TAMPEREEN TEKNILLINEN YLIOPISTO

**TAPIO AALTONEN**

# WAPPURADION ÄÄNENSIIRTOLINKKI IP-VERKON YLI

Diplomityö

Tarkastaja: Tommi Mikkonen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-
neuvoston kokouksessa 11.1.2012

# TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
**AALTONEN, TAPIO:** Wappuradion äänensiirtolinkki IP-verkon yli
Diplomityö, 45 sivua
Tammikuu 2014
Pääaine: Ohjelmistotuotanto
Tarkastaja: Tommi Mikkonen
Avainsanat: äänensiirto, wappuradio

Rakkauden Wappuradio on tampereen teekkarien paikallisradioprojekti, jonka puitteissa on lähetetty ohjelmaa vuosittain teekkariwapun yhteydessä noin viikon ajan. Ohjelma on ollut kuultavissa yleisradiotaajuuksilla Tampereella sekä MP3-virtana Internetissä. Radio on saavuttanut suuren suosion opiskelijoiden keskuudessa, ja sen innoittamina on aloitettu wappuradiolähetyksien teko monella muullakin Suomen suurella opiskelijapaikkakunnilla. Rakkauden Wappuradion studio on sijainnut Tampereen teknillisen yliopiston kampuksella, ja FM-lähetin Hervannan vesitornissa. Näiden välinen ohjelmansiirto on toteutettu aiemmilla lähetyskerroilla erilaisilla IP-pohjaisilla siirtojärjestelmillä.

Aiempiin äänensiirtoratkaisuihin on liittynyt jonkin verran ongelmia. Ne ovat pohjautuneet yleiskäyttöisiin tietokoneisiin ja niillä ajettaviin ohjelmiin. Nämä ratkaisut ovat osoittautuneet virhealttiiksi tai niihin tarvittavien laitteistokomponenttien saatavuus mahdollisimman vähäisin kustannuksin hankalaksi. Järjestelmien aiheuttamat usean sekunnin viiveet ääneen ovat hämmentäneet kuulijoita, ja vaikeuttaneet interaktiivisen radio-ohjelman tekoa. Lisäksi aiemmat järjestelmät ovat edellyttäneet Rakkauden Wappuradion teknisen henkilöstön jatkuvaa läsnäoloa ja valmiutta selvittää järjestelmän virhetilanteita.

Tässä työssä käydään läpi aiemmat ratkaisut ja niihin liittyvät ongelmat. Niiden perusteella muodostetaan vaatimukset sulautetulle äänensiirtojärjestelmälle, jota voidaan jatkossa käyttää Rakkauden Wappuradion ohjelmansiirtoon. Järjestelmä korjaa kaikki merkittävät aiempien ratkaisujen ongelmat, ja tarjoaa helppokäyttöisen ja luotettavan tavan siirtää ääntä IP-verkon yli.

Uuden järjestelmän suunnittelussa ja toteutuksessa onnistuttiin hyvin, ja järjestelmä täytti kaikki sille asetetut vaatimukset. Ääneen ei aiheutunut siirrosta juurikaan viivettä, ja järjestelmä toimi täysin automaattisesti. Sitä käytettiin Rakkauden Wappuradion ohjelmansiirrossa vuoden 2012 lähetyksessä, ja se toimi moitteetta.

# ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
**AALTONEN, TAPIO:** Audio over IP system for Wappuradio
Master of Science Thesis, 45 pages
January 2014
Major: Software engineering
Examiner: Tommi Mikkonen
Keywords: audio over ip, wappuradio

Rakkauden Wappuradio is a broadcast radio station project by students of Tampere University of Technology (TUT). A weeklong programme has been broadcasted during wappu festival using FM transmitter and MP3 stream on the Internet. The station has gained huge popularity among students, and many other student radio station projects in other Finnish university cities have been inspired by it. The studio of Rakkauden Wappuradio has been at campus of TUT, and FM transmitter has been placed at Hervanta water tower for optimal coverage. Programme has been transmitted between the sites by different audio over IP systems.

The audio over IP systems previously used in Rakkauden Wappuradio have suffered from somewhat subpar performance. Audio delay introduced by them has been at least multiple seconds, sometimes even dozens of seconds. The delay has been confusing for those who attempt to join some of the interactive programs of the station. The older systems have also been either somewhat unreliable or based on general-purpose desktop computer equipment which can be hard to get for the low budget student project. In addition a technical duty officer has been required to stay at the radio studio at all times and fix manually any faults in the system.

In this thesis previous audio over IP systems of Rakkauden Wappuradio are examined and requirements for a new embedded system are formed. The goal is to design and build a system which can be used by the project in the future. The new system must mitigate the issues in older systems and provide an easy and reliable way to transfer radio programme from the studio to the FM transmitter site.

Creation of the new audio over IP system for Rakkauden Wappuradio was successfull, and the system fullfilled all of its requirements. The audio transfer was almost with no delay, and the system was fully automatical. It was deployed in Rakkauden Wappuradio 2012, and it worked with great success.

# ALKUSANAT

Rakkauden Wappuradio on osoitus siitä, kuinka pienikin ryhmä voi innokkaalla yhteistoiminnalla saada aikaan paljon hyvää mieltä suurelle joukolle. Se on myös osoitus siitä, että hauskanpidon nimissä tehdystä vapaa-ajan aktiviteetista voi seurata mitä opettavaisin ja mielenkiintoisin opinnäytetyöprojekti, jossa pääsee käyttämään kaiken osaamisensa ja syventämään sitä aivan uudelle tasolle.

Haluan kiittää professori Tommi Mikkosta diplomityön tarkastamisesta ja ohjaamisesta sekä diplomi-insinööri Eero Alkkiomäkeä ja Baytems Holdings Oy:tä laitteen suunnittelusta ja koekappaleiden valmistamisesta. Lisäksi haluan kiittää Tampereen teekkarien radiokerhon jäseniä, erityisesti Joel Salmea ja Kimmo Jukaraista, tuesta ja avusta projektin aikana sekä Marjo Yli-Paavolaa oikoluvusta.

Tampereella 12. joulukuuta 2013

Tapio Aaltonen

# SISÄLLYS

# 1 JOHDANTO

Rakkauden Wappuradio on tamperelaisten teekkarien jo perinteeksi muodostunut tempausluontoinen projekti, jossa teekkariwapun ajaksi perustetaan opiskelijavoimin oma paikallisradioasema. Asema lähettää ohjelmaa noin viikon ajan yleisradiotaajuuksilla Tampereen alueella sekä Internetissä. Radioasema oli äänessä ensimmäisen kerran vuonna 2010, ja sen saavuttaman suuren suosion innoittamina projekti toteutettiin uudelleen myös seuraavana vuonna. Radio on saavuttanut Internet-lähetyksen ansiosta muidenkin kuin tamperelaisten opiskelijoiden keskuudessa suuren suosion, ja muidenkin suurten opiskelijapaikkakuntien teekkarit ovat perustaneet Rakkauden Wappuradion innoittamina omia wapun ajan paikallisradioasemiaan.

Tamperelaisten Rakkauden Wappuradion asema on sijainnut Tampereen teknillisen yliopiston kampuksella työmaaparakissa, jossa se on ollut keskeisellä paikalla helposti lähestyttävissä. FM-lähetin on ollut Tampereen teekkarien radiokerhon radiokilpailuasemalla Hervannan vesitornissa, josta on saavutettu hyvä kuuluvuus Hervannan ja Tampereen keskustan alueilla. Monilla autoradioilla ja muilla hyvillä ulkoisilla antenneilla varustetuilla vastaanottimilla Rakkauden Wappuradiota on kuunneltu jopa kymmenien kilometrien päässä Hervannasta. Radio-ohjelma on siirretty studiolta lähettimelle internet-yhteyttä pitkin, ja siirtoon on käytetty eri vuosina erilaisia ratkaisuja. Vuonna 2010 ääni tuotiin FM-lähettimelle soittamalla sille julkisesta Internet-lähetyksestä eriytettyä korkealaatuista MP3-ohjelmavirtaa. Lähettimen viereen sijoitettuun tietokoneeseen ei kuitenkaan etsinnöistä huolimatta löydetty MP3-virran soitto-ohjelmaa, joka olisi toiminut ongelmitta koko lähetyksen ajan. Testeissä parhaaksi osoittautunut ja tuotantokäyttöön valittu iTunes-ohjelmakin keskeytti soiton kahdesti lähetyksen aikana. Vuonna 2011 siirtoyhteys rakennettiin itse tehdyllä ohjelmalla, jota käytettiin yhteyden molemmissa päissä. Ohjelma saatiin toimimaan luotettavasti käyttämällä sen pohjana Applen Mac OS X -käyttöjärjestelmää ja lainaan saatuja Applen tietokoneita. Ohjelmansiirto onnistui hyvin, mutta sen aiheuttamaa viiden sekunnin siirtoviivettä pidettiin pitkänä ja tarvittujen laitteiden saantia projektin käyttöön jatkossa mahdollisesti ongelmallisena.

Tässä diplomityössä esitellään aiemmat äänensiirtoyhteydet ja toteutetaan niistä kertyneiden kokemusten pohjalta uusi sulautettu järjestelmä vuoden 2012 Rakkauden Wappuradion äänensiirtoa varten. Keskeisimmiksi tavoitteiksi asetetaan ai-

empaa lyhyempi siirtoviive ja yleiskäyttöisiä tietokoneita edullisempi ja yksinkertaisempi rakenne. Tavoitteena on myös tehdä järjestelmästä mahdollisimman automaattinen siten, että se ei vaadi käyttäjän toimenpiteitä toipuakseen häiriöistä, eli esimerkiksi käynnistyäkseen sähkökatkon jälkeen. Siirtoyhteyden vastaanottavan pään laitteeseen vaaditaan oheistoimintona automaattinen ennalta asetetun varatallenteen soitto, mikäli siirtoyhteys verkon yli katkeaa.

Tämä diplomityö on jaettu viiteen lukuun. Rakkauden Wappuradio ja sen aiemmat äänensiirtoyhteydet esitellään perusteellisesti luvussa kaksi. Luvussa kolme esitellään aiemmista järjestelmistä kertyneiden kokemusten perusteella muodostetut vaatimukset tässä työssä laadittavalle uudelle sulautetulle äänensiirtojärjestelmälle, ja esitetään sen suunnittelu ja toteutus. Uuden järjestelmän vaatimustenmukaisuutta ja soveltuvuutta Rakkauden Wappuradion tarpeisiin käsitellään luvussa neljä. Luvussa viisi tehdään lyhyt yhteenveto työstä.

# 2 LÄHTÖKOHDAT

Tässä luvussa esitellään Rakkauden Wappuradio, sen äänensiirtotarpeet ja aiemmat sitä varten kehitetyt äänensiirtoyhteysratkaisut. Niiden pohjalta muodostetaan vaatimukset tässä työssä toteutettavalle äänensiirtojärjestelmälle.

## 2.1 Rakkauden Wappuradio

Rakkauden Wappuradio on vuonna 2010 syntynyt Tampereen teekkarien paikallisradioprojekti. Sen alkuperäinen tavoite oli pitää teekkareiden omaa radioasemaa noin viikon ajan vuoden 2010 Tamperelaisen teekkariwapun [1] aikaan. Toimitettua ohjelmaa tehtiin päivisin puolesta päivästä puoleen yöhön, ja yöllä lähetettiin päivän ohjelmat uusintana. Lähetys oli kuultavissa radioteitse Tampereen alueella taajuudella sekä Internetin välityksellä ympäri maailman. Radioaseman kohdeyleisöksi valittiin erityisesti tulevat, nykyiset ja jo valmistuneet yliopiston opiskelijat, joille radio toi mahdollisuuden päästä wapputunnelmaan paikasta ja ajasta riippumatta. Selkeästä kohderyhmän valinnasta huolimatta ohjelmasisältö pyrittiin mahdollisuuksien mukaan pitämään myös muita kuin teekkareita kiinnostavana. Toiminnan keskeisenä periaatteena oli mielenkiintoisten uusien asioiden oppiminen yhdessä tekemällä ja hauskanpitoa unohtamatta. Kaupallisuutta pyrittiin välttämään, joten radioon myytiin ja tehtiin mainoksia vain välttämättömien kulujen kattamiseksi. Radioaseman nimeksi valittiin Rakkauden Wappuradio näitä tavoitteita korostamaan. Tavoitteisiin päästiin, ja radiosta tuli menestys. Sitä kehuttiin teekkareille tyypilliseksi tempaukseksi, jossa pieni joukko tuottaa vapaaehtoistyöllä paljon hyvää mieltä suurelle joukolle teekkareita ja erityisesti myös muita ihmisiä. Projekti oli ensimmäinen Pitkän Piipun Lupin perustaman ja myöhemmin Tampereen teknillisen yliopiston alumniyhdistys ry:n jaettavaksi siirtyneen Vuoden teekkariteko -palkinnon saaja. Radioprojektista tuli perinne, ja se on toteutettu uudelleen vuosittain. Perinnettä ylläpitämään sekä tietoa tekijäsukupolvelta toiselle välittämään perustettiin vuoden 2010 syksyllä Wappuradion tuki ry.

Vuonna 2010 Rakkauden Wappuradion projektiryhmä koostui Tampereen teekkarien perinneseuran ja Tampereen teekkarien radiokerhon jäsenistä. He kokoontuivat ensimmäisen kerran vuoden 2010 tammikuussa sittemmin päätoimittajaksi valitun Tomi Miikkulaisen koolle kutsumana pohtimaan oman lyhytaikaisen paikallisradioaseman tekemistä. Radiolähetyksen kuuluvuusalueeksi tavoiteltiin Tampereen Her-

vantaa, sekä muuta Tamperetta mahdollisuuksien mukaan. Lähetystä haluttiin jaella myös Internetin kautta, jolloin se olisi kuultavissa ympäri maailma. Radiokerhon radiokilpailuasema Hervannan vesitornissa oivallettiin hyväksi lähettimen sijoituspaikaksi, josta toivottu kuuluvuus radioaalloilla olisi mahdollista saavuttaa. Radion studioksi kaavailtiin aluksi matkailuvaunua, jolla olisi voitu liikkua wapun ajan ympäri kaupunkia teekkareiden tapahtumissa tekemässä niistä ohjelmaa. Vaunustudio tulisi vaatimaan toimiakseen sähköliittymän, joka arveltiin voitavan moniin paikkoihin järjestää. Lisäksi tuotettu ohjelmasignaali piti saada siirrettyä lähettimelle, missä puolestaan nähtiin ongelmia. Radiokerholaiset selvittivät vaihtoehtoja, ja esille nousivat ammattimaisten radioasemien käyttämät suorat radiolinkkiyhteydet sekä internetin avulla tapahtuva siirto. Radiolinkin arveltiin olevan melko vaikea saada viideksi vuorokaudeksi käyttöön niin pienin kustannuksin, että se olisi radion budjetin kannalta järkevää. Lisäksi radiolinkki olisi todennäköisesti ollut vaikea asentaa, ja ennalta tuntemattomana laitteistona mahdollisesti myös vaikea saada toimimaan luotettavasti. Näiden seikkojen valossa päätettiin äänensiirto toteuttaa internetin välityksellä. Studiolle tarvittiin näin myös varmatoiminen internet-yhteys, jonka järjestäminen liikkuvaan vaunuun todettiin projektin käytössä olevin voimavaroin lähes mahdottomaksi. Matkapuhelinverkkoja ei voitu tiedonsiirtoon käyttää, sillä ne olisivat ruuhkaisia paljon yleisöä keräävissä tapahtumissa, eikä yhteys olisi luotettava. Kaapeliyhteys olisi mahdollisesti saatavilla joihinkin kohteisiin, mutta edellyttäisi sielläkin suuren määrän ennakkovalmisteluita. Ajatus helposti liikuteltavasta studiosta jouduttiin näin hylkäämään. Tilalle kehiteltiin ajatus studiosta työmaaparakissa, joka voitaisiin sijoittaa Tampereen teknillisen yliopiston (TTY) kampukselle hyvien infrastruktuuripalveluiden viereen. Yliopiston hallinto, kuten monet muutkin yhteistyötahot, suhtautuivat ajatukseen myönteisesti. Projekti lähti liikkeelle, ja sille saatiin lainaan parakki ja lupa sen sijoittamiseen yliopiston kampukselle Tietotalon edustalle. Parakille saatiin myös kolmivaihesähkö ja kiinteä internet-yhteys yliopiston kautta. Viestintävirasto myönsi radioluvan, joka salli käyttää Hervannan vesitornissa 200 watin lähetystehoa taajuudella 101,6 MHz. Sillä lähetyksen arveltiin kuuluvan hyvin koko Hervannassa, ja Tampereen keskustassakin ainakin ulkoantennilla varustetuilla vastaanottimilla. Näin kävikin, ja parhaimmillaan Rakkauden Wappuradiota kuunneltiin autovastaanottimilla useiden kymmenien kilometrien päässä.

Vuonna 2011 radion puitteet olivat pääpiirteissään samat kuin ensimmäisellä kerralla, mutta pientä kehittymistä nähtiin monella saralla. Radion tekijöitä haettiin TTY:n piiristä avoimella haulla, ja mukaan saatiinkin useita uusia radioääniä. Ohjelmaa tehtiin ensimmäistä yötä lukuun ottamatta vuorokauden ympäri, ja lähetys kesti kokonaisuudessaan kuusi vuorokautta. Ensimmäisenä vuotena täynnä olleen studioparakin tilanahtautta helpottamaan hankittiin toinen työmaaparakki, joka sijoitettiin studiona toimineen parakin viereen tekniseksi tilaksi, taukotuvaksi ja ra-

dion tuotteiden myyntipisteeksi. Lähetys siirrettiin studioparakista analogista balansoitua linjaa pitkin tekniseen tilaan, jonne lähetystekniikka sijoitettiin. Näin saatiin vähennettyä studiossa olleiden laitteiden määrää, ja sen myötä vaimennettua ajoittain lähetykseenkin päässyttä studion taustahuminaa. Lähetystä kuunneltiin jälleen laajasti, ja palautteen sekä kertyneiden tilastojen perusteella arveltiin kuulijamäärän kasvaneen edellisestä vuodesta.

Tuore Tamperelainen teekkariperinne sai jatkoa jälleen vuonna 2012, kun Wappuradion tuki ry:n johdolla toteutettiin kaikkien aikojen pisin, kahdeksan vuorokautta yhtäjaksoista ohjelmaa käsittänyt Rakkauden Wappuradio. Mukaan lähtivät myös Oulun teekkarien radiokerhon aktiivit, jotka välittivät Tampereella tehtyä lähetystä radioaalloille Oulussa. Ouluun lähetystaajuudeksi myönnettiin 98,1 MHz, ja Tampereelle aiemmista vuosista poiketen 106,8 MHz. Radiossa kuultiin myös muutamia Oululaisten toimittamia Oulun teekkarikulttuuria esitteleviä ohjelmanpätkiä. Myös Aalto-yliopiston teekkareiden Rakkauden Wappuradion menestyksen innoittamina perustama Radio Wappu välitti ajoittain tamperelaisten ohjelmaa.

