

Julius Stenros

Sulautetun VoIP-puhelimen suunnittelu ja toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

10.5.2013

Tekijä Otsikko	Julius Stenros Sulautetun VoIP-puhelimen suunnittelu ja toteutus
Sivumäärä Aika	40 sivua 10.5.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	tietoverkkotekniikka
Ohjaaja	päätoiminen tuntiopettaja Juho Vesanen
<p>Insinööritöiden tarkoitus oli suunnitella ja toteuttaa nykypäiväisessä tietoverkossa toimiva puhelin. Työn ideana oli käyttää hyväksi mahdollisimman laajasti tietotekniikan eri suuntautumisvaihtoehtojen oppeja. Tavoitteena oli luoda toimiva prototyyppi, jonka avulla pystytään osoittamaan suunnitelman toteutettavuus.</p> <p>Projekti alkoi selvittämällä tarvittavat komponentit prototyypin rakentamiseen. Näihin kuului alusta jolle puhelinta alettiin rakentaa, äänen vastaanottoon ja toistamiseen vaadittavat komponentit, sekä eri tiedonsiirtoprotokollat.</p> <p>Prototyypin toteutuksen edetessä törmättiin haasteisiin, jotka asettivat rajoituksia alkupe- räisen suunnitelmaan. Muistin pienen määrän sekä prosessorin huonon suorituskyvyn vuoksi puhelinta ei pystytty saamaan yhteensopivaksi muiden valmistajien puhelinten kanssa.</p> <p>Vastoinikäymisistä huolimatta lopullinen prototyyppi saatiin toimimaan vakaasti hyväksyttävällä äänenlaadulla ja loistavalla vasteajalla. Projektin edetessä tuli opittua suunnattomasti tietoverkoissa toimivien puhelinten toimintatavoista ja standardeista.</p>	
Avainsanat	sulautettu, VoIP, Arduino, puhelin, A-Law, DAC

Author(s) Title	Julius Stenros Designing and implementing an embedded VoIP phone
Number of Pages Date	40 pages 10 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Computer Networks
Instructor	Juho Vesanen, Lecturer
<p>The purpose of the thesis was to design and implement a computer network capable phone. The idea of the project was to utilize as broadly as possible what was learned while studying different aspects of information technology. The goal was to create a working prototype which would serve as a proof-of-concept.</p> <p>The project started by researching the necessary components needed to build the prototype. Among these was the platform where the phone would be developed on, components for receiving and reproducing sound, and different communications protocols.</p> <p>Several obstacles were encountered in the process of creating the prototype – obstacles that resulted in restrictions on the original plan. Because of the limited amount of memory and CPU processing power, the phone could not be made compatible with the phones of other vendors.</p> <p>Despite all of the obstacles, the final prototype operated reliably with a decent sound quality and low latency. During the project a great deal was learned about how phones in general operate in computer networks and what kind of standards they use.</p>	
Keywords	embedded, VoIP, Arduino, phone, A-law, DAC

Sisällys

Lyhenteet

1	Johdanto	1
2	Prototyyppi	3
2.1	Komponentit	3
2.2	Alusta	3
3	Verkkoprotokollat	7
3.1	OSI-malli	7
3.2	Sovelluskerros	9
3.2.1	Esimerkkipuhelu sovelluskerroksen protokollilla	12
3.3	Testiympäristö	12
4	Äänen vastaanotto ja toisto Arduinossa	14
4.1	Vastaanotto	14
4.2	Toisto	17
4.3	Koodekki	19
4.4	Testaus	21
5	Arduinon rajoitteet	23
5.1	Kirjaston rajoitteet	23
5.1.1	Ominaisuuksien puute	23
5.1.2	Taaksepäin yhteensopimattomuus	24
5.2	Toimintavirheiden etsiminen	24
5.3	Muistirajoitteet	25
6	Oma protokolla	27
6.1	Rakenne	27
6.2	Logiikka	28
7	Lopullinen prototyyppi	30
7.1	Suorituskyky	30
7.2	Testipuhelu käytännössä	31
8	Pohdintaa	33

8.1	Käyttökohteet	33
8.2	Jatkokehityskohteet	33
8.3	Opittua	34
9	Yhteenveto	36
	Lähteet	38

Lyhenteet

ADC	Analogi-digitaalimuunnin. ADC muuttaa äänilähteestä tulevan värähtelysignaalin ykkösiksi ja nolliksi
DAC	Digitaali-analogimuunnin. DAC muuttaa ykköset ja nollat äänisignaaleiksi.
DC	<i>Direct Current</i> . Tasavirta, eli virta, joka kulkee virtapiirissä koko ajan samansuuntaisesti.
I/O	<i>Input/Output</i> . I/O-liitäntöjen kautta välitetään tietokoneelle ja sieltä pois informaatiota.
ICE	<i>In-Circuit Emulator</i> . Sulautetussa järjestelmässä käytetty virheenkorjaustyökalu.
IPv4	<i>Internet Protocol version 4</i> . Verkkokerroksen protokolla, jonka tarkoitus on mahdollistaa kommunikaatio eri verkkosegmenttien välillä.
ISO	<i>International Organization for Standardization</i> . Kansainvälinen standardoimisjärjestö.
MAC	<i>Media Access Control</i> . Ethernet-protokollan osa, joka hoitaa osoitteistuksen, sekä osan liikennöimisestä.
OSI-malli	<i>Open Systems Interconnection</i> in malli, jossa tiedonsiirtoprotokollat on esitetty kerroksittain.
PBX	<i>Private Branch Exchange</i> . Puhelinvaihde, joka välittää yhteyksiä puhelinten välillä.
PSoC	<i>Programmable System on a Chip</i> . Cypress Semiconductorin tuottama mikro-ohjain.
RTCP	<i>RTP Control Protocol</i> . RTP:n toimintaa seuraava ja ohjaava protokolla.

RTP	<i>Real-time Transport Protocol</i> . Internet Engineering Task Forcen suosittelema sovelluskerroksen VoIP-protokolla standardi, joka hoitaa osapuolten informaation kuljetuksen.
SCCP	<i>Skinny Call Control Protocol</i> . Cisco Systemsin laitevalmistajariippuvainen sovelluskerroksen VoIP-protokolla.
SDP	<i>Session Description Protocol</i> . Internet Engineering Task Forcen suosittelema sovelluskerroksen VoIP-protokolla standardi, joka määrittelee SIP:in parametreja.
SIP	<i>Session Initiation Protocol</i> . Internet Engineering Task Forcen suosittelema sovelluskerroksen VoIP-protokolla standardi, jolla määritellään keskustelun osapuolien istunnot.
TCP	<i>Transmission Control Protocol</i> . Luotettava kuljetuskerroksen protokolla.
UDP	<i>User Datagram Protocol</i> . Epäluotettava kuljetuskerroksen protokolla.
VoA	<i>Voice over Arduino</i> . Insinööriyötä varten kehitetty sovelluskerroksen verkkoprotokolla.
VoIP	<i>Voice over Internet Protocol</i> . Internetin yleisimmin käytetyn tiedonsiirto-protokollan (IP) avulla toteutettua puheensiirtojärjestelmää.

1 Johdanto

Ihmisten välinen kommunikaatio on mahdollistanut ihmiskunnan kehityksen nykyiseen muotoonsa. Verbaalinen viestintä toimii kulmakivenä ideoiden ja ajatusten välityksessä. Puhelimen keksiminen poisti rajoituksen olla fyysisesti samassa paikassa toisen osapuolen kanssa. Keksinnön jälkeen puhelimen kehitys on haarautunut moneen eri suuntaan. Lankapuhelimet ovat edelleen yleisiä ja matkapuhelimet ovat kokeneet räjähdysmäisen kasvun. Puhelimia alkaa löytyä yhä enemmän myös tietokoneilta.

Minua kiehtoi ajatus rakentaa puhelin yksinkertaisella tietokoneella, koska siinä yhdistyy tämän hetken modernein tekniikka suhteellisen vanhan keksinnön kanssa.

Tavoitteeni oli rakentaa toimiva, käsin kosketeltava käytännön malli sulautetusta tietoverkossa toimivasta puhelimesta. Työn sisältöön kuuluu tutkia millä tavalla on mielekästä toteuttaa eri mikro-ohjaimilla, verkkoprotokollilla ja pakkausalgoritmeilla modernissa tietoverkossa toimiva puhelin.

Aiheen valinta perustui myös henkilökohtaiseen haluun hyödyntää mahdollisimman laajasti tietotekniikan koulutusohjelman tarjoamia eri oppeja. Työssä yhdistyvät sulautetut järjestelmät, tietoverkot sekä ohjelmistotekniikka.

Sulautetut järjestelmät ovat laitteita, jotka on suunniteltu toteuttamaan tiettyä tehtävää ja niiden toimintaa ohjaa yksinkertainen tietokone. Tällaisia laitteita löytyy esimerkiksi autoista, viihde-elektroniikasta sekä kodinkoneista. Rakentamalla puhelimen yksinkertaisen tietokoneen ympärille voidaan puhua sulautetusta puhelimesta.

Tietoverkoilla tarkoitetaan eri koneiden välistä tiedonsiirtomenetelmää. Tietoverkkoihin kuuluu koneiden välisen kommunikaation toimintaperiaate sekä infrastruktuuri. Jotta sulautettu puhelin kykenisi siirtämään tietoa toiseen puhelimeen, täytyy sen toimintaan implementoida tuki tietoverkoissa liikennöimiseen.

Insinööriyön nimessä esiintyvä VoIP [1] viittaa termiin Voice over Internet Protocol. VoIP tarkoittaa Internetin yleisimmin käytetyn tiedonsiirtoprotokollan (IP) avulla toteutettua puheensiirtojärjestelmää. Protokolla [2] on joukko sääntöjä ja toimintatapoja, jotka määräävät eri laitteiden väliset keskustelun kulut sekä parametrit. Protokollat voi

mieltää kielenä, jossa molemmat ymmärtävät tosiaan puhuessaan samalla protokollalla.

Ohjelmistotekniikka käsittelee tietokoneen toimintatapaa ohjaavien käskyjen joukkoa, jota kutsutaan tietokoneohjelmaksi. Toimiakseen täytyy sulautetulle VoIP-puhelimelle antaa suunnaton määrä käskyjä ja ehtoja eri tilanteisiin, jotta se suoriutuu sille tarkoitusta tehtävästä. Ohjelmointi, jolla tarkoitetaan tietokoneohjelmien suunnittelua ja toteutusta, muodostaa suurimman osan tämän projektin työmäärästä.

Tietokoneet suorittavat ajattelematta kaikki niille annetut käskyt. Monimutkaisia ohjelmia kirjoittaessa on inhimillistä tehdä virheitä ja vaikea pitää kokonaisuus selkeänä. Hyvin suunnitellut ohjelmatkin saattavat toimia ei-halutulla tavalla pienen virheen vuoksi. Tavoitteena ohjelmoinnissa on kirjoittaa joukko käskyjä, jotka saavat laitteen toimimaan jokaisella käyttökerralla samalla tavalla ja sille tarkoitetun käyttötarkoituksen mukaisesti.

Uutta laitetta rakennettaessa ensimmäistä teknisen konseptin toimivuuden tarkistamista varten rakennettua versiota kutsutaan prototyypiksi. Prototyyppiä ei ole tarkoitettu lähetettäväksi loppuasiakkaille, vaan sen tarkoitus on toimia iteratiivisten parannusten lähtökohtana. Insinööriyön lähtökohtana on suunnitella prototyyppi.

Tämä dokumentti on jaettu useaan osaan, joista ensimmäisessä tutkitaan, mitä elementtejä tarvitaan, jotta voidaan rakentaa toimiva prototyyppi. Kun tarvittavat elementit on käyty läpi, siirrytään testiympäristön luomiseen ja käytännön implementaatioon. Projektin edetessä törmättiin seuraavaksi haasteisiin, jotka ratkaistiin, ja lopuksi esitellään lopullinen prototyyppi ja sen käyttö- ja kehityskohteet.

2 Prototyyppi

2.1 Komponentit

Jotta voidaan rakentaa toimiva puhelin, tarvitaan ensinnäkin äänilähteen taltioija sekä toistaja. Taltioijana käytetään analogi-digitaalimuunninta (ADC). ADC muuttaa äänilähteestä tulevan värähtelysignaalin ykkösiksi ja nolliksi. ADC:n toimintaa kuvataan dokumentin luvussa 4.1, mutta karkeasti sanoen voidaan tässä vaiheessa kuvitella sen muuntavan äänisignaalin konetulkittavaan muotoon.

