauksella voisi luoda. Olen vuosien ajan ollut äärimmäisen kiinnostunut langattomasta tiedonsiirrosta ja etenkin FM-radioista, mikä herätti kysymyksen siitä, voiko Arduinon kaltaisella laitteella luoda itse kotiolossa toimivan FM-radion. Runsaan internetin selaamisen jälkeen löysin Arduinoon kytkettävän kehitysalustan, joka mahdollistaisi sen muuttamisen FM-radioksi. Päädyin lopulta ostamaan tämän kehitysalustan, jonka jälkeen aloin toteuttamaan ensimmäistä alusta loppuun suunnittelemaani elektroniikkalaitetta.

Tässä työssä suunnitellaan ja toteutetaan Arduino-pohjainen FM-radio prototyyppitasolta valmiiksi laitteeksi asti. Luvussa 2 esitellään projektin osat yksitellen ja niiden merkitystä laitteen toiminnan kannalta avataan tarkemmin. Lisäksi tässä luvussa määritellään muutamia laskukaavoja, joita käytetään työn muissa vaiheissa.

Luvussa 3 siirrytään käsittelemään suunnittelun vaiheita laitteen ominaisuuksien määrittelystä aina viimeistellyn kotelon luontiin. Lisäksi kolmannessa luvussa testataan suunniteltuja lohkoja simulaatioiden avulla. Simulaatioiden jälkeen siirrytään kirjoittamaan laitteen ohjelmistoa, joka lopulta testataan koekytkentälevyjen avulla. Onnistuneiden testausten jälkeen suunnitellaan laitteelle piirilevy sekä ohjelmoidaan työhön tuleva mikrokontrolleri käyttäen Arduinoa ohjelmointityökaluna. Lopulta työ päätetään rakentamalla 3D-tulostettu kotelo laitteen elektroniikkaosien suojaksi.

FM-radio on keksintönä hyvin vanha ja monelle varsin arkinen laite. Radioita on markkinoilla nykyään suuri määrä ja niiden ominaisuudet vaihtelevat erittäin paljon. Tämän työn tavoitteena on vastata kysymykseen siitä, voiko itse suunniteltu radio vastata kaupasta ostettavaa laitetta. Samalla pyritään Arduino-alustaa käyttämään mahdollisimman monessa suunnittelun vaiheessa.

2. PROJEKTIN OSIEN ESITTELY

Ennen projektin aloittamista on tärkeää käydä läpi sen funktionaaliset lohkot. Elektroniikkalaitteen suunnittelussa kannattaa laitteen osiin suhtautua itsenäisinä lohkoinaan, jolloin suunnittelu on helpompi jakaa osiin. Tässä kappaleessa esitetään viimeistellyn laitteen rakentamiseen tarvitut osat, sekä perehdytään teoriatasolla niiden merkitykseen laitteen toiminnan kannalta.

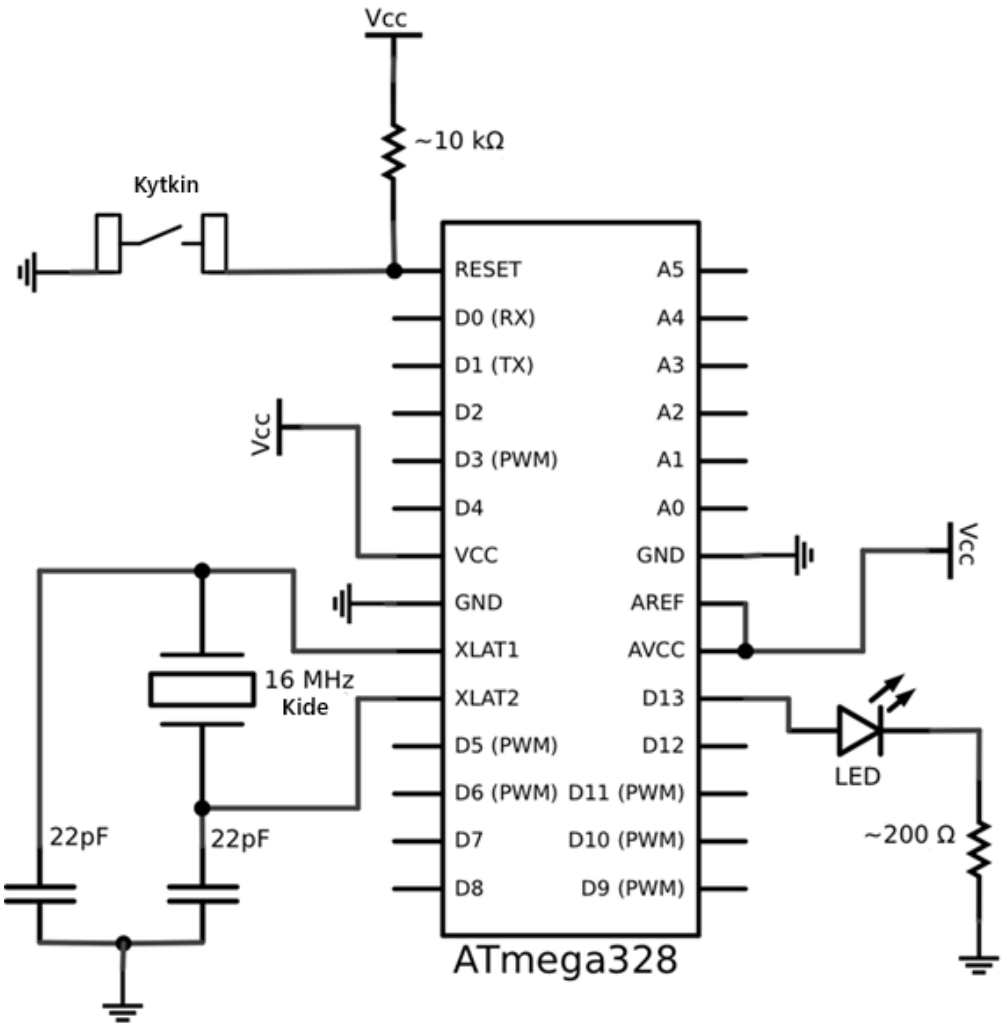
2.1 Arduino

Arduino on avoimen lähdekoodin elektroniikka-alusta, mikä mahdollistaa yksinkertaisten ja jossain tapauksessa monimutkaistenkin elektroniikkalaitteiden suunnittelun ja toteutuksen. Arduino-alusta on saavuttanut suurta suosiota varsinkin opetuskäytössä sen yksinkertaisuuden ja aloittelijaystävällisen käyttöliittymän vuoksi. [1] Samalla Arduino kuitenkin tarjoaa paljon mahdollisuuksia kokeneemmallekin elektroniikkaharrastajalle, joka tarvitsee alustaa esimerkiksi nopean prototyypin rakentamiseen. Arduinon suurta suosiota on lisännyt myös se, että siihen on saatavilla suuri määrä moduuleita ja lisäosia, joiden avulla Arduinosta itsestään on mahdollista toteuttaa esimerkiksi lämpömittari. Näitä moduuleita ja lisäosia on mahdollista käyttää monenlaisten laitteiden prototyyppien rakentamiseen, minkä ansiosta monet rakentavakin Arduinon ja koekytkentälevyn avulla elektroniikkalaitteen prototyyppiversion ja siihen kuuluvan mikrokontrolleriohjelmiston ennen varsinaisen laitteen valmistusta. Jossain tapauksessa Arduino saatetaan jopa jättää osaksi viimeisteltyä laitetta.

Arduinosta on olemassa monia erilaisia malleja ja valmistajia, mutta yleisesti Arduinolla kuitenkin viitataan Atmelin megaAVR-pohjaisen mikrokontrollerin omaaviin järjestelmiin [2, s.77]. Pohjimmiltaan Arduino on niin kutsuttu sovituslevy (breakout board) AVR-mikrokontrollerille. Tämän sovituslevyn päätehtävä on luoda kiinteä pakkaus, joka helpottaa mikrokontrollerin testausta sekä pääsyä sen eri pinneille tilanteessa, jossa komponentti on ns. pintaliitoskomponentti [3]. Sovituslevyt ovat erittäin tärkeitä varsinkin koekytkentälevyllä tehtävässä testauksessa, koska pintaliitoskomponentteja ei suoraan ole mahdollista kytkeä koekytkentälevyn reikiin.

Useassa tilanteessa kuten tässäkin työssä, Arduinoa käytetään laitteen ensimmäisen prototyypin rakentamiseen koekytkentälevylle ennen kuin tämä suunnitelma siirretään

PCB-levylle (Printed Circuit Board). Tämä onnistuu kopioimalla kuvassa 1 esitetty Arduinon arkkitehtuuri PCB-levylle käyttäen mikrokontrolleria ja siihen kytkettyjä oheislaitteita, joista merkittävimpänä on kvartsikide, joka toimii ulkoisena kellosignaalilähteenä. Arduinolla suunniteltu järjestelmä voidaan tällöin suoraan siirtää PCB-levylle.



Kuva 1. *Arduino Unon arkkitehtuuri DIP28-koteloinnille [mukaillen lähteestä 4].*

Keskeinen osa sulautettua järjestelmää on sen ohjelmiston suunnittelu. Arduinon tapauksessa ohjelmointi tapahtuu yleisesti Arduino IDE:n avulla. Arduino IDE on avoimen lähdekoodin maksuton ohjelmointi- ja kehitysympäristö, joka on kehitetty juuri Arduinojen ohjelmoimiseen [5]. Harjaantuneemmat käyttäjät siirtyvätkin useasti käyttämään kehittyneempiä ohjelmistoympäristöjä, kuten esimerkiksi Microchip Studiota, mitkä mahdollistavat mikrokontrollerien rekisterien tarkemman tarkastelun ja monia muita kehittyneempiä ohjelmistotyökaluja. Arduino IDE:llä on kuitenkin omanlaisiaan etuja, joita esimerkiksi

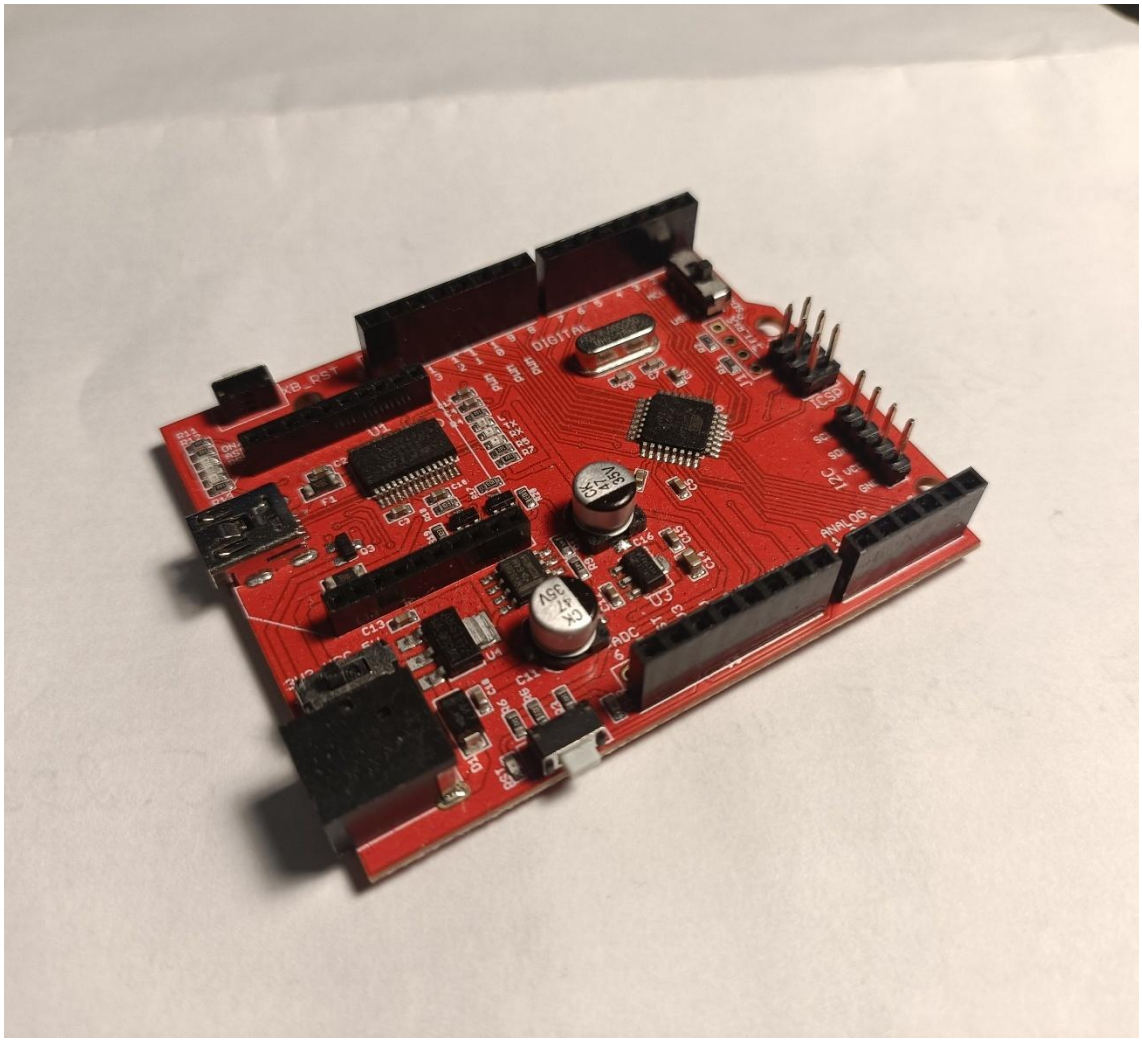
Microchip Studiossa ei ole. Arduino IDE:n edut keskittyvät nimenomaan sen yksinkertaisuuteen ja aloittelijaystävällisyyteen.

Käytännön ohjelmointi Arduinolla tapahtuu perinteisellä C++-ohjelmointikielellä, kuten monissa muissakin ohjelmointiympäristöissä, mutta se kuitenkin sisältää funktioita ja rakenteita, jotka yksinkertaistavat ja lyhentävät kirjoitettavaa syntaksia. Hyvänä esimerkkinä tällaisesta yksinkertaistuksesta on tämänkin työn kannalta tärkeä Analog to Digital Conversion eli ADC-muunnos. ADC-muunnosta käytetään tässä työssä äänenvoimakkuuden lukemiseen potentiometrin jännitearvosta. Atmega328p:n datalehden mukaisesti ADC-muunnos vaatii useita toimenpiteitä, kuten oikean ADC-kanavan valinta, ADC-muunnoksen käyttöönotto, mahdollinen keskeytyskäsittelijä, esijakaja sekä itse muunnoksen käynnistys [6, s. 205-220]. Arduino IDE:ssä tämä vaihe on kuitenkin luotu yhdeksi yksinkertaiseksi funktioksi nimeltään ANALOG_READ(), jota kutsumalla Arduino lukee parametrina annetun analogipinnin arvon ja muuntaa sen digitaaliseksi arvoksi. Halutessaan käyttäjä voi kuitenkin kirjoittaa ohjelman käyttäen puhdasta C++-kieltä edellä mainittujen vaiheiden mukaisesti, mutta omien funktioidensa avulla Arduino IDE helpottaa aloittelijoiden tutustumista ohjelmointiin.