Wappuradion tuki ry kokosi vuoden 2012 projektin päätöstilaisuutta varten Rakkauden Wappuradion tunnuslukuja vuosien varrelta. Luvut on esitetty taulukossa 2.1. Ne on kerätty useista lähteistä, eivätkä ole tarkkoja. Niistä kuitenkin ilmi projektin ja sen suosion kasvu vuodesta toiseen. Näin huolimatta siitä, että vuonna 2012 aloittivat teekkariwapun ystävien viihdyttämisen myös Otaniemeläisten teekkareiden Radio Wappu ja Lappeenrantalaisten kaksi viikkoa Internet-lähetystä lähettänyt Norpparadio. Muiden paikkakuntien opiskelijoiden innostamisen ja tamperelaisten kasvulukujen valossa voidaankin puhua tamperelaisten teekkareiden käynnistämästä wappuradioilmiöstä.

***Taulukko 2.1.*** *Rakkauden Wappuradio lukuina*

| | 2010 | 2011 | 2012 |
|---|---|---|---|
| Radion kesto (vrk) | 5 | 6 | 8 |
| Lähetyksen kesto (h) | 104 | 128 | 180 |
| Suoraa lähetystä (h) | 64 | 121 | 180 |
| Internet-kuulijoita keskim. | 250 | 300 | 450 |
| Facebook-faneja | 1300 | 1700 | 2200 |
| Verkkosivuilla kävijöitä | 7000 | 9200 | 17000 |

## 2.2 Aiemmat äänensiirtoyhteydet

Seuraavassa esitellään yksityiskohtaisesti Rakkauden Wappuradion äänensiirtoyhteydet studiolta lähettimelle vuosina 2010 ja 2011. Niiden perusteella hahmotellaan vaatimukset vuoden 2012 äänensiirtojärjestelmille.

### 2.2.1 Vuosi 2010

Vuoden 2010 Rakkauden Wappuradiota varten lähetystekniikkaa suunnitelleiden radiokerholaisten tavoitteena oli löytää mahdollisimman edullisia ja yksinkertaisia ratkaisuja äänen siirtoon, jossa siirto aiheuttaisi ääneen mahdollisimman vähän kuultavissa olevia muutoksia. Siirtoyhteyksiltä vaadittiin myös luotettavuutta, sillä äänen piti siirtyä yhtäjaksoisesti ja katkeamatta koko viiden vuorokauden mittaisen lähetysjakson ajan.

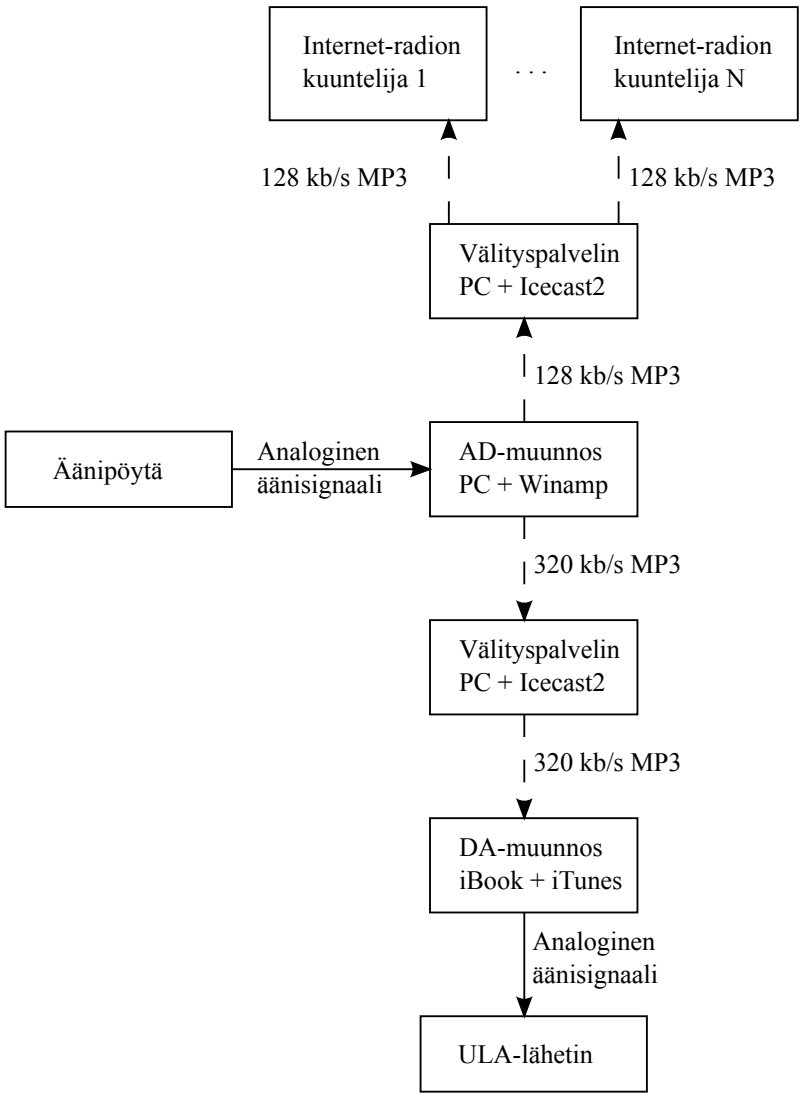
Internet-radion tekemisestä oli radiokerholaisilla aiempaa kokemusta. Sen pohjalta toteutettiin ensin järjestelmä Rakkauden Wappuradion Internet-lähetystä varten. Se koostui kahdesta osasta, studiolla tuotettua ohjelmasignaalia tallentavasta laitteesta ja nopean ja kuormitusta kestävän internet-yhteyden päässä olevasta jakelupalvelimesta. Palvelimen järjestämistä TTY:n kampukselle suunniteltiin, mutta lopulta valittiin kaupallisin periaattein toimivan yhteistyökumppanin tarjoama ratkaisu. Syynä olivat radioon kaavaillut mainokset, sekä Tampereen teknillisen yliopiston tietoverkon ja Suomen korkeakoulujen ja tutkimuksen tietoverkon (Finnish University and Research Network, FUNET) käyttösäännöt, jotka kielsivät kaupallisen toiminnan kyseisissä verkoissa. Äänensiirto studiolta jakelupalvelimelle päätettiin toteuttaa pienellä PC-tietokoneella, johon oli kytketty riittävän laadukkaaksi arveltu Sound Blaster Live -äänikortti. Ohjelmistoksi valittiin aiempien kokemusten perusteella vakaasti toimivaksi havaittu Winamp [2] -mediasoitin SHOUTcast DSP -nettiradiolisäkkeellä [3] varustettuna. Tällä kokoonpanolla ääni muunnettiin digitaaliseksi, pakattiin MP3-virraksi ja syötettiin jakelupalvelimelle. MP3-virran nimellisnopeudeksi valittiin 128 kbit/s, koska sen arveltiin tarjoavan riittävän äänenlaadun ollen kuitenkin samaan aikaan välitettävissä myös kolmannen sukupolven matkapuhelinverkkojen kautta. Palvelimella MP3-virran monistamiseen ja jakeluun käytettiin Icecast2-ohjelmaa [4]. Vaihtoehtoisten laite- tai ohjelmistokomponenttien harkitsemiseen tai kokeiluun ei käytetty aikaa valitusta kokoonpanosta olleiden aiempien hyvien kokemusten vuoksi.

Yksinkertaisimmaksi ratkaisuksi studion ja lähettimen väliseen äänensiirtoon havaittiin nopeasti Internet-radion soittaminen suoraan lähettimelle. Julkisen Internet-radion jakelupalvelin sijaitsi kuitenkin verkkoteknisesti kaukana, FUNET-verkon ulkopuolella, eikä studion ja lähettimen välisen äänensiirron haluttu riippuvan julkisen jakelupalvelimen toiminnasta. Ongelma ratkaistiin asentamalla TTY:n verkossa sijaitsevalle radiokerhon palvelimelle erillinen julkisista verkoista eristetty Icecast2-instanssi, jonka kautta ääni välitettäisiin vain lähettimelle. Lähettimen sijoituspaikassa radiokerhon asematilassa Hervannan vesitornissa oli käytettävissä 768 kbit/s nopeuteen pystyvä internet-yhteys, joten lähettimen ohjelmansiirtoon käytetylle välityspalvelimelle suunnatun MP3-virran nopeus voitiin nostaa äänenlaadun maksi-

moimiseksi suurimpaan MP3-pakkausohjelman tukemaan arvoon, joka oli 320 kbit/s. Asematilaan suunniteltiin sijoitettavaksi PC, joka vastaanottaisi tätä korkealaatuista MP3-virtaa ja soittaisi sitä lähettimen sisäänmenoon.

Suunnitelmaa ryhdyttiin kokeilemaan käytännössä lähetystä edeltäneellä viikolla. Järjestelmän osat toimivat testeissä hyvin MP3-virtaa lähettimelle soittavaa tietokonetta lukuun ottamatta. Kyseiselle järjestelmän osalle ei suunniteltaessa oltu laitettu suurta painoarvoa, vaan ajateltiin sen muodostuvan yksinkertaisesti Linux-PC:stä, laadukkaasta äänikortista ja MP3-virran soitto-ohjelmasta. Kyseessä ajateltiin olevan hyvin yleinen tietokoneen käyttötapaus, johon liittyvät ongelmat, kuten mahdolliset ohjelmavirheet, olisivat tulleet aikojen saatossa jo monesti esiin ja siten myös korjatuksi. Käytännön kokeet kuitenkin osoittivat, että vaikka laitteisto vaikutti toimivan ongelmitta ja moni ohjelma osasikin soittaa ääntä verkosta, yksikään niistä ei tulisi selviämään siitä viikkoa yhtäjaksoisesti. Kokeiltuja ohjelmia olivat muun muassa VLC [5], MPlayer [6] ja mpg321 [7]. Jokainen näistä oli kaatunut tai tullut käyttöjärjestelmän pysäyttämäksi muistinkäsittelyn virheen takia viimeistään vuorokauden kuluttua käynnistämisestään. Lähetyksen aloittamishetken lähestyessä uhkaavasti aikaa vaikeasti toistettavissa olevien ohjelmavirheiden paikallistamiseen ja korjaamiseen ei ollut, vaan tilalle päätettiin kokeilla Applen iBook -tietokonetta ja saman valmistajan iTunes-ohjelmaa. Yhdistelmää ei ehditty kokeilemaan kuin muutama vuorokausi, mutta se toimi ongelmitta koko testin ajan ollen selvästi paras käytettävissä olevista vaihtoehdoista. MP3-virran soittamien lähettimelle päätettiin toteuttaa tällä kokoonpanolla. Mahdollisten yhteyskatkosten varalta iTunesiin määriteltiin päättymätön soittolista, jossa verkon yli siirrettävän ohjelmavirran katketessa soitettiin siirtoyhteyden katkeamisesta kertova tiedote ja radion kanavatunnus. Näiden jälkeen yritettiin jälleen jatkaa MP3-virran soittoa. Lopullinen tuotantokäytössä ollut siirtoyhteysratkaisu on esitetty kuvassa 2.1.

Kaikkien aikojen ensimmäinen Tampereen teekkarien voimin toteutettu wappuradiolähetys käynnistyi maanantaina 26.4.2010 kello 10. Siirtoyhteys studiolta lähettimelle toimi hyvin, mutta lähetyksen edetessä lähettimelle menevä ääni jäi hitaasti jälkeen studiolta tulevasta äänestä. Aloitettaessa siirtoyhteyden viive oli muutamia sekunteja, mutta kahden vuorokauden jälkeen se oli jo yli viisitoista sekuntia. Viiveen syyksi arveltiin erilaista näytteenottotaajuutta nauhoittavalla ja toistavalla äänikortilla. Kortit olivat täysin erilaisia, sillä tallentava äänikortti oli PCI-liitäntäinen Creative SB Live, toistava puolestaan IEEE 1394 (FireWire) -liitäntäinen Swissonic Easy FireWire.




***Kuva 2.1.*** *Ohjelmansiirtoyhteydet vuonna 2010*

Olettamalla, että analogisesta äänestä otettujen näytteiden määrä $N_i$ oli sama kuin soitettujen näytteiden määrä $N_o$, ja että toistavan pään näytteenottotaajuus on asetetun arvon mukainen, saatiin tallentavan pään näytteenottotaajuuden virhe määritettyä seuraavasti.

$$N_i = N_o$$

$$(F_s + \Delta f)t = F_s(t + \Delta t)$$

$$\Delta f = \frac{F_s(t + \Delta t)}{(t - F_s)} \tag{2.1}$$

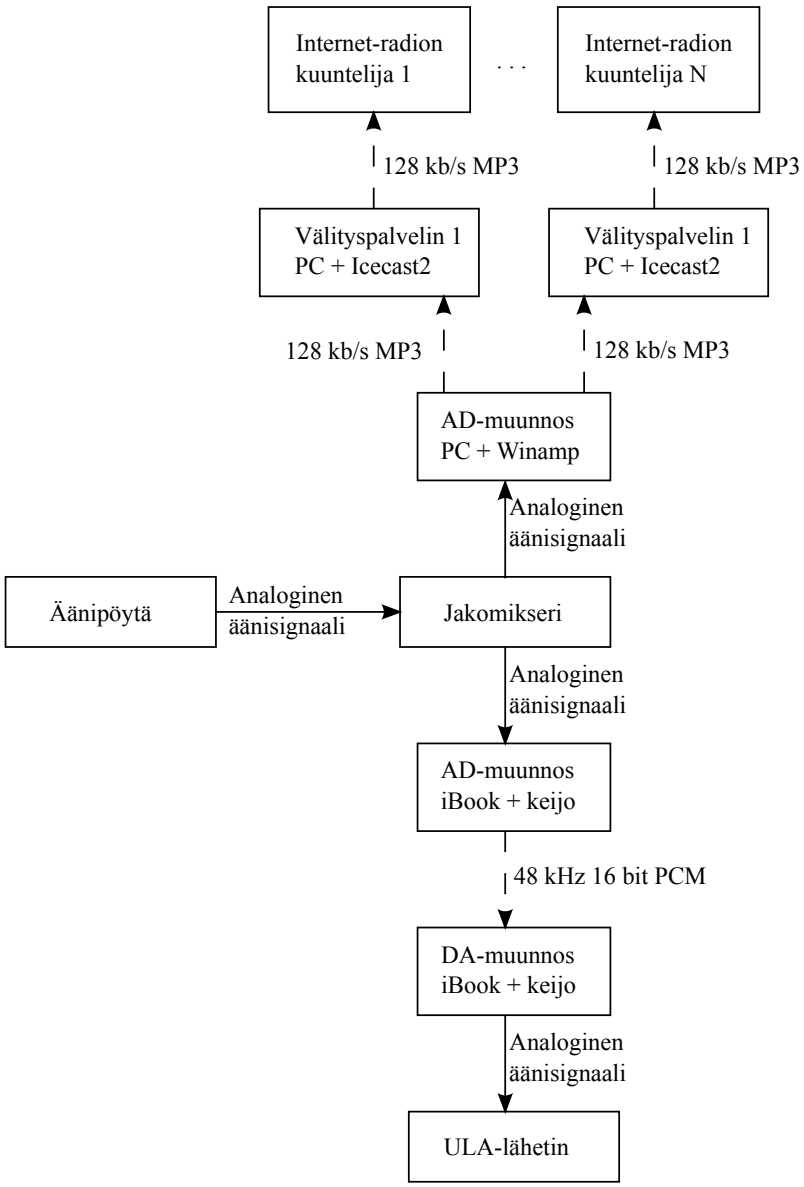
Sijoittamalla yhtälöön 2.1 oletettu näytteenottotaajuus $F_s$ 44100 Hz, lähetysaika $t$ 48 tuntia ja toistavan pään viive $\Delta t$ 15 sekuntia, saatiin tallentavan äänikortin näytteenottotaajuuden virheeksi $\Delta f$ 3,8 Hz. Taajuuden virhe olisi tässä tapauksessa 86 PPM, minkä arveltiin olevan kuluttajakäyttöön tarkoitettujen tuotteiden kyseessä ollessa hyvin mahdollista. Virhe ei kuitenkaan välttämättä johtunut pelkästään

yhdestä äänikortista, vaan pidettiin mahdollisena, että molempien äänikorttien taajuudessa saattaisi olla virhettä. Lisäksi toistettavien näytteiden määrä ei ehkä ollut sama kuin tallennettujen näytteiden määrä, sillä digitaalinen ääni pakattiin siirron ajaksi MP3-virraksi, jonka pakkaus- ja purkuohjelmat saattoivat käsitellä näytteitä keskenään eri tavoin. Laskelman ja päätelmien pohjalta todettiin, että siirtojärjestelmässä on virhe, jonka perimmäistä syytä ei lähetyksen aikana ryhdytä käytännön toimilla etsimään tai korjaamaan. Virheen tuottamaa viivettä voitaisiin tarvittaessa lyhentää katkaisemalla toisto viiveen mittaiseksi ajaksi esimerkiksi aamuyöllä tai muuten sellaisella tavalla, että se haittaa kuulijoita mahdollisimman vähän.

Noin 39 tunnin toiminnan jälkeen keskiviikkona kello 1.16 tapahtui ensimmäinen lähetyksen odottamaton katkeaminen. Syylliseksi paljastui lähetintä syöttävällä koneella siirtovirtaa soittanut iTunes-ohjelma, joka oli tuntemattomaksi jääneestä syystä siirtynyt soittotilasta taukotilaan. Ongelma saatiin korjattua muodostamalla etäyhteys koneelle ja antamalla soitto-ohjelmalle komentoriviltä play-komento. Samassa yhteydessä play-komento ajastettiin annettavaksi automaattisesti minuutin välein, jotta vastaavan ongelman toistuessa toipuminen tapahtuisi automaattisesti. Siirtoyhteyden viive lyhentyi katkon yhteydessä muutamaan sekuntiin, mutta alkoi jälleen kasvaa samalla nopeudella kuin ennen katkoa. Toinen odottamaton katko tapahtui lauantaina kello 1.42, mutta silloin edellisen katkon yhteydessä tehty automatiikka sai soiton jatkumaan alle minuutissa. Useampia katkoja tai muitakaan ongelmia ei lähetyksen aikana tapahtunut, joten kokonaisuutena arvioiden siirtoyhteyttä pidettiin onnistuneena.

### 2.2.2 Vuosi 2011

Vuoden 2011 Wappuradion tekniikan suunnittelu ja testaaminen aloitettiin ensimmäistä vuotta aiemmin, useampi kuukausi ennen lähetysviikkoa. Tavoitteena oli rakentaa itse ohjelma, jonka yksi instanssi nauhoittaisi stereoääntä ja lähettäisi sitä verkon kautta toiselle instanssille ulos soitettavaksi. Siirtoyhteys voitaisiin sen avulla toteuttaa tarkoitusta varten pyhitetyllä laitteistolla, jolloin antenniverkon radiolähetys ja Internet-radio olisivat mahdollisimman riippumattomia toisistaan. Siirtoyhteyssuunnitelma on esitetty kuvassa 2.2.




***Kuva 2.2.*** *Ohjelmansiirtoyhteydet vuonna 2011*

Internet-jakelu suunniteltiin toteutettavan edellisvuoden periaattein, mutta nyt yhden sijasta kahdella jakelupalvelimella, jotta kapasiteettiä olisi aiempaa useammalle yhtäaikaiselle internet-kuuntelijalle. Oman, keijoksi ristityn komentorivipohjaisen äänensiirto-ohjelman toteutuksen rinnalla selvitettiin myös jo olemassa olevien kaupallisten äänensiirtotuotteiden soveltuvuutta Rakkauden Wappuradion tarpeisiin. Esillä olivat muun muassa Mayahin [8] ja APT:n [9] tuotteet, mutta pienen selvityksen jälkeen todettiin, että sekä osto- että vuokrahinnat olivat selkeästi liian suuria radion budjettiin nähden. Lisäksi edullisimmatkin tuotemallit sisälsivät hyvin runsaasti hienostuneita ja monimutkaisia ominaisuuksia [10], joten niiden käytön opettelu ja hallinta mahdollisissa ongelmatilanteissa olisi ollut haastavaa. Näistä syistä radiokerholaiset hylkäsivät valmiit tuotteet, ja jatkoivat kehitystyötä

oman äänensiirtoyhteyden parissa.