Äänilähteen toistajana käytetään digitaali-analogimuunninta (DAC). DAC toimii päinvastaisesti ADC:n logiikkaan nähden, eli muuntaa ykköset ja nollat äänisignaaliksi.

Jotta ADC:stä tulevaa informaatiovirtaa voidaan käsitellä, tarvitaan mikro-ohjain. Mikro-ohjaimen tehtävä on käsitellä ja tulkita siihen liitettyjen komponenttien tarjoamaa informaatiota.

Jotta puhelimen tuottavaa ääntä voidaan lähettää ja vastaanottaa tietoverkkojen yli tarvitsee mikro-ohjaimelle luoda logiikka keskustella verkkolaitteitten kanssa. Tähän tehtävään tarvitaan useita eri verkkoprotokollia, jotka esitellään tarkemmin luvussa 3. Tärkeää näiden valinnassa on, että ne ovat yhteensopivia tämänhetkisten verkkolaitteiden kanssa. Puhelin on tarkoitus pystyä liittämään mihin tahansa moderniin tietoverkkoon.

Tietoverkkoihin liittäminen vaatii myös oman liitännänsä, mikä tarkoittaa sitä, että sen toteuttaminen vaatii myös omat komponenttinsa.

2.2 Alusta

Työn ensimmäinen askel oli valita, mille alustalle prototyyppiä aletaan rakentaa. Kaikkien sulautettujen järjestelmien sydämen muodostaa mikro-ohjain. Mikro-ohjain muodostuu prosessorista, muistista ja Input/Output (I/O) -liitännöistä. Joillekin mikro-ohjaimille on olemassa myös omia kehitysympäristöjä, jotka on luotu vähentämään suunnittelijan ohjelmointitaakkaa.

Hyvin valittu mikro-ohjain voi vähentää tarvittavaa työmäärä myös siltä osin, että ihan kaikkea laitteistoa ei tarvitse itse suunnitella ja toteuttaa.

Mikro-ohjainta valittaessa kiinnitettiin huomiota hintaan, suorituskykyyn, kehitysympäristöön, liitäntöihin sekä aikaisempaan kokemukseen alustasta.

Kolme kilpailijaa valittiin taistelemaan paikasta - Atmel megaAVR [3], Arduino [4] sekä Cypress Semiconductor Programmable System on a Chip (PSoC) [5]. Huomattakoon, että näistä kaikista on useita eri versioita, mutta vertailussa oletetaan käytettävän suorituskykyisintä projektin alussa olemassa olevaa mallia. Taulukossa 1 on listattuna mikro-ohjainten tiedot, joista hinta on Farnellin yksikköhinta. Farnell on elektronisten komponenttien verkkokauppa.

Taulukko 1. Eri alustojen tiedot.

Nimi	Kellotaajuus	Muisti (SRAM)	Digitaalisia liitäntöjä	Analogisia liitäntöjä	Hinta
PSoC	24 MHz	2 KB	64 In/Out	12 In, 4 Out	10,93 e
Atmel megaAVR	16 Mhz	8 KB	86 In/Out	16 In	18,48 e
Arduino Mega	16 MHz	8 KB	54 In/Out	16 In	32,64 e

Kaikista edellä mainituista alustoista tutuin oli PSoC. Olin suorittanut projekteja PSoC:ille aikaisemminkin, joten lähtökohtaisesti tunsin suurta vetovoimaa jatkaa samalla linjalla. PSoC:in vahvuutena toisiin alustoihin nähden on sen kyky ohjelmoida mikro-ohjaimen sisäisiä liitäntöjä ohjelmallisesti. PSoC:ista on useita eri versioita, mutta tähän projektiin soveltuvassa versiossa oli sisäänrakennettu ohjelmoitava DAC ja ADC, joiden liitännät pystyivät ohjelmallisesti määrittämään eri piirilevyllä sijaitseviin pinneihin. PSoC sisältää myös helposti lähestyttävän graafisen kehitysympäristön sen eri sisäisten komponenttien sekä ajastimien hallintaan.

PSoC:in heikkoutena on aikaisemmissa projekteissa huomattu sen tehottomuus. Vaikka PSoC:in sydämenä toimii vertailukohteita korkeammalla kellotaajuudella toimiva mikroprosessori, on sille annettavien käskyjen optimoinnissa toivomisen varaa. Suuri osa tehosta häviää siihen, että prosessori joutuu käyttämään useita kellojaksoja yksittäisten käskyjen toteuttamiseen. Käskyllä tarkoitetaan prosessorin operaatiota, joka voi olla esimerkiksi muistilohkosta lukemista tai kahden luvun vertailemista keskenään.

Käskyn monimutkaisuudesta riippuen pahimmassa tapauksessa PSoC:illa menee 13 kellojaksoa toteuttaa yksi käsky [6]. Kontrastina on AVR, jossa yhden käskyn toteuttamiseen menee korkeintaan 3 kellojaksoa.

Alustan tuli pystyä prosessoimaan huomattava määrä käskyjä sekunnissa, koska sen täytyi vastaanottaa ja lähettää ääntä, joka muodostuu useita tuhansia kertoja sekunnissa mitattavista amplitudilukemista. Dokumentin luvussa 4 kuvataan tarkemmin äänen vastaanotto ja lähetys. Alustan piti pystyä myös kommunikoidaan verkkolaitteiden kanssa, jotka lähettävät ja vastaanottavat erilaisia viestejä useita kymmeniä, ellei satoja kertoja sekunnissa.

PSoC suljettiin pois kilpailusta, koska pelko sen suorituskäytännömyydestä tässä tehtävässä oli liian suuri.

Jäljelle jäi Atmelin megaAVR ja Arduino. Arduino-mikro-ohjain käyttää myös Atmel megaAVR:ää, joten kysymykseksi jäi, tarjoavatko Arduinon lisäominaisuudet projektiin soveltuvaa lisäarvoa. Arduinon vahvuudet löytyvät siihen myytävistä laajennuskorteista ja kehitysympäristössä sijaitsevista kirjastoista [7]. Kirjastolla tässä tapauksessa tarkoitetaan kokoelmaa aliohjelmia, jotka on kirjoitettu valmiiksi ohjelmoijan työtaakan keventämiseksi.

Yksi Arduinon myytävä laajennuskortti on Ethernet shield, joka sisältää tarvittavat fyysiset komponentit Arduinon liittämiseen tietoverkkoon. Koska Ethernet shield on Arduinon myytävä virallinen laajennuskortti, on sille olemassa myös kehitysympäristöstä valmiiksi kirjastot, jotka sisältävät alemman tason logiikan verkkoyhteyksien muodostamiseen ja ylläpitoon. Jo pelkästään tämän takia Arduino voitti kilpailun pelkkää AVR:ää vastaan, koska työmäärä tietoverkkokomponenttien toteuttamiseen itse on suuri.

Arduinon Ethernet shieldin keskeisenä komponenttina toimii Wiznet W5100-mikro-ohjain [8], josta minulla oli aikaisempaa kokemusta PSoC:ille kehitettävistä projekteista. Minulla oli epäily, että vaikka Arduino tarjoaa kirjastot Ethernet shieldin hallintaan, saattaisi tulla vastaan tilanne, jossa joutuisin selvittämään vikaa pintaa syvemältä. Se taas tarkoittaisi vianselvitystä Wiznet W5100:n ja Arduinon väliltä, mutta koska mikro-ohjain oli ennestään tuttu, olisin varautunut kohtaamaan mahdollisesti tulevat haasteet.

Myöhemmin projektissa huomattiin, että Arduinon aluksi vetovoimainen kehitysympäristö osoittautui rajoittavaksi tekijäksi (käydään läpi luvussa 5) muun muassa ohjelman toimintavirheiden etsimisessä (tarkemmin: ei-halutun toiminnallisuuden kitkemisessä) sekä kehitysympäristön kirjastojen eri versioiden taaksepäin yhteensopimattomuudessa ja rajoituksissa.

3 Verkkoprotokollat

3.1 OSI-malli

Hierarkkista International Organization for Standardizationin (ISO) Open Systems Interconnection (OSI) -mallia [9] hyväksi käyttäen voidaan esittää verkkoprotokollat eri kerroksissa. Protokollapinon alimmaisena toimii fyysinen kerros, joka määrittää tietoverkoissa kulkevan tiedon median. Tämä voi olla esimerkiksi kupari- tai valokaapeli.

Pinoa ylemmäs mentäessä lisääntyy verkossa kulkevan tiedon abstraktio. Toisin sanoen alimmalla tasolla määritellään, miten ykköset ja nollat kulkevat kaapelissa, toisella tasolla määritellään, miten ykkösistä ja nolista muodostuvat tietoryppäät kulkevat kahden verkkolaitteen välillä, kolmannella tasolla määritellään, miten alemmalta tasolta saatu tieto reititetään eri verkkolohkosta toiseen ja niin edelleen.

OSI-mallin eri kerrokset ovat riippumattomia toisistaan, mikä mahdollistaa sen, että eri kerrosten väliset rajapinnat ovat aina yhteensopivia. Tämän takia eri kerroksissa olevat standardit voidaan valita käyttötarkoituksesta riippuen aina vapaasti ilman ristiriitaisuuksia muiden kerrosten kanssa.

Arduinon lisätty Ethernet shield toteuttaa OSI-mallissa käytettävät neljä alinta kerrosta, koska siihen on sisäänrakennettu yhteensopivuus tiettyjen OSI-mallin standardien kanssa.

OSI-mallin alimmalla eli *fyysisellä* kerroksella Ethernet shield toimii 100BASE-TX standardilla. 100BASE-TX standardi on yhteensopiva käytännössä kaikkien kuluttajille myytävien langallisten verkkolaitteiden kanssa. Nopeudeltaan 100BASE-TX on enemmän kuin riittävä, sillä 100 megabitin sekuntivauhti kattaa helposti tarvittavan yleisenä ohjenuorana pidetyn puheelle tarvittavan 64 kilobitin sekuntivauhdin [10].

OSI-mallin toisella eli *siirtoyhteyserroksella* Ethernet shield toimii IEEE 802.3 standardilla [11], joka on yleisemmin tunnettu nimellä ethernet. Ethernet-standardin yhteensopivuus on sama kuin edellä mainitun 100BASE-TX:n. Standardi sisältää toimintalogiikan kahden verkkolaitteen väliseen kommunikaatioon. Ethernet jakaa fyysisen tason tarjoaman datavirran pieniin osiin. Dataosien ympärille lisätään myös lähettävän ja vastaanottavan verkkolaitteen MAC-osoite (Media Access Control) eli fyysinen osoite sekä

tiedon korruptoitumisen havaitsemiseen käytettävä tieto, eli tarkistussumma. Datan ja sitä ympäröivien tietokenttien kokonaisuutta kutsutaan kehyksiksi (frame).

OSI-mallin kolmannella eli *verkkokerroksella* käytetään IP-protokollan (Internet Protocol) neljättä versiota (IPv4). IPv4 on tällä hetkellä internetin käytetyin protokolla [12]. Se mahdollistaa eri verkkosegmenttien kommunikoinnin keskenään. IPv4 koteloi siirtoyhteyserrokselta saamansa datan omilla tietokentillään, jotka sisältävät muun muassa lähettäjän ja vastaanottajan IP-osoitteen, eli verkkolaitteen loogisen osoitteen, sekä Ethernetistä tutun tarkistussumman. Datan ja sitä ympäröivien tietokenttien kokonaisuutta kutsutaan paketeiksi (packet). Ethernet shield tukee IPv4 protokollaa.

OSI-mallin neljännen eli *kuljetuskerroksen* kaksi tunnetuinta protokollaa ovat Transmission Control Protocol (TCP) ja User Datagram Protocol (UDP). TCP on niin sanottu yhteydellinen ja UDP yhteydetön protokolla. Yhteydellinen TCP muodostaa kahden verkkolaitteen välille keskustelun, joka täytyy avata ja purkaa. TCP haluaa tietyin väliajoin vastauksen lähettämiinsä viesteihin ja jos se huomaa, että jokin paketti on hukunut matkan varrella, pyytää se keskustelun toista osapuolta lähettämään paketin uudestaan. TCP pitää myös kirjaa siitä, että paketit saapuvat kohteeseen oikeassa järjestyksessä. Tietoverkoissa perättäiset paketit saattavat reitittyä eri teitä pitkin kohteeseen ja reitistä riippuen ne saattavat saapua eri viiveillä.