Toinen aloittelijoille suunnattu ominaisuus, jonka Arduino IDE sisältää, on sen sisäänrakennetut ohjelmointikirjastot. Tällaisia kirjastoja ovat esimerkiksi tässäkin työssä käytettävä LiquidCrystal-kirjasto. Tämä kirjasto on suunnattu LCD-näyttöjä (Liquid Crystal Display) varten ja se luo helpon ja yksinkertaisen tavan esittää erilaisia tietoja ohjelman toiminnasta. Useassa tapauksessa nämä kirjastotkin on koodattu mahdollisimman käyttäjäystävällisiksi, siten että esimerkiksi juuri LiquidCrystal-kirjaston osalta käyttäjän tarvitsee vain alustaa ne Arduinon pinnit, joihin LCD-näyttö on kiinnitetty. Tämän jälkeen käyttäjän tarvitsee vain kutsua jotain kirjaston funktiota sen enempää miettimättä, mitä LCD-näytön toimiessa oikeastaan edes tapahtuu. Lisäksi Arduinon suuren suosion ansiosta internetistä löytyy paljon muiden käyttäjien ja yritysten luomia kirjastoja juuri tietyille Arduino-malleille, mitkä tekevät ulkoisten moduulien ja lisälaitteiden lisäämisen vielä entistäkin helpommaksi. Tässä työssä käytettävä Si4703-kehitysalusta hyödyntää juuri tämän tapaista kirjastoa.

Arduino on avoimen lähdekoodin sovellus, joten siitä on saatavilla kaikki tarvittavat tiedot aina lähdekoodista piirikaavioihin asti. Tämän avoimuuden ansiosta kuka tahansa voi siis rakentaa oman Arduinonsa, mikä on suurin syy sille, että Arduino-klooneja on saatavilla runsaasti [7, luku 1]. Tässä työssä on ohjelmiston ja elektroniikkapuolen suunnitteluun käytetty Crowduino UNO v. 1.1:stä, joka on esitetty alla olevassa kuvassa 2. Kyseessä on siis Arduino UNOa vastaava klooni.

Joissakin klooneissa ominaisuudet eroavat alkuperäisestä Arduino-mallista hyvinkin paljon. Tässäkin tapauksessa Crowduino UNO sisältää paljon ominaisuuksia, joita perinteisestä Arduino UNOsta ei löydy. Tärkein tällainen ominaisuus tämän työn kannalta on säädettävä logiikkataso. Tämä siitä syystä, että Si4703-kehitysalusta, joka mahdollistaa radion toiminnan, käyttää 3,3 V:n jännite- ja logiikkatasoja [8]. Perinteisessä Arduino UNOssa käyttöjännite ja logiikkataso on asetettu olemaan aina 5 V, mikä kuitenkin pahimmassa tapauksessa tuhoaisi Si4703-kehitysalustan.



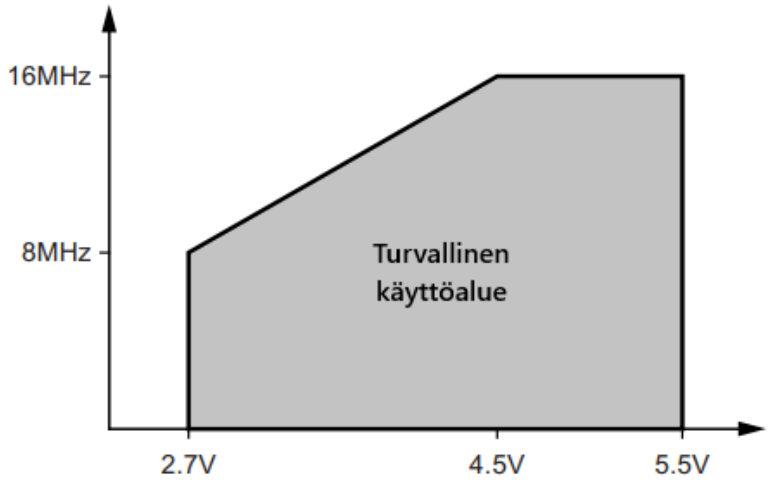
Kuva 2. *Crowduino v1.1.*

Tilanteissa, jossa Arduinolla kehitetty projekti halutaan siirtää käytännön toteutukseen piirilevylle, tulee ottaa huomioon muutamia asioita. Näistä keskeisimpiä on käyttöjännite ja käytetty kellotaajuus. Datalehtensä mukaisesti Atmega328p-mikrokontrolleri voi operoida 2,7–5,5 V käyttöjännitteellä ja kellotaajuus voi olla välillä 0-16 MHz [6, s. 2]. Atmega328p:n tapauksessa kellolähteeksi voidaan valita joko ulkoinen tai sisäinen kello-

lähde. Perinteisessä Arduino UNOssa käytetään ulkoisena kellolähteenä 16 MHz kvartsikidettä [4], mutta Atmega328p tarjoaa myös mahdollisuuden sisäisen RC-oskillaattorin käyttöön kellolähteenä. RC-oskillaattoreita on Atmega328p:n tapauksessa kaksi erilaista, 8 MHz ja 128 kHz [6, s. 25], mitkä on mahdollista jakaa vielä pienempiin arvoihin esijakajan avulla.

Mikäli siirtymisen Arduinolta piirilevylle haluaa tehdä mahdollisimman suoraviivaisesti, kannattaa Arduinon arkkitehtuuri kopioida suoraan suunniteltuun laitteeseen. Toisin sanoen, jos projektin kehityksessä on käytetty Arduino UNOa, joka operoi 5 V:n käyttöjännitteellä ja 16 MHz kellotaajuudella, kannattaa tämä arkkitehtuuri kopioida lopulliseen kokoonpanoon, jotta siirtyminen on mahdollisimman suoraviivainen. Tämän projektin osalta tämä siirtymä ei kuitenkaan ollut näin suoraviivainen johtuen siitä, että järjestelmän käyttöjännite tulee olla 3,3 V. Si4703-kehitysalustan asennusohjeessa neuvotaan käyttämään Arduino UNOn sijasta Arduino Pro Miniä järjestelmän mikrokontrollerina, jolloin tarvetta logiikkatasomuuntimelle ei ole [8]. Arduino UNO ja Pro Mini molemmat käyttävät Atmega328p:tä mikrokontrollerinaan, mutta suurin ero niiden välillä syntyy siitä, että Pro Mini on saatavilla myös 3,3 V:n ja 8 MHz versiona [9]. Tässä työssä Arduinon arkkitehtuuri on siis itse asiassa kopioitu Arduino Pro Ministä, vaikka itse laitteen testaus koekytkentälevyllä on tehty käyttäen Arduino UNOn kloonia.

Lopullisessa kokoonpanossa Atmega328p operoi siis käyttäen 3,3 V:n käyttöjännitettä. Tämä jännitearvo asettuu hyvin datalehden mukaiseen 2,7-5,5 V:n käyttöjännitteeseen. [6, s. 2] Kellolähteen määrittämiseen taas on useampia mahdollisuuksia. Kellolähde määrittää siis jaksonajan, jolla mikrokontrolleri operoi. Datalehdellä esitetyn kuvaajan (kuva 3) mukaisesti voidaan määrittää käyttöjännite ja kellotaajuus, jolla Atmega328p operoi luotettavasti [6, s. 260]. Kuvan 3 mukaisesti 3,3 V:n käyttöjännitteellä 8 MHz kellotaajuus mahdollistaa luotettavan toiminnan. Vastaavasti Arduino UNOn käyttämä kellotaajuus eli 16 MHz olisi taas liian suuri käytettyyn käyttöjännitteeseen verrattuna, jolloin mikrokontrolleri ei toimisi enää luotettavasti turvallisen käyttöalueen sisällä.



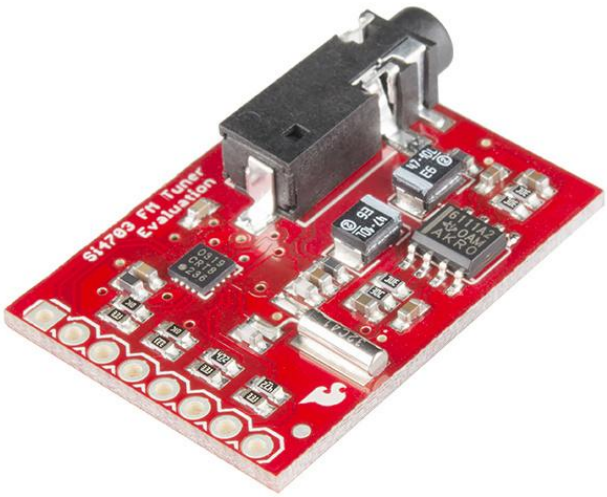
Kuva 3. *Kellotaajuuden suuruus käyttöjännitteen funktiona [mukaillen lähteestä 6].*

Edellisten päätelmien mukaisesti 8 MHz näyttäisi olevan hyvä valinta kellolähteen luomiseen 3,3 V:n käyttöjännitteellä. Vielä tulee kuitenkin valita, tuleeko lopullisessa projektissa käyttöön ulkoinen 8 MHz kellolähde vai Atmega328p:n oma sisäinen 8 MHz oskillaattori. Tämän työn osalta on päädytty käyttämään ulkoista 8 MHz kvartsikidettä, koska yleisesti niitä pidetään luotettavampina ja niiden tarkkuus on parempi kuin sisäisen oskillaattorin. Tähän syynä on se, että ulkoisen kellolähteen, eli tässä tilanteessa kvartsikiteen lupaama virhetoleranssi on ±30 ppm (Parts per million) [10]. Toisin sanoen virhettä voi 8 MHz taajuudella syntyä maksimissaan ±240 Hz. Vastaavasti Atmega328p:n datalehdellä on ilmoitettu RC-oskillaattorin tarkkuudeksi huoneenlämmössä 3 V:n käyttöjännitteellä ±2 % [6, s. 260]. Jos tämä arvo muunnettaisiin edelliseen tapaan taajuusarvoksi, vastaisi se virhetoleranssia ±160 kHz. Näiden päätelmien avulla voidaan todeta, että kvartsikiteen lisääminen kellolähteeksi on erittäin järkevää, sillä sen avulla voidaan virhetoleranssia laskea moninkertaisesti. Mikrokontrollerin kellosignaalin tarkkuus tässä tilanteessa on äärimmäisen tärkeä osa-alue, koska mikrokontrolleri kommunikoi Si4703-kehitysalustan kanssa käyttäen I2C-datalinjaa (Inter-Intergrated Circuit). Ilman kunnollista kellosignaalia tiedonsiirto ei onnistu tai pahimmassa tapauksessa se on virheellistä.

2.2 Si4703 FM-tuner

Si4703 on FM-vastaanotin IC (Integrated Circuit) erilaisiin kannettaviin sovelluksiin, kuten MP3-soittimiin ja kannettaviin radioihin [11, s.1]. Tässä työssä ei kuitenkaan suoranaisesti käytetä Si4703 IC:tä, vaan SparkFun Electronicsin valmistamaa kuvan 4 mu-

kaista Si4703-kehitysalustaa, joka on Arduinon lailla sovituslevy Si4703 IC:lle. Tästä kehitysalustasta on olemassa kahta erilaista mallia, joista toinen sisältää oman audiovahvistimensa ja toinen ei. Tässä työssä käytetty malli on ensimmäisenä mainittu eli se siis sisältää jo itsessään oman audiovahvistimensa. Tämä malli on valikoitu tähän työhön myös sen vuoksi, että se sisältää valmiiksi 3.5 mm audioliittimen, joka helpottaa laitteen suunnittelua ja testausta koekytkentälevyllä. Tämä audioliitin mahdollistaa sen, että kehitysalustaan on mahdollista kytkeä esimerkiksi nappikuulokkeet, joiden kautta voi suoraan kuunnella radioasemia. Käytännössä kehitysalusta hoitaa vastaanotetun FM-signaalin demoduloinnin ja vahvistamisen, mutta se ei itsessään sisällä minkäänlaisia ohjaimia. Kehitysalustan ohjaaminen toteutetaan käyttäen I2C-sarjaliikennettä, mikä tässä tilanteessa tehdään mikrokontrollerin avulla.



Kuva 4. *Si4703-kehitysalusta [8].*

I2C on alun perin Philipsin kehittämä sarjaliikenneprotokolla, joka on tarkoitettu mikroprosessorien ja niihin kytkettyjen oheislaitteiden väliseen kommunikointiin. I2C-väylä koostuu kahdesta kaksisuuntaisesta tietolinjasta eli sarjadatalinjasta (Serial Data) ja kellosignaalista (Serial Clock), jotka on yhdistetty isäntälaitteen (master) ja orjalaitteen (slave) välille. [12, s. 169] Useimmissa tilanteissa isäntälaitteena toimii juuri mikrokontrolleri ja orjalaitteena oheislaite, jota halutaan ohjata tai josta halutaan vastaanottaa tietoa. Yhteen isäntälaitteeseen on mahdollista lisätä useampi kuin yksi orjalaite kerrallaan, koska jokaisella I2C-orjalaitteella on oma uniikki laiteosoitteensa. Tässä työssä Atmega328p toimii I2C-väylän isäntälaitteena ja Si4703-kehitysalusta orjalaitteena.

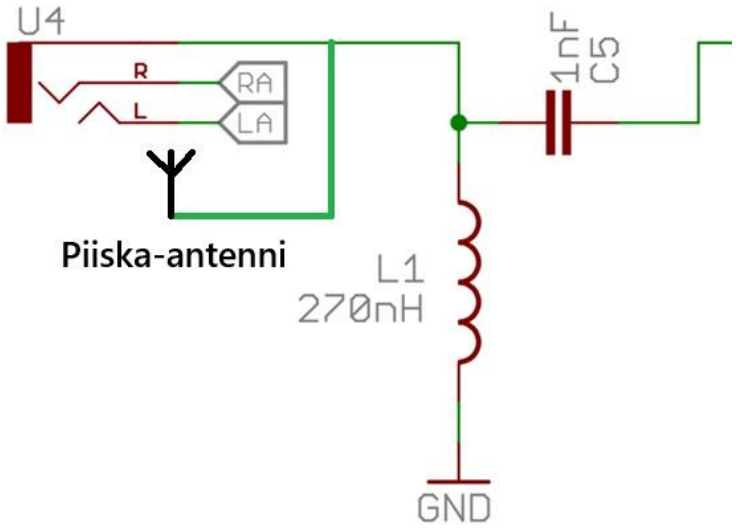
I2C-tiedonsiirron alustuksessa isäntälaite pyrkii ottamaan yhteyden orjalaitteeseen lähettämällä datalinjaan orjalaitteen laiteosoitetta ja sen perässä R/W-bittiä (read/write), eli tietoa siitä haluaako isäntä lukea vai kirjoittaa dataa orjalaitteeseen. Mikäli laiteosoitetta vastaava orjalaite löytyy, vastaa orjalaite isäntälaitteelle ACK-bitin avulla (Acknowledge), jonka jälkeen itse tiedonsiirto voi alkaa. [12, s.169] Datasiirron alkaessa data isännän ja orjan välillä alkaa siirtymään 8 bitin kehyksessä riippuen yhteyden alussa määritetystä R/W-bitin arvosta, jonka jälkeen kehys lopetetaan orjan lähettämällä ACK-bitillä [2, s. 95]. Mikäli R/W-bitin arvo oli yhteyden alustuksen aikana yksi, tarkoittaa tämä sitä, että isäntä haluaa lukea dataa orjalaitteesta. Vastaavasti, jos R/W-bitti oli yhteyden alussa nolla, haluaa isäntälaite kirjoittaa dataa orjalaitteeseen. Keskeistä I2C-kommunikoinissa on se, että se on toteutettu noudattamaan ns. active low-periaatetta, jolloin signaalin arvo on aina korkealla, ellei sitä erikseen vedetä alas ulkoisen vastuksen avulla arvoksi nolla. Tämä johtuu siitä, että laitteiden konfiguroinnissa on käytetty avokollektoritransistoreja, jotka johtavassa tilassa vetävät datalinjan nollaan imaistessaan virtaa ulkoisen vastuksen läpi [2, s.94-95].