Äänensiirron toteuttavan ohjelman tärkein vaatimus oli luotettavuus, sillä edellisen vuoden siirtoyhteyskatkoksien ei haluttu toistuvan. Luotettavuuden tavoittelussa nähtiin parhaaksi keinoksi pitää toteutus niin yksinkertaisena kuin mahdollista. Tällöin mahdollisten virheiden löytäminen ja korjaaminen olisi vielä testausvaiheessakin helppoa. Ohjelman käyttöliittymään ei ollut tarvetta rakentaa interaktiota käyttäjän kanssa, mutta näytölle haluttiin tulostaa toiminnan seuraamista helpottavia tilatietoja. Radiokerhon asematilaan Hervannan vesitorniin tiedettiin olevan tulossa ennen radiolähetyksen alkua aiempaa nopeampi 2 Mbit/s kaistanleveyteen kykenevä internet-yhteys. Digitaalisen äänen bittinopeus määritettiin seuraavasti.

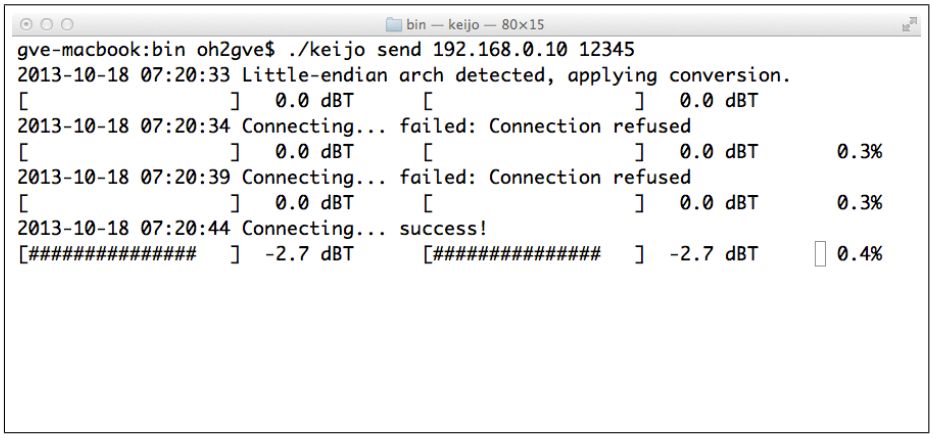
$$\text{Bittinopeus} = \text{Kanavien määrä} \times \text{Näytteen koko} \times \text{Näytteenottotaajuus} \quad (2.2)$$

Valitsemalla siirrettävän stereoäänen näytteenottotaajuudeksi 48 kHz ja näytteen kooksi 16 bittiä kanavaa kohti saatiin yhtälön 2.2 avulla bittinopeudeksi 1,536 Mbit/s. Tämä tulisi mahtumaan protokollakehystenkin kanssa uuden internet-yhteyden läpi ongelmitta, minkä ansiosta äänensiirto-ohjelmaan ei ollut tarpeen suunnitella lainkaan äänen pakkausta.

Toinen tärkeä vaatimus oli äänenlaadun säilyminen siirrossa. Laadun varmistamiseksi siirtokerros oli toteutettava siten, etteivät mahdolliset pakettikytkentäisen verkon lieveilmiöt kuten pakettien katoaminen tai niiden järjestyksen muuttuminen aiheuttaisi äänen kuultavissa olevia häiriöitä. Lähetyksen siirtoon käytettävän uuden internet-yhteyden laadusta näiltä osin ei ollut tietoja, joten erilaisiin häiriöihin oli varauduttava. Siirtokerroksen protokollaksi harkittiin UDP:tä ja TCP:tä, sillä ne olivat potentiaalisille ohjelman toteuttajille ennalta tuttuja. UDP:n etu olisi ollut lähes vakiomittainen ja lyhyt siirtoviive, mutta siirron virheettömyys olisi jouduttu takaamaan sovelluskerroksen ratkaisuilla. TCP:ssä siirtoyhteyden viive voisi olla ajoittain useita sekunteja, mutta protokollan sisäänrakennettu luotettavuus mahdollistaisi hyvin yksinkertaisen ohjelman toimintalogiikan. Siirtoviive olisi kuitenkin myös TCP:n tapauksessa pienempi kuin edellisen vuoden siirtoyhteysratkaisussa, joten TCP valittiin käyttöön. Siirtoviiveen vaihteluun varauduttiin suunnittelemalla ääntä toistavaan instanssiin puskuri, johon kerätään viiden sekunnin edestä äänidataa ennen toiston aloittamista. Mikäli puskuri menisi toiston aikana tyhjäksi, soitettaisiin ennalta tehtyä tallennetta koneen paikalliselta levyltä kunnes puskurissa olisi jälleen viiden sekunnin edestä äänidataa.

Kolmas vaatimus oli studiolta tulevan analogisen äänen voimakkuuden mittaaminen AD-muunnoksen yhteydessä. Mittarin avulla voitaisiin tarkkailla, että ääni ei säröydy AD-muunnoksessa liiallisen voimakkuuden takia. Samalla varmistuttaisiin siitä, että äänisignaali on kuitenkin tarpeeksi voimakas AD-muuntimen dynaamisen

alueen [11, s. 5907] täysimittaiseksi hyödyntämiseksi. Lähetystekniikkaa valvovat radiokerholaiset voisivat lisäksi mittarin avulla tukea radion studiossa olevaan tuottajaa, jonka tehtäviin äänenvoimakkuuden hallinta lukeutuu. Mittari päätettiin toteuttaa samoin kuin Internet-lähetyksen tekemiseen käytetyssä ohjelmistossa, jossa se näytti hetkellisen äänenvoimakkuuden desibeleinä suhteessa digitaalisen näytteen maksimiarvoon (desibeliä yli täyden voimakkuuden, dBT). Tämä oli toteutettavissa helposti lisäämällä ohjelmaan digitaalista ääntä lähettävässä instanssissa käytettäväksi koodilohko, joka etsi joka kymmenennen lähetettävän TCP-paketin hyötykuormasta itseisarvoltaan suurimmat näytearvot sekä vasemmalta että oikealta kanavalta, ja vertasi niitä 16-bittisen etumerkillisen luvun suurimpaan mahdolliseen itseisarvoon 32767 ja ottamalla osamäärästä kymmenkantaisen logaritmin. Saadun tuloksen perusteella päivitettiin ohjelman pääteikkunassa olevia lukuarvoja ja ristikkomerkein piirrettyjä ”mittarinäyttöjä”. Esimerkki pääteikkunan sisällöstä ääntä verkkoon lähettävän ohjelman käynnistyttyä nähdään kuvassa 2.3.




***Kuva 2.3.*** *Ohjelman käyttöliittymä siirtoyhteyden lähettävässä päässä*

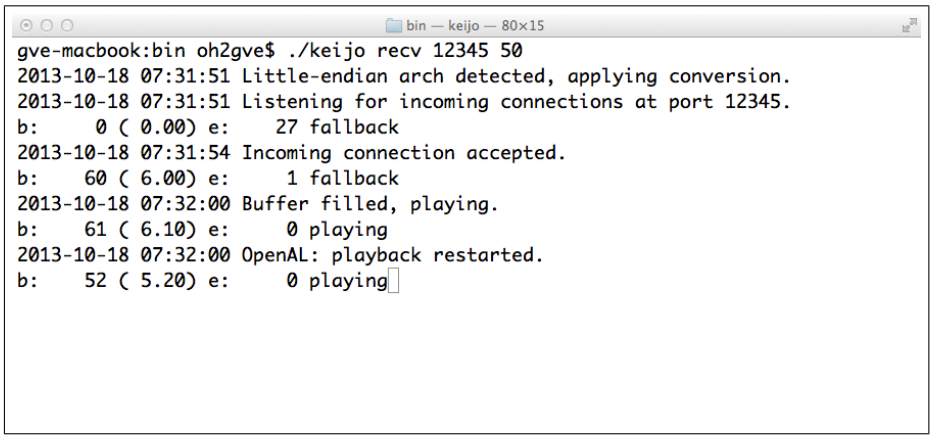
Toteutuksen laitealustaksi valittiin aluksi Linux-PC, joita radiokerholla oli käytettävissä runsaasti. Radiokerholaiset olivat hankkineet edellisen Wappuradion yhteydessä laadukkaiksi havaittuja Swissonic Easy FireWire -äänikortteja itselleen, ja niitä saatiin heiltä lainaan siirtoyhteysprojektia varten. Osoittautui, että Linux-käyttöjärjestelmässä ei ole ajuritukea FireWire-äänikorteille, mutta niiden käyttäminen on mahdollista Linuxin yleiseen FireWire-laitteistorajapintaan kytkeytyvällä FFADO-kirjastolla [12]. Kirjasto yhdistää äänikortin JACK-äänikehykseen [13], jonka toiminta perustuu ääntä käyttäville ohjelmille palveluita tarjoavaan reaaliaikaprioriteetilla ajettavaan palvelinprosessiin. Kehykseen toteutettiin oma TCP-siirtomoduli, joka siirsi ääntä JACK-palvelinprosessilta toiselle verkon yli. Lyhyissä, muutamien minuuttien mittaisissa kokeiluissa ratkaisu vaikutti toimivan, mutta pidempiaikaisissa siirtotesteissä ääntä tallentavan pään palvelinprosessi kaatui aina viimeistään kuuden tunnin toiminnan jälkeen. Ongelmien aiheuttajaksi pal-

jastui FFADO-kirjasto, jonka vianetsintätulosteista havaittiin muistinkäsittelyvirheisiin viittaavia ilmoituksia, esimerkiksi osoittimiin liittyviä assertation failed -virheitä. Kirjaston ohjelmakoodia ei näiden kokemusten valossa pidetty riittävän kypsänä ajateltuun käyttöön, joten FireWire-äänikorteista jouduttiin luopumaan.

Toista toteutusyritystä varten PC-laitteisiin vaihdettiin PCI-väyläiset SoundBlaster-äänikortit. Uusille äänikorteille löytyi ajurit suoraan Linux-ytimestä, joten niitä päästiin käyttämään Linuxin oman äänirajapinnan (Advanced Linux Sound Architecture, ALSA) [14] kautta. Tätä rajapintaa käyttäen toteutettiin suunnitellut toiminnallisuudet toteuttava ohjelma. Testit sujuivat jälleen aluksi hyvin, mutta pidempikestoisissa testeissä ohjelma kaatui aina viimeistään muutamien tuntien jälkeen. Ohjelman tilan kaatumishetkellä tallentaneita tiedostoja analysoimalla syy paikallistettiin poikkeuksetta ALSA-rajapinnan tarjoavan kirjaston sisään, jossa tapahtui muistinkäsittelyvirhe. Tämän seurauksena käyttöjärjestelmä katkaisi prosessin suorituksen. Virheitä analysoitaessa todettiin, että pitkäkestoinen yhtäjaksoinen äänen tallennus on ilmeisesti käyttötapaus, jota useimmat PC:n käyttäjät eivät tarvitse. Siihen liittyvät virheet eivät näin ollen ole tulleet riittävästi esille tullakseen tekijöidensä korjaamaksi.

Avoimen lähdekoodin komponenttien aiheuttamista pettymyksistä huolimatta ajatusta radiolähettimen erillisestä äänensiirtoyhteydestä omalla ohjelmalla ei oltu vielä valmiita hylkäämään. Edellisen vuoden toteutus olisi kuitenkin edelleen käytettävissä viimeisenä vaihtoehtona, mutta ennen siihen turvautumista päätettiin kokeilla vielä kolmatta laite- ja ohjelmointirajapintayhdistelmää siirtoyhteyden toteutusalustaksi. Applen Mac OS X -käyttöjärjestelmän tiedettiin edellisen vuoden kokemuksesta toimivan ongelmitta FireWire-äänikorttien kanssa. Pikainen katsaus käyttöjärjestelmän tarjoamiin ääniohjelmointirajapintoihin paljasti, että dokumentaatiota oli löydettävissä helpoiten OpenAL-rajapinnasta [15], joten sitä päädyttiin käyttämään. Nopeat savutestit rajapintaa käyttäen onnistuivat, joten niiden pohjalta päätettiin tehdä vielä kolmas äänensiirto-ohjelman toteutus. Usko luovilla aloilla työskentelevien käytössä usein nähdyn kaupallisen alustan toimivuuteen oli vahva, joten ohjelmaan toteutettiin heti alkuun myös suunnitelmien mukainen kevyt käyttöliittymä, joka näytti tallentavassa päässä digitoitavan signaalin äänenvoimakkuuden ja toistavassa päässä puskurin pituuden sekunteina. Käyttöliittymään päivitettiin äänenvoimakkuustietojen lisäksi mahdollista vianetsintää varten OpenAL:n kiinteänkokoisen tallennuspuskurin täyttöastetta, joka alustavissa kokeissa näytti pysyvän lähellä nollaa. Ääntä verkosta toistavaa instanssia varten mahdollisuus soittaa tiedostoa paikalliselta levyltä jatkuvassa silmukassa, mikäli verkosta ei tulisi äänidataa. Datavirran palatessa ohjelma asetettiin puskuroimaan sitä noin komentoriviltä annetun sekunnin kymmenysmäärän pituinen pätkä, jonka jälkeen tiedoston soittaminen vaihdettiin verkosta tulleen datan soittamiseen. Valmis siirto-

yhteyden vastaanottavan pään käyttöliittymä on esitetty kuvassa 2.4. Siirtoyhteyttä testattiin muutaman onnistuneen pikakokeen jälkeen yön yli, ja tällä kertaa tulokset olivat hyviä. Katkoksia ei ollut tapahtunut, eikä soittopuskurin pituus ollut kasvanut tai vähentynyt alkuarvona olleesta viidestä sekunnista. Järjestelmää testattiin saman tien useiden vuorokausien ajan, ja se osoittautui luotettavaksi. FM-lähettimen äänensiirto vuonna 2011 päätettiin toteuttaa tällä ratkaisulla.




***Kuva 2.4.*** *Ohjelman käyttöliittymä siirtoyhteyden vastaanottavassa päässä*

Äänensiirto toimi lähetyksen ajan ongelmitta. Odottamattomia katkoksia ei tullut, eikä soittopuskurin koko muuttunut viidestä sekunnista useammankaan vuorokauden yhtäjaksoisen soiton aikana. Äänikorttien näytteistystaajuuksien arveltiin olevan mitä ilmeisimmin vakaita ja lähes samoja. Ohjelmasta löydettiin kuitenkin suunnitteluvirhe. Pääteikkunassa toimivia tilatietoindikaattoreita päivitettiin samassa ohjelman säikeessä, missä kaikki muutkin ohjelman toiminnot tehtiin. Tämän seurauksena pääteikkunaa tarkastelleen operaattorin näppäilyvirheestä johtunut viiden sekunnin pysähdys käyttöliittymän päivityksessä sai koko ohjelman ja siten myös äänensiirron pysähtymään. Siirtoyhteyden toistavan pään puskuri ehti tyhjentyä, ja varatallenteen soitto alkaa. Kun tallentava pää jatkoi jälleen toimintaansa, se lähetti kaiken käyttöjärjestelmän äänipuskureihin tallentuneen datan kerralla toistavalle päälle. Tapauksen seurauksena lähettimeltä kuultiin kesken ohjelman joidenkin sekuntien ajan varatallennetta, jonka jälkeen ohjelma jatkui siitä mistä se oli katkennut. Siirtoyhteyden palautuessa siirrosta aiheutuva viive oli lähes kymmenen sekuntia. Toinen lyhyt katko kuultiin varhain seuraavana aamuna, kun tallentava pää käynnistettiin uudelleen, jotta viive saatiin palautettua viiteen sekuntiin. Näistä pienistä katkoksista huolimatta vuoden 2011 äänensiirtoa voitiin pitää hyvin onnistuneena ja teknisesti edellistä vuotta parempana.

## 2.3 Yhteenveto

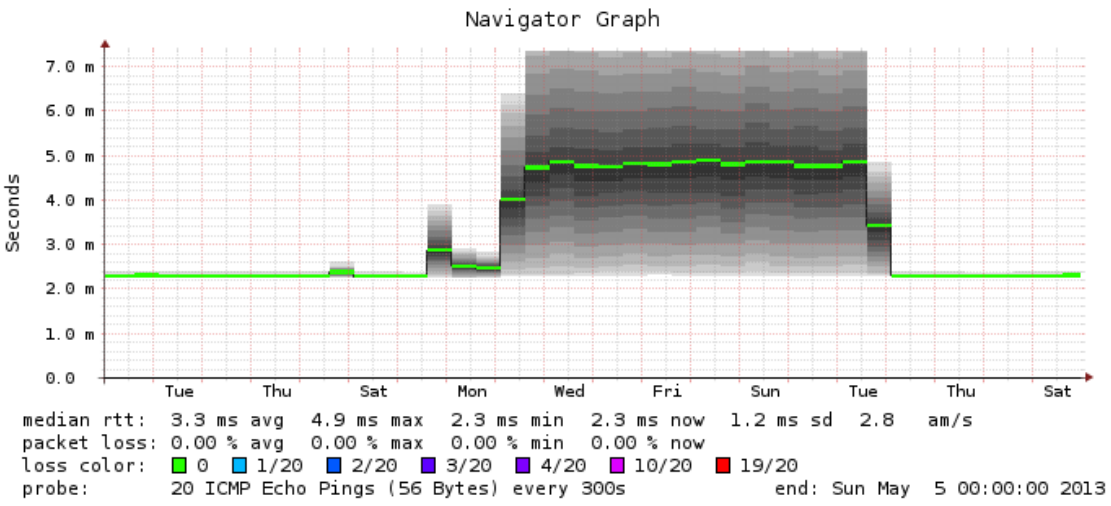
Vertailtaessa vuosien 2010 ja 2011 Rakkauden Wappuradioiden äänensiirtoyhteyksiä vuoden 2011 projektin palautetilaisuudessa voitiin todeta, että siirtoyhteys oli kehittynyt paitsi viiveen käyttäytymisen myös toimintavarmuuden osalta. Parannukseksi laskettiin myös uuden järjestelmän suoraviivaisuus, sillä siinä siirrettävää signaalia ei pakattu tai muutenkaan käsitelty, vaan äänidata vain siirrettiin sellaisenaan TCP-yhteyttä pitkin. Vuosien 2010 ja 2011 äänensiirtojärjestelmien tärkeimpiä ominaisuuksia on vertailtu taulukossa 2.2.