UDP:ssa ei ole mekanismeja havaita pakettihävikkiä tai pitää kirjaa pakettien järjestyksestä. UDP on yksinkertaisempi protokolla ja tästä syystä myös kevyempi siltä osin, että sen lisäämät tietokentät ovat kooltaan pienempiä, kuin TCP:n. Tässä vaiheessa on tärkeää huomata, että vaikka UDP ei pidä kirjaa pakettien järjestyksestä, ei se tarkoita sitä etteikö sellaista palvelua voisi tarjota ylemmillä kerroksilla.

Reaaliaikaisissa sovelluksissa, kuten esimerkiksi puheluissa, Internet-radioissa, verkkopeleissä tai videoneuvotteluissa ei ole mielekästä käyttää TCP:tä sen tarjoaman pakettihävikkimekanismin vuoksi. Vastaanottajalla on väistämättä viive huomata pakettihävikin tapahtuneen ja siinä vaiheessa, kun lähettäjältä pyydetty kadotettu paketti saapuu uudestaan, on kulunut jo niin pitkään, että se olisi pitänyt jo toimittaa sitä käyttävälle sovellukselle. Toisin sanoen vastaanottaja ei hyödy mitään siitä, että se vastaanottaa ääninäytteen, joka olisi pitänyt jo toistaa.

Esimerkiksi liikkuvan kuvan tapauksessa ei ole mitään hyötyä esittää jälkikäteen jotain kuvasta puuttuvaa osaa myöhemmin tulevassa kuvassa. Tästä syystä TCP on käyttökeltoton reaaliaikaisissa sovelluksissa ja projektinkin tapauksessa päädyttiin käyttämään UDP:ta. Ethernet shield tukee sekä UDP:ta, että TCP:tä.

3.2 Sovelluskerros

OSI-mallin ylimmällä kerroksella eli sovelluskerroksella määritellään nimensä mukaisesti sovellusriippuvaisia palveluita. VoIP-protokollia on useita (monella laitevalmistajalla omansa), joista listaan sovellusten käyttämistä protokollista tunnetuimmat: Skinny Call Control Protocol (SCCP), H.323 [13], ja Session Initiation Protocol (SIP) [14]. Listan ensimmäinen on laitevalmistajariippuvainen ja kaksi jälkimmäistä ovat kahden eri standardiorganisaation suosittelemia protokollia. Projektin VoIP-protokollaksi valittiin SIP sen avoimuuden ja levinneisyyden vuoksi.

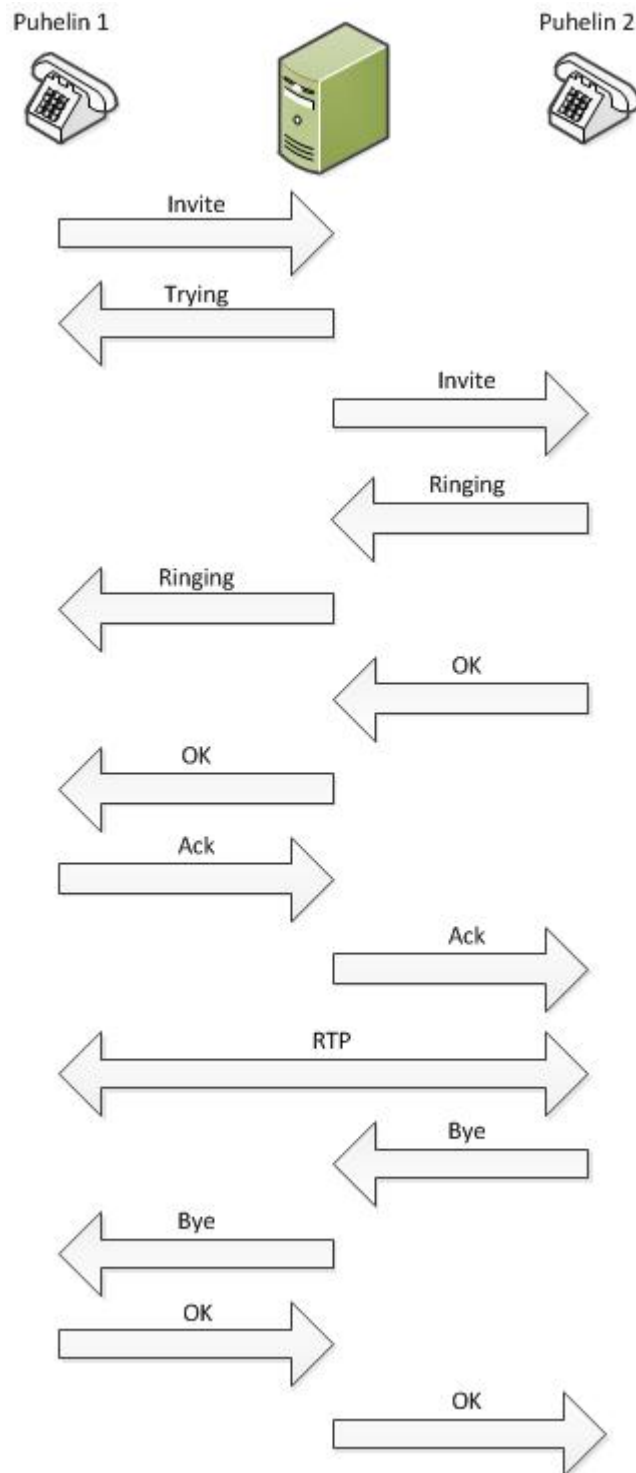
SIP-protokolla

SIP vaatii toimiakseen kolme muuta sovelluskerroksen protokollaa, jotka ovat Session Description Protocol (SDP) [15], Real-time Transport Protocol (RTP) [16] sekä RTP Control Protocol (RTCP) [15]. Edellä mainittujen protokollien käyttötarkoitus käydään läpi yksi kerrallaan.

SIP käyttää kahdenlaisia viestejä – pyyntöjä (request) ja vastauksia (response). SIP-pyyntöihin vastataan SIP-vastauksilla. SIP-viestit kulkevat PBX:n eli Private Branch Exchangen (PBX) läpi. Puhelinvaihteen toimintaan kuuluvat muun muassa pitää kirjata puhelinnumeroista ja puhelujen muodostamisesta. SIP-viesteihin kuuluu esimerkiksi: "puhelin soi", "varattu", "puheluun vastattiin" jne. Kuva 1 sisältää esimerkkipuhelun SIP-viestein.

Puhelin 1 lähettää ensin Invite-pyyntönsä PBX:lle ilmoittaen, että se haluaa aloittaa puhelun Puhelimen 2 kanssa. PBX kuittaa saaneensa pyynnön Puhelimelta 1 lähettämällä Trying-vastauksen ja välittää Invite-pyyntönsä Puhelimelle 2. Puhelin 2 ilmoittaa PBX:lle, että puhelin soi lähettämällä Ringing-vastauksen. Vastaus välitetään Puhelimelle 1. Puhelin 2 lähettää OK-vastauksen, kun puhelimen kuuloke nostetaan. OK-vastaus menee PBX:n läpi Puhelimelle 1, joka lähettää Ack-pyyntönsä PBX:n läpi Puhe-

limelle 2. Ack-pyyntö on viimeinen viesti, joka lähetetään ennen puhelun muodostusta ja sitä seuraa kaksisuuntainen tiedonsiirto verkkoprotokollalla, joka käydään läpi myöhemmin tässä luvussa. Puhelu katkeaa, kun luuri lasketaan alas ja Puhelin 2 lähettää PBX:n kautta Bye-pyyntön, joka kuitataan OK-vastauksella.



Kuva 1. SIP-viestit esimerkipuhelussa.

SDP-protokolla

SDP määrittelee puhelun tai videoneuvottelun parametrit. Näihin parametreihin kuuluvat muun muassa yhteyden käyttämä media (audio/video), median pakkausformaatti (esitelty luvussa 4.3) ja salausavain (esitelty luvussa 8.2). Yksi määriteltävistä parametreista on myös se, miten media kuljetetaan vastaanottajan ja lähettäjän välillä – projektin tapauksessa RTP:llä.

RTP-protokolla

RTP:tä käytetään median kuljetuksessa. Kuten aikaisemmin mainittiin, ei UDP:lla ole olemassa mitään mekanismia havaita pakettihävikkiä tai pitää kirjaa pakettien järjestyksestä. RTP:n jokaisessa viestissä on järjestysnumero sekä aikaleima, jolla paikataan UDP:n puutteet.

Mikäli pakettihävikkiä huomataan, toisin sanoen saadun paketin järjestysnumero on paljon korkeampi kuin edellisen, turvaudutaan yleensä näyttämään esimerkiksi videon tapauksessa edellistä kuvaa. Pakettijärjestyksestä huolen pitäminen on tärkeää, jotta ääni tai video tulee esitettyä oikealla tavalla.

Aikaleimaa käytetään havaitsemaan verkkoviiveen vaihtelua eli jitteriä. Verkkoviive on dynaaminen asia, ja VoIP-sovellus voi verkkoviiveen vaihtelun määrän perusteella säädellä esimerkiksi joko äänen tai kuvan laatua huonommaksi, jotta se veisi vähemmän kaistaa tai kasvattaa puskuria suuremmaksi, jotta pätkimistä ei pääse tapahtumaan. Aikaleimaa käytetään hyväksi myös ryhmäpuheluihin osallistuvien osapuolten keskenään synkronoinnissa.

Se, miksi pelkkää aikaleimaa ei voi käyttää pakettihävikin seurantaan on että puheluisa tulee hetkiä, jolloin jompikumpi tai kumpikaan eivät puhu mitään. Näinä hetkinä kaistan säästämiseksi ei lähetetä ollenkaan viestejä, ja mikäli seurattaisiin pelkkää aikaleimaa, tehtäisiin oletus, että paketteja on hävinnyt matkan varrella.

RTCP-protokolla

RTCP pitää kirjaa RTP:n kuljettamista paketeista. RTCP lähettää raportteja, joissa näkyy yhteenvetona muun muassa kuinka paljon pakettihävikkiä on tapahtunut, kuinka

paljon verkkoviivettä on havaittavissa ja kuinka paljon verkkoviive vaihtelee. RTCP laskee nämä käyttäen hyväksi RTP:n järjestysnumeroa ja aikaleimaa [16].

3.2.1 Esimerkkipuhelu sovelluserroksen protokollilla

Seuraavassa on esimerkki puhelun kulusta: puhelun aloittaja soittaa puhelimellaan toimivaan puhelinnumeroon. Aloittajan puhelin lähettää SIP-viestin PBX:lle, joka osoittaa, että puhelu halutaan muodostaa edellä mainittuun puhelinnumeroon. SIP-viestin jatkeeksi lisätään SDP-viesti, joka sisältää keskustelun parametrit – tässä tapauksessa, että puhelin, josta soitetaan vastaanottaa ja lähettää ääntä, joka on näytteistetty 8000 Hz:n taajuudella, ja se kuljetetaan hyväksi käyttäen RTP:tä.