Si4703-kehitysalusta sisältää käyttöjännite- ja I2C-pinnien lisäksi myös muita pinnejä useita eri käyttötarkoituksia varten. Tällaisia pinnejä ovat SEN, RST, GPIO1 ja GPIO2. SEN-pinni on otettu kehitysalustaan omaksi pinnikseen, sillä se mahdollistaa kommunikoinnin laitteen kanssa I2C:n lisäksi käyttäen kolmijohtoyhteyttä (3-Wire) [8]. Kolmijohtoyhteys tarkoittaa siis kokonaan eri kommunikointimenetelmää kuin I2C, koska siinä kommunikointiin käytetään kahden johtimen sijasta kolmea. RST-pinni on nimensä mukaisesti varattu kehitysalustan resetointia varten [8]. Tämän työn osalta RST-pinni on kytketty Atmega328p:n pinniin PD2. Tämä siitä syystä, että RST-pinnillä on merkitys kehitysalustan käynnistyksessä, koska sen avulla kehitysalusta asetetaan toimimaan käyttäen I2C-tiedonsiirtoa kolmijohtoyhteyden sijasta [11, s.19]. Erona tässä SEN-pinniin on se, että RST-pinnin tilaa tulee vaihtaa alustuksen aikana korkean (1) ja matalan tilan (0) välillä. SEN-pinnin kohdalla riittää johtoyhteyden valintaan joko korkea tai matala tila. Tässä tilanteessa kehitysalustaan on lisätty valmiiksi 10 kΩ ylösvetovastus, mikä mahdollistaa sen [8], että SEN-pinnin signaali on vakiona aina korkea. Tästä seuraa se, että SEN-pinniin ei tarvitse kytkeä johdinta, mikäli käyttöön halutaan ottaa I2C-kommunikointi. Kehitysalustan kaksi viimeistä pinniä GPIO1 ja GPIO2 ovat yleiskäyttöisiä tulo- ja lähtöpinnejä (General Input/Output pins), joita voidaan käyttää useaan eri tarkoitukseen [8]. Yleisesti näitä kahta pinniä voidaan käyttää indikaattorina siitä, että onko RDS-data (Radio Data System) valmista luettavaksi tai onko kuunneltu kanava stereo- vai monokanava [11, s.16].

Eräs syy siihen, miksi juuri Si4703-kehitysalusta on valittu tähän työhön, on se, että siitä on saatavilla valmiina oleva Arduino-kirjasto, mikä helpottaa vastaanottimen ohjaamista huomattavasti. Toisin sanoen tämä kirjasto sisältää valmiit komennot, joilla kehitysalustaa ohjataan. Tämän kirjaston takana on sama tuotantoryhmä, joka on suunnitellut Si4703-kehitysalustan eli SparkFun Electronics [13]. Arduino-kirjasto helpottaa laitteen ohjelmointia suuresti, koska ohjelmiston puolesta suurin työ on tehty jo valmiiksi. Jos kuitenkin kirjaston sisältöä haluaa muokata tai luoda kokonaan oman Arduino-kirjastonsa Si4703:n hallintaan, on sekin täysin mahdollista. Si4703-datalehdellä on esitetty kaikki toimenpiteet ja niitä koskevat rekisterit, joita muokkaamalla voi muokata IC:n toimintaa. Tämän työn osalta Si4703-kirjasto on jätetty melko koskemattomaksi yhtä rekisteriä lukuun ottamatta. Tästä muutoksesta lisää luvussa 3.3.

Koska Si4703 IC on suunniteltu kannettaviin sovelluskohteisiin, on sen eräs erikoisuus se, että se ei tarvitse ulkoista antennia vaan antennin virkaa hoitaa ulkoinen kuulokeliitäntä. Tässä ongelmaksi muodostuu kuitenkin se, että Si4703 IC ei sisällä suoraa valmiutta ulkoisen antennin yhdistämiseen. [14, s. 1] Koska Si4703 käyttää pääsääntöisesti kuulokejohtoa ulkoisena antennina, on kehitysalustaan kytketty sitä varten kaksi lisäkomponenttia, jotka mahdollistavat FM-taajuusalueen parhaan mahdollisen kuulumisen. Näitä ovat kuvan 5 mukaisesti 270 nH käämi sekä AC-kytkentä, joka on toteutettu käyttäen 1 nF kondensaattoria [15]. AC-kytkennän tarkoitus on suodattaa vastaanotetusta radiosignaalista DC-komponentti (DC-offset) [14, s. 19]. Vastaanotetun radiosignaalin tulisi olla mahdollisimman puhdas AC-signaali, koska DC-komponentit aiheuttavat laitteen toimintaa haittaavia vääristymiä. Kondensaattorin lisäksi kytkennässä on 270 nH käämi, joka pyrkii maksimoimaan jännitevahvistuksen koko FM-taajuusalueella [14, s.19].

Kuvan 5 mukaisen käämin L1 ja kondensaattorin C5 arvo on määritetty käytettäväksi silloin, kun antennina käytetään kuulokejohtoa. Tässä työssä tilanne on kuitenkin erilainen, koska antennin tehtävää hoitaa ulkoinen piiska-antenni. Tällöin piiska-antennin kohdalla olisi käämin L1 ja kondensaattorin C5 arvot pitänyt määrittää uudelleen, jotta vastaanotettu signaali olisi ollut laadultaan paras mahdollinen. Tämän työn osalta arvojen uudelleenmääritys on jätetty tekemättä, koska käytännön testauksen perusteella päästiin kuuluvuuden osalta hyviin tuloksiin kytkemällä piiska-antenni suoraan kuulokeliitäntään alla olevan kuvan 5 mukaisesti. Piiska-antenni on kytketty siis suoraan 3.5 mm audioliittimen maajohtimeen.



Kuva 5. *Antenni ja sen lisäkomponentit [mukaillen lähteestä 15*].

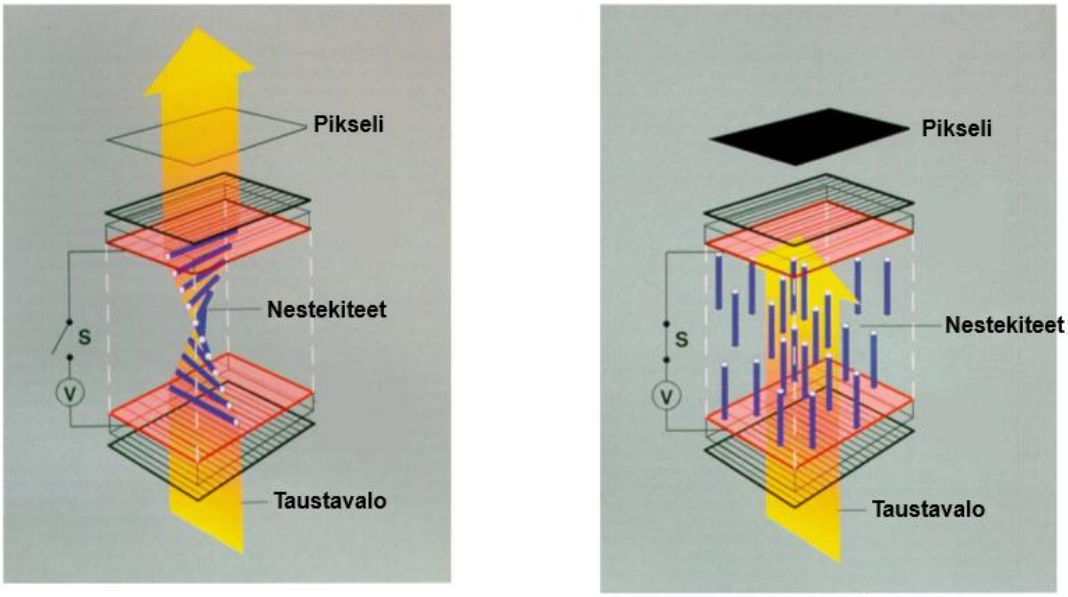
Si4703 IC:n mielenkiintoinen ominaisuus on se, että siihen on luotu valmius RDS-datan prosessointiin. RDS on lyhenne sanoista Radio Data System, joka nimensä mukaisesti tarjoaa mahdollisuuden radioaseman datan prosessointiin. Radioasemien lähettämä data sisältää esimerkiksi tiedot siitä, mikä on toistettavan aseman tai soitettavan kappaleen nimi. [11, s.15] Tämän työn osalta RDS-datan prosessointi on jätetty pois käsittelystä aihealueen rajaamiseksi. Laiteeseen on kuitenkin sisällytetty valmius RDS-datan prosessointiin ja näin se olisi mahdollista lisätä osaksi laitetta tulevaisuudessa tekemällä muutoksia ohjelmistoon.

2.3 LCD-näyttö

Radion toiminnan kannalta on äärimmäisen tärkeää, että käyttäjä saa jonkinlaista informaatiota siitä, mitä kehitysalusta ja mikrokontrolleri ovat suorittamassa. Tällaista tietoa on esimerkiksi soitossa oleva FM-taajuus sekä kanavan äänenvoimakkuus. Näiden tietojen näyttämiseen tarvitaan siis jonkinlainen näyttö. Tässä työssä tähän tehtävään on valittu LCD-näyttö.

LCD:n toiminta perustuu nimensä mukaisesti nestekiteisiin (liquid crystal), jotka on suljettu kahden polarisoivan levyn väliin. Keskeistä kahdessa polarisoivassa levyssä on se, että ne on sijoitettu 90° kulmaan toisiinsa nähden, jolloin valo voi polarisoitua vain näiden kahden levyn polarisaatiosuunnan mukaisesti. [16, s. 43] Polarisoimattoman valon komponenteista pääsevät siis läpi vain ne, jotka ovat yhdensuuntaisia polarisoivien levyjen

kanssa. Valon polarisointi mahdollistaa sen, että kun nestekidetason yli kytketään jännite, kääntyvät nestekiteet sähkökentän suuntaisesti, jolloin polarisoitu valo ei pääse läpi toisesta polarisoivasta levystä [16, s. 44]. Tällöin kyseinen pikseli esiintyy pimeänä. Vastaavasti, jos jännite kytketään pois päältä, pystyvät nestekiteet liikkumaan vapaasti, jolloin taustavalolta saapuva valo pääsee etenemään molempien polarisoivien levyjen läpi. Tällöin kyseinen pikseli kytkeytyy päälle. Näin ollen jokaisen pikselin päällä oloa voidaan kontrolloida yksinkertaisesti kytkemällä jännite nestekidetasoon. Ohjaamalla yksittäisiä pikseleitä ja niiden joukkoja, voidaan muodostaa merkkejä sekä numeroita. Edellä kuvattu LCD-pikselin toiminta on havainnollistettu alla olevassa kuvassa 6.

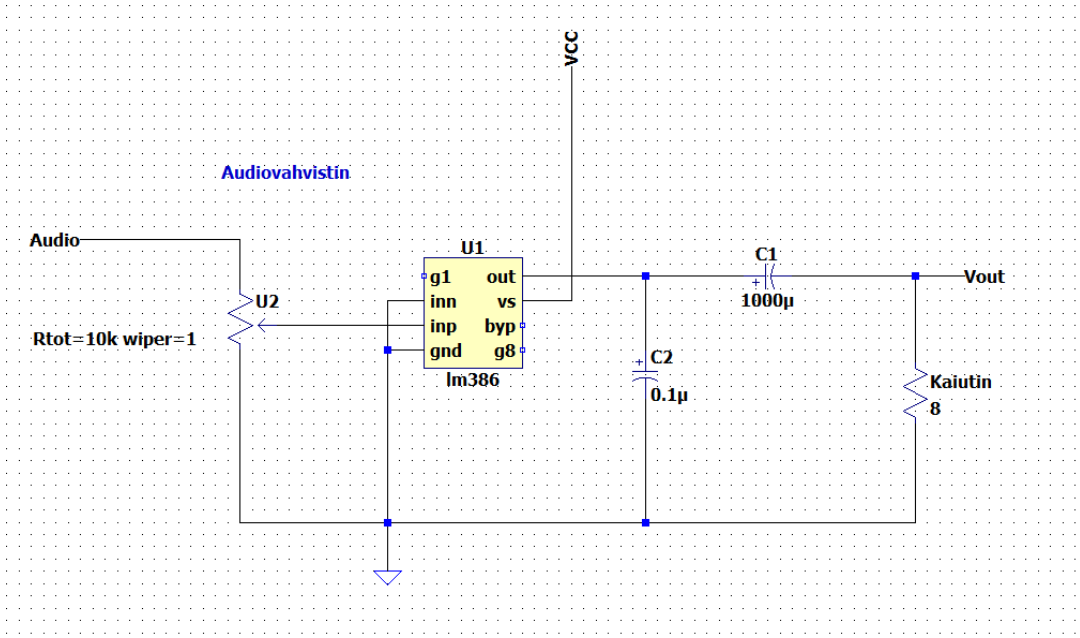


Kuva 6. *Yksittäisen LCD-pikselin toiminta. Vasemmalla esitetty tilanne, jossa pikseli on päällä ja oikealla tilanne, jossa pikseli on kiinni [mukaillen lähteestä 17].*

Pikseleiden ohjaukseen on monia tapoja, mutta tässä työssä ohjaus on toteutettu käyttäen Hitachi HD44780-ohjainpiiriä [18]. Ohjainpiirin tehtävä on siis hoitaa edellä mainittu yksittäisten pikseleiden päälle kytkentä ja tällä tavoin mahdollistaa merkkien osoittaminen käyttäjälle. Ohjainpiiri on siis osa LCD-moduulia, jolloin sen ohjaaminen on mahdollista kytkemällä se suoraan mikrokontrolleriin. LCD-kytkentää laitteeseen yksinkertaistaa myös se, että samaan tapaan kuin Si4703-kehitysalustan kanssa, löytyy Hitachi HD44780-ohjainpiiriä käyttäviin LCD-näyttöihin valmis Arduino-kirjasto nimeltään LiquidCrystal [18]. Si4703-kehitysalustan tavoin tämä kirjasto siis sisältää valmiit funktiot, joiden avulla tekstin saaminen LCD-näytölle ja sitä kautta käyttäjälle onnistuu äärimmäisen kätevästi.