***Taulukko 2.2.*** *Äänensiirtojärjestelmien ominaisuudet vuosina 2010 ja 2011*

| | Vuosi 2010 | Vuosi 2011 |
|---|---|---|
| Viive (s) | 3-30 | 5-10 |
| Spontaanit yhteyskatkot | 2 | 0 |
| Äänen pakkaus | 320 kb/s MP3 | ei pakkausta |
| Näytteenottotaajuus (Hz) | 44100 | 48000 |
| IP-siirron kaistanleveys (Mb/s) | 0,5 | 1,5 |
| AD- ja DA-äänikortit | erilaiset | samanlaiset |
| Ohjelmisto | yleiskäyttöinen, monimutkainen | pelkistetty, helppokäyttöinen |

Vuoden aikana tapahtuneeseen kehitykseen oltiin tyytyväisiä, mutta järjestelmässä nähtiin edelleen myös jatkokehitysmahdollisuuksia. Siirtoviiveen lyhentäminen edelleen nähtiin tavoittelemisen arvoisena, sillä esimerkiksi osa lähetykseen soittaneista kuulijoista oletti kuulevansa äänensä välittömästi radiosta, ja yllättyi havaitessaan siinä olleen viiden sekunnin viiveen. Lisäksi kahden megabitin internetyhteys lähettimen sijaintipaikkaan Hervannan vesitorniin oli radiolähetyksen aikaisella kuormituksellakin havaittu siirtovirhe- ja viivemittauksissa lähes virheettömäksi. Tulokset olivat hyvin samanlaisia, kuin kuvassa 2.5 nähtävät vuoden 2013 lähetyksen aikaiset mittaustulokset. Ainoa kuormituksen aiheuttama muutos oli siirtoviiveen kasvu hieman yli kahdesta vajaaseen viiteen millisekuntiin. Yhteyden läpi voitaisiin siis kuljettaa dataa myös yksinkertaisemmallakin protokollalla, jotka ei toteuta siirron onnistumisen varmistusta eikä datan järjestyksen säilymistä, mutta mahdollistaa hyvin lyhyen viiveen. Myös vuonna 2011 käytössä ollut kalusto koettiin jatkossa mahdollisesti ongelmalliseksi. Järjestelmä nojasi vahvasti yleiskäyttöisiin ja käytettynäkin arvokkaisiin Applen tietokoneisiin, joiden järjestely radion käyttöön tuotti ja tulisi luultavasti tuottamaan jatkossakin ylimääräistä vaivaa tai vaihtoehtoisesti huomattavia kustannuksia. Lisäksi niiden operointiin ja valvontaan tarvittiin

vuorokauden ympäri henkilö, joka osasi virhetilanteiden, kuten esimerkiksi sähkökatkon sattuessa konfiguroida järjestelmän takaisin käyttöön. Esiin nostettiin ajatus kokonaan oman sulautettuun laitteistoon pohjautuvan äänensiirtojärjestelmän toteuttamisesta mahdollista seuraavaa Rakkauden Wappuradiota varten. Näin saataisiin radioprojektin käyttöön varmasti saatavilla olevat laitteet, joiden toiminnot voitaisiin valita ja toteuttaa kaikilta osin itse projektin tarpeista lähtien ja pelkästään Rakkauden Wappuradiota ajatellen.




***Kuva 2.5.*** *Verkkoyhteyden laatu vuoden 2013 lähetyksen aikana*

# 3 SUUNNITTELU JA TOTEUTUS

Tässä luvussa esitellään vuoden 2012 Rakkauden Wappuradiota varten valmistetun sulautetun IP-pohjaisen äänensiirtojärjestelmän suunnittelu ja toteutus. Laitteen ja koteloinnin on suunnitellut sekä koekappaleet koonnut DI Eero Alkkiomäki Baytems Holdings Oy:stä.

## 3.1 Vaatimukset

Valmistettavaan järjestelmään kohdistuvat vaatimukset tunnistettiin aiempien Rakkauden Wappuradion äänensiirtojärjestelmien ja niistä kertyneiden kokemusten pohjalta. Kaikkein tärkeimmäksi vaatimukseksi nostettiin luotettavuus. Mikäli uuden järjestelmän toimivuuteen ei voitaisi luottaa, käytettäisiin mielummin vanhoja kauttaaltaan tunnettuja ja luotettavaksi tiedettyjä ratkaisuja. Jotta uuteen järjestelmään olisi helppo perehtyä ja vakuuttua sen toimintaedellytyksistä ja luotettavuudesta, päätettiin pitäytyä mahdollisimman yksinkertaisissa suunnitteluratkaisuissa. Järjestelmään päätettiin toteuttaa vain kaikkein välttämättömimmät toiminnot, jotta kokonaisuus säilyisi helposti hahmotettavana ja valmistuskustannukset pysyisivät mahdollisimman pieninä. Välttämättömiksi toiminnoiksi oli aiemmilla Rakkauden Wappuradion lähetyskerroilla todettu analogisen äänen siirto IP-verkon yli paikasta toiseen, ennalta asetetun tallenteen soittaminen, mikäli verkkoyhteys katkeaa ja siirrettävän äänen voimakkuuden mittaus. Tallenteen soittaminen verkkoyhteyden katketessa oli yleisön palvelun kannalta välttämätöntä, sillä tallenteella voitaisiin kertoa kuulijoille kyseessä olevan tilapäinen häiriötilanne. Pelkän hiljaisuuden lähettäminen tai lähettimen sammuttaminen eivät mahdollistaisi sitä, ja saattaisivat johtaa väärinkäsityksiin kuuntelijoiden keskuudessa. Siirrettävän äänen voimakkuuden mittaus oli havaittu tarpeelliseksi siksi, jotta siirtojärjestelmän dynaaminen alue saataisiin mahdollisimman laajasti käyttöön. Tämä oli tehtävissä säätämällä järjestelmään syötettävän äänen voimakkuus mittauksen avulla mahdollisimman suureksi ilman, että se ylittää suurimmat sallitut arvot. Nämä toiminnot päätettiin toteuttaa yksinkertaisella kahdesta laitteesta koostuvalla sulautetulla järjestelmällä. Laitteista toinen sijoitettaisiin Rakkauden Wappuradion studioon lähettämään digitaaliseksi muutettua ääntä tietoverkkoon, ja toinen radiolähettimen läheisyyteen vastaanottamaan ohjelmaa verkosta ja muuttamaan se takaisin analogiseksi signaaliksi lähetintä varten.

Muut järjestelmään kohdistuvat vaatimukset muodostuivat monilta osin halusta parantaa aiempien äänensiirtojärjestelmien ominaisuuksia. Tärkeimpänä näistä pidettiin äänensiirrosta aiheutuvaa viivettä, joka oli aiemmin ollut tyypillisesti yli viisi sekuntia. Tämä oli ajoittain koettu ongelmalliseksi radio-ohjelman tekijöiden ja kuuntelijoiden välisissä vuorovaikutustilanteissa. Esimerkiksi studioon soittaneiden kuuntelijoiden ja heidän seurassaan olleiden oli havaittu hämmentyvän, kun he eivät odotustensa vastaisesti kuulleetkaan puhelimitse välitettyä ääntä radiosta heti toimittajan vastattua puhelimeen. Uuden järjestelmän vaatimukseksi asetettiin alle sadan millisekunnin pituinen siirtoviive, sillä sen arveltiin signaalinkäsittelyn asiantuntijan kanssa käydyn keskustelun [16] perusteella riittävän interaktiivisen radionteon tarpeisiin. Aiempien järjestelmien pidempi siirtoviive oli aiheutunut suurelta osin siirtoyhteyden vastaanottavassa päässä tehdystä puskuroinnista. Puskuroinnin tarkoituksena oli ollut varautua kuljetuskerroksen protokollana käytetyn TCP:n siirtoviiveen vaihteluun. Viiveen vaihtelu johtui protokollaan kuuluvien datan eheyden ja järjestyksen takaavien uudelleenlähetysmekanismien toiminnasta IP-siirron virhetilanteissa. Näin oli varmistettu äänensiirron virheettömyys, vaikka käytössä oli siirtovirheitä aiheuttanut tietoverkko. Varautumisvaatimuksesta voitiin kuitenkin nyt luopua, sillä käytettävän tietoverkkoyhteyden laatu oli edellisen Rakkauden Wappuradion lähetyksen yhteydessä havaittu erinomaiseksi. Siirtoyhteyden nopeus oli sen hitaimmalla osuudella edelleen kaksi megabittiä sekunnissa, joten äänensiirron datavirran oli uudessakin järjestelmässä jäätävä bittinopeudeltaan tämän alle.

Laitteiden käytettävyyttä haluttiin parantaa aiempiin toteutuksiin nähden. Aiemmat ratkaisut olivat vaatineet operaattoreilta laajaa perehtyneisyyttä ja tarkkaavaisuutta järjestelmää käynnistettäessä ja vikatilanteista toivuttaessa. Uuden järjestelmän haluttiin toimivan täysin automaattisesti siten, ettei käyttäjän tarvitse kaapeleiden kytkennän jälkeen koskea laitteeseen lainkaan. Mahdollisimman monesta virhetilanteesta oli siis toivuttava täysin automaattisesti. Laitteisiin päätettiin rakentaa käyttöliittymäksi yksinkertainen RS-232-sarjaportin kautta käytettävä konsoli äänenvoimakkuuden ja laitteen tilatietojen raportointia sekä ohjelmistokehityksen aikaista vianetsintää varten. Konsolin tuli kuitenkin tarkoitus toimia vain laitteen tuottaman tiedon esittämiseen, eikä sen kautta saanut olla lainkaan mahdollista puuttua järjestelmän toimintaan millään tavoin. Linjatasoisen signaalin audioliitäntöjen tiedettiin olevan studiotekniikassa yleensä balansoituja ja niissä käytettävän XLR-tyyppisiä [17] liittimiä. Balansoitu signaali tiedettiin kuluttajalaitteissa usein käytettyä balansoimatonta signaalia häiriösietoisemmaksi [18, s. 203] ja XLR-liitin kaikin puolin hyväksi, joten niitä päätettiin käyttää fyysisenä äänirajapintana myös tässä laitteistossa. Laitesuunnittelun ja toteutuksen helpottamiseksi kaikista laiteyksilöistä päätettiin tehdä samanlaisia, ja toteuttaa toiminnallisuuksien erot ohjelmakoodissa. Jokaisessa laitteessa olisi siis sekä äänen tulo- että lähtöliittimet

riippumatta siitä, missä roolissa se toimii. Ainakin siirtoyhteyden vastaanottavan pään laite tulisi sijaitsemaan tehokkaan radiolähettimen välittömässä läheisyydessä, joten häiriösietoisuuteen päätettiin kiinnittää huomiota muidenkin kuin ääniliitäntöjen osalta.

Järjestelmän ohjelmisto suunniteltiin laitteiston tavoin yksinkertaiseksi, vain välttämättömät toiminnot toteuttavaksi kokonaisuudeksi. Ohjelmakoodi päätettiin jakaa laitteiden kesken siten, että siirtoyhteyden lähettävään ja vastaanottavaan päähän käännetään erilliset ohjelmabinäärit. Tällöin kaikki ohjelmien tarvitsema asetustieto voitiin sijoittaa helposti ohjelmankoodin oheen, minkä seurauksena ohjelman ei tarvitsisi missään vaiheessa kysellä asetuksia käyttäjältä. Mitään interaktiota käyttäjän kanssa ei näin ollut tarpeen toteuttaa. Ohjelmabinäärit oli käännettävä koodin tuplaantumisen välttämiseksi mahdollisimman suurelta osin samoista lähdekooditiedostoista, ja vain eriävät osat tuli sijoittaa omiin tiedostoihinsa. Ohjelmien perustana päätettiin käyttää jotakin vapaasti saatavilla olevaa reaaliaikakäyttöjärjestelmää, mihin olisi saatavissa valmis TCP/IP-protokollapinon toteutus. Mikäli näitä komponentteja ei löydettäisi valmiiksi saatavilla olevina, tulisi ohjelmiston toteuttaminen olemaan liian työlästä tämän projektin puitteissa. Valmiin TCP/IP-tuen lisäksi käyttöjärjestelmän haluttiin tarjoavan jonkin ratkaisun ohjelmakoodin rinnakkaistamiseen, jotta ohjelman toiminnot voitaisiin jakaa koodin eri osiin mahdollisimman itsenäisiksi kokonaisuuksiksi. Siirtoyhteyden vastaanottavan pään ohjelmistoon oli tarpeen kiinnittää erityistä huomiota, sillä siinä oli varauduttava mahdollisuuksien mukaan erilaisiin tiedonsiirron häiriöihin. Jotta aivan pienet häiriöt eivät aiheuttaisi heti varatallenteen soittamisen tarvetta, päätettiin ohjelmaan toteuttaa pieni, alle sadan millisekunnin edestä ääntä sisältävä puskuri. Yksittäisen paketin puuttumiseen olisi näin mahdollista reagoida keräämällä puskuriin hieman ääntä verkosta ennen sen soittamisen aloittamista, ja paketin puuttuessa antaa puskurin lyhentyä. Esimerkiksi laitteiden sijoituspaikkojen lämpötilaeroista johtuvista pienestä kellotaajuuksien erosta selvittäisiin antamalla puskurin tyhjentyä tai täyttyä hitaasti ajan myötä. Puskurin käsittelyn hoitava algoritmi oli siis toteutettava siten, että se tekee puskurin esitäytön noin puoleen kokonaispituudesta ja sen jälkeen pyrkii tarkkailemaan puskurin täyttöastetta ja pitämään sen vakiona.

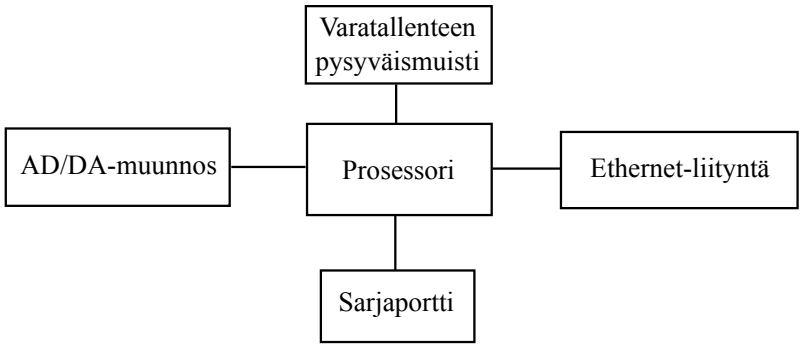
Suunnittelutyön tuloksena saatiin luettelo ominaisuuksista, jotka järjestelmässä on vähintään oltava, jotta sitä voidaan käyttää äänen siirtoon Rakkauden Wappuradion toiminnassa.

- Järjestelmään toteutetaan vain Rakkauden Wappuradion äänensiirrossa välttämättömät toiminnot, jotta järjestelmästä ei tule tarpeettoman monimutkaista.

- Äänensiirrosta aiheutuva viive on saatava alle sadan millisekunnin mittaiseksi, jotta toimittajien ja kuuntelijoiden reaaliaikainen interaktio on vaivatonta.
- Analogisen äänen liitynnät ovat balansoituja, ja niissä käytetään XLR-liittimiä. FM-lähettimen radiotaajuinen säteily tai muukaan odotettavissa oleva elektromagneettinen häiriö ei saa vaikuttaa järjestelmän toimintaan.
- Laitteiden tulee toimia kokonaan ilman vuorovaikutusta käyttäjän kanssa. Myös vikatilanteesta toipuminen on tapahduttava ilman käyttäjän toimia.
- Järjestelmän tulee siirtää digitaalinen ääni IP-protokollaa käyttäen, jotta tietoliikenne voidaan reitittää verkosta toiseen. Digitaalisen äänen bittinopeuden on oltava alle kaksi megabittiä sekunnissa.
- Verkkoyhteyden katketessa FM-lähetintä ohjaavan laitteen on soitettava massamuistiin ennalta ladattua äänidataa, kunnes verkkoyhteys palautuu.
- Siirtoyhteyden molempien pään laitteiden on oltava samanlaiset. Toiminnallisuuksien erot on toteutettava ohjelmakoodissa.

## 3.2 Laitteisto

Laitteiston tärkeimmät osat ovat mikroprosessori, AD- ja DA-muunnokset tekevä stereokodekki ja ethernet-lähetinvastaanotin. Lisäksi keskeisiä ovat verkkoyhteyden katketessa soitettavan varatallenteen säilyttämiseen käytettävä flash-muisti sekä äänenvoimakkuus- ja tilatietojen näyttämiseen tarvittava sarjaporttiliityntä. Nämä osat ja niiden suhteet on esitelty kuvassa 3.1.

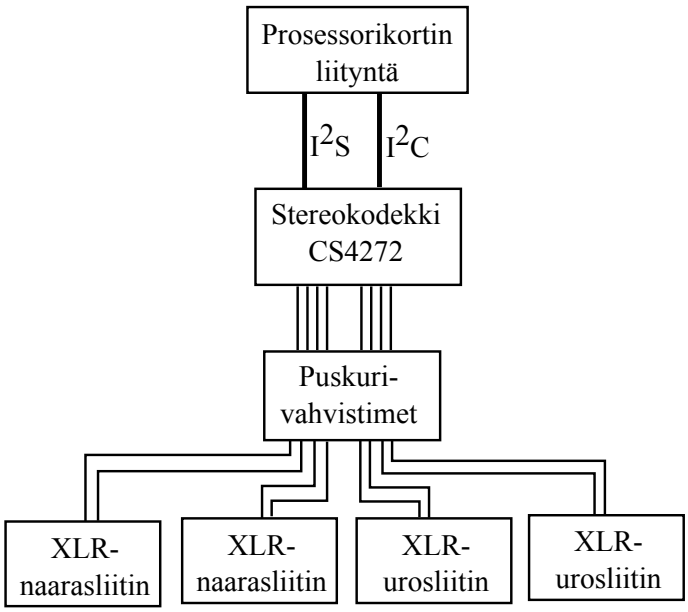



***Kuva 3.1.*** *Laitteen tärkeimmät komponentit*

Mikroprosessori ohjaa laitteen toimintaketjua, jossa ohjelmasta riippuen yleensä joko vastaanotetaan AD-muuntimelta analogista ääntä digitaaliseksi ja pilkotaan se paketteihin IP-verkossa siirtämistä varten tai vastaanotetaan IP-verkosta äänidataa sisältäviä paketteja ja siirretään ne DA-muuntimelle analogiseksi ääneksi muuntamista varten. Mikäli IP-verkosta ei saada äänidataa, DA-muuntimelle lähetetään

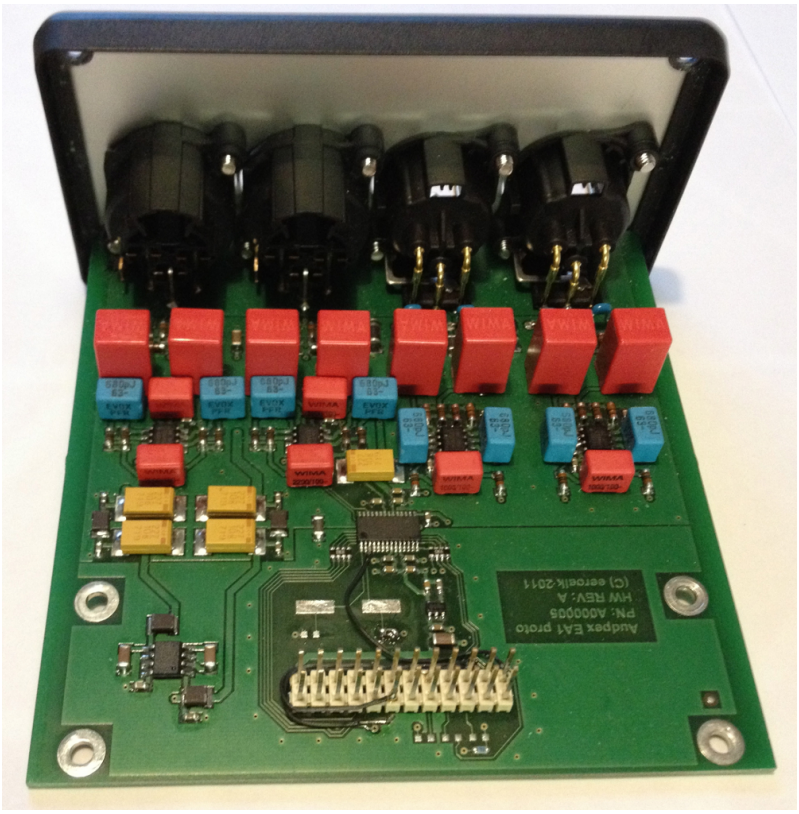
flash-muistiin ennalta tallennettua äänidataa. Siirtoyhteyden lähettävän pään laitteessa mikrokontrolleri määrittää jatkuvasti välittämänsä äänen voimakkuutta ja lähettää siitä tietoja käyttäjän päätteelle sarjaportin kautta.