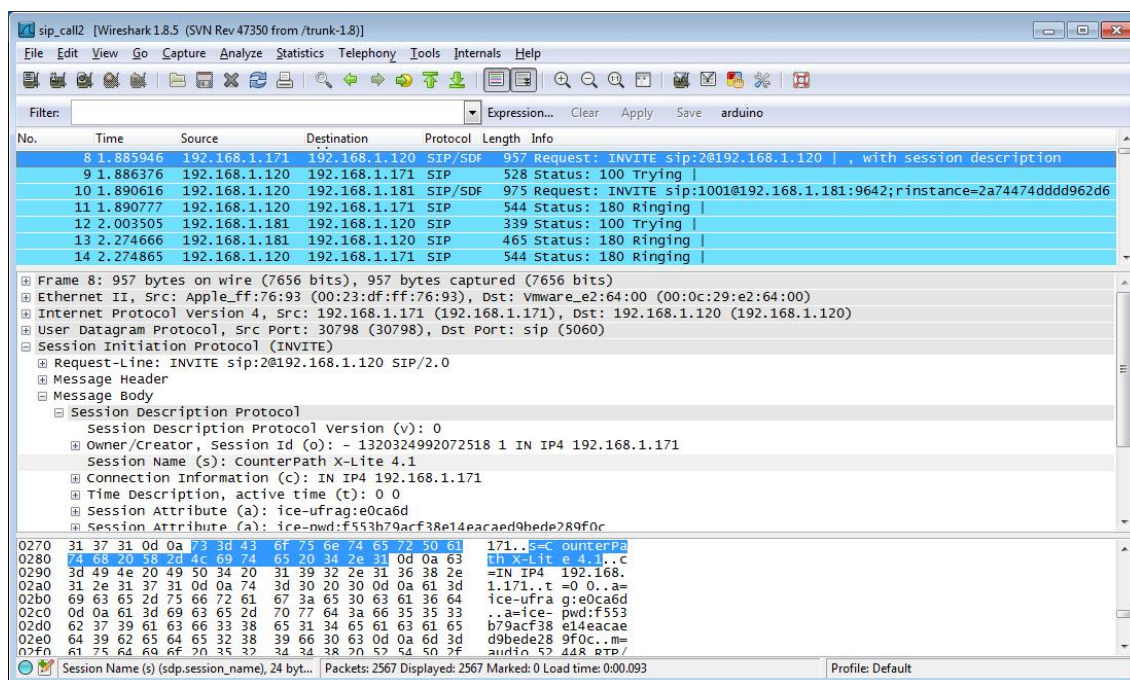
PBX kuittaa puhelun aloittajalle SIP-viestillä, että puhelu yritetään muodostaa. PBX välittää puhelun aloittajan soittopyynnön puhelinluettelostaan löytyvälle puhelimelle. PBX välittää puhelun aloittajalle SIP-viestin, jossa kerrotaan, että vastaanottajan puhelin soi. Vastaanottaja vastaa puheluun ja tästä lähtee SIP-viesti PBX:lle. PBX välittää viestejä puhelinten välillä siihen asti, kunnes puhelu on muodostettu. Puhelun muodostuttua RTCP lähettää ensimmäisen raporttinsa ja alkaa pitää kirjaa RTP-paketeista. RTP-paketit kuljettavat ääntä sisältävän informaation puhelinten välillä, kunnes puhelu päätetään SIP-viestillä (joka kulkee PBX:n läpi). Kaikki edellä mainittu liikenne kulkee myös UDP:n, IP:n ja ethernetin avulla.

3.3 Testiympäristö

Paras tapa ymmärtää, miten puhelu muodostuu, on tutkia, miten muiden valmistajien puhelimet toimivat. Loin testiympäristön, jossa pystyin kaappaamaan puhelun aikana muodostuvaa liikennettä. Testiympäristöni koostui virtuaalikoneelle asentamastani Ubuntu-käyttöjärjestelmästä [17], johon oli asennettu Asterisk [18], kahdelle Mac OS X-käyttöjärjestelmälle [19] asennetusta CounterPath X-Lite [20] -VoIP-sovelluksesta ja Wireshark [21] -pakettikaappaus- ja -analysointityökalusta.

Asterisk on vapaan lähdekoodin PBX-sovellus, joka tukee suuren määrän eri verkko-protokollia ja äänenpakkausalgoritmeja. Asteriskille piti määrittää käytössä olevat puhelinnumerot ja se mitä sovellustason protokollaa puhelimet käyttävät puhelujen muodostamiseen, tässä tapauksessa SIP:iä.

X-Lite-sovelluksille määriteltiin puhelinnumero sekä käytettävä sovellustason protokolla (SIP). Testipuhelun liikenne taltioitiin Wireshark-ohjelmalla. Wireshark kuuntelee kaikkea koneen läpi kulkevaa liikennettä ja esittää sen suodatettavien tavoin. Testipuhelusta kaapattu Wireshark-tuloste toimi tulevaisuudessa referenssinä ja testiympäristö nimen-
sä mukaan valmiina prototyypin edetessä toiminnan testauksessa. Kuvassa 2 on ruu-
dunkaappaus Wiresharkilla taltioidusta puhelusta.



Kuva 2. Testipuhelun tuloste Wiresharkissa.

4 Äänen vastaanotto ja toisto Arduinossa

Äänisignaali koostuu näytteistä. Näyte on signaali hetkittäinen amplitudin tila, joka esitetään numerona. Amplitudi on ääni- tai muun värähtelyn suuruuden mitta. Näytteistystaajuus määrittelee, kuinka usein äänisignaalista otetaan näyte ja ITU-T:n G.711-standardi [22] määrittelee puheelle 8000 Hz taajuuden (8000 näytettä sekunnissa). Toinen näytteistyksessä määritetty arvo on resoluution eli mittauksessa käytetty erotteilykyky. Toisin sanoen resoluutio määrittää, kuinka tarkkaan näyte pystytään mittaamaan. ITU-T:n G.711-standardina puheluissa on pidetty 8 bittiä. 8 bittiä pyöristää näytteen numeraaliseen muotoon 0:n ja 255:n välillä. Yhdellä bitillä esitetään joko nolla tai ykkönen, ja kahdeksalla bitillä pystytään esittämään $2^8 = 256$ eri tilaa.

4.1 Vastaanotto

Arduinon sisäänrakennettu ADC näytteistää vastaanottamansa signaalin jännitetason 10 bitillä (0 – 1023), joka on enemmän kuin tarpeeksi, jos miettii ITU-T:n G.711-standardia. Arduinon kutsuessa funktiota, joka lukee sille määritetyn vastaanottopinnan jännitteen arvon, täytyy sille kertoa, mihin muuttujaan tieto talletetaan. Muuttujat ovat muistista varattuja lohkoja lukujen tallentamiseen, jotka voidaan nimetä niiden toimintatarkoituksen mukaisesti.

Arduinon sisääntulo mittaa jännitettä 0:n ja +5 voltin (V) väliltä. Analoginen äänisignaali on kuitenkin aaltoliikettä, jonka huiput vaihtelevat negatiivisesta positiiviseen. Tästä syystä signaalia tulee käsitellä passiivisten komponenttien avulla ennen sen esittämistä Arduinolle. Passiiviset komponentit ovat elektroniikan komponentteja, jotka eivät vaadi energiaa toimiakseen. Passiiviset komponentit eivät myöskään voi itsessään tuottaa tehoa, mutta niiden avulla voi lisätä tai vähentää virtaa tai jännitettä (teho on virran ja jännitteen tulo).

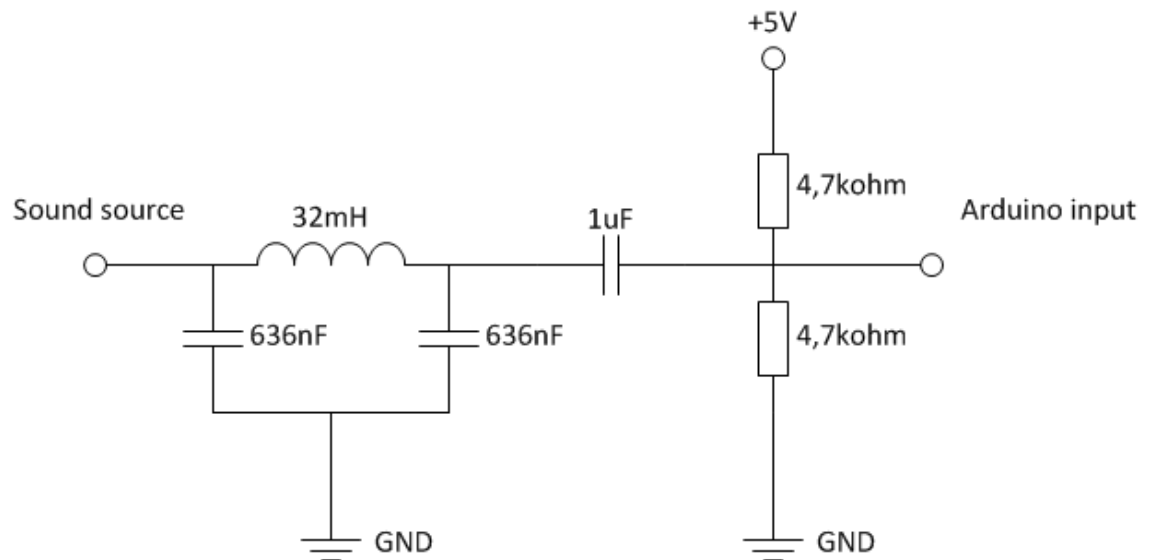
Ensimmäisenä halutaan signaalista poistaa sen mahdollinen tasajännite eli Direct Current (DC) -komponentti. Mikäli signaalissa esiintyy DC-komponentti, voi se vääristää signaalin nollatasoa. Äänisignaalin nollataso eli sen muodostama keskiarvo tulisi olla mahdollisimman lähellä nollaa. DC-komponentista pääsee eroon laittamalla kondensaattorin signaalin vastaanottimen eteen. Tavanomaisen tietokoneen stereo-ulostulon signaalitason huiput vaihtelevat negatiivisen ja positiivisen välillä. Testikokoonpanossa

käytettiin kannettavan tietokoneen linjaulostuloa, jossa jännite vaihteli -5 mV:n ja +5 mV:n välillä. Helpoin tapa muuttaa tietokoneen ulostulo Arduinon mittaamaan 0 V:n ja 5 V:n välille on nostaa signaalin tasoa +2,5 V:n tasajännitekomponentilla ja käyttää signaalin nollakohtana +2,5 V. Tämän voi toteuttaa tuomalla Arduinosta sen toimintajännitteen (+5 V) vastuksen läpi. Vastus tulee mitoittaa niin, että siihen kuluu +2,5 V ja audiosignaaliin jää täten lisättäväksi +2,5 V.

Näytteistystaajuuden ollessa vain 8000 Hz täytyy ottaa huomioon laskostuminen (aliasing) [23]. Laskostuminen eli korkeataajuuselementtien muodostama signaalin vääristymä on ei-haluttu ilmiö, joka tapahtuu näytteenottotaajuuden ollessa liian pieni signaaliin verrattuna. Nyquistin teoreeman [23] mukaisesti näytteistettävän signaalin taajuus saa olla enintään puolet näytteistystaajuudesta, jotta signaali voidaan esittää alkuperäisessä muodossaan. Käytännössä tämä tarkoittaa sitä, että kaikki 4000 Hz:ä korkeammat taajuudet aiheuttavat vääristymää 8000 Hz:n näytteenottotaajuudella.

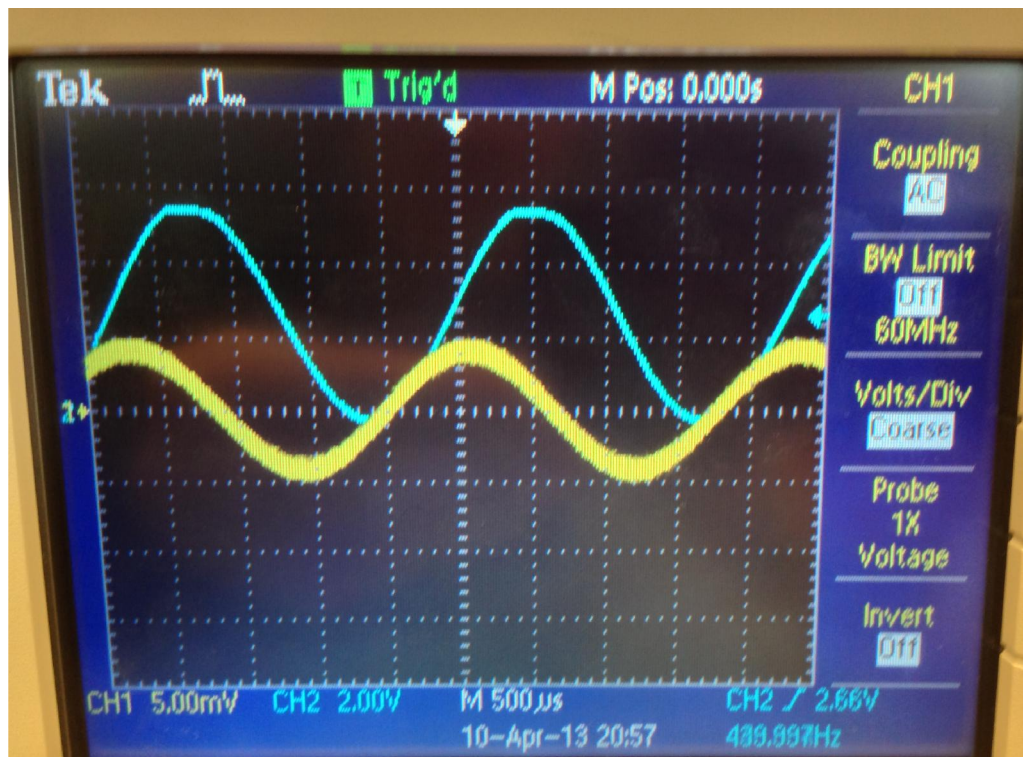
Edellä mainitusta syystä signaalista halutaan karsia yli 4000 Hz:n taajuudet. Tämä voidaan toteuttaa jakosuotimella, joka suodattaa vain halutut taajuudet lävitseen. Korkeita taajuuksia suodattaessa halutaan käyttää alipäästösuodinta, jonka voi rakentaa kondensaattoreilla ja keloilla.