2.4 Audiovahvistinkytkentä

Si4703-kehitysalusta mahdollistaa FM-taajuuksien demoduloinnin audiosignaaliksi. Tämä audiosignaali on kuitenkin varsin heikko, jolloin se tulee vahvistaa jotenkin, jotta se on mahdollista toistaa kaiuttimesta. Tähän tehtävään avuksi tulee audiovahvistinkytkentä. Tässä työssä yksinkertainen audiovahvistin on toteutettu käyttäen LM386 IC:tä. LM386 on matalan käyttöjännitteen audiovahvistin [19, s.1], joka siis tämän työn kohdalla vahvistaa Si4703-kehitysalustalta tulevan audiosignaalin kaiutinta varten. Kuvassa 7 on esitettynä tässä työssä käytetty kytkentä.



Kuva 7. *LM386-audiovahvistinkytkentä LT Spice-ohjelmassa esitettynä.*

Kuvassa 7 olevassa kytkennässä on otettu vaikutteita LM386-datalehden mallikytkennästä. Tällä kyseisellä kytkennällä vahvistus on 26 dB, joka tässä sovelluskohteessa riittää mainiosti. [19 s. 10] Kytkentää on kuitenkin muokattu hieman mahdollistamaan paras mahdollinen toiminta. Keskeisin tällainen muutos on nähtävissä kuvan 7 kondensaattorissa C1. Datalehden mallissa tämän kondensaattorin arvoksi on määritetty 250 µF. Tämä arvo on kuitenkin aivan liian matala musiikin kuunteluun, koska tällaisessa tilanteessa vahvistimen ulostulo toimii eräänlaisena ylipäästösuotimena. Toisin sanoen toistettavat musiikin alimmat taajuudet määräytyvät tämän kondensaattorin mukaan ja mikäli tämän kondensaattorin arvo on liian matala, suodattuvat matalat bassotaajuudet pois musiikista.

Audiovahvistimen -3 dB-rajataajuus voidaan määrittää seuraavanlaisilla laskukaavoilla

$$f_{-3dB} = \frac{1}{2\pi RC} \tag{1}$$

$$C = \frac{1}{2\pi R f_{-3dB}} \tag{2}$$

, jossa *R* on vahvistimen kuorma eli tässä tilanteessa kaiuttimen vastus ohmeina, f_{-3dB} on audiovahvistimen -3 dB-rajataajuus ja *C* on ulostulossa olevan kondensaattorin C1 kapasitanssi [20, luku 5.8]. Koska kaiuttimen halutaan toistavan audiosignaalin taajuuksia mahdollisimman laajalta alueelta, tulee kondensaattorille C1 valita mahdollisimman suuri kapasitanssin arvo tiettyjen viitearvojen sisältä. Tässä tilanteessa nämä viitearvot määrittää ihmisen kuuloalue.

Musiikin bassotaajuudet, joita audiovahvistimen tulee toistaa, sijoittuvat välille 20Hz-300Hz [21, luku 1.12]. Kondensaattoria C1 määrittäessä kannattaa sen arvo valita siten, että vahvistinpiiri itsessään ei rajoita näiden taajuuksien toistamista. Näin ollen voidaan kaavan 2 mukaisesti laskettuna todeta, että ulostulokondensaattorin arvo tulisi olla 20 Hz:n taajuudella noin 1000 µF. Tällä ulostulokondensaattorin C1 arvolla myös alimmat bassotaajuudet kuuluvat nyt audiovahvistimen ulostulon taajuusalueeseen. Huomioitavaa tässä on kuitenkin se, että datalehden mukaisella 250 µF arvolla kaiuttimen –3 dB-rajataajuus olisi sen sijaan n. 80Hz kohdalla. Tällä kondensaattorin arvolla audiovahvistin siis suodattaisi pois suuren osan musiikin alimmista bassotaajuuksista.

Muuta huomioitavaa tässä audiovahvistinkytkennässä on se, että lopullista äänenvoimakkuuden säätöä ei toteuteta kuvassa 7 näkyvällä säätövastuksella, vaan digitaalisesti luodun ohjelmiston avulla. Kuvan 7 säätövastus on kuitenkin sijoitettu lopulliseen kokoonpanoon hienosäätöelementiksi eli sen avulla voidaan varmistaa, että ulostuleva ääni ei säröydy, vaikka audiovahvistimeen saapuva ääni olisi suurin mahdollinen. Säätövastuksen avulla voidaan siis luoda ikään kuin rajoitin, joka varmistaa suurimman mahdollisen äänenvoimakkuuden valinnan.

2.5 Regulaattorikytkentä

Käyttöjännitteen määrittely on erittäin tärkeä osa elektroniikkalaitteen suunnittelua. Tässä tilanteessa käyttöjännitteen määrittely on melko yksinkertaista, koska Si4703-kehitysalusta käyttää käyttöjännitteenään ja logiikkatasonaan 3,3 V:n jännitettä. Datalehden mukaisesti Si4703-kehitysalustaa on mahdollista käyttää 5 V:n käyttöjännitteelle, mutta moduulin logiikkatasona on tässäkin tapauksessa 3,3 V:n jännite. Mikäli kehitysalusta kytkettäisiin suoraan 5 V:n jännitteeseen, voisi tämä jännite pahimmassa tapauksessa tuhota laitteen. Jos kehitysalustaa jostain syystä haluttaisiin käyttää 5 V:n jännitteellä tulisi logiikkatasojen väliin kytkeä niin sanottu logiikkatasomuunnin. [8] Tällaisen logiikkatasomuuntimen tarve voidaan kuitenkin poistaa suunnittelemalla laite käyttämään koko ajan 3,3 V:n jännitettä, koska tällaisessa tilanteessa laitteen muut osat eivät vaadi edellä mainittua 5 V:n jännitettä. Tässä työssä laite saa jännitteensä neljältä sarjaan kytketyltä AA-paristolta. Tällaisen paristokytkennän napojen välinen jännite on 6 V. Tämä jännite on kuitenkin liian suuri syötettäväksi suoraan järjestelmään, minkä vuoksi se tulee reguloida.

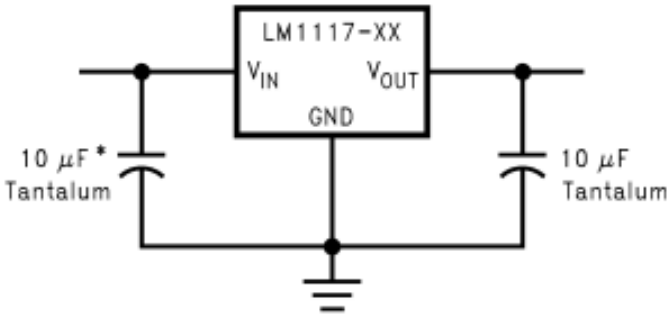
Jänniteregulaation keskeisin tavoite on pitää laitteen käyttöjännite vakiona kuormavirran, ympäristötekijöiden tai tulojännitteen muuttuessa [2, s.239-240]. Tämän työn osalta kuormavirta pysyy suhteellisen vakiona laitteen operoidessa, mutta laitteen tulojännite ja ympäristötekijät ovat muuttuvia parametrejä, jotka tulee huomioida suunnittelussa. Varsinkin laitteen tulojännite muuttuu hyvinkin paljon paristojen tyhjentyessä. Vastaavasti ympäristötekijät muuttuvat riippuen siitä, missä lämpötilassa laitetta käytetään.

Pääsääntöisesti regulointimenetelmät jaetaan kahteen eri luokkaan riippuen siitä, onko reguloitava kuorma rinnan- vai sarjaankytketty regulaation suorittavan säätöelementin kanssa [23, s.300]. Nykyaikana kuitenkin yleisempi menetelmä varsinkin regulaattori IC:den kohdalla on sarjaregulointi, jossa säätöelementti on toteutettu transistoripiirin avulla [2, s.240]. Tässä työssä käytetty regulaattorikytkentä on lineaarinen, joka tarkoittaa sitä, että säätöelementti toimii lineaarisella alueella [23, s.301]. Toisin sanoen regulaattorin toimiessa sen käyttöalueella, muuttaa säätöelementti toimintaansa lineaarisesti siten, että lähtöjännite pysyy vakiona kuormavirran tai tulojännitteen muuttuessa. Lineaaristen regulaattorien negatiivisena puolena on kuitenkin tulon ja lähdön välisestä potentiaalierosta syntyvät tehohäviöt ja siten alhaisempi hyötysuhde [2, s.244].

Lineaarisen regulaattorin sijasta olisi mahdollista käyttää niin kutsuttua hakkuriteholähdettä (switching regulator), jonka hyötysuhde on lineaarisia regulaattoreita parempi [23, s.323]. Hakkuriteholähteiden haittana taas on se, että ne ovat useasti rakenteeltaan mo-

nimutkaisempia [23, s.323]. Tämän edellä mainitun seikan vuoksi on tässä työssä päädytty käyttämään lineaarista teholähdettä niiden yksinkertaisuuden ja halvan hinnan vuoksi. Lisäksi potentiaaliero regulaattorin tulon ja lähdön välillä sekä kuormavirta (n. 50 mA) ovat tässä työssä niin pieniä, että merkittävää tehohäviötä ei synny, vaikka kyseessä onkin lineaarinen regulaattorikytkentä.

Tähän työhön valikoitunut regulaattori on lineaarinen Texas Instrumentsin valmistama LM1117 IC, jolla luotu kytkentä on esitetty alla olevassa kuvassa 8. Tätä kyseistä regulaattoria valmistetaan useammalle eri jännitearvolle, mutta koska tässä tilanteessa jännitteen halutaan pysyvän koko ajan 3,3 V:n arvossa, voidaan valita malli, missä jännite on “lukittu” tähän arvoon.



Kuva 8. *Regulaattorikytkentä [22, s.16].*

Regulaatiopiiriä suunnitellessa on kuitenkin tärkeää muistaa, että jännitteen alentamisesta syntyy aina tehohäviötä, joka muuttuu regulaattori IC:n kohdalla lämmöksi. Tällöin tulojännitteen valinnassa kannattaa pyrkiä siihen, että potentiaaliero tulo- ja lähtöjännitteen välillä on mahdollisimman pieni. Tässä työssä kuormavirta on hyvin pieni, minkä vuoksi LM1117 IC:n jäähdytyksestä ei tarvitse välittää. Kuitenkin jos potentiaaliero tulo- ja lähtöjännitteen välillä olisi suurempi, voisi pienikin kuormavirta aiheuttaa IC:n merkittävää lämpenemistä [2, s.242]. Joka tapauksessa kannattaa hukkatehon määrä aina minimoida. Tulon ja lähdön välinen potentiaaliero ei myöskään saa olla liian pieni, koska tällöin regulaattori ei kykene luotettavaan toimintaan.

Regulaattorien tapauksessa tästä minimipotentiaalierosta käytetään nimitystä pudotusjännite (*dropout voltage)*. LM1117:n datalehdellä on mainittu, että 100 mA kuormavirralla tulo- ja lähtöjännitteen ero tulee olla on ainakin 1,2 V [22, s.6]. Tästä syystä työhön on valittu jännitelähteeksi 6 V:n paristokytkentä, sillä 6 V:n jännite on tarpeeksi suuri mahdollistamaan regulaattorin varma toiminta, mutta samalla 6 V:n ja 3,3 V:n välinen potentiaaliero on niin pieni, että merkittävää tehohäviötä ei synny.

Tämän työn simulaatio-osiota varten on tärkeää määrittää laskukaava regulaattorikytkennän tehohäviölle. Ohjeet tehohäviön laskemiseen on esitetty yksityiskohtaisesti LM1117-datalehdellä. Tältä datalehdeltä löytyvät alla esitetyt laskukaavat 3 ja 4, jotka mahdollistavat tehohäviön laskemisen. [22, s.18]

$$I_{in} = I_L + I_G \tag{3}$$

$$P_D = (V_{in} - V_{out})I_L + V_{in}I_G \tag{4}$$

Kaavassa 3 I_{in} on tulovirta, joka samalla Kirchoffin virtalain mukaisesti on kuormavirran I_L ja maajohtimen virran I_G summa. Kaavassa 4 P_D on regulaattorin lämmöksi haihtuva tehohäviö, V_{in} on tulojännite ja V_{out} on lähtöjännite. Lisää regulaattorikytkennän simuloidusta tehohäviöistä on kerrottu luvussa 3.2.

3. SUUNNITTELUN VAIHEET

Elektroniikkalaitteen suunnittelu kannattaa aloittaa sen ominaisuuksien määrittelystä. Suunnittelun alkuvaiheessa kannattaa siis tarkkaan miettiä, mitä laitteen tulee pystyä suorittamaan. Tämän työn osalta tämä vaihe on melko suoraviivainen, koska FM-radio laitteena on monelle jo entuudestaan hyvin tuttu. Näin ollen laitteen määrittely on suurimmalta osin tehty jo valmiiksi.

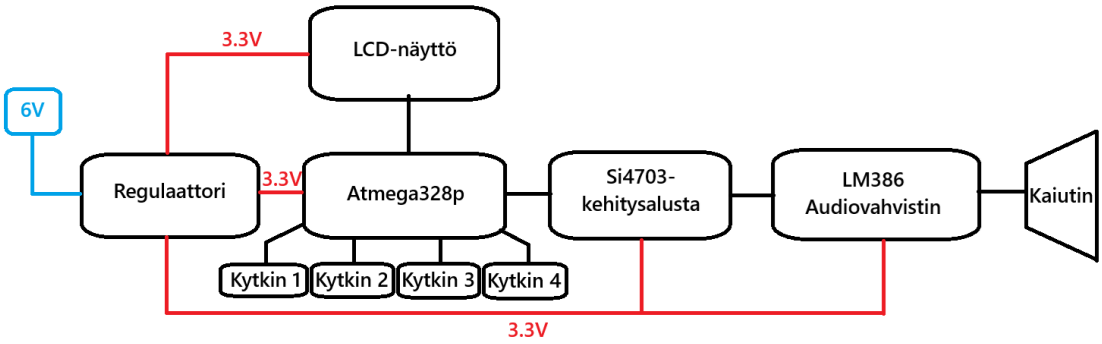
3.1 Laitteen määrittely

FM-radion keskeisin tehtävä on demoduloida ja toistaa FM-kantoaallolla välitetty viesti eli FM-radioaseman lähettämä ohjelma. Kun vastaanotettu lähetys on saatu demoduloitua, tulee tämä sisältö vahvistaa ja toistaa kuuntelijalle. FM-signaalin demoduloinnin ja vahvistamisen suorittaa tässä työssä Si4703-kehitysalusta. Demoduloinnin jälkeen audiosignaali tulee saada siirrettyä kaiuttimelle käyttäjän kuunneltavaksi. Tähän tehtävään tarvitaan siis jonkinlainen audiovahvistin sekä kaiutin, joka toistaa vastaanotetun sanoman. Kaikissa tilanteissa audiosignaalia ei haluta toistaa samalla voimakkuudella, minkä vuoksi äänenvoimakkuuden säätö tulee myös ottaa osaksi laitteen suunnittelua.