Laitteen suunnittelussa oli tarkoitus hyödyntää Baytems Holdings Oy:n prototyyppiasteella ollutta yleiskäyttöistä prosessorikorttia, joka perustui Atmelin valmistamaan ARM9-perheen prosessoriin. Prosessorikortin pariksi suunniteltiin stereokodekkipiirin ja muun audioelektroniikan sisältävä erillinen äänikortti. Kortin tärkeimmät osat on esitetty kuvassa 3.2.




***Kuva 3.2.*** *Äänikortin keskeisimmät komponentit*

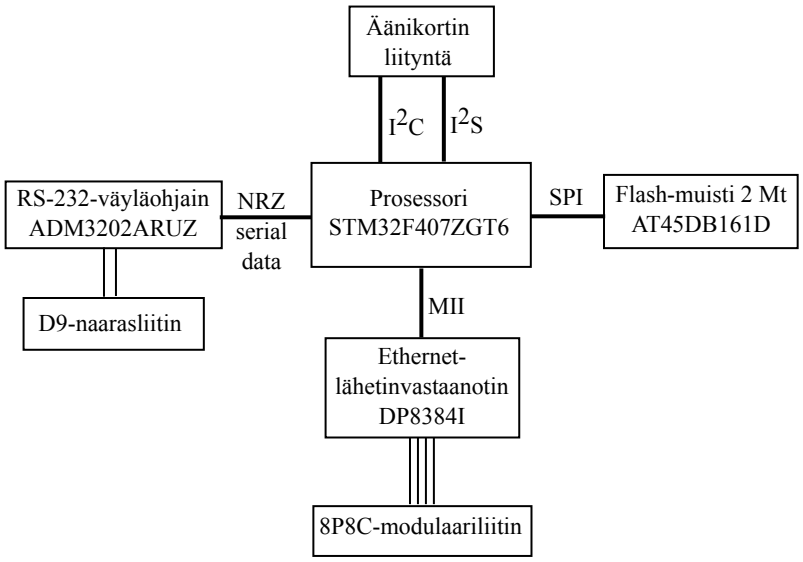
Äänen AD- ja DA-muunnokset tekee Cirrus Logicin valmistama stereokodekkipiiri CS4272. Piiri kykenee näytteistämään stereoäänen 16- tai 24-bittisenä 48, 96 tai 192 kilohertsin näytteenottotaajuudella, ja sen dynaaminen alue on 111 dB [19, s. 14]. Piiri konfiguroidaan käynnistyksen yhteydessä piirienvälistä väylää (Inter-Integrated Circuit, $I^2C$) käyttäen ja digitaalinen ääni siirretään kodekin ja prosessorin välillä piirikortin sisäistä äänensiirtoa varten luotua väylää (Inter-IC Sound, $I^2S$) [20] pitkin. Kellolähteenä piiri käyttää prosessorilta $I^2S$-väylän ohessa tulevaa erillistä kellolinjaa. Stereokodekkipiirin lisäksi kortille suunniteltiin piirin koekytkentälevyn kytkentäkaaviossa esitettyjen [21, s. 29] mallien mukaiset puskurivahvistimet. Ne sovittavat operaatiovahvistinten avulla kodekkipiirin ääniliitäntöjen impedanssit järjestelmään kytkettävien laitteiden impedansseihin. Äänikortti saa käyttöjännitteensä prosessorikortin viiden voltin pääjännitteestä, josta muodostetaan kodekkipiirin ja operaatiovahvistimien tarvitseman jännitteet. Ääni- ja prosessorikorttien yhteenliittämistä varten kortilla on yksinkertainen piikkirimaliitin. Kuva 3.3 esittelee kootun äänikortin laitteen kotelon päätylevyyn kiinnitettynä.



***Kuva 3.3.*** *Äänikortti koottuna*

Tutkittaessa prosessorikortin ja äänikortin yhteensovittamista huomattiin, että prosessorikortin Atmelin ARM9 -prosessorin I$^2$S-toteutus ei ollut täysin yhteensopiva valitun äänikodekkipiirin kanssa. Lisäksi prosessorikortin prototyypissä oli havaittu joitakin ongelmia, jotka olivat äänensiirtojärjestelmän laitesuunnittelun aikaan yhä selvitystyön alla. Näiden syiden takia nähtiin helpommaksi suunnitella laitteen osaksi uusi nimenomaan tämän järjestelmän tarpeisiin sopiva prosessorikortti. Kortille voitiin näin valita helposti tässä järjestelmässä tarpeelliset osat ja joitakin mahdollista jatkokehitystä tukevia komponetteja. Prosessoriksi valittiin STMicroelectronicsin valmistama ARM Cortex-M -perheen neljättä sukupolvea edustava STM32F407ZG [22]. Cortex-M-perheen prosessorit on suunnattu mikrokontrollerityyppiseen käyttöön, jossa yleensä vaativan laskennan sijaan tehdään tiedonsiirtoa ja muuta oheislaitteiden hallintaa. Ne kuluttavat sähköä maltillisesti, täydellä 168 MHz:n kellotaajuudella tyypillinen sähkönkulutus on valmistajan mukaan [23, s. 77] alle 0,3 wattia. Laskentatehoa arveltiin kuitenkin olevan moninkertaisesti yli tarpeen, joka koostui lähinnä väyläoperaatioiden kontrolloinnista ja datan kopionnista. Prosessorista löytyvät valmiit laitteistolohkot tavanomaisten I$^2$C- ja lisälaitesarjaväylän (Serial Peripheral Interface, SPI) lisäksi muun muassa I$^2$S-väylälle ja ethernet-lähetinvastaanottimien liittämiseen tarkoitettua mediariippumatonta väylää (Media Independent Interface, MII) varten. Niiden ansiosta valitun prosessorin

katsottiin soveltuvan erinomaisesti hallitsemaan äänensiirtojärjestelmän toimintaa sen keskeisimpänä komponenttina. Prosessorikortin muut keskeisimmät osat on esitetty kuvassa 3.4.

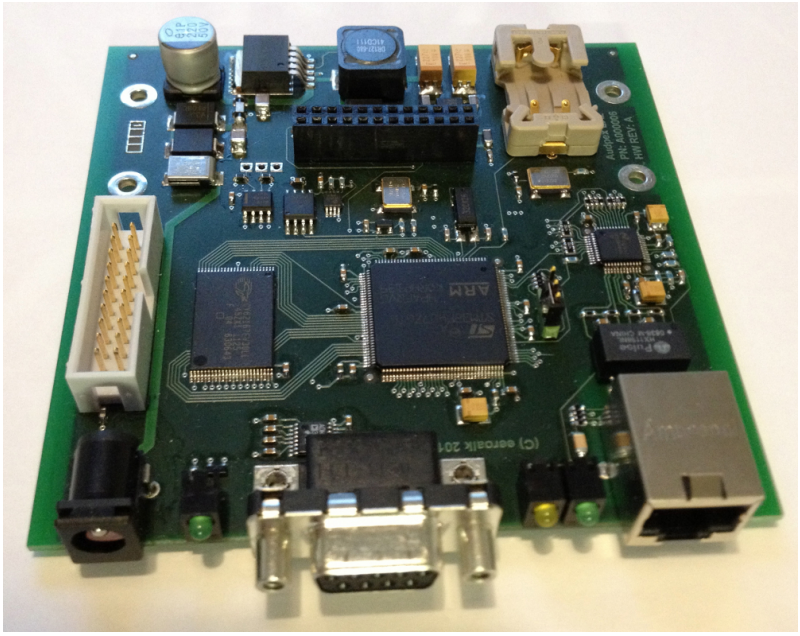



***Kuva 3.4.*** *Prosessorikortin keskeisimmät komponentit*

Kortille sijoitettiin verkkoyhteyden katketessa soitettavan äänen tallentamista varten Atmelin valmistama kahden megatavun SPI-väyläinen flash-muisti AT45DB161D. Prosessorin MII-väylään kytkettiin National Semiconductorin valmistama ethernet-lähetinvastaanotin DP83848I, josta ethernet-signaalit siirtyvät erotusmuuntajan läpi 8P8C-modulaariliittimelle eli niin sanotulle RJ45-liittimelle. Verkkoyhteyden toiminnan seuraamista varten liittimen viereen sijoitettiin ethernet-linkkiä ja -aktiviteettiä osoittavat ledit, joita lähetinvastaanotin ohjaa automaattisesti. Äänenvoimakkuudesta ja ohjelman tilasta kertovien tietojen välittämiseksi ja ohjelmointivaiheen vianetsinnän helpottamiseksi laitteeseen toteutettiin PC-tietokoneyhteensopiva sarjaportti kytkemällä yhteen prosessorin yleisistä sarjaliikennelohkoista (Universal Synchronous Asynchronous Receiver Transmitter, USART) RS-232-tasomuunninpiiri ja D9-liitin.

Kuvassa 3.4 nähtävien komponenttien lisäksi kortille sijoitettiin muiden komponenttien toiminnan kannalta välttämättömiä ja varmuuden vuoksi myös joitakin mahdollisessa jatkokehityksessä tarpeellisia tai ohjelmistokehitystä helpottavia osia. Myöhemmin mahdollisesti tarvittavaa konfiguraatiotietojen tallentamista varten lisättiin kortille Microchipin valmistama kahdeksan kilotavun $I^2C$-väyläinen sähköisesti tyhjennettävä ohjelmoitava muisti (Electronically Erasable Programmable Read-Only Memory, EEPROM) 24LC64. Prosessorin 192 kilotavun RAM-muistin arveltiin riittävän melko monimutkaisellekin ohjelmalle, mutta kortille sijoitettiin myös kaksi megatavua ulkoista RAM-muistia. Prosessorin reaaliaikakello-

lohkoa (Real-Time Clock, RTC) varten kortille sijoitettiin kolmen voltin CR2032-paristolle tarkoitettu pidike. Prosessorin ohjelmointia ja vianetsintää varten sen IEEE 1149.1-standardin (Joint Test Action Group, JTAG) mukaiseen debug-lohkoon liitettiin ARM-järjestelmäsuositusten mukainen 20-piikkinen piikkirimaliitin [24]. Yksinkertaista laitteen tilan indikointia varten kortille sijoitettiin suoraan prosessorilla ohjattavissa oleva vihreä ledi. Käyttöjännitteiden käsittelyyn kiinnitettiin erityistä huomiota, jotta laitteen käyttöjännitteelle ei tarvitsisi asettaa tiukkoja reunaehtoja. Korttia ajateltiin syötettävän 12 voltin tasajännitteellä, mutta monipuolisen suodatuksen ja reguloinnin ansiosta jännite voi olla välillä 10-30 volttia, eivätkä esimerkiksi edullisten hakkurivirtalähteiden mahdollisesti aiheuttamat vaihtojännitekomponentit haittaa laitteen toimintaa. Valmis koottu prosessorikortti esitellään kuvassa 3.5.



***Kuva 3.5.*** *Koottu prosessorikortti*

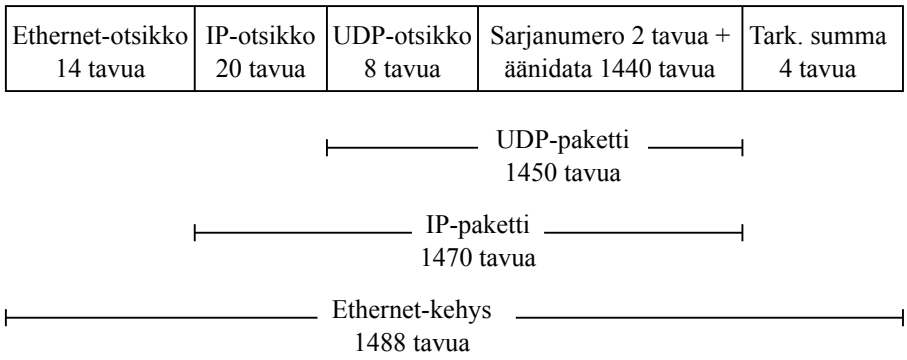
Prosessori- ja äänikortti liitettiin toisiinsa ruuveilla, ja näin saatu valmis laite koteloitiin mittatilaustyönä teetettyyn alumiinikoteloon. Laitteita koottiin heti kaksi kappaletta, jotta ohjelmistokehitys testeineen voitiin tehdä helposti samanaikaisesti siirtoyhteyden molempien päiden osalta. Valmiiksi koottu ja koteloitu laite esitellään kuvassa 3.6.


***Kuva 3.6.*** *Valmis laite*

## 3.3 Tietoliikenneprotokolla

Järjestelmän suunnitteluvaatimuksissa oli lähdetty alle sadan millisekunnin siirtoviiveestä äänessä. Lisäksi vaadittiin IP-protokollan käyttöä, jotta tietoliikennettä voitaisiin reitittää. Aiemmista järjestelmistä poiketen tiedonsiirron virheisiin ei ollut tässä järjestelmässä tarpeen varautua. Näiden vaatimusten pohjalta harkittiin lyhyesti äänen kuljettamista suoraan IP-paketeissa. Muiden kuin tunnettua kuljetuskerroksen protokollaa sisältävien IP-pakettien arveltiin kuitenkin joutuvan helposti palomuurien tai muiden verkon turvalaitteiden torjumaksi. Lisäksi järjestelmän haluttiin toimivan tarpeen vaatiessa myös sellaisissa verkoissa, joissa on käytössä kuljetuskerroksen porttinumeroon perustuva osoitteenmuutos (Network Address and Port Translation, NAPT) tai pakettien suodatus. Näistä syistä pelkän IP-paketin käytöstä luovuttiin. Kuljetuskerroksen protokollista mahdollisena vaihtoehtona pidettiin lähinnä UDP:tä, sillä TCP oli aiemmin havaittu siirtoviiveeltään liian pitkäksi. Muut valmiit protokollat olisivat tuntemattomia paitsi mahdollisesti verkkolaitteille, myös toteuttajille, eikä niiden soveltuvuuden analysointiin tai opiskeluun haluttu käyttää projektin puitteissa aikaa. UDP valittiin näin laitteiden välisen tietoliikenneprotokollan pohjaksi.

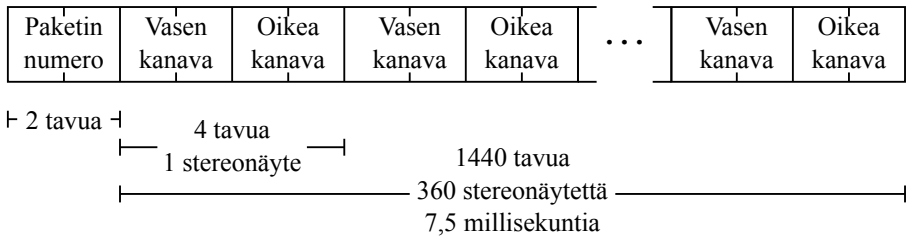
Siirrettävien UDP-pakettien katoamiseen tai niiden järjestyksen muuttumiseen ei tarvinnut reagoida, joten pakettien uudelleenlähetyksiä tai järjestyksen vaihtoa ei ollut tarpeen toteuttaa. Tämän seurauksena todettiin riittäväksi vain lähettää äänidataa siirtoyhteyden läpi ilman mitään paluukanavaa vastaanottavasta päästä lähettävään. Protokolla oli siis toteutettavissa täysin yksisuuntaisesti. Paketteihin päätettiin kuitenkin epätodennäköisiksi muuttuneiden mutta silti edelleen mahdollisten verkkovirheiden havaitsemiseksi sijoittaa sarjanumerot. Yhteyden vastaanottava pää voisi näin ilmaista verkkovirheen tapahtuneen, vaikka se ei korjaavia toimenpiteitä tekisikään. Sarjanumero toteutettiin UDP-paketin hyötykuormaosan alkuun sijoitettavalla 16-bittisellä etumerkittömällä kokonaisluvulla, jota kasvatettiin yhdellä aina paketin lähettämisen yhteydessä. Varsinainen äänidata päätettiin sijoittaa UDP-paketteihin sellaisenaan ilman pakkausta tai muutakaan käsittelyä. Yhteen pakettiin mahtuvan äänidatan määrän laskemiseksi oletettiin ethernet-kehyksen maksimikooksi 1500 tavua. Hyötykuorman suurin mahdollinen koko saatiin vähentämällä ethernet-kehyksen maksimikoosta sen otsikkotietojen pituus 14 tavua [18, s. 508], IP-paketin otsikkotietojen pituus 20 tavua [18, s. 593] ja UDP-paketin otsikkotietojen pituus 8 tavua [18, s. 701]. Tulokseksi saatuun 1458 tavuun laskettiin mahtuvan neljän tavun mittaisia 16-bittisiä näytteitä stereoäänestä yhteensä enimmillään 364,5 kpl, kun paketin numeron vaatimaa kahta tavua ei otettu huomioon. Laskutoimenpiteiden helpottamiseksi yhteen pakettiin päätettiin sijoittaa 360 näytettä, jolloin myös pakettinumerokentälle jäi tilaa. Näin muodostettu kokonaisuus on esitelty kuvassa 3.7.




***Kuva 3.7.*** *Järjestelmän lähettämän ethernet-kehyksen sisältö*

Valitulla 48 kHz näytteenottotaajuudella saatiin yhden paketin sisältämien 360 näytteen aikavastaavuudeksi 7,5 millisekuntia ja siirtoyhteyden vaatimaksi kaistanleveydeksi 1,55 megabittiä sekunnissa. Siirtoyhteyden kapasiteetin ollessa kaksi megabittiä sekunnissa voitiin järjestelmän kaistanleveysvaatimus todeta riittävän pieneksi, sillä yhteyteen jäisi äänensiirron aikanakin kapasiteettia esimerkiksi verkonvalvonta- ja hallintayhteyksiä varten. Stereokodekin tuottama digitaalinen äänidata sijoitettiin UDP-paketteihin kokonaisina 16-bittisinä stereonäytteinä siten,

että vasemman kanavan tavut tulivat ennen oikean kanavan tavuja. Tavut olivat valmiiksi verkon tavujärjestyksessä (big endian). Sijoittuminen on esitetty kuvassa 3.8.