Kuva 3 esittää Arduinoon suunnitellun kytkentäkaavion, joka poistaa vääristävän DC-komponentin, leikkaa aggressiivisesti korkeat taajuudet sekä nostaa signaalin positiiviselle puolelle.



Kuva 3. Arduinon sisääntulon signaalinkäsittely passiivikomponentein.

Oskilloskooppia hyväksi käyttäen pystyttiin seuraamaan signaalinkäsittelyn kulkua kytkennässä. Testaus aloitettiin toistamalla PC:stä siniaaltoja. Siniaalto toimii hyvänä referenssisignaalinä, koska siitä on helppo tulkita signaalin passiivikomponenttien aiheuttamat signaalin muutokset. Siniaallon aallonpituutta muuttamalla pystytään käytännössä testaamaan jakosuotimien toimivuus eri taajuuksilla. Kuvassa 4 on syötetty 440 Hz testisignaalia kytkennän läpi. Kuva osoittaa, että keltaisena esitetty alkuperäinen signaali muunnetaan Arduinolle luettavaan muotoon sinisenä signaalina.



Kuva 4. Arduinon sisääntulon ennen ja jälkeen käsittelyn.

4.2 Toisto

Arduinossa on mallista riippuen eri määrä ulostulopinnejä. Jokainen pinni voi olla kahdessa tilassa 0 V (low) ja +5 V (high). Nämä tilat vastaavat digitaalisia tiloja nolla ja yksi.

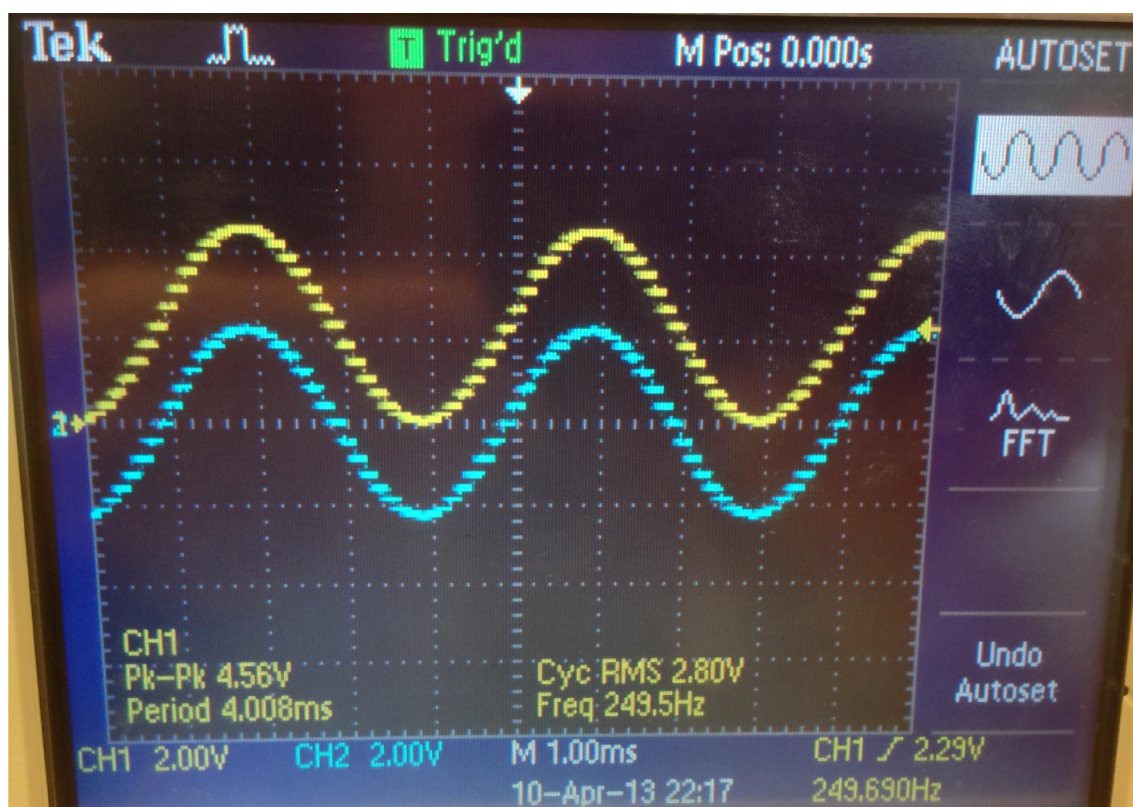
Bittien järjestyksessä on jokaisella bitillä oma painoarvo, joka johtuu siitä, miten tietokoneet käsittelevät tietoa. Koska tietokoneet käsittelevät kaiken ykkösinä ja nollina täytyy suurempia lukuja esittäessä käyttää useaa nollaa ja ykköstä niiden esittämiseen. Vähiten merkitsevä bitti esittää luvun nollan ja yhden välillä, toiseksi merkityksetön bitti esittää luvun nolla tai kaksi, kolmanneksi merkityksetön bitti esittää luvun nolla tai neljä jne. Esimerkiksi $10 = 2$, $11 = 3$ ja $101 = 5$. Suurin kahdeksalla bitillä esitetty luku on $11111111 = 255$. Kahdeksan bittiä muodostaa tavun.

Aikaisemmin mainittu 8 bitin resoluutio saadaan aikaiseksi käyttämällä kahdeksaa ulostulopinniä. Merkitsevimmän bitin arvo lopullisessa signaalissa tulee olla 0 V:n ja +2,5 V:n välillä, toiseksi merkitsevimmän bitin arvo puolet edellisestä, eli 0 V:n ja +1,25 V:n

välillä ja sitä seuraavan puolet edellisestä jne. Kahdeksan pinnin yhteenlaskettu kokonaisarvo tulee olemaan yhteensä noin 0 V:n ja +5 V:n välillä.

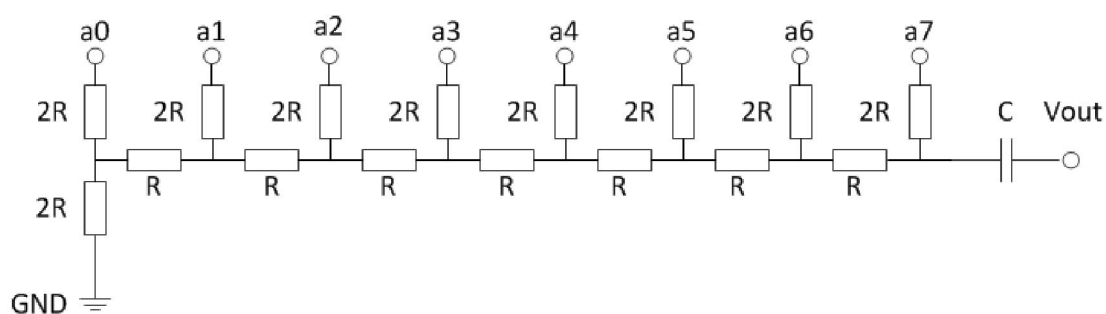
Jännitteelliset painoarvot ulostuloille voidaan toteuttaa vastustikapuilla, eli resistor ladderilla. Niin sanotun R-2R-tikapuun [24] toimintatapa perustuu siihen, että ulostulosta tulevan signaalin jännitetaso pudotetaan merkitsevimmän bitin kohdalla puoleen, toiseksi merkitsevimmän bitin kohdalla puoleen edellisestä jne. R-2R-tikapuun toteutus on halpa, yksinkertainen ja nopea, joskin sen toiminta on riippuvainen komponenttien laadusta. Huonoilla vastuksilla on suuret toleranssit, joten niiden resistanssi saattaa vaihdella komponenttien välillä suuresti, tämän seurauksena jokainen ulostulo ei välttämättä ole painotettu oikein.

R-2R-tikapuun ulostulo tuottaa signaalin 0 V:n ja +5 V:n välillä, kaiuttimille syötettävän signaalin tulisi kuitenkin vaihdella -2,5 V:n ja +2,5 V:n välillä. Toisin sanoen signaalilla on +2,5 V:n DC-komponentti, josta täytyy päästä eroon. Ulos tuleva signaali on myös hyvin kanticas huonon resoluution takia, kuten on havaittavissa kuvassa 5. Keltainen signaali esittää Arduinon ulostulosta tuotettua 250 Hz testisignaalia ja sininen samaa signaalia kondensaattorin jälkeen.



Kuva 5. Arduinon ulostulo kondensaattoria ennen ja jälkeen.

Kuvassa 6 on esitetty R-2R-tikapuun kytkentäkaavio. Terminaalit a0 – a7 esittävät Arduinosta tulevia ulostuloja, jossa a7 on merkitsevin bitti. Vout esittää kaiuttimille menevää lopullista äänisignaalia.



Kuva 6. R-2R-tikapuun kytkentäkaavio.

4.3 Koodekki

VoIP-sovellukset pakkaavat sisään tulevan signaalin viemään vähemmän tilaa. Rajoitettu kaista pitää aina ottaa huomioon internetin yli liikennöimisessä, ja tämän takia

puhesovelluksissakin käytetään matemaattista kaavaa, joka muuttaa signaalin vähemmän tilaa vievään muotoon. Tällaista matemaattista algoritmia kutsutaan koodeksi (codec).

Euroopassa puhesignaalin pakkauksen standardina toimii A-law-algoritmi [25]. Useimmiten informaatiota pakattaessa tapahtuu väistämättä informaatiohäviötä. Jos alkuperäinen signaali on esitetty 12 bitin resoluutiolla ja se tulisi pakata 8 bittiin, on osan tiedosta pakko kadota johonkin. On olemassa yleisiä pakkausalgoritmeja, jotka pystyvät säilyttämään kaiken alkuperäisen tiedon vähemmän tilaa vievässä muodossa (lossless compression) [26]. Tällaisten algoritmien kohdalla muodostetaan esimerkiksi lista useimmiten esiintyvistä luvuista ja merkitään kyseiset luvut vähemmän tilaa vievillä symboleilla ja pakkausta purkaessa tarkistetaan listasta, mitä mikäkin symboli edustaa.

Puheen tapauksessa tällainen ei ikävä kyllä toimi, mutta A-law:in toimintatapa on älykkäämpi, kuin vain yksinkertaisesti pudottaa vähiten merkitsevät bitit pois. Ihmisen puhe on hyvin dynaamista, toisin sanoen amplitudivaihtelu on suuri, jossa esimerkiksi eri konsonanttien äänenvoimakkuus on vokaaleja huomattavasti suurempi [27]. Jos vähiten merkitsevimmät bitit jätettäisiin vain huomioimatta, kärsisi hiljaa puhutun äänen laatu huomattavasti.

A-law määrittelee eniten merkitsevimmällä (7) bitillään, onko signaali negatiivinen vai positiivinen ja kolmella seuraavalla bitillään (6–4) signaalin suurpiirteisen amplituditason ja lopuilla neljällä bitillään (3–0) amplituditason tarkennuksen. Tällä tavalla saadaan pidettyä signaalin erottelukyky tasaisena eri amplitudeilla. Taulukossa 2 on esitetty A-law-pakkauksen käännöstaulukko. Aakkoset a, b, c ja d edustavat signaalin amplituditason tarkennuksia ja x signaalista pudotettavia bittejä.