Äänenvoimakkuuden säätämisen lisäksi tulee soitettavaa FM-taajuutta pystyä muuttamaan. Taajuuden vaihto tapahtuu kolmen erillisen kytkimen avulla, joiden avulla FM-taajuusalueelta voidaan etsiä kanavia liikkuen ylös- ja alaspäin yhden desimaalin tarkkuudella. FM-taajuuden säätö yhden desimaalin tarkkuudella on tärkeä lisä laitteeseen, koska ylös- ja alaspäin etsittäessä Si4703-kehitysalusta etsii kanavia niiden signaaliteho perusteella. Tällöin on mahdollista, että kehitysalusta ohittaa joitakin kanavapaikkoja, jos niiden signaaliteho on liian heikko. Lisäämällä laitteeseen mahdollisuuden liikkua FM-taajuusalueella yhden desimaalin tarkkuudella, saadaan näin katettua koko FM-taajuusalue.

FM-taajuuden säätämisen lisäksi sisältää laite vielä neljännen kytkimen, jonka tarkoitus on resetoida laite. Resetointi on mikrokontrollerijärjestelmissä tärkeä ominaisuus, koska se mahdollistaa vikatilanteiden korjauksen laitteen uudelleenkäynnistyksellä. Tässä työssä radio alkaa toistamaan ohjelmistossa määritettyä kanavaa heti käynnistyksen jälkeen. Tällöin jos yksittäinen kanava on asetettu ohjelmistossa käynnistymään aina laitteen toiminnan alkaessa, voidaan RESET-kytkimellä siirtyä kuuntelemaan tätä kanavaa.

Viimeisimpänä tarvitaan laitteeseen jonkinlainen näyttö, jotta käyttäjä pystyy seuraamaan laitteen toiminnan vaiheita eli toistettavaa FM-taajuutta ja äänenvoimakkuuden tasoa. Nämä tiedot ovat käyttäjälle äärimmäisen tärkeitä, koska ne ovat tämän työn osalta ainoita parametreja, joita käyttäjä voi muuttaa laitteen toiminnan aikana. Kuvassa 9 on esitetty lohkokaavio, johon on sisällytetty laitteen kaikki funktionaaliset lohkot.



Kuva 9. *FM-vastaanottimen lohkokaavio.*

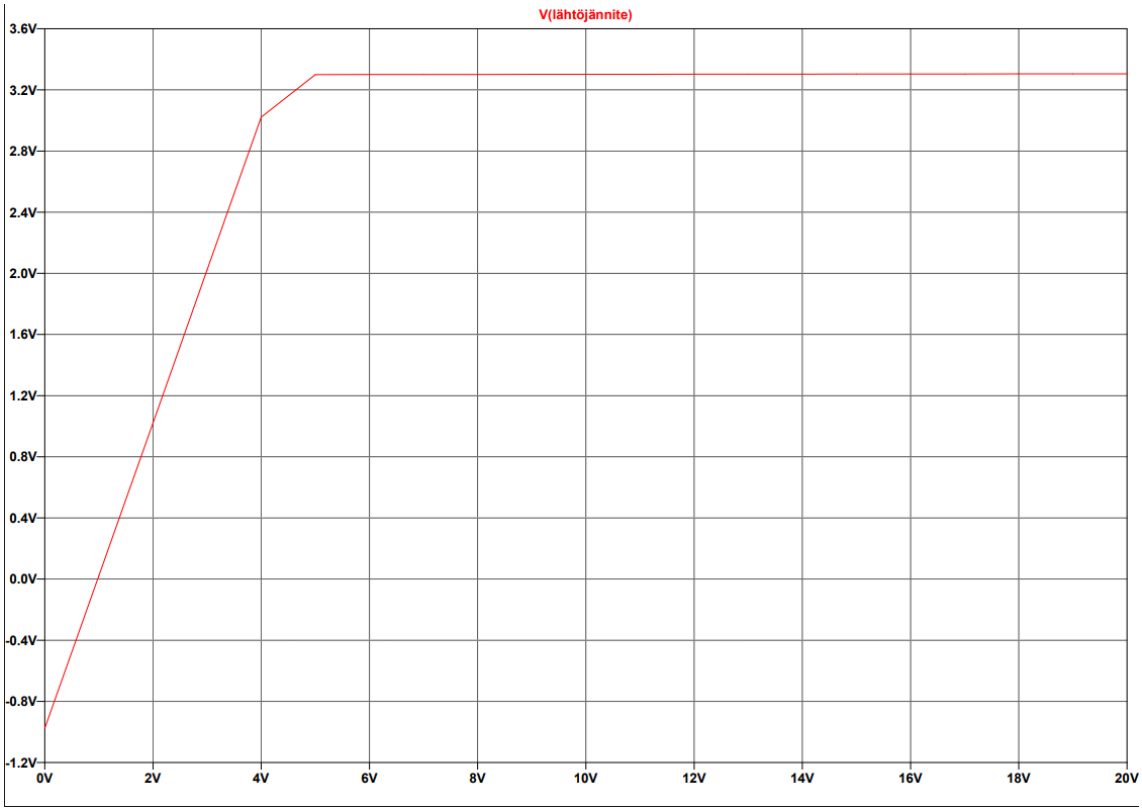
3.2 Osien ja lohkojen simulointi

Tämä projekti on ollut erittäin käytännönläheinen eli monia asioita on suunniteltu suoraan koekytkentälevyllä kokeilemalla. Tämän työn osalta tällainen testaus on mahdollista, koska käytössä olevien lohkojen osalta on ollut saatavilla runsaasti dokumentaatiota ja ohjeita, joidenka pohjalta laitetta on ollut mahdollista rakentaa. Tämän tapaisessa testauksessa on kuitenkin riskinsä, sillä jos kytkentöihin ja ohjeistuksiin ei kiinnitä tarpeeksi huomiota, on mahdollista että laite tai sen osia tuhoutuu.

Suurin ongelmien aiheuttaja tässä työssä on käyttöjännitteen suuruus. Ongelma muodostuu siitä, että perinteinen Arduino UNO käyttää 5 V:n käyttöjännitettä ja logiikkatasoja, kun taas tämän laitteen muut osat käyttävät 3,3 V:n käyttöjännitettä. Tätä ongelmaa ratkaisemaan kehitettiin teoriaosuudessa kuvassa 8 esitelty regulaattorikytkentä. Ennen laitteen lopullista rakentamista tulee kuitenkin simuloida, toimiiko tämä regulaattorikytkentä odotetulla tavalla. Tällaisia simuloitavia aiheita on esimerkiksi jo teoriaosuudessa mainittu hukkatehon määrä sekä itse regulaation onnistuminen.

Tässä työssä kaikki simulaatiot on toteutettu käyttäen LT Spice-simulaatio-ohjelmaa. Regulaattoripiiriä simuloitaessa on käytetty LT Spice-ohjelman valmista komponenttia LT1117-3.3, joka vastaa erittäin paljon työssä käytettävää LM1117-3.3 komponenttia [24]. Lisäksi kytkennän kuormaksi on asetettu ideaalinen virtalähde, joka simuloi kytken-

nän koekytkentälevyltä mitattua 50 mA kuormavirtaa. Ensimmäisessä simulaatiossa testattiin komponentin reguloitiominaisuuksia pyyhkäisemällä tulojännitteen arvoa välillä 0-20 V, milloin saatiin lähtöjännitteelle alla olevan kuvan 10 mukaisia tuloksia.

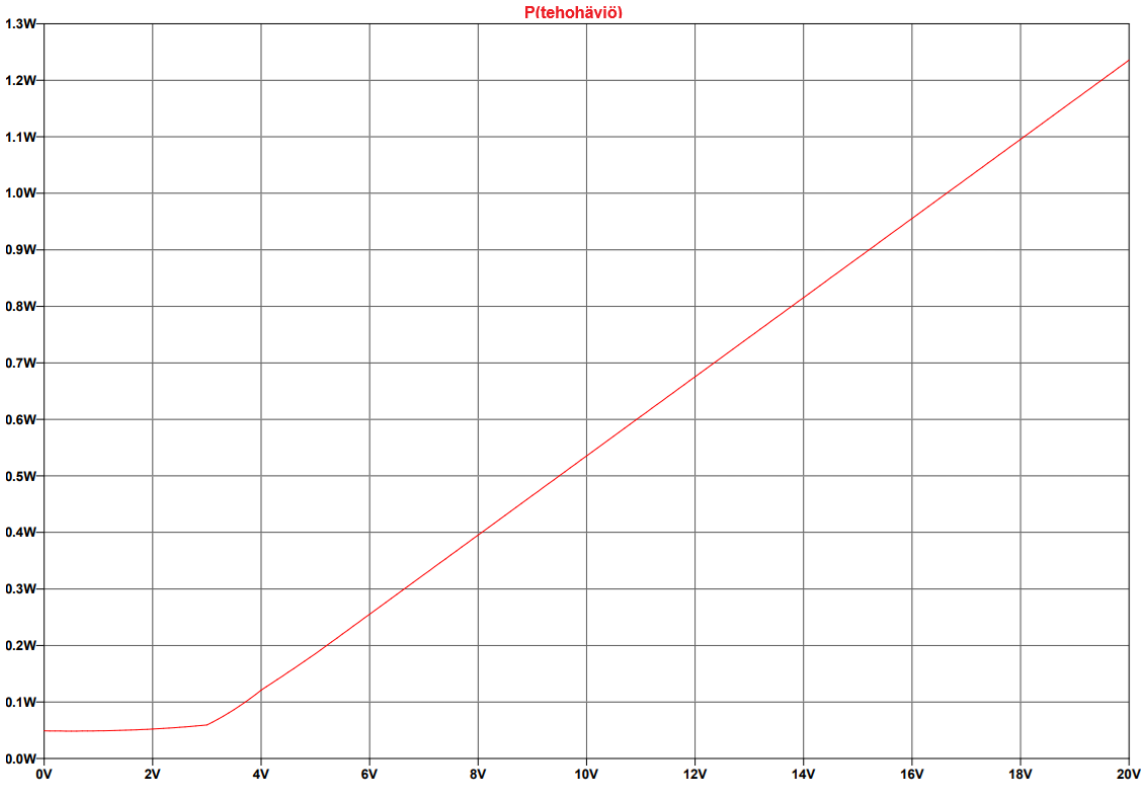


Kuva 10. *Regulaattorikytkennän lähtöjännite tulojännitteen funktiona.*

Kuvassa 10 esitettyjen tulosten perusteella voidaan päätellä, että regulaattoripiiri todella reguloi paristoilta saadun 6 V:n jännitteen laitteelle sopivaan 3,3 V:n jännitteeseen. Muuta huomioitavaa kuvassa 10 esitetyssä simulaatiossa on se, että siitä on nähtävillä myös luvussa 2.5 mainitun pudotusjännitteen vaikutus. Kuten kuvasta 10 voidaan huomioida, on lähtöjännitteen arvo alle 3,3 V, kunnes tulojännite saavuttaa 5 V:n jännitteen. Toisin sanoen regulaattori ei kykene saavuttamaan 3,3 V jännitearvoa, jos paristokytkennän jännite on alle lähtöjännitteen ja pudotusjännitteen summan verran. Laskennallisesti LM1117:n pienin tulojännite, jolla regulaattorin tulisi vielä toimia on 4,5 V, jos laskuissa käytetään pudotusjännitteen arvoa 1,2 V [22, s.6]. Simulaation mukaan pienin arvo, jolla regulaattori toimii on 5 V, josta voidaan tehdä päätelmä, että laskennallinen ja simuloitu arvo vastaavat toisiaan melko hyvin. Tärkeimpänä huomiona voidaan kuitenkin todeta, että suunniteltu 6 V:n paristokytkentä on riittävä laitteen toiminnan takaamiseksi.

Muita ominaisuuksia, mitä regulaattorikytkennästä kannattaa simuloida on sen tehohäviö. Kappaleen 2.5 mukaisesti, lineaarisen regulaattorin tehohäviö muuttuu komponentin

lämmöksi. Simulaation avulla voidaan siis varmistua siitä, että LM1117-komponentti ei lämpene liikaa laitteen lopullisessa versiossa. Monet regulaattori-IC:t, kuten myös LM1117 sisältää sisäänrakennetun sulkuominaisuuden, joka käytännössä tarkoittaa sitä, että komponentti sammuttaa itsensä mikäli sen lämpötila kasvaa liian suureksi [22, s.18]. Tästä syystä on tärkeää tarkistaa, että komponentti ei lämpene liikaa.

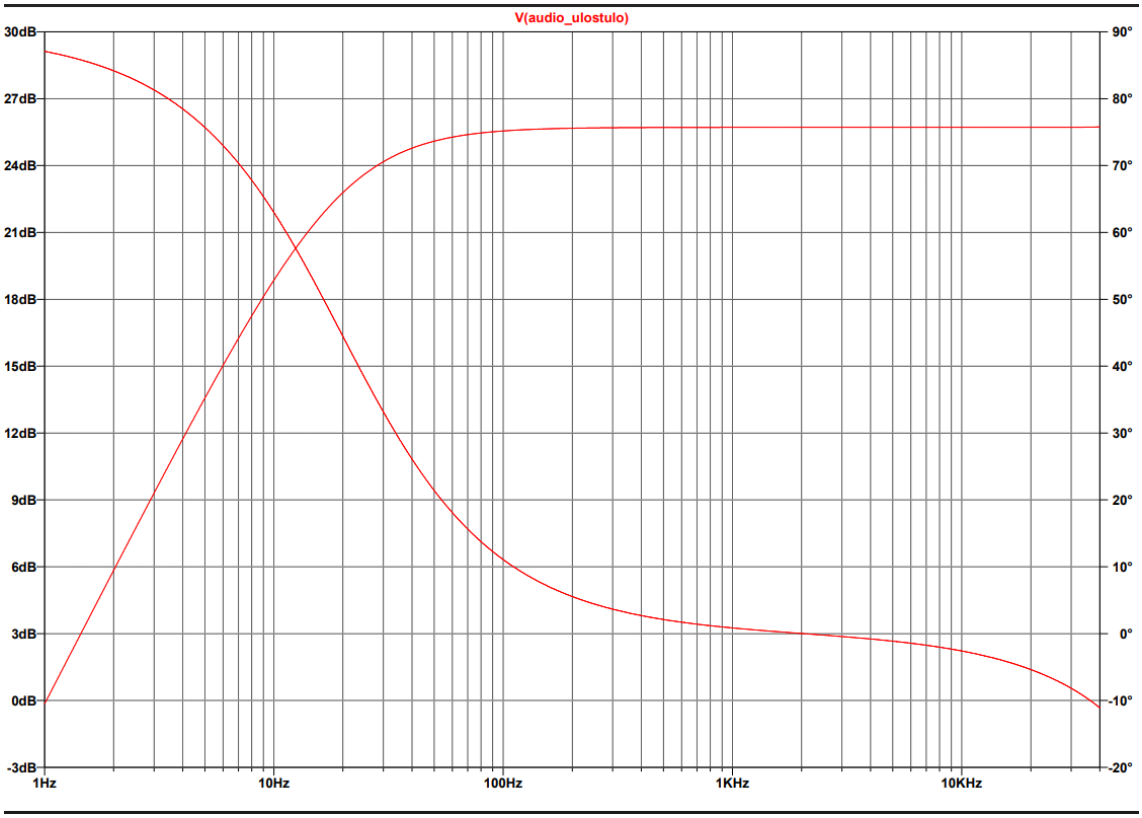


Kuva 11. *Regulaattorikytkennän tehohäviö tulojännitteen funktiona.*

Yllä olevasta kuvasta 11 Voidaan nähdä, että 6 V:n tulojännitteellä regulaattorikytkennän tehohäviö on n. 0.25 W. Tästä voidaan tehdä päätelmä, että regulaattori ei lämpene tämän laitteen toimintatarkoituksissa suuresti, joten sen jäähdyttämisestä ei tarvitse huolehtia. Merkittävä huomio kuvasta 11 on kuitenkin se, että jopa 50 mA kuormavirralla regulaattorin tehohäviö pystyy nousemaan jo yli yhden watin. Tämä on hyvä esimerkki kappaleessa 2.5 tehdystä huomiosta, jonka mukaan lineaaristen regulaattorien hyötysuhteet ovat alhaiset, jos potentiaaliero tulon ja lähdön välillä kasvaa suureksi.