***Kuva 3.8.*** *Äänidatan sijoittuminen UDP-pakettiin*

Siirtoyhteyden vastaanottavan pään laitteeseen varattiin mahdollisia pieniä verkon siirtoviiveen vaihteluita varten kymmenen paketin äänidatan kokoinen puskuri. Puskurin koko muistissa oli hieman yli 14 kilotavua, joten se mahtui ongelmitta prosessorin omaan 192 kilotavun RAM-muistiin käyttöjärjestelmän ja ohjelmakoodin oheen. Puskurin toiminta konfiguroitiin siten, että laitteen käynnistyessä se täytetään puolilleen, eli siihen kerätään viiden paketin sisältämä äänidata. Tämän jälkeen aloitetaan äänen lähettäminen kodekkipiirille soitettavaksi. Puskurin arveltiin näin selviytyvän parhaiten mahdollisista verkon häiriöistä. Tällä konfiguraatiolla tyypilliseksi äänensiirrosta aiheutuvaksi viiveeksi muodostui hieman alle 40 ms, jonka arveltiin olevan niin pieni, ettei sen olemassaoloa voi helposti havaita.

## 3.4 Ohjelmisto

Järjestelmän ohjelmisto jakautui vaatimusten mukaisesti kahteen ohjelmaan. Kahden tuotantokäytössä tarvittavan ohjelman lisäksi laitetta varten tehtiin työkaluohjelma, jolla verkosta vastaanotettava äänidata voitiin analogiseksi muuntamisen sijaan tallentaa laitteen flash-muistiin. Sen parina toimi PC-tietokoneelle SDL-kirjastoa [25] käyttäen toteutettu työkaluohjelma, joka lähetti käyttäjän valitseman tiedoston sisällön sellaisenaan laitteiden välillä käytettävää tietoliikenneprokollaa käyttäen käyttäjän antamaan IP-osoitteeseen. Mainittujen ohjelmien keskeiset tiedot on koottu taulukkoon 3.1.

***Taulukko 3.1.*** *Järjestelmän ohjelmat*

| koodinimi | kyosti | poysti | neponen | repomies |
|---|---|---|---|---|
| käyttöpaikka | studio | lähetinkoppi | huolto | huolto |
| rooli | tuotanto | tuotanto | työkalu | työkalu |
| tehtävä | AD-muunnos | DA-muunnos | flash-muistin täyttö | tiedoston lähetys |
| laitealusta | ARM CM4 | ARM CM4 | ARM CM4 | x86 PC |

### 3.4.1 Jaetut komponentit

Ohjelmiston suunnittelussa otettiin työmäärän rajaamiseksi lähtökohdaksi valmiiden komponenttien mahdollisimman laaja hyödyntäminen. Tärkeimpiä olivat reaaliaikakäyttöjärjestelmä ja TCP/IP-protokollapinon toteutus, joiden laatiminen itse ei olisi tämän työn puitteissa mahdollista. Valmiita komponentteja etsittiin ensisijaisesti vapaasti saatavilla olevien tuotteiden joukosta, koska järjestelmän kokonaiskustannukset haluttiin pitää mahdollisimman pieninä. Ohjelmiston työstäminen aloitettiin käyttöjärjestelmän valinnalla. Tavoitteena oli löytää vapaasti saatavilla oleva tuote, jossa olisi mahdollisimman hyvä tuki valitulle prosessoriarkkitehtuurille ja prosessorimallille. Toissijaisena valintakriteerinä päätettiin käyttää säikeistystukea, sillä sen arveltiin helpottavan rinnakkaisten toimintojen ohjelmointia. Ensimmäiseksi tutkittiin Baytems Holdings Oy:n aiemmissa projekteissa käytössä ollutta RTEMS:iä [26]. Se oli julkaistu muutamien erillisten avoimen lähdekoodin lisenssien alaisuudessa, ja siinä oli säikeistystuki ja valmiiksi integroituna BSD Unix -järjestelmistä kopioitu TCP/IP-protokollapinon toteutus. Nopeasti kuitenkin havaittiin, että vaikka käyttöjärjestelmä tuki ARM Cortex-M -prosessoriperheen käyttämää ARM thumb -käskykantaa, ja olisi siten ajettavissa valitulla prosessorilla, siinä ei ollut mitään valmiita rakenteita STM32F4-prosessoreiden runsaiden lisälaitteiden käyttämiseen. Mahdollisen käyttöjärjestelmän etsintää päätettiin jatkaa laajemmalla laitetuella varustetun vaihtoehdon löytämiseksi.

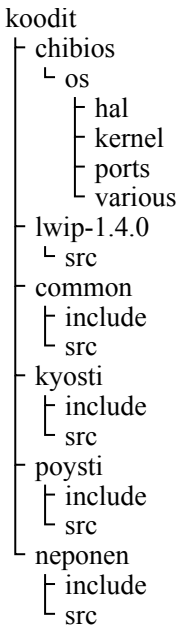
Internet-haut prosessorin mallisarjan nimellä STM32F4 ja RTOS-avainsanalla tuottivat hakutuloksiksi kaksi uutta käyttöjärjestelmäkandidaattia. FreeRTOS [27] ja ChibiOS/RT [28] osoittautuivat molemmat RTEMS:in tavoin soveltuviksi valitulle prosessorille, ja molemmissa oli käytettävissä säikeet sekä sulautettuihin järjestelmiin suunnattu TCP/IP-protokollapinon toteutus lwIP [29]. ChibiOS/RT:ssä protokollapinon toteutus oli tosin liitettävä käyttöjärjestelmään itse, mutta toimenpiteestä oli saatavilla ohjeistusta ja ohjelmaesimerkki. Vertailtaessa näitä kahta käyttöjärjestelmäkandidaattia todettiin nopeasti, että ChibiOS/RT:n oheislaitetuki oli merkittävästi parempi. Siinä oli erillinen laitteiston abstrahointikerros (Hardware Abstraction Layer, HAL), joka sisälsi laiteajurit niin $I^2C$-, SPI-, kuin MII-väyläohjaimiakin varten. Lisäksi prosessorissa oli yhdistetty $I^2S$- ja SPI-väylien käsittely samoihin laitteistolohkoihin, joten käyttöjärjestelmän sisältämästä SPI-ajurista arveltiin olevan vähäisin muutoksin tehtävissä siitä vielä puuttunut $I^2S$-ajuri. FreeRTOS ei sisältänyt valmiita laiteajureita, joten ne olisi jouduttu toteuttamaan kyseiseen käyttöjärjestelmään itse. Käyttöjärjestelmien tarjoamat rajapinnat vaikuttivat nopean selvityksen perusteella olevan dokumentoituja kaikissa kandidaateissa. RTEMS:ssä ja ChibiOS/RT:ssä dokumentointiin oli käytetty Doxygen-järjestelmää [30], FreeRTOS:ssa projektin omia menetelmiä. Käyttöjärjestelmistä

kerätyt tiedot on koottu taulukkoon 3.2. Kandidaateista asetetut kriteerit täytti selvästi parhaiten ChibiOS/RT, joten se valittiin ohjelmien toteuttamisen pohjaksi. Käyttöjärjestelmä mahdollisti ohjelmointikielivaihtoehdoiksi C:n ja C++:n, mutta C++-rajapintakääreiden tuoreuden ja mahdollisen virhealttiuden vuoksi myös oma ohjelmisto päätettiin toteuttaa C-kielellä suoraan käyttöjärjestelmän alkuperäisiä rajapintoja käyttäen.

***Taulukko 3.2.*** *Käyttöjärjestelmävaihtoehtojen vertailu*

| Nimi | RTEMS | FreeRTOS | ChibiOS/RT |
|---|---|---|---|
| Lisensointi | GPL2, BSD | muokattu GPL | GPL3 |
| Säikeistys | kyllä | kyllä | kyllä |
| TCP/IP-totetus | BSD | lwIP | lwIP |
| Laiteajureita | ei | ei | kyllä |
| Dokumentaatio | Doxygen | kyllä | Doxygen |

Käyttöjärjestelmäksi valitun ChibiOS/RT:n havaittiin tukevan useita erilaisia kääntäjiä ja käännöksen hallintaan tehtyjä järjestelmiä. Yksinkertaisin niistä oli GNU Make [31], jota käyttäen käyttöjärjestelmä oli mahdollista kääntää komentoriviltä ilman graafisia kehitysympäristöjä. Järjestelmän käyttämät Makefile-ohjaustiedosto oli paloiteltu hierarkisesti siten, että kussakin käyttöjärjestelmän lähdekoodipuun hakemistossa oli sen hakemiston sisältämien kooditiedostojen kääntämiseen tarvittavat tiedot omassa lyhyessä makefile.mk-tiedostossaan. Näiden lisäksi jokaisessa käyttöjärjestelmän mukana toimitetussa esimerkkiohjelmassa oli oma varsinainen Makefile, johon tehtyjen konfiguraatioiden perusteella käyttöjärjestelmän koodista käännettiin ja linkitettiin esimerkkiohjelman binääritiedostoon vain kussakin esimerkissä tarpeelliset osat. Yksi esimerkeistä käsitteli lwIP-komponentin liittämistä käyttöjärjestelmään, ja liitos saatiin tehtyä vaivatta tätä esimerkkiä mukaillen. Omien ohjelmabinäärien teko oli ChibiOS/RT:n Makefile-järjestelmää käyttäen erittäin helppoa, sillä omien omat Makefile-tiedostot voitiin rakentaa esimerkkien avulla ja käyttäjärjestelmän mk-tiedostoja apuna käyttäen. Järjestelmän laitteella ajettavat ohjelmat rakentuivat siis käyttöjärjestelmän koodista, kaikille ohjelmille yhteisistä alustuskoodeista ja kunkin ohjelman oman toiminnallisuuden toteuttavasta koodista. Nämä lähdekoodit järjesteltiin kuvan 3.9 mukaisesti.




***Kuva 3.9.*** *Ohjelmakoodin hakemistorakenne*

Kunkin ohjelmabinäärin yksilölliset lähdekoodit koottiin omiin hakemistoihinsa yhdessä niiden kääntämisen määrittelevän Makefile-tiedoston kanssa. Käyttöjärjestelmän eri osien ja lwIP:n asetukset sisältäneet otsikkotiedostot sijoitettiin kaikille ohjelmille yhteiseen common-hakemistoon. Myöhemmin sinne lisättiin myös äänikortilla sijaineen stereokodekkipiirin konfiguroivat funktiot sisältävä ohjelmakoodi. Common-hakemistoon sijoitettiin myös käyttöjärjestelmän kääntämisen spesifioiva Makefile-katkelma. Tämä tiedosto sisälsi esimerkiksi kääntäjän ja linkittäjän parametrit. Kokoamalla mahdollisimman suuri osa määrityksistä yhteiseen tiedostoon saatiin ohjelmakohtaiset Makefile-tiedostot pidettyä erittäin lyhyinä ja yksinkertaisina. Esimerkiksi kyosti-ohjelman Makefile-tiedosto on esitelty kokonaisuudessaan listauksessa 3.1.

Ohjelmakohtainen koodi koostui yhteisissä koodeissa olleiden alustusrutiinien kutsumisesta ja ohjelman varsinaisia tehtäviä hoitavien säikeiden käynnistämisestä. Alustustoimenpiteisiin kuuluivat käyttöjärjestelmän laitteistokerroksen konfigurointi, ytimen käynnistys sekä järjestelmän muiden piirien konfigurointi ja käynnistys. Laitteistokerroksen asetuksissa käyttöjärjestelmälle kerrottiin kaikkien käytössä olleiden prosessoripiirin pinnien toimintatila ja asetettiin prosessorin kellopuuhun tarvittavien kertoimien ja jakajien arvot. Suurin osa prosessorin pinneistä asettui prosessorin käynnistyessä yleiskäyttöiseen tilaan (General purpose input/ouput, GPIO), josta ne vaihdettiin tarpeen mukaan oheislaitelohkojen käyttöön. Prosessorin kellopuu konfiguroitiin siten, että prosessorikortilla olleesta 25 MHz kidepohjaisesta taajuusreferenssistä saatiin prosessorin kellotaajuudeksi sen suurin sallittu arvo 168 MHz. Tarvetta pienemmän kellotaajuuden käyttöön ei ollut, sillä proses-

sori ei vähäisen sähkönkulutuksensa takia juurikaan lämmennyt maksimikellotaajuudellakaan. Prosessorin kellopuusta generoitiin myös AD- ja DA-muunnokset tehneen stereokodekkipiirin tarvitsemat pääkello- ja kanavanvaihtokellosignaalit, jotka lähetettiin kodekkipiirille I$^2$S-väyläohjaimen kautta. Kellopuun I$^2$S-haaran parametrit asetettiin siten, että näytteenottotaajuus saatiin mahdollisimman lähelle 48 kilohertsiä. Prosessorin käyttöohjekirjan [32, s. 694] mukaan käyttöön valituilla optimaalisilla arvoilla todellinen näytteenottotaajuus oli 47991 hertsiä, eli noin 186 PPM tavoitearvoa vähemmän. Virhe oli kuitenkin kaikissa laitteissa ja ohjelmissa systemaattisesti sama, joten sen arveltiin olevan järjestelmän toiminnan kannalta merkityksetön.

```
# $Id: Makefile 377 2013-11-22 16:36:09Z oh2gve $

# Omat yhteiset.
include ../common/makefile.mk

# Binäärin nimi.
PROJECT = kyosti

# Binäärikohtaiset headerit.
UINCDIR += include

# Binäärikohtaiset koodit.
TCSRC += src/main.c src/aani.c src/db-taulu.c src/puskuri.c

# Chibioksen päämakefile.
include $(CHIBIOS)/os/ports/GCC/ARMCMx/rules.mk
```

***Listaus 3.1.*** *Kyosti-ohjelman Makefile-tiedosto.*

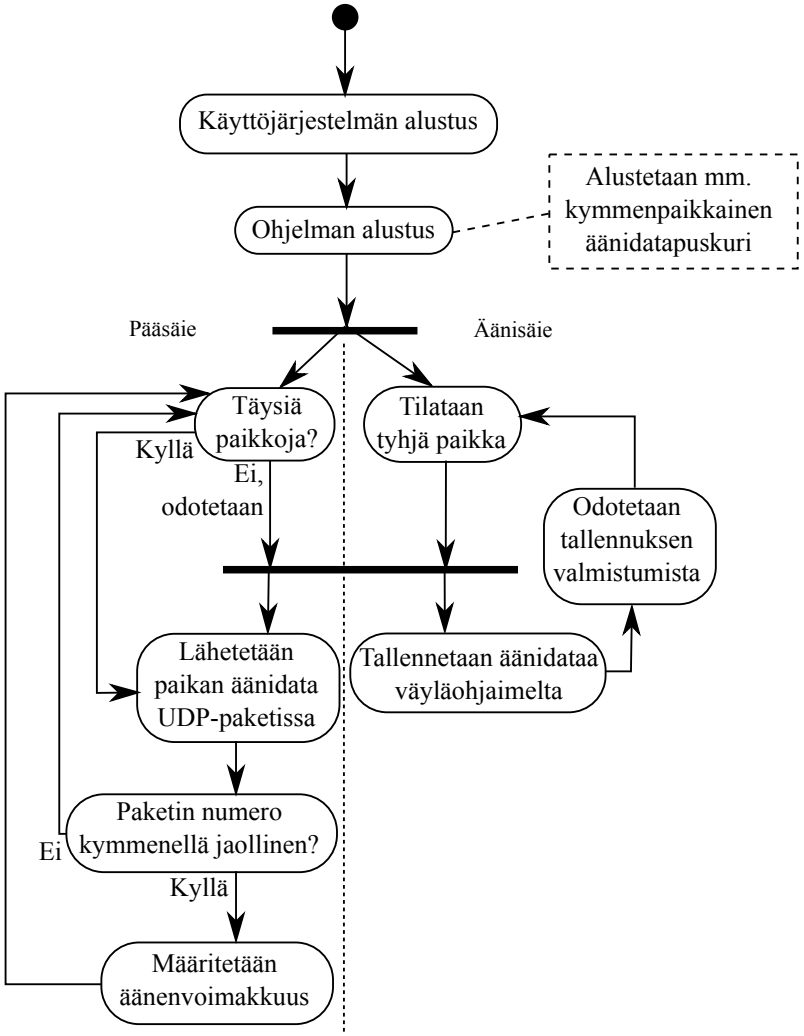
Prosessorin alustusten jälkeen käynnistettiin käyttöjärjestelmän ydin, alustettiin tarpeelliset oheislaitelohkot ja niiden avulla laitteen muut piirit. Ensimmäisenä alustettiin laitteen sarjaporttia ohjannut USART-lohko, jonka avulla saatiin lähetettyä sarjaporttiin kytketyn PC-tietokoneen kautta käyttäjälle tietoja ohjelman käynnistymisestä. Stereokodekkia varten käynnistettiin I$^2$S-lohko, jolloin kodekkipiiri sai tarvitsemansa kellosignaalit ja käynnistyi. Kodekin asetukset määriteltiin erillisenä komentokanavana toimineen I$^2$C-väylän kautta, jonka jälkeen kodekki aloitti varsinaisen toimintansa. Prosessorin ethernet-käsittelyn hoitava lohko (Medium Access Controller, MAC) ja ethernet-lähetinvastaanotin alustettiin, ja niitä käyttävä lwIP-komponetti käynnistetiin omaan säikeeseensä. Ohjelman toiminnan sisältäessä varatallenteen käsittelyä käynnistettiin myös varatallenteen käyttämä flash-muisti ja sen ohjaamiseen käytetty SPI-oheislaitelohko. Kaikille ohjelmille yhteisten alus-

tustoimien jälkeen käynnistettiin ohjelmakohtaisen toiminnon toteuttaneet säikeet. Flash-muistin täyttämiseen laadittu neponen-ohjelma toimi yhdessä säikeessä, mutta tuotantokäytössä olleissa ohjelmissa toimintoja eroteltiin useampaan säikeeseen. Näin pyrittiin pitämään ohjelman osien keskinäiset riippuvuudet mahdollisimman vähäisinä, jotta esimerkiksi ongelmat käyttöliittymän käsittelyssä eivät johtaisi äänensiirron häiriintymiseen. Toimintojen eriyttämisessä auttoi myös ChibiOS/RT:n erinomainen laitteistoabstraktiokerros, jossa datan siirto prosessorin oheislaitelohkon ja keskusmuistin välillä tehtiin aina suoraa muistiosoitusta (Direct Memory Access, DMA) käyttäen. Tiedonsiirron käynnistämiseen käytetyt järjestelmäkutsut eivät jääneet odottamaan siirron valmistumista, vaan palauttivat kontrollin välittömästi. Siirron suoritti käytännössä prosessorin erillinen DMA-ohjain, joka vapautti siirtoon kuluvan prosessoriajan muihin tarkoituksiin. Käyttöjärjestelmä ilmoitti siirtojen valmistumisesta ja siihen varattujen laiteresurssien vapautumisesta takaisinkutsufunktioiden avulla, jolloin ohjelmassa riitti käynnistää uusi siirto ja palata sen jälkeen tekemään muita asioita. Mahdollisiin alustustoimenpiteiden aikana tapahtuneisiin virheisiin reagoitiin tulostamalla käyttäjälle tilanteen salliessa virheilmoitus sarjaportin kautta ja pysäyttämällä ohjelman eteneminen saattamalla se tyhjään ikuiseen silmukkaan.