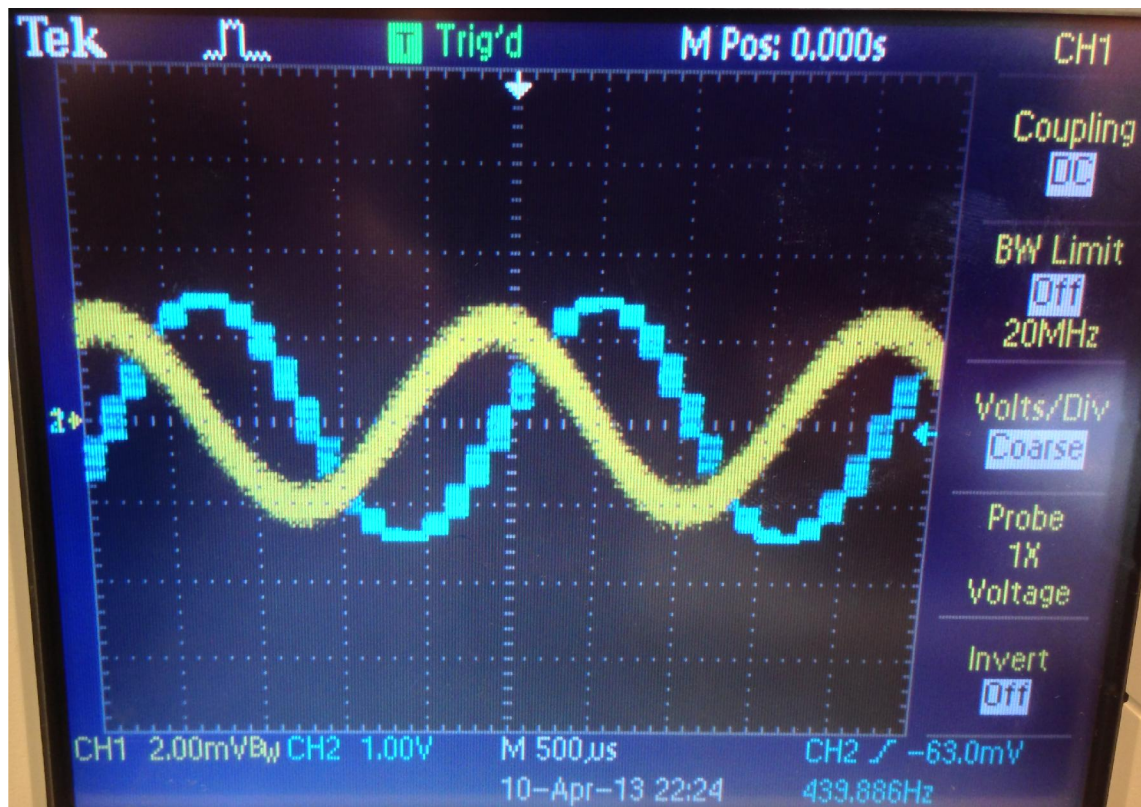
Taulukko 2. A-law pakkaustaulukko.

	Sisääntulo												Ulostulo							
Bit	11	10	9	8	7	6	5	4	3	2	1	0	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	a	b	c	d	x	0	0	0	a	b	c	d	
	0	0	0	0	0	0	1	a	b	c	d	x	0	0	1	a	b	c	d	
	0	0	0	0	0	1	a	b	c	d	x	x	0	1	0	a	b	c	d	
	0	0	0	0	1	a	b	c	d	x	x	x	0	1	1	a	b	c	d	
	0	0	0	1	a	b	c	d	x	x	x	x	1	0	0	a	b	c	d	
	0	0	1	a	b	c	d	x	x	x	x	x	1	0	1	a	b	c	d	
	0	1	a	b	c	d	x	x	x	x	x	x	1	1	0	a	b	c	d	
	1	a	b	c	d	x	x	x	x	x	x	x	1	1	1	a	b	c	d	

4.4 Testaus

Kun vastaanottoon ja toistoon tarvittavat kytkennät oli tehty ja koodekin logiikka koodattu, täytyi vielä testata kokonaisuuden toimivuus yhdessä. Kaikkea kolmea eri prototyyppin osa-aluetta testattiin luomalla Arduinoon koodinpätkä, joka vastaanottaa PC:n äänilähdössä linjatasoista signaalia, pakkaa sen A-law-koodekilla, purkaa pakkauksen ja toistaa signaalin R-2R-tikapuun kautta kaiuttimiin.

Kuvassa 7 keltainen käyrä esittää tietokoneen ulostulosta syötettyä siniaaltoa ja sininen Arduinosta kaiuttimille syötettävää lopullista aaltoa. Kuvasta näkee, että signaali vääristyy kaiken käsittelyn jälkeen, mutta pysyy muodoltaan samana.



Kuva 7. Arduinoon sisääntuleva signaali verrattuna kaiken käsittelyn jälkeen ulostuloon.

Todettuani siniaaltotestin onnistuneeksi vaihdoin testisignaalin PC:ltä musiikkikappaleeksi. Musiikkikappale valittiin testisignaaliksi sen takia, että siinä esiintyy laajalla taajuusalueella eri instrumenttien aiheuttamia sointuja sekä puhelun kannalta tärkeä elementti eli ihmisääni. Musiikkikappaleen kohdalla pystyi myös testaamaan erikokoisia kondensaattoreita sekä keloja, joiden vaikutus kaiuttimista tulevaan ääneen pystyttiin kohdentamaan myös kuuloa käyttäen, sillä tähän asti ne oli valittu laskemalla matemaattisesti eri kaavoilla teoreettisesti sopivia kokoja.

Musiikkikappaletesti päätettiin, kun kaiuttimista tulevasta äänestä sai hyvin selvää, mitä laulaja lausuu. Prototyypissä oli täten toimiva äänen vastaanotto, toisto ja pakkaus. Jäljelle jäi verkkopuolen toteutus.

5 Arduinon rajoitteet

Projektin edetessä törmättiin useaan Arduinon rajoitukseen. Tässä luvussa käydään läpi kirjastoihin ja muistiin liittyvät ongelmat sekä ohjelman toimintavirheiden etsimisen vaikeus. Työn aikana kohdattiin muitakin Arduinoon liittyviä haasteita, mutta suurimmat ongelmat muodostuivat edellä mainituista.

5.1 Kirjaston rajoitteet

Sen lisäksi, että Arduinon kirjastot olivat puutteellisia, eivätkä suoraan soveltuneet toteuttamaan projektissa vaadittavia toimintoja, ovat ne taaksepäin yhteensopimattomia.

5.1.1 Ominaisuuksien puute

Arduinon yksi lukuisista kirjastoista on Ethernet-kirjasto [7]. Se sisältää aliohjelmia, joiden kuului helpottaa verkkokommunikaation kanssa. Käytännössä kirjasto sisältää useita aliohjelmia, jotka keskustelevat Wiznet W5100 mikro-ohjaimen kanssa. W5100 sisältää itsessään tuen kommunikoida UDP:lla IP:n ja Ethernetin yli.

W5100:n hyväksi käyttö verkkokommunikaatiossa suoraan on hyvin työlästä, sillä sen toimintaa ohjataan ennalta määrätyillä muistilohkoilla. Muistilohkojen sijainti ja yleisesti W5100 mikro-ohjaimen toimintalogiikka on tarkasti esitelty sen datalehdessä (datasheet) [8]. Esimerkiksi yhden UDP-viestin lähetys vaatii suunnattoman määrän muistilohkojen sisällön muuttamista ja lohkojen tilan seuraamista. Tästä syystä on hienoa, että Arduinosta löytyy valmiit aliohjelmat, jotka automatisoivat kaiken edellä mainitun.

Ongelmaksi kuitenkin muodostui se, että prototyypin käyttämät eri verkkoprotokollat käyttivät useaa porttia. Portit ovat UDP:n (ja TCP:n) käyttämä tunniste siitä, minkä sovelluksen kanssa yritetään keskustella verkon välityksellä. Koska jokaisella verkkolaitteella on karkeasti yleistettynä vain yksi IP-osoite, mutta se sisältää usean sovelluksen, jotka haluavat keskustella verkon välityksellä, käytetään portteja erittelemään kaikki nämä eri keskustelut toisistaan. IP-osoite, portti ja siirtokerroksen protokolla muodostavat yhdessä pistokkeen (socket).

W5100 datasheetin mukaan se tukee neljää yhtäaikaista pistoketta, mikä riittää aikaisemmin valittujen verkkoprotokollien yhtäaikaiseen käyttöön. Arduinon kirjastossa on kuitenkin tuki vain yhdelle kerrallaan.

Arduinon kirjastoa täytyi täten kirjoittaa uusiksi. Oli kohtuullisen työlästä muuttaa Arduinon Ethernet-kirjastoa sen laajuuden takia ja siksi, että sen oli ohjelmoinut joku muu. Kirjasto saatiin lopuksi tukemaan neljää yhtäaikaista pistoketta.

5.1.2 Taaksepäin yhteensopimattomuus

Arduinon kehitysympäristö koostuu kääntäjästä, kirjastoista ja ajureista eri Arduino-versioille. Uuden kehitysympäristöversion julkaisussa muuttuu siis suuri määrä eri elementtejä [28]. Mikäli uudessa versiossa on korjattu kääntäjään kohdistuvia ohjelmointivirheitä, mutta kirjastot ovat muuttuneet huomattavasti, voi olla, että ohjelmoija ei voi päivittää uuteen versioon. Uuden version kirjastossa on voitu muuttaa radikaalisti eri aliohjelmien nimiä ja toimintatapoja.

Tämän projektin tapauksessa kävi niin, että uudessa kehitysympäristöversiossa oli korjattu suuri muistinvaraukseen liittyvä ongelma, mutta Ethernet-kirjasto oli kirjoitettu melkein kokonaan uudestaan. Koska olin tehnyt suunnattomia muutoksia vanhaan ethernet-kirjastoon, en voinut päivittää uudempaan versioon tekemättä muutoksia uuteen kirjastoon sekä omaan ohjelmaani.

5.2 Toimintavirheiden etsiminen

Arduinossa on hyvin rajattu tapa etsiä niitä kohtia koodista, jotka aiheuttavat ei-haluttua toimintaa. PSoCin tapauksessa on mahdollista hankkia In-Circuit Emulator (ICE) [29], joka antaa ohjelmoijan esimerkiksi pysäyttää ajossa oleva ohjelma milloin tahansa, mennä eteenpäin koodissa yhden askeleen kerralla ja tarkistaa mitä missäkin muuttujassa on talletettuna reaaliajassa.

Arduinossa ei tällainen ole mahdollista, vaan ainoa tapa etsiä kohtia, joissa koodi alkaa tempuilla, on lisätä epäilyyn vika-alueen ympärille käskyjä, jotka lähettävät tulosteen kehitysympäristössä olevalle sarjaliikennekuuntelijalle. Tuloste voi sisältää eri muuttuji-

en sen hetkistä sisältöä tai ihan pelkästään testitekstiä, jonka tarkoitus on kertoa, mihin asti koodissa päästään ennen ohjelman kaatumista.

5.3 Muistirajoitteet

Hyvin aikaisessa vaiheessa huomasin, että Arduinon muistin käsittelyssä tulee olla varovainen. Sen lisäksi, että käytettävissä ei ole kovin paljoa muistia (josta mm. Arduinon kirjastot haukkaavat vielä suuren osan) ei Arduino ilmoita, jos muisti loppuu kesken, eikä kirjastosta löydy edes valmista rutiinia selvittämään, kuinka paljon sitä on jäljellä.

Muistia varatessa tuli huomattua, että calloc-aliohjelma ei toiminut, kuten se oli kuvattu. Calloc-aliohjelman kuuluisi varata muistista lohko ja alustaa se. Ajon aikana huomasin, että calloc varaa kyllä muistista lohkon, mutta ei alusta sitä välillä ollenkaan. Tästä syystä oli pakko kirjoittaa kaikkiin varattuihin muistilohkoihin aluksi tyhjää, jotta voitiin välttää ohjelman sattumanvarainen toiminta.

Mikäli ajossa oleva ohjelma yrittää varata enemmän muistia, kun sitä on tarjolla (kysymättä ensin onko sitä jäljellä), alkaa ohjelma alusta asti uudestaan. Tämän vuoksi yksi ensimmäisistä aliohjelmista, jota täytyi kirjoittaa, oli rutiini, joka palauttaa jäljellä olevan muistin määrän. Aliohjelma varaa niin paljon muistia kuin pystyy (kysymällä aina ensin onko sitä jäljellä), kunnes sitä ei ole enää jäljellä, ja vapauttaa tämän jälkeen varaimansa muistin ja palauttaa lukuarvon siitä, kuinka paljon pystyi varaamaan.

Olin kirjoittanut ja testannut SIP- ja RTP/RTCP-protokollien kehyksen ja SIP:in logiikan kokonaan, kunnes SDP-pakettien muodostuksen aliohjelmassa huomasin, että Arduinossa ei yksinkertaisesti ole tarpeeksi muistia. Jo ohjelman alkaessa, ennen kuin muuttujille varataan muistia, oli Arduinossani noin 1500 tavua muistia jäljellä ja yksi SDP-viesti vie kokonaisen kilotavun (1024 tavua).