Viimeisimpänä simuloinnin kohteena on laitteessa käytettävä audiovahvistinkytkentä. Audiovahvistinkytkennän simulointi on mielekästä, sillä sen avulla on mahdollista määrittää vahvistimen suuntaa antava taajuusvaste. Tässä työssä käytetty audiovahvistin on LM386 IC, jolle valmistaja Texas Instruments ei kuitenkaan suoraan ole luonut LT Spice-simulointimallia. Internetin foorumeilla on kuitenkin saatavilla muiden käyttäjien luomia

LT Spice-malleja kyseiselle komponentille [25], jonka avulla kytkennän toimintaa voidaan testata. Saatuihin tuloksiin tulee kuitenkin suhtautua varoen, koska kyseessä ei ole Texas Instrumentsin virallinen simulointimalli. Tämän mallin avulla on kuitenkin mahdollista simuloida, vastaavatko laskennallinen ja simuloitu -3 dB-rajataajuus toisiaan. Audiovahvistinkytkennän simulaatiotulokset on esitetty alla olevassa kuvassa 12.



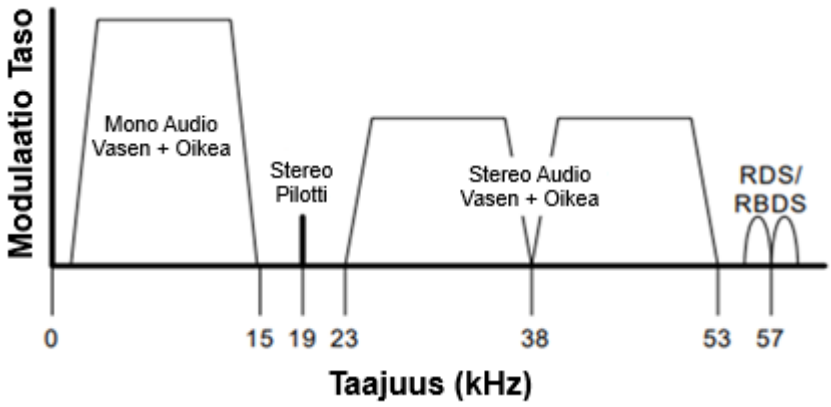
Kuva 12. *Audiovahvistinkytkennän taajuusvaste.*

Kuvassa 12 on 0 dB kohdalta alkava käyrä ulostulosignaalin amplitudi. Toinen käyrä esittää vaiheen muutosta taajuuden funktiona. Amplitudi on yllä olevassa kuvassa skaalattu 3 dB välein, jolloin -3 dB -pisteen etsiminen helpottuu. Kuvasta 12 voidaan siis nähdä, että audiovahvistimen -3 dB -piste on noin 20 Hz kohdalla. Tämä tulos vastaa kappaleessa 2.4 laskettua arvoa ja näin ollen voidaan todeta, että ulostulokondensaattorin uudelleenmitoituksella todella on vaikutusta audiovahvistimen taajuusvasteeseen. Kuten edellä mainittiin, on tämä simulaatio kuitenkin vain suuntaa antava sen vuoksi, että simulaatiomallin oikeellisuudesta ei voida tässä tilanteessa olla aivan varmoja. LM386-datalehden mukaan tässä työssä käytetyn audiovahvistinkytkennän vahvistus tulisi olla n. 26 dB [19, s.10], mistä voidaan päätellä, että simulaation tulokset vaikuttavat loogisilta.

3.3 Ohjelmiston suunnittelu

Työtä varten luotu ohjelmisto on rakenteeltaan hyvin yksinkertainen ja kompakti. Syynä tälle on se, että ohjelmiston rakentamisessa on voitu tehokkaasti hyödyntää Si4703 IC:lle ja LCD-näytölle luotuja valmiita ohjelmistokirjastoja. Tästä seuraa se, että funktiot, joita laitteen operointiin tarvitaan, on jo määritetty valmiiksi ohjelmistokirjastoihin. LCD-näytön ohjaukseen tarkoitettu LiquidCrystal-kirjasto on tässä työssä jätetty koskemattomaksi, mutta Si4703-kirjastoon on sen sijaan tehty hieman muutoksia, jotka parantavat laitteen kuuluvuutta.

Si4703-kirjastoon tehty muutos liittyy laitteen alustukseen, jossa määritellään se, miten Si4703 IC prosessoi vastaanotettua FM-signaalia. 1960-luvulta asti radiosignaalilähetykset on toteutettu käyttäen niin kutsuttua stereo multipleksointia (MPX), jossa mono ja stereo audiosignaali on jaettu stereo pilottisignaalin molemmille puolille [11, s.16]. MPX-signaali on kuvattuna alla olevassa kuvassa 13.



Kuva 13. *MPX-signaalin rakenne [mukaillen lähteestä 11, s.16].*

Koska MPX-signaali sisältää sekä stereo- että monosignaalin, pystyy Si4703 IC tekemään signaalin vahvuuden perusteella päätöksen siitä, kumpaa signaalia se päättää käyttää [11, s.16-17]. Tämän työn osalta on kuitenkin käytännöllisempää pakottaa IC toistamaan pelkästään monosignaalia, koska itse laitteessa kaiuttimia on vain yksi. Toisin sanoen stereoprosessointi vie IC:ltä turhia resursseja, vaikka itse stereoaudiota ei voida toistaa. IC:n pakottaminen toistamaan monosignaalia on hyvin yksinkertaista, koska se vaatii ainoastaan yhden laiterekisterin muokkaamista. Tämä rekisteri on datalehdellä mainittu POWERCFG, josta muuttamalla 13. bitin arvon ykköseksi, voidaan IC pakottaa toimimaan monona [11, s.24].

Kun Si4703 IC:n alustus on saatu valmiiksi, voidaan aloittaa itse ohjelmakoodin kirjoitus. Ohjelmakoodin kirjoitus on melko suoraviivaista, koska mikrokontrolleri käytännössä

seuraa vain neljän kytkimen tilaa ja tekee tarvittavat toimenpiteet niiden mukaan. Kytkinten lisäksi ohjelma tekee ADC-muunnoksen potentiometrin arvolle, jolla muutetaan äänenvoimakkuuden tasoa. Alla olevassa ohjelmassa 1 on esitetty pääfunktio kokonaisuudessaan. Lopullinen ohjelmakoodi on esitetty liitteessä D.

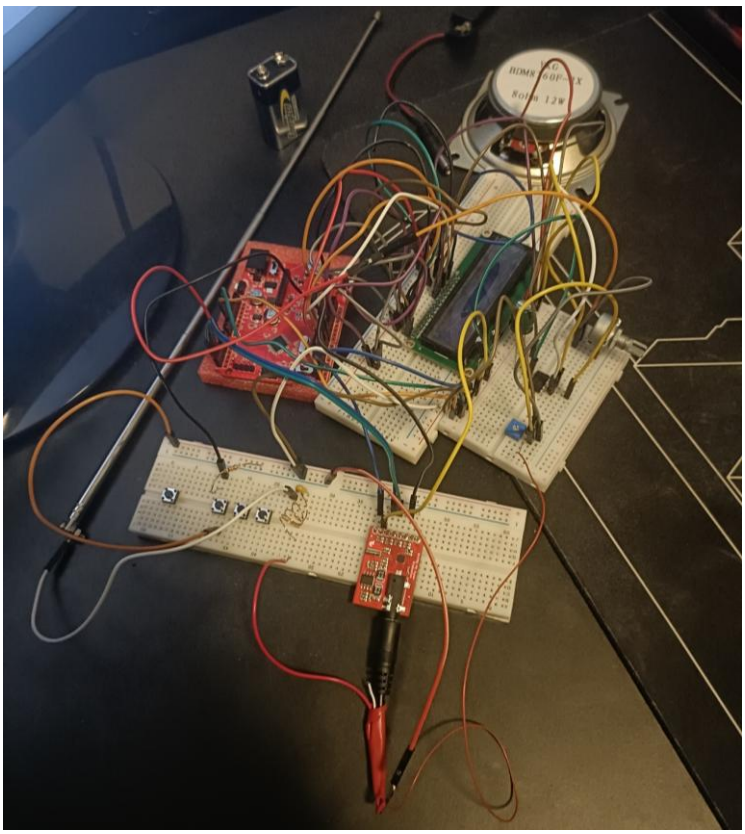
```
void loop()
{
  //Lukee potentiometrin arvon ja asettaa äänenvoimakkuuden
  lue_volume();

  //Odottaa, että SEEK UP-nappia painetaan
  if (digitalRead(8)){
    kanava = radio.seekUp();
  }

  //Odottaa, että SEEK DOWN-nappia painetaan
  if (digitalRead(9)){
    kanava = radio.seekDown();
  }

  //Odottaa, että FINE TUNE-nappia painetaan
  if (digitalRead(10)){
    kanava = kanava+1;
    radio.setChannel(kanava);
  }

  //Rajaa taajuudet välille 87,5-108 MHz
  if (kanava == 0){
    kanava = 875;
  }

  //Päivittää LCD-näytön arvon
  naytto();

  //200 ms viive toiminnan varmistamiseksi
  delay(200);
}
```

Ohjelma 1. *Luodun ohjelmiston pääfunktio.*

3.4 Prototyypin rakentaminen koekytkentälevyillä

Ennen piirilevyn suunnittelua ja lopullisen laitteen kasaamista tulee laitteen ja sen ohjelmiston toimivuus testata käytännössä. Tämän tyylinen testaus saadaan parhaiten toteutettua käyttäen koekytkentälevyjä ja siihen sopivia johtimia. Regulaattoripiiriä lukuun ottamatta laite sisältää jo kaikki osat ja lohkot, jotka sisältyvät myös lopulliseen laitteeseen.

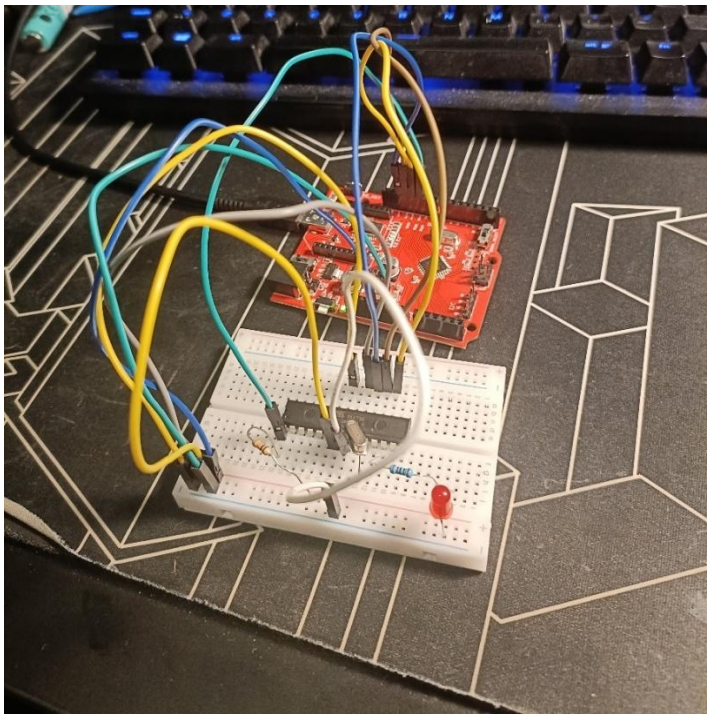
Prototyyppien valmistus koekytkentälevyillä käy usein vaikeaksi, jos suunniteltava laite sisältää paljon lohkoja ja toiminnallisuuksia. Tämän työn osalta tätä sekavuutta on pyritty vähentämään lohkojen järjestelmällisellä sijoittelulla. Kuvan 15 mukaisesti pienelle koekytkentälevylle on sijoitettu audiovahvistinkytkentä, sillä se toimii itsenäisesti omana lohkonaan. Vastaavalla tavalla Si4703-kehitysalusta ja LCD-näyttö on sijoitettu omille koekytkentälevyilleen. Arduino on erotettu omaksi lohkokseen sijoittamalla se hieman sivuun muusta laitteesta.



Kuva 15. *FM-vastaanottimen prototyyppi koekytkentälevyillä.*

3.5 Arduinon käyttö ISP-ohjelmointityökaluna

Kun ohjelmisto on todettu toimivaksi koekytkentälevyllä, tulee se ladata piirilevylle juotettavaan mikrokontrolleriin. Ohjelmiston lataamiseen mikrokontrollerille on monia tapoja, mutta koska eräs tämän työn tavoitteista on käyttää Arduinoa mahdollisimman monessa työvaiheessa, on lopullinen ohjelma päätetty ladata mikrokontrolleriin käyttäen Arduinoa itsessään ohjelmointityökaluna. Kuva ohjelmointiprosessista on esitetty alla olevassa kuvassa 14.



Kuva 14. *Arduinon käyttö ISP-ohjelmointityökaluna.*

Jotta Atmega328p saadaan ohjelmoitua Arduino-pohjaisella koodilla, tulee siihen ensin ohjelmoida Arduino-bootloader [26]. Käytännössä bootloader siis alustaa asetukset, joilla mikrokontrolleri toimii. Tällaisia asetuksia ovat esimerkiksi käytetty kellotaajuus sekä käyttöjännite. Kun Arduinoa käytetään ISP-ohjelmointityökaluna (In-System Programming), voidaan näitä asetuksia säätää käyttötarkoituksen mukaan. Tässä työssä asetetaan käyttöjännitteen arvoksi 3,3 V ja kellotaajuudeksi 8 MHz.