### 3.4.2 Varsinaiset ohjelmat

Ohjelmien kehitystyö aloitettiin siirtoyhteyden lähettävän pään kyosti-ohjelmasta. Sen tehtävä oli vastaanottaa digitaalista ääntä stereokodekkipiiriltä ja lähettää sitä UDP-paketeissa verkkoon. Digitaalisen äänen voimakkuutta oli myös määritettävä jatkuvasti ja päivitettävä tietoa siitä sarjaportin kautta käyttäjälle. Ohjelman keskeisin toiminta on esitelty kuvassa 3.10. Ohjelmakohtainen toiminnallisuus jaettiin kahteen säikeeseen, joista toinen keskittyi äänikortin ohjaamiseen ja toinen digitaalisen äänidatan jatkokäsittelyyn ja käyttöliittymän ylläpitoon. Ohjelmistokehityksen ohessa ilmeni, että prosessorin kolmesta SPI/I2S-lohkosta tässä kytkennässä käytössä ollut I2S3-lohko kykeni muista poiketen vain yhtäaikaiseen molempisuuntaiseen tiedonsiirtoon (Full Duplex, FD). Käytännössä tämä havaittiin siitä, että lohko lähetti $I^2S$-väylälle kellosignaaleita vain, mikäli sillä oli myös DA-muunnettavaksi lähetettävää äänidataa. Koska lähetettävää dataa ei kehitteillä olleen ohjelman käyttötapauksessa ollut, ei kellosignaalien lähetys eikä myöskään stereokodekkipiiri käynnistyneet lainkaan. Ongelma korjattiin toteuttamalla ohjelman äänisäikeeseen ylimääräinen ikuinen silmukka, joka lähetti jatkuvasti hiljaisuutta I2S3-lohkon kautta stereokodekille. Näin saatiin kellosignaalit toimimaan, ja kodekki käyntiin muuttamaan analogista ääntä digitaaliseksi. Äänisäie vastaanotti dataa $I^2$2S-väyläohjaimelta 16-bittisinä etumerkillisinä kokonaislukuina suoraan UDP-paketin hyötykuorman kokoisissa paloissa ja puskuroi niitä pääsäikeelle. Pääsäie tyhjensi puskuria paikka ker-

rallaan sijoittaen yhdestä puskurin paikasta saamansa äänidatan sellaisenaan UDP-paketin hyötykuormaosaan, jonka kahteen ensimmäiseen tavuun oli paketin luonnin yhteydessä sijoitettu paketin juokseva numero. Paketti luovutettiin tämän jälkeen verkkosäikeelle lähetettäväksi.




***Kuva 3.10.*** *Ääntä verkkoon lähettävän kyosti-ohjelman toiminta*

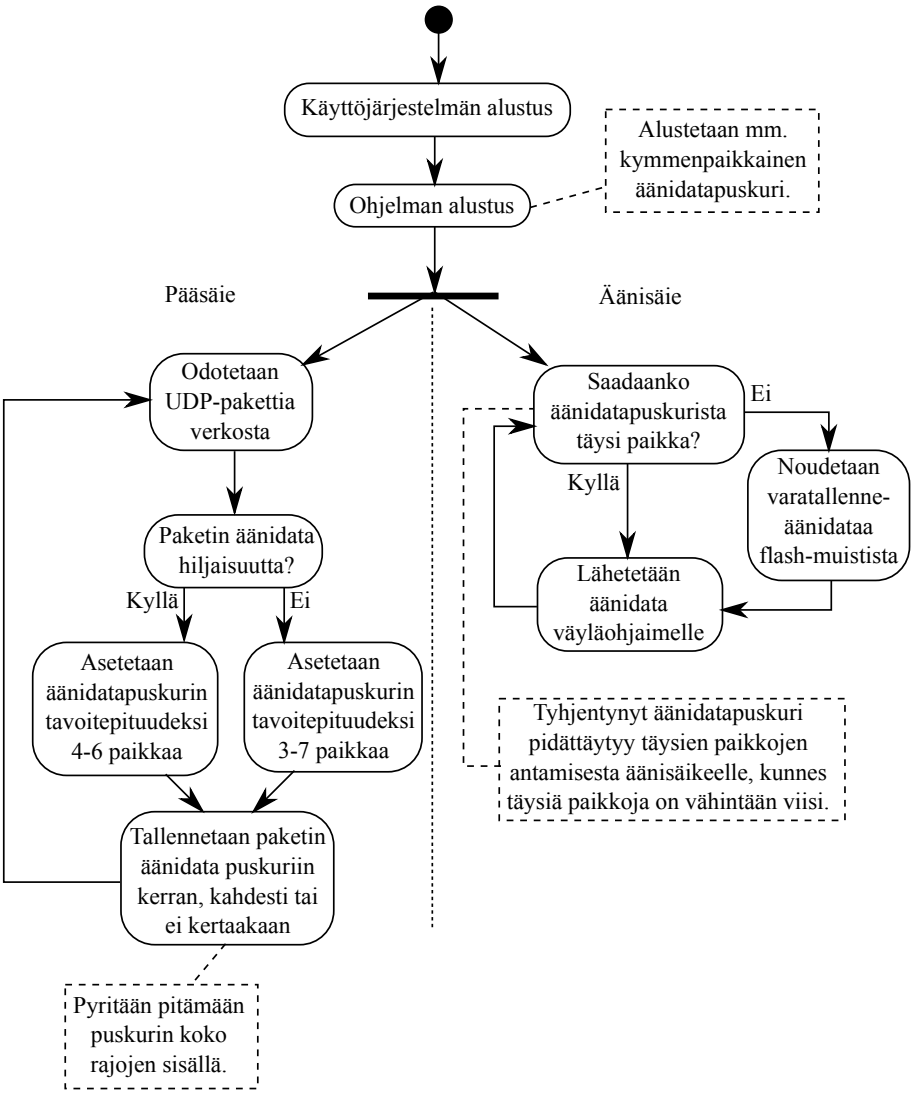
Joka kymmenennen paketin lähettämisen jälkeen pakettiin sijoitettu äänidata käytiin läpi etsien erikseen vasemman ja oikean stereokanavan suurin äänestä otetun näytteen arvo. Arvo muutettiin ohjelmakoodin sekaan sijoitetun kiinteän muunnostaulukon avulla desibeleiksi suhteessa näytteen maksimiarvon (decibels relative to full scale, dBFS) sekä graafiseksi esitykseksi, jossa äänenvoimakkuutta kuvattiin yhdelle riville ladotuilla yhdestä kolmeenkymmeneenviiteen peräkkäisellä ristikkomerkillä. Tiedot lähetettiin lopuksi sarjaportin kautta käyttäjän nähtäville. Sarjaportin sisääntuloa ei ohjelmassa luettu, eikä käyttäjälle tarjottu mitään muutakaan mahdollisuutta puuttua ohjelman toimintaan. Mahdollisiin ajonaikaisiin virhetilanteisiin reagoitiin joko vain tulostamalla virheilmoitus ja siirtymällä ohjelmassa eteen-

päin tai vakavimmissa tapauksissa pysäyttämällä ohjelman suoritus ohjaamalla se tyhjään ikuiseen silmukkaan.

Siirtoyhteyden vastaanottavan pään laitteessa toiminut poysti-ohjelma vastaanotti UDP-paketteja verkosta ja siirsi niiden kuljettaman äänidatan stereokodekkipiirille DA-muunnettavaksi. Lisäksi ohjelma soitti varatallennetta flash-muistilta, mikäli verkosta ei tullut UDP-paketteja. Toiminta on esitetty kuvassa 3.11. Ohjelma jaettiin kyosti-ohjelman tapaan kahteen säikeeseen, joista pääsäie huolehti datan siirrosta lwIP-komponentin ja äänisäikeen välillä, ja äänisäie äänidatan siirrosta stereokodekkipiirille ja varatallenteen käytöstä. Äänisäikeeseen toteutettiin kymmenpaikkainen puskuri, johon pääsäie tallensi UDP-paketeista purkamansa äänidatan. Paketin äänidata siirrettiin sellaisenaan yhteen puskurin paikkaan. Sarjanumeroa verrattiin edellisen vastaanotetun paketin sarjanumeroon, ja mikäli numerot eivät olleet peräkkäisiä, tulostettiin virheilmoitus sarjaporttiin. Toiminnon tarkoitus oli pelkästään auttaa järjestelmän käyttäjää kartoittamaan siirtovirheiden määrää, ei korjata ilmennyttä tiedonsiirron virhettä. Äänisäikeessä äänidataa purettiin puskurista paikka kerrallaan $I^2S$-väylälle lähetettäväksi. Mikäli puskuri tyhjeni, pyydettiin varatallenteen sisältävältä flash-muistilta jatkuvaa lukukomentoa käyttäen yhden muistisivun sisältö, muunnettiin se mono-muodosta stereoksi kahdentamalla jokainen näyte ja lähetettiin näin saatu data stereokodekkipiirille. Muistipiiri luki dataa pyydettäessä aina automaattisesti seuraavan sivun, ja siirtyi muistin loppuun päästyään automaattisesti muistin alkuun. Kun verkosta saapuvan äänidatan puskurista saatiin jälleen dataa, palattiin soittamaan sitä, ja lähetettiin flash-muistille komento siirtyä ensimmäisen muistisivun alkuun seuraavaa käyttötarvetta varten. Nopeat vaihtelut verkkodatan ja varatallenteen välillä estettiin toteuttamalla verkkodata puskuri siten, että se antoi äänidataa soitettavaksi vasta kun se oli tyhjentymisen jälkeen täyttynyt vähintään puolilleen.

Toteutuksen alkuvaiheessa pääsäikeen tehtäväksi suunniteltiin vain äänidatan purkaminen UDP-paketeista ja puskurointi äänisäiettä varten. Heti ensimmäisissä testeissä kuitenkin huomattiin, että puskuri tyhjenee ajan myötä siten, että jo viidentoista minuutin käytön jälkeen ohjelma soitti hetken varatallennetta täydentääkseen puskuria. Ilmiön nopeus yllätti, sillä laitteet olivat testin ajan samassa tilassa vieretysten, joten niiden oletettiin olevan samassa lämpötilassa ja käyttäytyvän muutenkin keskenään samalla tavalla. Käyttäytymisen eron perimmäistä syytä ei tässä yhteydessä ryhdytty selvittämään, sillä nopeasti ymmärrettiin, että eroja tulisi hyvin todennäköisesti olemaan tuotantokäytössä joka tapauksessa, kun laitteet sijaitsisivat erilaisissa olosuhteissa. Puskurin täyttöasteen seurantaan kehitettiin tilanteen korjaamiseksi mekanismi, joka muodosti pitkällä, usean minuutin aikavälillä kuvan puskurin täyttöasteen kehityksestä, ja pyrki sisääntulevien UDP-pakettien äänidataa tarpeen mukaan hylkäämällä tai monistamalla pitämään puskurin täyttö-

asteen ennalta määrättyjen rajojen sisällä. Kun yhden paketin sisältämä data edusti 7,5 millisekunnin ääntä, arveltiin sellaisen poisjäännin tai tuplaantumisen aiheuttavan äänisignaaliin selvän epäjatkuvuuskohdan. Vaikka nämä häiriöt olisivat verraten pieniä ja harvinaisia, päätettiin mekanismi rakentaa kaksiportaiseksi siten, että kunkin verkosta tulevan UDP-paketin sisältämän äänen voimakkuus määritettiin, ja korjausoperaatiot pyrittiin tekemään hiljaisuutta sisältäneiden pakettien kohdalla. Rajat asetettiin siten, että yhden paketin äänidata tallennettiin puskuriin kahdesti, jos puskurin keskimääräinen pituus oli laskenut alle kolmen paikan, tai hylättiin, mikäli pituus oli yli seitsemän paikkaa. Kuitenkin jos paketin sisältämän äänen voimakkuus oli enintään hiljaisuuskynnykseksi asetettu -33 dBFS, nostettiin alaraja neljään ja laskettiin yläraja kuuteen paikkaan.

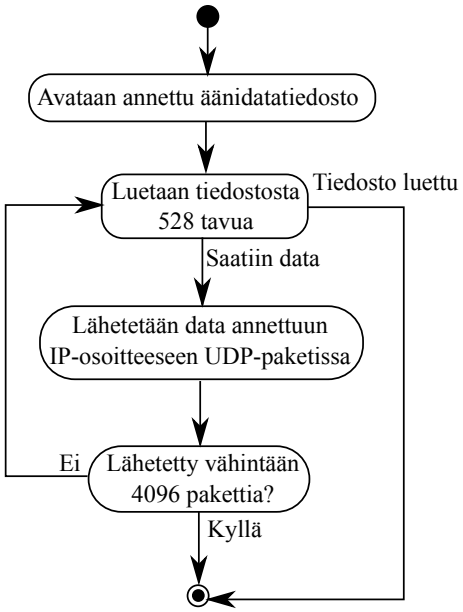



***Kuva 3.11.*** *Ääntä verkosta vastaanottavan poysti-ohjelman toiminta yksinkertaistettuna*

Puskurin pituuden keskiarvo määritettiin tallentamalla sen hetkellinen arvo 7,5 sekunnin välein ja laskemalla viidenkymmenen edellisen arvon keskiarvo. Näin saatiin käyttöön pitkän, noin kuuden minuutin aikajakson keskiarvotieto, jossa yksittäiset pakettien katoamiset tai muut lyhytaikaiset tietoverkon häiriöt eivät vaikuttaneet. Korjausmekanismin toiminnan nopeuttamiseksi edelliset viisikymmentä pituuden hetkellisarvoa alustettiin kuitenkin aina paketin hylkäämisen tai monistamisen yhteydessä puskurin senhetkiseen pituuden arvoon.

### 3.4.3 Työkaluohjelmat

Järjestelmää varten toteutettiin kaksi työkaluohjelmaa, joiden avulla laitteen flash-muistiin voitiin kirjoittaa varatallenteena käytettävä äänidata. Toinen ohjelmista oli itse laitteessa oleva neponen, jonka tehtävänä oli purkaa data UDP-paketista ja kirjoittaa se muistipiirille. Neponen-ohjelman pariksi toteutettiin Linux-PC:llä ajettavaksi tarkoitettu repomies-ohjelma, joka luki äänidatan tiedostosta ja lähetti sen laitteelle. Flash-muistin sivukoko oli 528 tavua ja piirillä oli 4096 muistisivua, joten tallennuskapasiteettia oli 2112 kilotavua. Siihen saatiin tallennettua järjestelmän 48 kHz näyteenottotaajuudella ja 16-bittisillä näytteillä noin 22,5 sekuntia monoääntä. Äänidataa PC-tietokoneelta lähettävän repomies-apuohjelman toiminta on esitelty kuvassa 3.12.

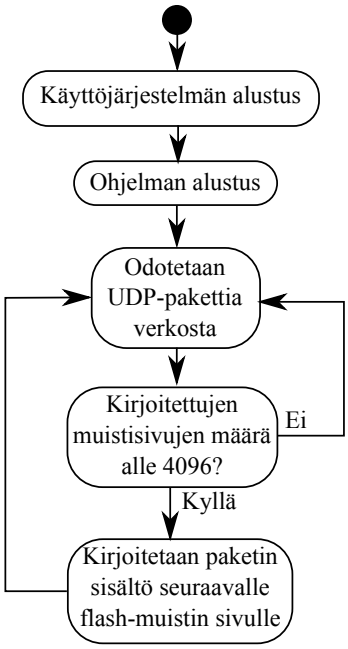



***Kuva 3.12.*** *PC-tietokoneella ajettavan repomies-apuohjelman toiminta*

Ohjelma toteutettiin C++-kielellä SDL-kirjastoa käyttäen, koska tästä yhdistelmästä oli aiempaa kokemusta UDP-pohjaisen tiedonsiirron toteutusvälineenä. Ohjelmalle annettiin komentoriviparemetreinä neponen-ohjelmaa ajavan laitteen IP-osoite tai nimi, UDP-portin numero ja tiedoston nimi. Tiedostossa tuli olla ha-

luttu varatallenne 48 kHz taajuudella näytteistettynä monoäänenä, joka on tallennettu raakana 16-bittisten näytteiden jonona käyttämättä enkoodausta tai mitään otsikkotietoja. Tiedostoa luettiin laitteen flash-muistin sivun eli 528 tavun kokoisissa pätkissä, jotka pakattiin UDP-pakettiin muuttaen näytteiden tavujärjestys x86-järjestelmän little endian -muodosta IP-verkossa ja kohdelaitteessa käytettyyn big endian -muotoon. Paketteja lähetettiin kohdelaitteelle kahdenkymmenen millisekunnin välein, jotta laite ehti kirjoittaa aina paketin vastaanotettuaan sen sisällön flash-muistiin. Ohjelman suoritus lopetettiin, kun tiedosto oli luettu ja lähetetty kokonaisuudessaan tai viimeistään kun kaikkiin flash-muistin 4096:een sivuun oli lähetetty sisältö. Kohdelaitteessa neponen-ohjelma yksinkertaisesti kirjoitti vastaanottamiensa UDP-pakettien sisällön sellaisenaan flash-muistin sivuihin.

Neponen-ohjelman ainoana tarkoituksena oli ladata prosessorikortin flash-muistille haluttu varatallenne, jota poysti-ohjelma soittaisi mahdollisissa siirtoyhteyden ongelmatilanteissa. Toiminta on esitetty kuvassa 3.13. Ohjelma varmisti, että vastaanotetun UDP-paketin hyötykuorman koko vastasi muistisivun kokoa, ja että kaikkia muistisivuja ei ollut vielä ohjelmoitu. Mikäli ehdot täyttyivät, UDP-paketin sisältämä data lähetettiin flash-muistin kirjoituspuskuriin ja annettiin kirjoituskomento. Kirjoituskomennon antamisen jälkeen ohjelma odotti kirjoituksen valmistumista ennen seuraavan UDP-paketin käsittelyyn ottoa. Paketin saapumisesta ja kirjoituksen valmistumisesta tulostettiin ilmoitukset sarjaporttiin, joiden avulla varmistettiin, että kaikki lähetetyt sivut ehdittiin kirjoittaa.




***Kuva 3.13.*** *Varatallenteen flash-muistiin tallentavan neponen-ohjelman toiminta*

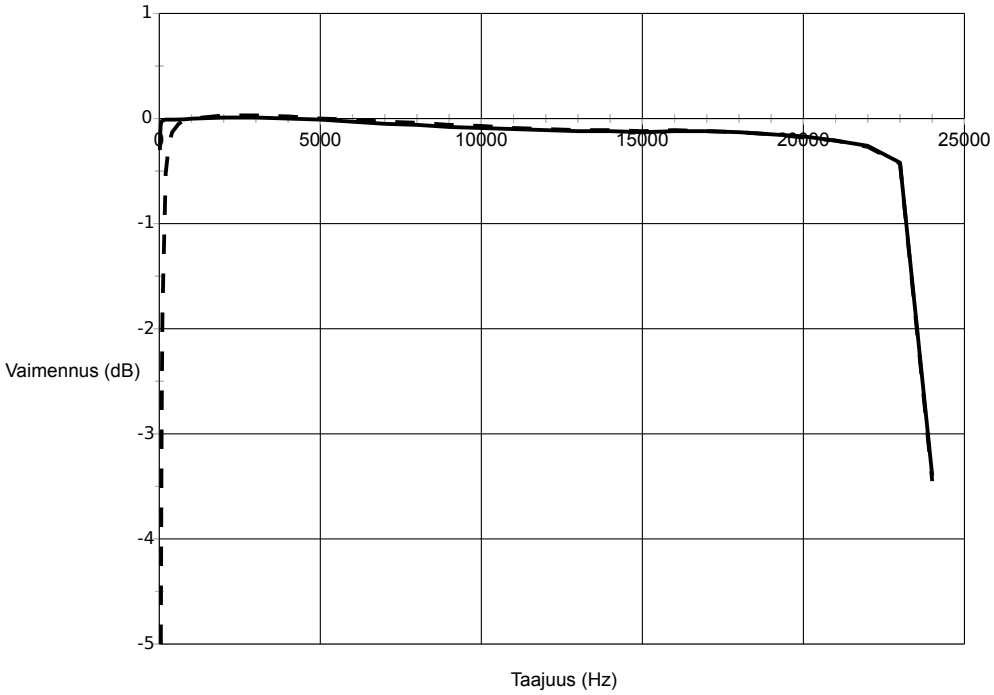
# 4 ARVIOINTI

Tässä luvussa arvioidaan toteutetun järjestelmän vaatimustenmukaisuutta ja soveltumista ajateltuun tarkoitukseen. Lisäksi esitetään joitakin jatkokehitysajatuksia.