SDP-viestit ovat yksinkertaisesti niin pitkiä, että Arduino ei pysty pitämään muistissaan kahta sellaista kerrallaan. Viestejä ei voida edes pilkkoa osiin, koska standardi on hyvin tarkka niiden rakenteesta eikä Arduinosta ole olemassa versiota, jossa on enemmän muistia. Vaikka SDP-viestit olisivat vähän lyhempiä, olisi huomattavan vaikeaa saada muisti riittämään sittenkään. Muistia pitäisi olla sen verran käytössä, että eri viestejä voi

pitää sisään tulevassa ja ulosmenevässä puskurissa samalla, kun seuraavia viestejä koostetaan ja ääninäytteitä kerätään koko ajan.

Tämän takia oli pakko hylätä alkuperäinen suunnitelma ja keksiä jokin muistin kannalta ystävällisempi tapa liikennöidä ääntä verkon välityksellä.

6 Oma protokolla

6.1 Rakenne

Oman protokollarakenteen suunnittelu osoittautui helpoksi, koska olin tutkinut suuren määrän jo olemassa olevia protokollia. Tiesin, mitkä eri protokollien ominaisuuksista olisi välttämättömiä puheliikenteen onnistuneeseen liikennöimiseen.

Arduinon muistirajoitteiden vuoksi oman protokollan tärkeimpiä ominaisuuksia on pieni muistin kuormitus. Protokollan nimeksi valitsin Voice over Arduino (VoA). Aloitin VoA:n suunnittelun sillä, että listasin kaikki toiminnan kannalta pakolliset ominaisuudet muistiin ja yritin keksiä niiden ympärille viestikehyksen. Implementoin SIP:istä tutun puhelinten välisen kommunikaation siitä, kuinka osapuolet keskustelevat soittamisesta, vastaamisesta jne. Jokainen näistä pyynnöistä vastaa ennalta määrättyä numeroa ja viestikehys alkaa tällä numerolla, jotta se on ensimmäinen asia mitä luetaan vastaantulevasta paketista.

Mikäli kyseessä on paketti, joka sisältää puhesignaalia on toinen kenttä varattu sille. Viimeisenä kenttänä on paketin järjestysnumero, jonka avulla voidaan järjestää paketit oikeaan järjestykseen ja huomata, jos paketteja on kadonnut matkan varrella.

Datakentän koossa tuli miettiä sen suhdetta loppuun pakettia. Jos jokaista yksittäistä näytettä kohden lähetettäisiin yksi viesti, olisi lähetettävän informaation osuus koko viestistä huomattavan pieni ja täten hyvin tehoton tapa kommunikoida. Yksi tavu varattuna viestin tyypille, toinen datalle ja kolmas järjestysnumerolle. Tässä tapauksessa informaation osuus koko paketista on vain kolmasosa.

Toisaalta jos lähetetään suunnaton määrä näytteitä kerralla, aiheutuu siitä pitkä vasteaika, koska näytteitä joudutaan kerryttämään puskuriin pitkään [30]. Kompromissiksi päätin lähettää 160 ääinäytettä kerralla. 8000 Hz:n näytteistystaajuudella 160 ääinäytettä kerrytetään puskuriin 20 ms, joka on hyväksyttävä vasteaika puheelle. Verkoviive ja Arduinon sisäinen käsittelyaika aiheuttaa vielä lisää viivettä, mutta nekin mukaan luettuna päästään helposti alle 150 ms, jota pidetään rajana, jolloin kuulija ei huomaa viivettä ITU-T G.114-standardin luvun 4 mukaan [31]. Kuvassa 8 on esitetty VoA-paketin rakenne.

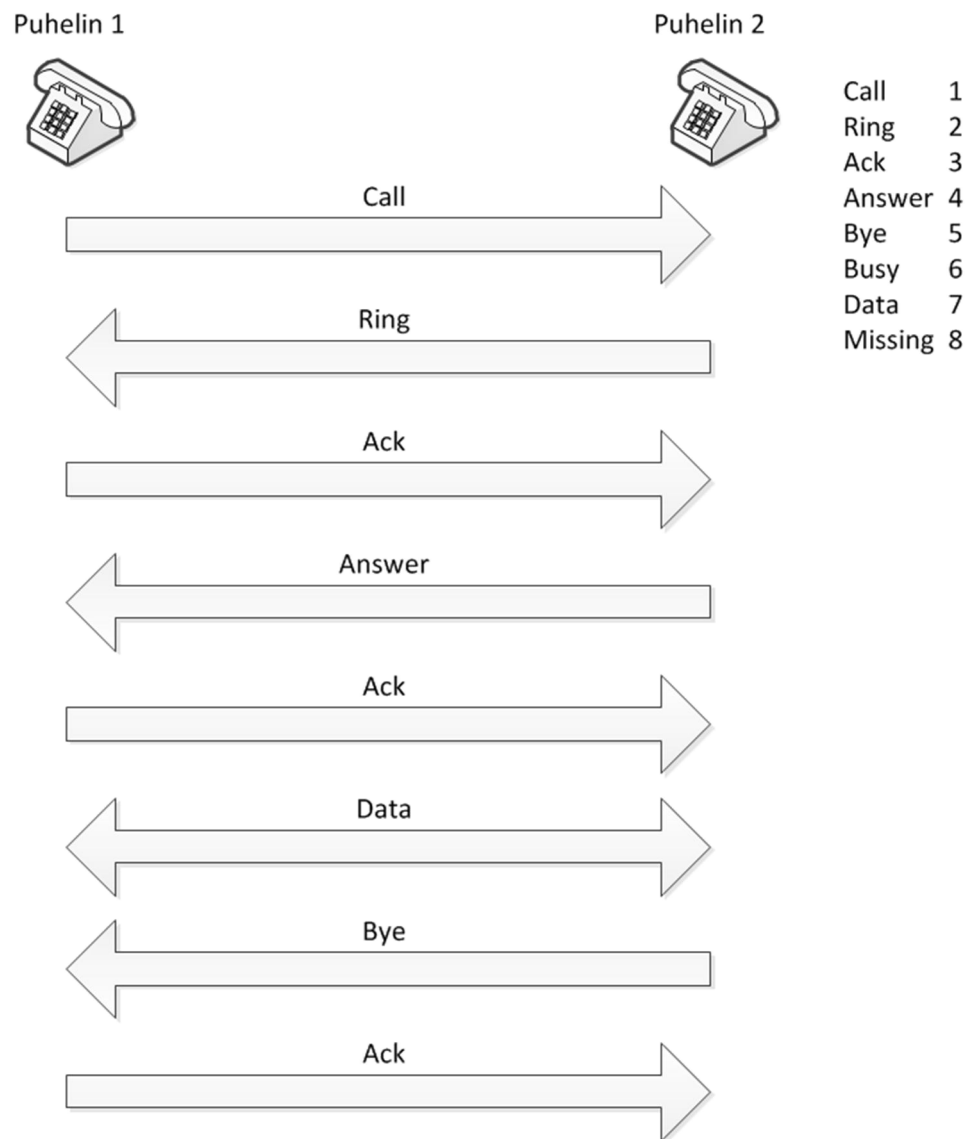
Tavut:	1	160	1
Kenttä:	Request	Data	Sequence

Kuva 8. VoA paketin rakenne.

6.2 Logiikka

Kuvassa 9 on esitetty VoA-protokollan eri pyynnöt ja esimerkkipuhelu. Jokaista pyyntöä kohden täytyy lähettää Ack-viesti (lyhennys acknowledgementista, suom. kuittaus), jonka tarkoitus on ilmoittaa vastapuolelle, että viesti on tullut perille. Kuvan esimerkkipuhelu alkaa soittoilmoituksella, joka kuitataan saaduksi. Seuraavaksi puhelun vastaanottaja lähettää viestin, että puhelin soi, joka kuitataan. Puhelin 2 vastaa puheluun, jolloin lähetetään viesti, että puhelun voi aloittaa, mikä kuitataan. Tätä seuraa molemminpuolinen puheliikenne, joka päättyy siihen, kunnes jompikumpi osapuoli lopettaa puhelun, joka jälleen kerran kuitataan.

Mikäli vastapuolella on puhelu jo käynnissä, lähettää se varattu-viestin. Jos huomataan pakettihävikkiä, lähetetään missing-viesti.



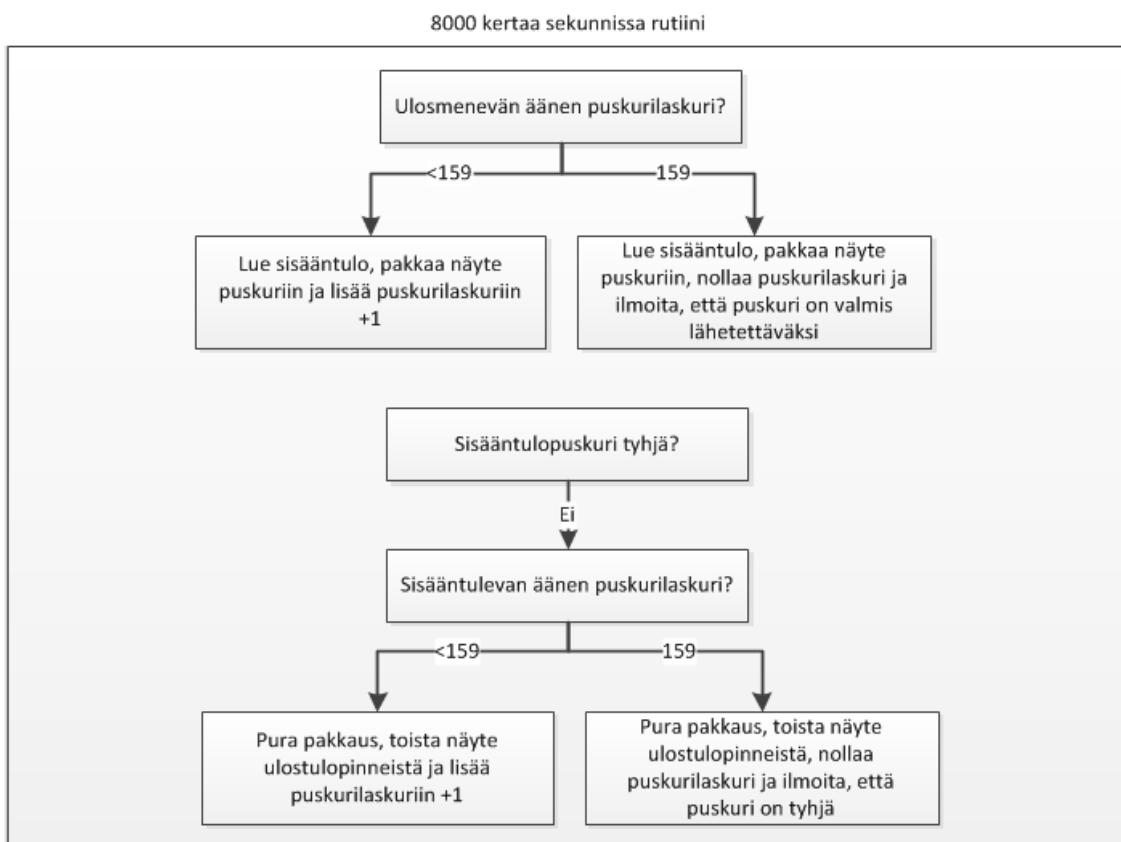
Kuva 9. VoA-protokollalla toteutettu esimerkkipuhelu.