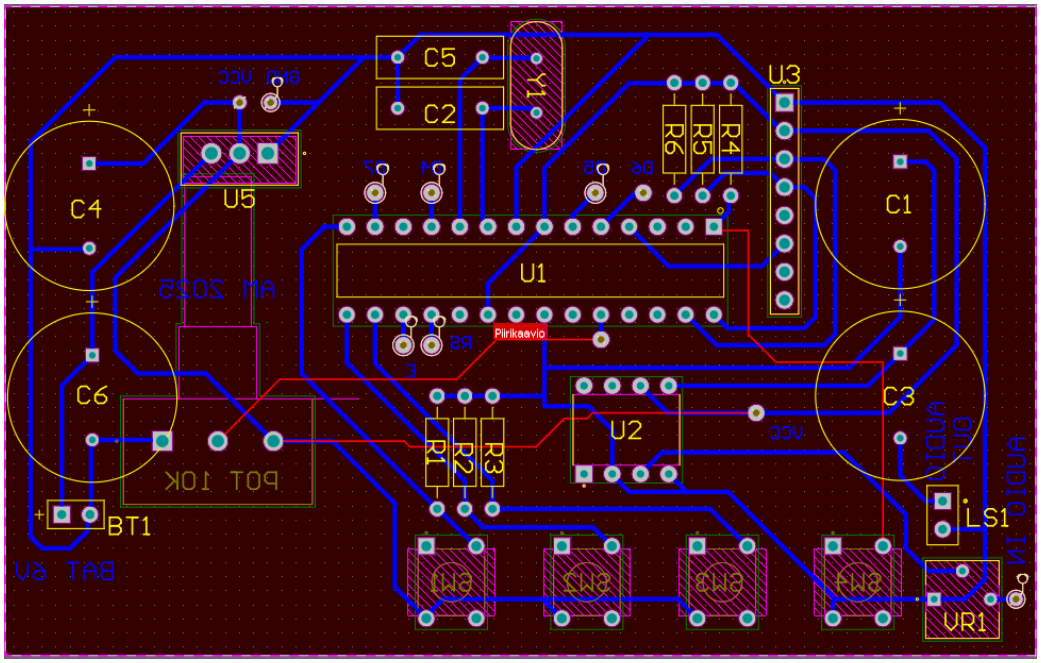
Muita muutettavia asetuksia on BOD (Brown-out-detector), joka määrittää jännitteen jolla Atmega328p alkaa resetoimaan itseään [6, s.40]. Tämän tarkoituksena on estää mikrokontrollerin virheellinen toiminta tilanteessa, jossa sen käyttöjännite laskee alle sallitun rajan. Tämä on tärkeä ominaisuus varsinkin silloin kun mikrokontrollerin muistien sisältö halutaan pitää eheänä. Tässä työssä ohjelma ei kuitenkaan tallenna mikrokontrollerin pysyvään muistiin mitään, joten on tämä ominaisuus turha. Datalehden ohjeistuksen mukaisesti on BOD siksi poistettu käytöstä [6, s.37].

Kun bootloader-asetukset (Fuses) on saatu valittua, voidaan itse bootloader ja luotu ohjelma ladata Atmega328p-mikrokontrolleriin. Ohjelmaa ladattaessa tulee varmistaa se, että mikrokontrolleriin on kytketty sen vaatimat oheislaitteet. Nämä oheislaitteet ovat esitetty teoriaosuuden kuvassa 1, joista tärkein on 8 MHz kvartsikide. Mikäli kellolähde ei ole kytkettynä mikrokontrolleriin, antaa Arduino IDE ilmoituksen virheen merkiksi. Tämä kellolähde on nähtävillä myös kuvan 14 tilanteessa.

3.6 PCB-suunnittelu

Mikrokontrollerin ohjelmoimisen ja onnistuneiden koekytkentälevykokeiden jälkeen voidaan luotua suunnitelmaa lähteä siirtämään PCB-levylle. Tässä tilanteessa PCB-levyn tavoite on saada pakattua työn komponentit mahdollisimman pieneen tilaan, jotta se on lopulta mahdollista pakata koteloon. Kytkinten ja potentiometrin sijoittelu oli tässä työssä haastavin vaihe PCB-suunnittelussa, koska niiden sijainnilla olisi suora vaikutus kotelon ulkonäköön ja toimivuuteen. Lopulta päädyttiin ratkaisuun, jossa kytkimet ja potentiometri juotetaan samaan tasoon johdinvetojen kanssa piirilevyn pohjatasoon. Yksinkertaisuuden vuoksi on tässä työssä piirilevy tehty yksipuoleiselle valoherkälle PCB-levylle, jolloin kaikki johdinvedot on mahdollista tehdä yhdelle puolelle. Luotu PCB-layout on esitetty alla olevassa kuvassa 16.

Kun kytkinten ja potentiometrin sijainnit oli saatu määriteltyä, voitiin muut komponentit sijoitella piirilevylle siten, että kokonaispinta-alasta tulisi mahdollisimman pieni. Tässäkin tilanteessa on kuitenkin pyritty järjestelmällisyyteen siinä mielessä, että samantyyppiset komponentit, kuten vastukset ja kondensaattorit, on aseteltu omiin ryhmiinsä piirilevyllä. IC:den kohdalla tuli tarkasti miettiä, miten ne tulisi sijoitella, koska niihin on kytkettynä suuri määrä johdinvetoja. Myös Si4703-kehitysalustan sijoittelu tuotti vaikeuksia, koska se sisältää itsessään jo pienen PCB-levyn. Ongelmat muodostuivat siitä, että PCB piti saada sijoiteltua piirilevyn päälle siten, että se ei osu muihin ulospäinsuuntautuviin komponentteihin, kuten kondensaattoreihin.



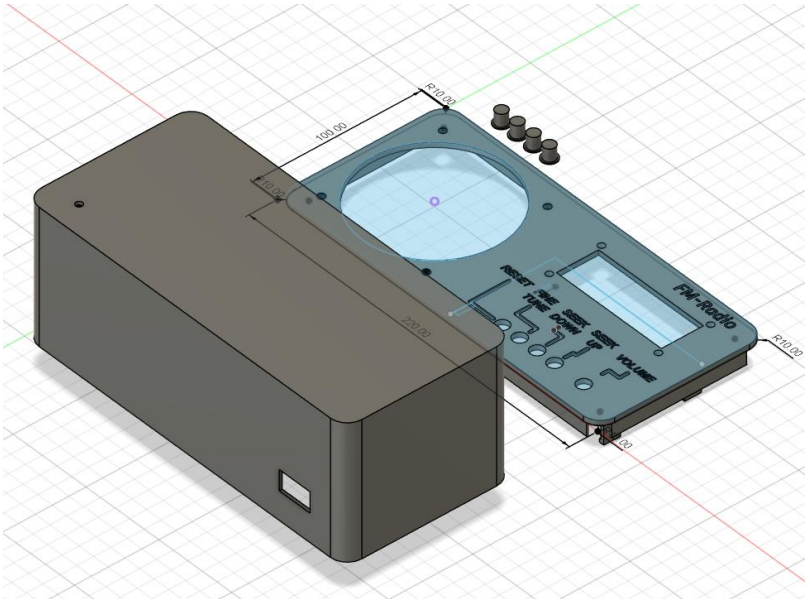
Kuva 16. *Altium Designer-ohjelmalla luotu PCB-layout.*

Keskeinen osa PCB-suunnittelua on johdinvetojen leveyden määrittäminen. Mil on piirilevysuunnittelussa yleisesti käytetty mittayksikkö, joka vastaa suuruudeltaan tuuman tuhannesosaa. Tällöin yksi mil on suuruudeltaan 0,0254 millimetriä. IPC-2221A-standardin mukaan 10 milin johdinleveys riittäisi jopa 400 mA virrankestoon [27, s.152], mikä tämän työn osalta olisi aivan riittävä arvo. Koska tämän työn piirilevy kuitenkin käsin valmistetaan etsaamalla**,** on tämä 10 milin arvo käytännön testauksen perusteella aivan liian pieni. PCB:n uudemmissa versioissa nostettiin johdinvetojen leveys ja niiden väliset etäisyydet 20 miliin, jolloin käsin valmistuksessa tapahtuvat virheet eivät haittaa laitteen toimintaa.

Koska kaikki komponentit tässä työssä ovat läpiladottavia, tulee niille porata reiät piirilevyyn. Perinteisesti PCB-suunnittelussa määritetään niin kutsuttu porauskartta, jossa on esitetty porattavien reikien paikat ja halkaisijat. Tässä työssä tarvetta porauskartalle ei kuitenkaan ole, koska potentiometriä lukuun ottamatta kaikkien komponenttien jaloille tarkoitetuille paikoille porataan reikä. Lisäksi nämä reiät on tehty kaikki käyttäen 1 mm poranterää, jolloin terän suuruus pysyy vakiona reikiä porattaessa.

3.7 Kotelointi

Kun laitteen elektroniikkaosuus on saatu valmiiksi, tulee näiden elektroniikkaosien suojaksi luoda jonkinlainen kotelo. Samalla kotelosuunnittelulla voidaan parantaa laitteen akustisia ominaisuuksia. Kotelo päädyttiin tekemään 3D-tulostamalla, koska se on tässä tilanteessa edullisin ja tehokkain tapa luoda yksinkertainen suoja laitteen komponenteille.



Kuva 17. *Luotu 3D-malli Autodesk Fusion-ohjelmassa esitettynä.*

Jotta kotelo on mahdollista 3D-tulostaa, tulee siitä ensin luoda 3D-malli. Tässä työssä on 3D-mallin luomisessa käytetty Autodesk Fusion-ohjelmaa. Viimeistelty 3D-malli on esitetty yllä olevassa kuvassa 17. 3D-mallin luomisessa on tärkeää mitata ja mallintaa laitteen komponentit sekä niiden dimensiot mahdollisimman tarkasti. Tässä työssä ongelmia aiheuttavia osia ovat nimenomaan LCD-näyttö, kytkimien paikat sekä itse kaiutin. Koska elektroniikkaosuus on kuitenkin luotu jo valmiiksi, on näiden komponenttien dimensiot helppo mitata työntömitan avulla. Kaikki elektroniikkaosat on kiinnitetty tässä työssä kanteen, jolloin kotelon avaaminen esimerkiksi paristojen vaihtoa varten on helppoa. Kaiutin ja LCD-näyttö on kiinnitetty kanteen käyttäen M3-pultteja ja muttereita. Lisäksi kannen sisäpuolella on lovet, joihin piirilevy asettuu tiukasti paikalleen. Lisäksi piirilevy kiristetään vielä kannen ulkopuolelta potentiometrin jenkoihin menevällä mutterilla. Alla olevassa kuvassa 18 on esitelty laitteen lopullinen 3D-tulostettu kotelo ja siihen kiinnitetyt elektroniikkaosat.

Kuva 18. *Viimeistelty FM-vastaanotin.*

4. YHTEENVETO

Koteloinnin myötä on FM-radio saatu valmiiksi. Laite löytää valtaosan Tampereen radiokanavista ja luodut ominaisuudet, kuten kanavan etsintä, hienosäätö ja äänenvoimakkuuden säätö, toimivat loistavasti. Olen tyytyväinen työn lopputulokseen ottaen huomioon, että tämä on ensimmäinen elektroniikkalaite, jonka suunnittelen alusta loppuun itse. Samalla opin paljon uusia asioita elektroniikkalaitteen suunnittelusta ja sen eri vaiheista.

Laite toimii tällä hetkellä hyvin, mutta varaa jatkokehityksen löytyy vielä. Esimerkiksi kappaleessa 2.2 mainittu RDS-dataprosessointi voisi olla hyvä lisä laitteen toimintaan. Tässä työssä kotelon akustisiin ominaisuuksiin ei keskitytty mitenkään suuresti työn aihealueen rajaamiseksi. Kotelon jatkokehitys on kuitenkin vielä mahdollista, koska sen kansi ja pohja ovat toisistaan irrallisia osia. Tällöin esimerkiksi refleksiaukon lisääminen pohjaosaan voisi parantaa varsinkin bassotaajuuksien toistamista.

Työn johdannossa esitettiin kysymys siitä, voiko kotioloissa luoda vertailukelpoisen FM-radion kaupallisiin radioihin verrattuna. Vastaus kysymykseen riippuu hyvin paljon siitä, minkälaisia ominaisuuksia radioon haluaa. Mikäli käyttäjä haluaa radion, joka yksinkertaisesti vain toistaa halutun FM-taajuuden kannattaa tällöin valita kaupallinen tuote niiden halvan hinnan vuoksi. Mikäli käyttäjä haluaa RDS-prosessointiin kykenevän ja mahdollisesti vain nappikuulokkeilla toimivan radion, kannattaa tällöin valita Si4703-kehitysalustalla tehty radio. Vaikka rahallisesti kotioloissa tehty FM-radio ei olisi kilpailukykyinen kaupallisten radioiden kanssa, voi itse tehdessä laitteen ominaisuudet räätälöidä omiin käyttötarkoituksiin sopivaksi.

Elektroniikkalaitteen suunnittelussa ja rakentamisessa tulee ottaa huomioon, että se voi tarjota tekijälleen kokemuksia ja ideoita, joita ei välttämättä voi rahallisesti mitata. Arduino-alusta kykenee tehokkaasti toimimaan lähes jokaisessa elektroniikkalaitteen suunnittelun vaiheessa aina suunnittelusta lopullisen kokoonpanon ohjelmointiin. Joissakin tilanteissa saatetaan Arduino jopa jättää osaksi viimeisteltyä laitetta sen halvan hinnan vuoksi. Arduino on työkaluna monipuolinen ja käyttäjäläheinen, minkä vuoksi sen menestys ja suuri suosio elektroniikan harrastajien ja ammattilaisten keskuudessa ei ole ihme.

LÄHTEET:

[1] What is Arduino?, Arduino, 2018. Saatavissa (viitattu 31.3.2025): https://www.arduino.cc/en/Guide/Introduction/

[2] K. Silvonen, Elektroniikka ja sähkötekniikka, Gaudeamus, Otaniemi, 2018, 504 s.

[3] RayMing, What Are Breakout Boards / SMT Breakout PCB ?, RayPCB, 2025. Saatavissa (viitattu 31.3.2025): https://www.raypcb.com/smt-breakout-boards/

[4] Jeff, Schematic of a Breadboard Arduino, Fiz-ix, 2012. Saatavissa (viitattu 31.3.2025): https://fiz-ix.com/2012/11/schematic-of-a-breadboard-arduino/

[5] Arduino Integrated Development Environment (IDE) v1, Arduino, 2025. Saatavissa (viitattu 8.5.2025): https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/

[6] Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet, Atmel, 2015. Saatavissa (viitattu 31.3.2025): https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[7] D. Wheat, Arduino Internals, 1st ed. 2011, Apress, Berkeley, 2011, 329 s.