## 4.1 Vaatimusten täyttyminen

Valmistettuun järjestelmään saatiin toteutettua kaikki vaaditut toiminnot. Järjestelmä siirsi ääntä paikasta toiseen, soitti varatallennetta siirtoyhteyden katketessa ja näytti yhteyden lähettävässä päässä signaalin äänenvoimakkuuden. Järjestelmän toiminta oli täysin automaattista, eikä käyttämiseen tarvittu Rakkauden Wappuradion lähetysteknisen henkilökunnan toimenpiteitä. Äänen kulku järjestelmän läpi aiheutti siihen yleensä noin 40 millisekunnin ja enintään 75 millisekunnin mittaisen viiveen, joka oli alle vaatimuksena olleen sadan millisekunnin. Järjestelmän verkkoliikenne oli toteutettu UDP-protokollaa käyttäen, joten se reitittyi helposti verkkojen välillä. Laitteessa oli kehittynyt häiriöiden suodatus ja studiotekniikassa usein käytetyt XLR-liitännät balansoidun äänisignaalin kytkemistä varten. Prototyyppilaitteet olivat keskenään samanlaiset, eikä ohjelmien kannalta ollut merkitystä, kummassa laiteyksilössä niitä ajettiin.

Järjestelmän taajuusvaste mitattiin Standford Research Systemsin DS345-funktiogeneraattorilla ja Agilentin 34401A-yleismittarilla sekä 600 Ω että 10 kΩ kuormia käyttäen. Mittausalueeksi asetettiin hieman laajennettu ihmisen kuuloalue 20 hertsistä 24 kilohertsiin. Testisignaali syötettiin järjestelmään balansoimattomana ja sen referenssitasoksi asetettiin 1 $V_{RMS}$ 1000 Hz taajuudella. Mittaukset suoritettiin sataan hertsiin asti kymmenen hertsin välein, sadan hertsin ja yhden kilohertsin välillä sadan hertsin välein ja suuremmilla taajuuksilla yhden kilohertsin välein. Ennen varsinaisia mittauksia tarkistettiin järjestelmän kohinataso ilman signaalilähdettä, ja se oli koko taajuuskaistalla alle 60 $mV_{RMS}$. Mittaustulokset on esitetty kuvassa 4.1.




***Kuva 4.1.*** *Järjestelmän taajuusvaste. Katkoviiva edustaa 600 Ω ja yhtenäinen viiva 10 kΩ kuormaa.*

Taajuusvaste on mittaustulosten perusteella hyvin tasainen. Vaimennus referenssitasoon nähden on 10 kΩ kuormaimpedanssilla 20 Hz taajuudella 0,3 dB ja 20 kHz taajuudella 0,17 dB. Pienemmällä 600 Ω kuormalla taajuusvaste on matalemmilla taajuuksilla hieman vaimeampi kolmen desibelin rajan ylittyessä noin 70 hertsin kohdalla, mutta muulla taajuuskaistalla vaste on lähes täsmälleen sama kuin suuremmalla kuormalla. Vaimentuma johtuu äänikortin ulostulossa olevista tasavirtakomponentin suodattavista sarjakondensaattoreista, ja on sinänsä normaali. Linjatasoisen signaalin liittymien impedanssin oli havaittu olevan monissa studiolaitteissa 10 kΩ, joten järjestelmän taajuusvasteeseen oltiin erittäin tyytyväisiä. Taajuusvastemittausten yhteydessä tarkkailtiin myös järjestelmän virrankulutusta, joka oli kummallakin laitteella 12 voltin käyttöjännitteellä tasainen ja erittäin kohtuullinen 160 mA.

Tässä työssä toteutettua järjestelmää on vertailtu aiempien vuosien järjestelmiin taulukossa 4.1. Taulukkoon on koottu ominaisuuksia, joiden osalta järjestelmien välillä on merkittäviä eroja.

***Taulukko 4.1.*** *Uuden äänensiirtojärjestelmän vertailu edellisiin*

| Vuosi | 2010 | 2011 | 2012 |
|---|---|---|---|
| Laitteiston arvo uutena | tuhansia euroja | tuhansia euroja | noin 400 euroa |
| Tehonkulutus | 100 - 200 W | 100 - 200 W | 4 W |
| Kylmäkäynnistys | käsin | käsin | automaattisesti |
| Siirtoviive | 3 - 30 s | 5 - 10 s | 30 - 75 ms |
| Asetusten muuttaminen | graafisesti | komentoriviltä | ei |

Aiempien vuosien järjestelmät olivat pohjautuneet Rakkauden Wappuradion käyttöön lainattuihin yleiskäyttöisiin tietokoneisiin, joiden hankintahinnat uutena olivat yhteensä useita tuhansia euroja. Yhden tässä työssä valmistetun laitteen kustannukset olivat noin kaksisataa euroa. Yleiskäyttöisten tietokoneiden tehonkulutuksen arveltiin olleen sadan watin luokkaa per laite, kun tässä työssä valmistettu laite käytti toimiessaan hieman alle kaksi wattia sähköä. Käytettävyydessä erot olivat myös merkittäviä, sillä aiempien vuosien käsin tehtävää konfigurointia vaatineet järjestelmät eivät käynnistyneet automaattisesti esimerkiksi sähkökatkon jälkeen. Uuden järjestelmän käynnistyminen oli puolestaan täysin automaattista, eikä operaattorin tarvinnut laitteiden paikalleen kytkemisen jälkeen puuttua niiden toimintaan millään tavalla. Merkittävimmäksi uuden järjestelmän puutteeksi edellisiin nähden voitiin lukea asetustietojen, esimerkiksi IP-osoitteiden muuttamisen vaikeus, sillä niiden vaihtamiseen ei ollut mitään keinoa ilman laitteiden uudelleenohjelmointia.

Työkaluohjelmat toteutettiin roolinsa vuoksi nopeasti vain vähimmäisvaatimukset täyttäväksi, joten niiden käytössä oli monia rajoitteita. Laitteen flash-muistia ei esimerkiksi tyhjennetty kokonaisuudessaan, vaan vasta kirjoittamisoperaatioiden yhteydessä sivu kerrallaan. Varatallennetiedoston oli aina oltava vähintään koko muistin eli 2112 kilotavun mittainen, ettei mahdollista flash-muistin aiempaa sisältöä soitettaisi varatallenteena. Varatallennetta asetettaessa tallennetiedosto piti käsitellä tietokoneella ennalta juuri oikeaan muotoon, koska laitteelle siirtäneet ohjelmat eivät kovertoineet tai tarkistaneet dataa mitenkään. Nämä eivät kuitenkaan haitanneet Rakkauden Wappuradion tapauksessa järjestelmän käyttöä, sillä käyttöönottovalmistelut tehtiin kehitysprojektin osana järjestelmän valmistuttua.

Järjestelmää oli kehityksen aikana testattu jatkuvasti, kehitystyön loppuvaiheessa myös useiden vuorokausien ajan yhtäjaksoisesti. Testeissä oli löydetty virheitä ja ne oli korjattu, joten luottamus järjestelmän toimintaedellytyksiin ja -varmuuteen oli hyvä. Vuoden 2012 Rakkauden Wappuradion FM-lähettimen äänensiirtoyhteys päätettiin toteuttaa uutta järjestelmää ja sen prototyyppilaiteyksilöitä käyttäen. Järjestelmän toiminta oli koko lähetyksen ajan moitteetonta.

## 4.2 Kehitysideoita

Järjestelmää kohtaan on esitetty paljon mielenkiintoa jo sen kehitystyön alkuvaiheista lähtien, ja se on saanut osakseen myös paljon kehitysideoita useilta eri tahoilta. Tärkeimpänä voidaan pitää mahdollisuutta muuttaa laitteen asetuksia ilman uudelleenohjelmointia. Prosessorikortille on jo prototyyppilaitteissa sijoitettu EEPROM-muistipiiri, johon asetustieto voitaisiin tallentaa. Asetuskäyttöliittymän toteuttamiseen tulisi kiinnittää erityistä huomiota, jotta sen avulla ei olisi mahdollista häiritä laitteen varsinaista toimintaa.

Laitteiden kytkentää Rakkauden Wappuradion lähetystekniikan valvontajärjestelmään on toivottu. Esimerkiksi yksinkertaista verkonhallintaprotokollaa (Simple Network Management Protocol, SNMP) käyttäen voisi olla mahdollista kysellä ja lähettää laitteista tietoja, joiden perusteella voidaan tehdä automaattisia toimenpiteitä ja hälyttää operaattoreita. Esimerkiksi välitettävän äänisignaalin voimakkuustiedon perusteella olisi mahdollista tehdä hälytyksiä liian suuresta äänenvoimakkuudesta tai siitä, että tuottaja on unohtanut avata mikrofonit musiikkikappaleen soittamisen jälkeen.

Järjestelmää on ajaltetu sovellettavan muuallakin kuin studion ja lähettimen välisessä äänensiirrossa. Ajatus teekkariwapun tapahtumissa kiertämisestä on ollut läsnä Rakkauden Wappuradion alkuajoista lähtien, ja sitä on suunniteltu toteutettavaksi esimerkiksi pienen etälähetysyksikön muodossa. Rakennettu järjestelmä olisi mahdollisesti sovellettavissa äänen siirtoon etäyksikön ja päästudion välillä, ja siihen voitaisiin rakentaa nykyisessä käytössä käyttämättä jäävien XLR-liitinten kautta komentokanava päästudiolta etäyksikköön.

Monet kehitysideat tulisivat mahdolliseksi toteuttaa, jos järjestelmän tietoliikenne ei edellyttäisi lähes virheetöntä siirtoyhteyttä. Pienillä virheenkorjaustoiminnoilla voitaisiin mahdollisesti parantaa vikasietoisuutta niin, että esimerkiksi langattoman lähiverkon käyttö siirtoyhteytenä olisi mahdollista. Mikäli vikasietoisuus saataisiin erittäin hyvälle tasolle, olisi järjestelmän käyttäminen mobiiliverkkojen yli mahdollista. Tällöin voitaisiin harkita esimerkiksi Rakkauden Wappuradion kiertävän toimittajan varustamista järjestelmän reppuun mahtuvalla kannettavalla laitteistolla, jolloin toimittajan tekemä ohjelma välittyisi studiolle vaivattomasti ja äänenlaadultaan erinomaisena.

Yhden pisteestä pisteeseen etenevän yhteyden lisäksi järjestelmää voitaisiin soveltaa myös yhdestä pisteestä moneen etenevän äänen siirtämiseen. Esimerkiksi erillisiin huoneisiin tai halleihin jakaantuvissa messutiloissa olisi mahdollisuus pystyttää nopeasti ja vaivattomasti langaton kenttäkuulutusjärjestelmä WLAN-verkoon kytkettyjen yhden lähettävän ja monen vastaanottavan laitteen avulla.

# 5 YHTEENVETO

Tässä työssä esiteltiin Rakkauden Wappuradio ja käytiin perusteellisesti läpi sen käytössä olleet studion ja lähettimen väliset äänensiirtoyhteydet. Läpikäynnin pohjalta muodostettiin vaatimukset sulautetulle järjestelmälle. Niistä keskeisimpiä olivat luotettava äänensiirto paikasta toiseen IP-verkon yli, alle sadan millisekunnin siirtoviive, yleiskäyttöisiä tietokoneita edullisempi ja yksinkertaisempi laitteisto sekä automaattinen käynnistyminen ja virhetilanteista toipuminen. Lisäksi lähettimeen kytketyn laitteen oli siirtoyhteyden katketessa soitettava ennalta asetettua varatallennetta yhteyden palautumiseen asti.

Järjestelmän laitteisto koostuu kahdesta samanlaisesta laitteesta, jotka sijoitetaan siirtoyhteyden päihin. Laitteessa on yhdessä kotelossa toisiinsa liitettynä ARM Cortex M4 -prosessoriin perustuva prosessorikortti, sekä AD- ja DA-muunnokset tekevän stereokodekkipiirin ja äänielektroniikan sisältävä äänikortti. Laitteisiin toteutettiin kolme ohjelmaa, joista kahta käytetään äänensiirtokäytössä yhteyden eri päissä ja kolmatta varatallenteen tallentamiseen ennalta laitteen flash-muistiin. Lisäksi toteutettiin Linux-PC:llä käytettävä apuohjelma, jolla varatallenne lähetetään laitteelle. Laitteessa ajettavien ohjelmien pohjana käytettiin ChibiOS/RT-reaaliaikakäyttöjärjestelmää ja lwIP-nimistä TCP/IP-protokollapinon toteutusta. Kaikki konfiguraatiotieto tallennettiin ohjelmiin käännösaikaisesti, jotta se oli käytettävissä heti ohjelman käynnistyessä ilman, että erillistä asetusten tekemiseen soveltuvaa käyttöliittymää oli tarpeen toteuttaa.

Järjestelmän toteutuksessa pystyttiin vastaamaan kaikkiin vaatimuksiin, ja lopputuloksena saatiin hyvin toimiva sulautettu äänensiirtojärjestelmä. Kehitystyön aikaisissa testeissä ja valmiilla laitteilla tehdyissä mittauksissa järjestelmä havaittiin erittäin vakaaksi ja suorituskykyiseksi. Se soveltui Rakkauden Wappuradion käyttöön erinomaisesti, ja toimi radion käytössä vuoden 2012 ja 2013 lähetysten ajan moitteetta.

# LÄHTEET

[1] Wappu [WWW], [viitattu 3.2.2012]. Saatavissa: http://www.students.tut.fi/~wappu/.

[2] Winamp Media Player [WWW], [viitattu 3.2.2012]. Saatavissa: http://www.winamp.com/.

[3] SHOUTcast-lähetystyökalut [WWW], [viitattu 3.2.2012]. Saatavissa: http://www.shoutcast.com/broadcast-tools/.

[4] Icecast Streaming Multimedia Server [WWW], [viitattu 22.9.2012]. Saatavissa: http://www.icecast.org/.

[5] VLC media player [WWW], [viitattu 6.2.2012]. Saatavissa: http://videolan.org/vlc/.

[6] MPlayer - The Movie Player [WWW], [viitattu 6.2.2012]. Saatavissa: http://mplayerhq.hu/.

[7] mpg321 komentorivipohjainen MP3-toisto-ohjelma [WWW], [viitattu 6.2.2012]. Saatavissa: http://mpg321.sourceforge.net/.

[8] Mayah Centaur III [WWW], [viitattu 18.10.2013]. Saatavissa: http://www.mayah.com/products/CIII/index.htm.

[9] APT Codecsin IP-kodekkien tuoteluettelo [WWW], [viitattu 3.2.2012]. Saatavissa: http://www.aptcodecs.com/radio-32.html.

[10] Mayah Centaur III ominaisuudet [WWW], [viitattu 3.2.2012]. Saatavissa: http://www.mayah.com/products/CIII/cent3-features.htm.

[11] R. Ellingson and M. Dille, "Dynamic range considerations when designing pc sound card based audiometric systems to test human hearing," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 5907–5910.

[12] FFADO - Free Firewire Audio Drivers [WWW], [viitattu 17.5.2012]. Saatavissa: http://www.ffado.org/.

[13] Jack Audio Connection Kit [WWW], [viitattu 12.12.2013]. Saatavissa http://jackaudio.org/.

[14] Advanced Linux Sound Architecture [WWW], [viitattu 12.12.2013]. Saatavissa http://www.alsa-project.org/.

[15] Open Audio Layer [WWW], [viitattu 9.6.2012]. Saatavissa: http://openal.org.

[16] Rämö, Anssi. Diplomi-insinööri, Senior Researcher, Voice Applications at Nokia Research Center Finland. Tampere. Haastattelu 17.2.2012.

[17] IEC 61076-2-103. Connectors for electronic equipment - Part 2-103: Circular connectors - Detail specification for a range of multipole connectors (type 'XLR'). Sveitsi 2004, International Electrotechnical Commission. 69 s.

[18] W. Stallings, *Data and Computer Communications*, 7th ed. Pearson Education, 2004.

[19] Cirrus Logic CS4272 -kodekkipiirin datalehti [WWW], [viitattu 13.3.2013]. Saatavissa: http://www.cirrus.com/en/pubs/proDatasheet/CS4272_F1.pdf.

[20] I²S-väylän määrittelydokumentti [WWW], [viitattu 21.10.2013]. Saatavissa: https://sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf.

[21] Cirrus Logic CS4272 -kodekkipiirin koekytkentälevyn datalehti [WWW], [viitattu 24.10.2013]. Saatavissa: http://www.cirrus.com/en/pubs/rdDatasheet/CDB4272-2.pdf.

[22] STM32F407ZG-mikroprosessorin tuote-esittely [WWW], [viitattu 10.3.2013]. Saatavissa: http://www.st.com/internet/mcu/product/252136.jsp.

[23] STM32F405xx- ja STM32F047xx-mikroprosessoreiden datalehti [WWW], [viitattu 15.3.2013]. Saatavissa: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf.

[24] ARM JTAG 20 -liityntä [WWW], [viitattu 24.3.2013]. Saatavissa: http://infocenter.arm.com/help/topic/com.arm.doc.dui0499b/BEHEIHCE.html.

[25] Simple DirectMedia Layer [WWW], [viitattu 26.10.2013]. Saatavissa: http://www.libsdl.org/.

[26] RTEMS Real Time Operatin System (RTOS) [WWW], [viitattu 24.3.2013]. Saatavissa: http://www.rtems.org/.

[27] FreeRTOS - Market leading RTOS for embedded systems [WWW], [viitattu 24.3.2013]. Saatavissa: http://www.freertos.org/.

[28] ChibiOS/RT free embedded RTOS [WWW], [viitattu 24.3.2013]. Saatavissa: http://www.chibios.org/, viitattu 24.3.2013.

[29] lwIP - A Lightweight TCP/IP stack [WWW], [viitattu 31.3.2013]. Saatavissa: http://savannah.nongnu.org/projects/lwip/.

[30] Doxygen-dokumentaation generointijärjestelmä [WWW], [viitattu 23.11.2013]. Saatavissa http://www.doxygen.org/.

[31] GNU Make [WWW], [viitattu 22.11.2013]. Saatavissa http://www.gnu.org/software/make/.

[32] STM32F407ZG-mikroprosessorin käyttöohjekirja [WWW], [viitattu 24.11.2013]. Saatavissa http://www.st.com/web/en/resource/technical/document/reference_manual/DM00031020.pdf.