[8] Si4703 FM Radio Receiver Hookup Guide, SparkFun Learn, 2025. Saatavissa (viitattu 1.4.2025): https://learn.sparkfun.com/tutorials/si4703-fm-radio-receiver-hookup-guide?_ga=1.110826

[9] Arduino Pro Mini, Arduino, 2025, Saatavissa (viitattu 8.5.2025): https://docs.arduino.cc/retired/boards/arduino-pro-mini/

[10] KIDE 8,000000MHz HC49/4H (matala), PARTCO, 2025. Saatavissa (viitattu 11.5.2025): https://www.partco.fi/fi/elektroniikan-komponentit/passiivit/kiteetoskillaattorit/kiteet-tht/6145-kide-8mhz-4h.html

[11] Si4702-03-C19, Silicon Labs, 2009. Saatavissa (viitattu 8.4.2025): https://cdn.sparkfun.com/assets/f/0/9/0/1/Si4702-03-C19-1.pdf

[12] D. Russell, Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment. Cham, Switzerland, 2010, 264 s.

[13] Toni Klopfenstein, Si4703_FM_Tuner_Evaluation_Board/Libraries/Arduino/src, Sparkfun, 2015, päivitetty 12.9.2018. Saatavissa (viitattu 28.4.2025): https://github.com/sparkfun/Si4703_FM_Tuner_Evaluation_Board/tree/master/Libraries/Arduino/src

[14] AN383: Si47xx Antenna, Schematic, Layout, and Design Guidelines, Skyworks Solutions Inc.,2007, päivitetty 23.1.2025. Saatavissa (viitattu 8.4.2025): https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/application-notes/AN383.pdf

[15] N. Seidle, Toni Klopfenstein, Si4703_Eval_v13.sch, 2014, päivitetty 1.7.2024. Saatavissa (viitattu 28.4.2025): https://cdn.sparkfun.com/datasheets/Wireless/General/Si4703_Eval_v13.pdf

[16] D. J. R. Cristaldi, S. Pennisi, ja F. Pulvirenti, Liquid crystal display drivers: techniques and circuits, Dordrecht: Springer, 2009, 295 s.

[17] TN-LCD-schematic-MS-208kB.png, Wikimedia Commons, 2007. Saatavissa (viitattu 11.5.2025): https://commons.wikimedia.org/wiki/File:TN-LCD-schematic-MS-208kB.png

[18] Liquid Crystal Displays (LCD) with Arduino, Arduino, 2025. Saatavissa (viitattu 11.5.2025): https://docs.arduino.cc/learn/electronics/lcd-displays/

[19] LM386 Low Voltage Audio Power Amplifier, Texas Instruments, 2004, päivitetty 08/2023. Saatavissa (viitattu 8.4.2025): https://www.ti.com/lit/ds/symlink/lm386.pdf

[20] Lam JCM, Analog Audio Amplifier Design, 1st ed, Oxford: Taylor & Francis Group, 2024, 598 s.

[21] B. Duncan, High perfomance audio power amplifiers for music performance and reproduction, Oxford Boston: Newnes, 1996, 463 s.

[22] LM1117 800-mA, Low-Dropout Linear Regulator, Texas Instruments, 2000, päivitetty 01/2023. Saatavissa (viitattu 8.4.2025): https://www.ti.com/lit/ds/symlink/lm1117.pdf

[23] J. Fiore, Operational Amplifiers & Linear Integrated Circuits: Theory and Application, 3e p, Itsejulkaisu, 2018, 556 s.

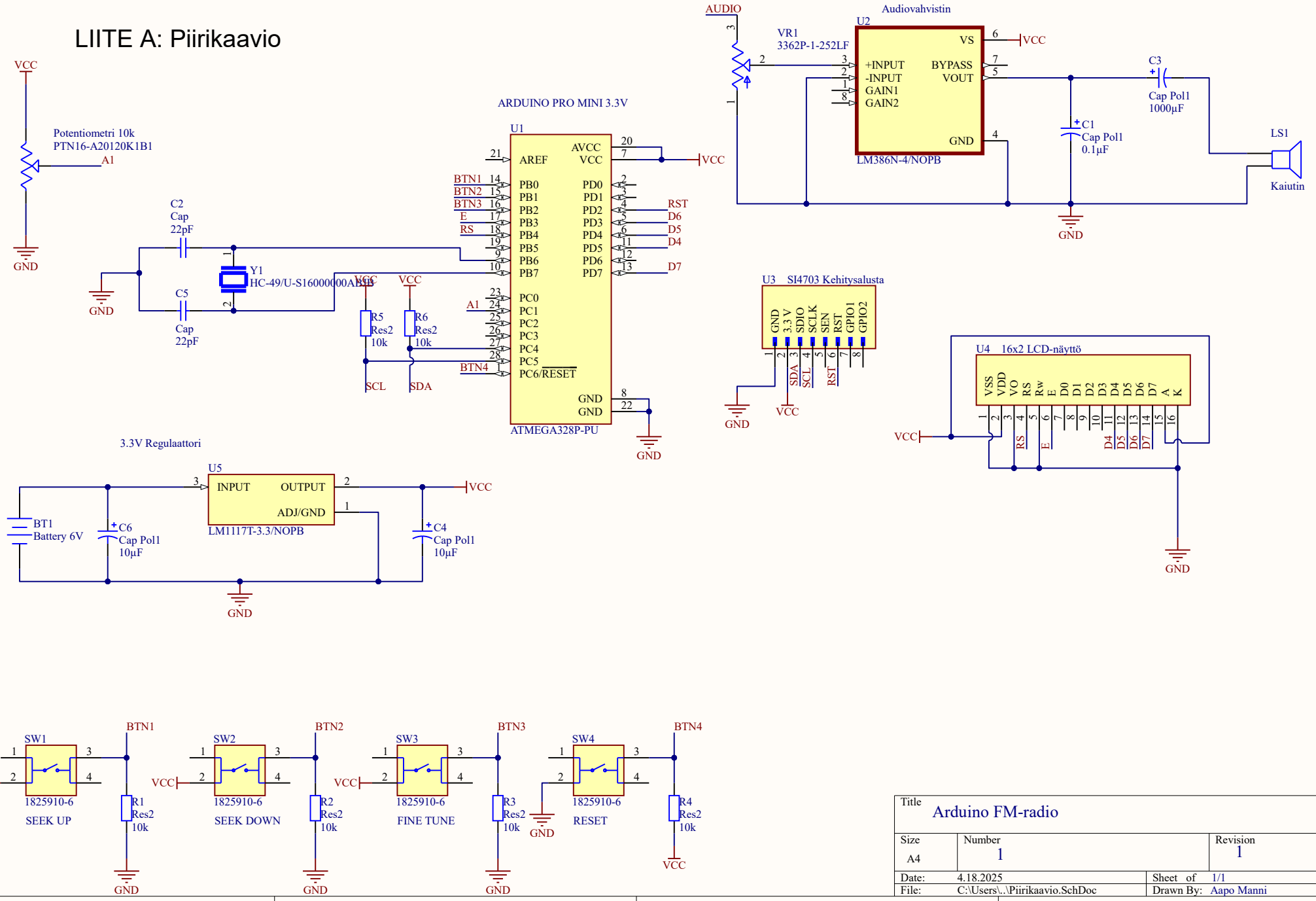
[24] LT1117 Datasheet and Product Info, Linear Technology, 1993, päivitetty 04/2010. Saatavissa (viitattu 3.5.2025): https://www.analog.com/media/en/technical-documentation/data-sheets/1117fd.pdf

[25] Roff, Spice model LM386, Electro-Tech-Online, 2006. Saatavissa (viitattu 11.5.2025): https://www.electro-tech-online.com/threads/spice-model-lm386.23296/

[26] Arduino as ISP and Arduino Bootloaders, Arduino, 2025. Saatavissa (viitattu 8.5.2025): https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoISP/

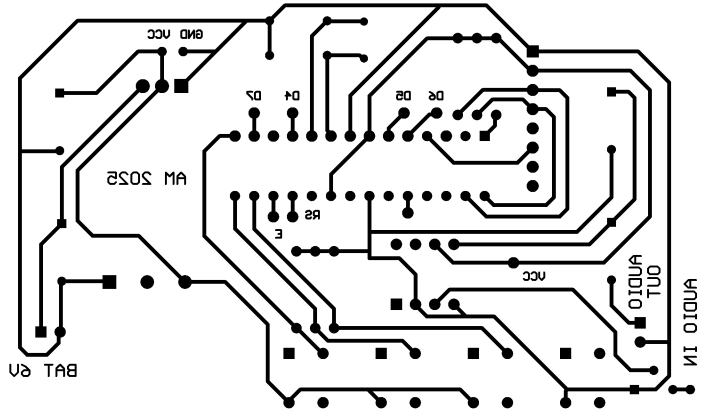
[27] K. Mitzner, Complete PCB Design Using Orcad Capture and Layout, Elsevier Science & Technology, Chantilly, USA, 2007, 529 s.

LIITE A: Piirikaavio



Title	Arduino FM-radio		
Size: A4	Number: 1		Revision: 1
Date: 4.18.2025		Sheet of 1/1	
File: C:\Users\..\Piirikaavio.SchDoc		Drawn By: Aapo Manni	

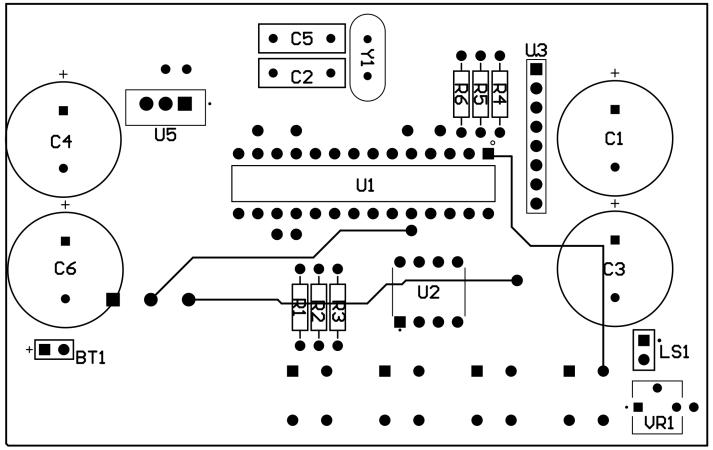
LIITE B: Valotusmaski



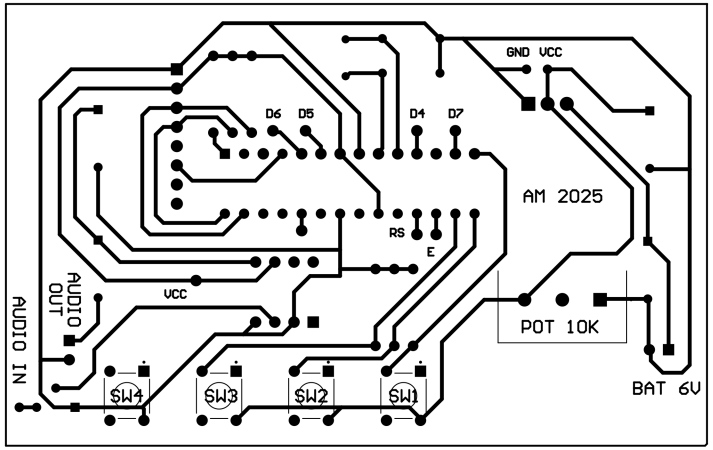
Altium Limited 12a Rodborough Rd Frenchs Forest NSW 2086	ENGINEER: Aapo Manni	TITLE: Arduino FM-radio	
	PCB DESIGNER: Aapo Manni		
	DATE: 18.4.2025	PART NO.:	REV: 1
	FILE NAME: pcb_v1.PcbDoc	DWG NO:	SCALE: 1:1

LIITE C: Osasijoittelukuvat

Yläpuoli



Alapuoli



Altium Limited 12a Rodborough Rd Frenchs Forest NSW 2086	ENGINEER: Aapo Manni	TITLE: Arduino FM-radio	
	PCB DESIGNER: Aapo Manni		
	DATE: 18.4.2025	PART NO.:	REV: 1
	FILE NAME: pcb_v1.PcbDoc	DWG NO:	SCALE: 1:1

```
#include <LiquidCrystal.h>
#include <Si4703_Breakout.h>
#include <Wire.h>

//Aapo Manni 2025
//Arduino-pohjainen koodi Si4703-kehitysalustan ohjaamiseen.

int resetPinni = 2;
int SDIO = A4;
int SCLK = A5;

Si4703_Breakout radio(resetPinni, SDIO, SCLK);
int kanava = 937; //aloituskanava
int volume;

//Kytkimiä vastaavien pinnien määritys
const int seek_up_pinni = 8;
const int seek_down_pinni = 9;
const int fine_tune_pinni = 10;


//Pinnien määritys LCD:tä varten.
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


void setup()
{
  //Si4703 alustus
  radio.powerOn();

  //Aloituskanavan asettaminen
  radio.setChannel(937);

  //LCD:n alustus
  lcd.begin(16,2);

}


//Palauttaa kokonaisluvun välillä 0-15 riippuen siitä mille
//tasolle äänenvoimakkuus halutaan asettaa
int lue_volume(){

  //Lukee potentiometrin arvon.
  int vol = analogRead(A1);

  //Skaalaa luvun välille 0-15.
  volume = (vol-22)/61.5;

  //Määrittää suurimmaksi arvoksi 15.
  if (volume >= 16){
    volume = 15;
  }

  //Asettaa radion äänenvoimakkuuden
  radio.setVolume(volume);
  }


//Pääfunktio
void loop()
{
  //Lukee potentiometrin arvon ja asettaa äänenvoimakkuuden
  lue_volume();

  //Odottaa, että SEEK UP-nappia painetaan
  if (digitalRead(8)){
    kanava = radio.seekUp();
  }

  //Odottaa, että SEEK DOWN-nappia painetaan
  if (digitalRead(9)){
```

```
      kanava = radio.seekDown();
    }

    //Odottaa, että FINE TUNE-nappia painetaan
    if (digitalRead(10)){
      kanava = kanava+1;
      radio.setChannel(kanava);
    }

    //Rajaa taajuudet välille 87,5-108 MHz
    if (kanava == 0){
      kanava = 875;
    }

    //Päivittää LCD-näytön arvon
    naytto();

    //200 ms viive toiminnan varmistamiseksi
    delay(200);
  }

  //Tämä funktio päivittää LCD-näytön tietoja.
  void naytto()
  {
    //Muuntaa kanavan arvon desimaaliksi. Esim. 937 -> 93.7
    float desimaali = kanava / 10.0;

    //Asettaa soitettavan taajuuden LCD:n ensimmäiselle riville.
    lcd.setCursor(0, 0);
    lcd.print("FREQ: ");
    lcd.print(desimaali, 1);
    lcd.print(" MHz ");

    //Asettaa äänenvoimakkuuden toiselle LCD:n riville.
    lcd.setCursor(0, 1);
    lcd.print("VOLUME: ");


    //Jos äänenvoimakkuus saavuttaa maksimin se tulostetaan MAX LCD:lle.
    if (volume == 15){
      lcd.print("MAX");
      return;
    }

    //Jos äänenvoimakkuus saavuttaa minimin se tulostetaan MIN LCD:lle.
    if (volume == 0){
      lcd.print("MIN");
      return;
    }

    // Muissa tilanteissa tulostetaan äänenvoimakkuus numeroarvona.
    lcd.print(volume);
    lcd.print("   ");

  }
```