7 Lopullinen prototyyppi

7.1 Suorituskyky

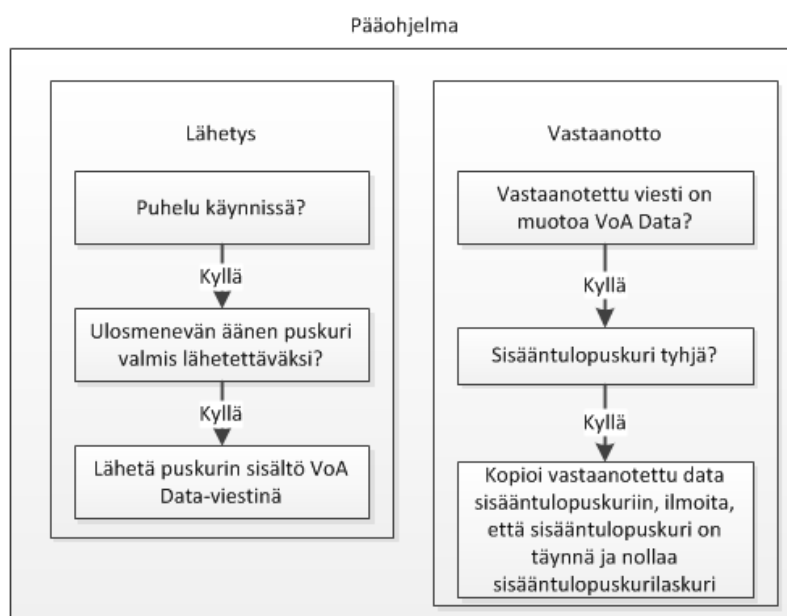
VoA-protokollan implementoinnin jälkeen kävi ilmi, että näytteistystaajuutta täytyi pudottaa, koska suorituskyky ei riittänyt käsittelemään 8000 näytettä sekunnissa. Arduinolla kestää 100 mikrosekuntia lukea ADC:n arvo muistiinsa, mikä tarkoittaa käytännössä sitä, että teoreettinen maksiminäytteenottotaajuus on 10000 Hz [32]. Valittu 8000 Hz:n näytteenottotaajuus on lähellä maksimia, ja koska mm. VoA:n sekä koodin purku ja pakkaus vievät oman siivunsa suorituskyvystä, ei Arduino ehdi käsitellä tarvittavia käskyjä tarpeeksi nopeasti.

Prototyyppi pysäyttää pääohjelman ajon 8000 kertaa sekunnissa suorittaakseen rutiinin, joka esitetään kuvassa 10. Pääohjelman pysäyttäviä tapahtumia kutsutaan keskeytysrutiineiksi (interrupt routine). Keskeytysrutiinin aikana vastaanotetaan ja toistetaan ääntä riippuen eri ehdoista.



Kuva 10. Keskeytysrutiini.

Pääohjelmassa vastaanotetaan ja lähetetään verkon välityksellä äänidataa sisältäviä viestejä, mikäli puhelu on käynnissä, kuten on esitetty kuvassa 11. Testauksen aikana keskeytysrutiinien ja täten myös samalla näytteistystaajuuden ollessa 8000 kertaa sekunnissa käyttäytyi prototyypin tuottama ääni ei-halutulla tavalla. Keskeytysrutiinien määrän pudottaminen 5000 kertaan sekunnissa mahdollisti Arduinon ehtimään suorittamaan kaikki tarvittavat käskyt.



Kuva 11. Pääohjelman rutiini

Äänen laatu oli siedettävänä. Toisin sanoen puheesta saa hyvin selvää, mutta korkeampi näytteistystaajuus tuottaisi paremman äänenlaadun. Aikaisemmissa testeissä (luvussa 4) pelkän äänen vastaanoton ja toiston yhteydessä pystyttiin käyttämään 8000 Hz:n näytteistystaajuutta, mutta verkkoprotokollan lisäyksen jälkeen Arduinosta loppui suoritustehot kesken.

7.2 Testipuhelu käytännössä

Käytössäni oli kaksi Arduinoa – Arduino Mega 2560 ethernet shieldillä, sekä Arduino ethernet. Arduino ethernetin ongelma on, että siinä ei ole tarpeeksi ulostulopinnejä käytettävissä, jotta voitaisiin ajaa 8 bittistä DAC:ia. Tästä syystä testaus toteutettiin lähettämällä Arduino Mega 2560:stä VoA-paketit Arduino ethernetiin, josta ne palautettiin Mega 2560:een takaisin.

Tällä tavoin saatiin aikaiseksi toteutettavuuden osoitus (Proof-of-concept). Testi todistaa, että yksittäinen Arduino Mega 2560 pystyy lähettämään, vastaanottamaan ja prosessoimaan äänet. Tulos olisi käytännössä sama, vaikka käytössä olisikin kaksi Arduino Mega 2560:tä.

Lopputuloksena oli toimiva A-law-pakkausstandardia ja omaa protokollaa käyttävä su-lautettu VoIP-puhelin. Puhelun äänestä sai selvää ja prototyyppi toimii missä tahansa modernissa tietoverkossa.

8 Pohdintaa

8.1 Käyttökohteet

Prototyyppiä voi soveltaa esimerkiksi pienten yritysten sisäisessä kommunikaatiossa, jossa toimistot ovat maantieteellisesti kaukana toisistaan. Koska VoIP-puhelin toimii missä tahansa modernissa tietoverkossa, on sen käyttämä infrastruktuuri jo valmiiksi rakennettuna.

Puhelinta voi myös käyttää osana kuulutusjärjestelmää, jossa toinen puhelin asennetaan kaiuttimeen kiinni. Esimerkiksi kouluissa ja rautatieasemilla on tarve välittää tietoa asiakkaille, jotka ovat kaukana kuuluttajasta ja/tai hajautettuna suurelle alueelle.

Ehkä paras käyttökohde puhelimelle on käyttää tätä projektia esimerkkinä Metropolia Ammattikorkeakoulun tietoliikennetekniikan tai sulautettujen järjestelmien kursseissa. Toivon, että tämä projekti inspiroi muita opiskelijoita kehittämään joko Arduinolle jotain verkkosovelluksia tai jollekin muulle alustalle sulautettua VoIP-puhelinta.

8.2 Jatkokehityskohteet

Suorituskyvyn takaamiseksi VoA-protokollasta karsittiin paljon muiden sovelluskerroksen protokollien tarjoamia lisäominaisuuksia. VoA-protokollaa voisi tulevaisuudessa laajentaa tukemaan ryhmäpuheluita sekä viestien salausta.

Ryhmäpuhelut voisi toteuttaa käyttämällä ryhmälähetystä (multicast), joka tarkoittaa yhden viestin lähettämistä monelle kerrallaan. Ryhmälähetyksessä on yksi lähettäjä ja vastaanottajille luodaan ryhmä, johon voi halutessaan liittyä. VoA toimii tällä hetkellä käyttäen täsmälähetystä (unicast), jossa viestin lähettäjiä ja vastaanottajia on vain yksi.

Tietoturvan kannalta salaus olisi ehdoton lisäominaisuus. VoA ei käytä mitään salausalgoritmia, mikä tarkoittaa sitä, että jos tapahtuu niin sanottu välistävetohyökkäys (man-in-the-middle attack) [33], pystyy luvaton urkkija kuuntelemaan puheluita ilman esteitä, eikä sitä voida estää. Välistävetohyökkäys tapahtuu, kun viestin lähettäjän ja vastaanottajan väliin asettuu kolmas osapuoli, joka voi halutessaan kuunnella ja jopa

muuttaa viestien sisältöä. Kyseistä hyökkäystä vastaan voidaan käyttää kahta menetelmää.

Ensimmäinen suoja on salata lähetettävä tieto jollain yksisuuntaisella funktiolla (one-way function) [34]. Yksisuuntainen funktio muuntaa salattavan tiedon näennäisesti selvittämättömään muotoon käyttäen hyväksi salausavainta, joka on yleensä suuri alkuluku. Yksisuuntainen funktio on nopea laskea vain toiseen suuntaan. Salausavainta hyväksi käyttäen funktio on myös nopea laskea takaisin alkuperäiseen muotoon, mutta ilman sitä sen purkamiseen kestää huomattavan kauan. Mikäli viestit salataan, ei se haittaa, vaikka hyökkääjä saakin tiedon käsiinsä, koska se ei pysty muuntamaan sitä ymmärrettävään muotoon. Yksisuuntaisen funktion voi toki pitkän ajan kanssa murtaa, mutta salausavaimia vaihtamalla säännöllisesti voidaan taata, että hyökkääjä ei koskaan pysy perässä.

Toinen suoja on käyttää julkisen avaimen salausta. Julkisen avaimen salaus on niin sanottu epäsymmetrinen salaus, jossa viestin salaukseen ja purkamiseen käytetään kahta eri avainta. Mikäli julkisena avaimena (kaikille jaettava ja tiedossa oleva avain) pidetään salaavaa avainta, voidaan vastaanottajalle lähettää salattuja viestejä ilman, että kukaan pystyy lukemaan niitä matkan varrella. Jos taas julkisena avaimena pidetään purkavaa avainta, voidaan taata, että viestejä ei muunnella matkan varrella.

Salauksen käytön huonona puolena on sen intensiivinen suoritustehon käyttö. Kryptografisten algoritmien laskeminen kuormittaisi jo valmiiksi äärirajoilla operoivaa Arduinon suoritinta. Toisaalta yksinkertaisempien algoritmien käyttö mahdollistaisi sen, että ne olisi helposti murrettavissa modernilla PC:llä.

8.3 Opittua

Projektin aikana tuli tutustuttua erittäin syvällisesti VoIP:iin liittyviin sovelluskerroksen protokoliin, takaisinmallinnukseen (reverse engineering), C++-ohjelmointiin sekä virtapiirien suunnitteluun.

Sovelluskerroksen protokollat ja niiden kuvausten monimutkaisuus teknisissä dokumenteissa yllätti. Ilman Wireshark-ohjelmaa olisi ollut huomattavan paljon vaikeampaa hahmottaa, miten eri protokollat käyttäytyvät.

Arduinolle ohjelmoidessa tuli opittua suunnattomasti muun muassa muistin hallintaa. Rajoitetussa ympäristössä ei voi tuhata resursseja, vaan esimerkiksi kaikki käytettävät muistilohkot täytyy metodisesti ensin käydä varmistamassa, että ne ovat saatavilla, varata, alustaa ja lopuksi vapauttaa.

Siitä huolimatta, että projektissa ei tavoitettu täyttä yhteensopivuutta VoIP-sovellusten kanssa, olin kokonaisuudessaan erittäin tyytyväinen lopputulokseen. Pystyin todistamaan, että Arduinoa hyväksi käyttäen on mahdollista toteuttaa VoIP-puhelin.

9 Yhteenveto

Insinööriyö alkoi idealla käyttää hyväksi mahdollisimman laajasti tietotekniikan eri osa-alueiden oppeja hyväksi. Sulautettussa VoIP-puhelimessa yhdistyvät sulautettujen järjestelmien, ohjelmistotekniikan sekä tietoliikennetekniikan osa-alueet.

Toimivan prototyypin rakentamisen ensimmäinen askel oli valita, mille alustalle ja millä komponenteilla se voidaan rakentaa. Eri vaihtoehtojen vertailun jälkeen päädyttiin toteuttamaan projekti Arduinolle käyttäen hyväksi useita eri verkkoprotokollia sekä ADC:tä ja DAC:tä ääneen vastaanottoon ja lähetykseen.

Verkkoprotokollien valinnassa eroteltiin eri protokollat OSI-mallin kerroksiin. Arduino ethernetissä löytyi sisäänrakennettu tuki neljälle alimmalle kerrokselle, ja sovelluskerroksen protokollissa tehtiin syväluotaavaa tutkimusta, jonka seurauksena päädyttiin neljään protokollaan – SIP, SDP, RTP ja RTCP. Sovelluskerroksen protokollien toimintatavan tutkimukseen rakennettiin testiympäristö, jossa voitiin tutkia ja testata niiden käytännön toimintaa.

Äänen vastaanotossa käytettiin hyväksi Arduinon sisäistä ADC:tä, joka vaati sille syötettävän signaalin eteen rakennetun passiivikomponenttikytken. Äänen toistoon rakennettiin DAC R-2R-vastustikapuulla. Ennen äänen lähetystä ja vastaanoton jälkeen pakattiin tai purettiin signaali A-law-koodekillä.

Sovelluskerroksen verkkoprotokollia implementoidessa törmättiin Arduinon rajoituksiin, joihin kuului kirjaston ominaisuuksien puute, taaksepäin yhteensopimattomuus, toimintavirheiden etsimisen vaikeus, sekä muistirajoitteet. Loppujen lopuksi muistirajoitteet oli se tekijä, jonka takia jouduttiin luopumaan alkuperäisestä suunnitelmasta käyttää hyväksi standardijärjestöjen sovelluskerroksen verkkoprotokollia.

Päätettiin suunnitella ja toteuttaa oma sovelluskerroksen verkkoprotokolla – VoA. Sen suunnittelussa käytettiin hyväksi standardiprotokollista opittuja asioita, joskin sen rakenne tehtiin mahdollisimman yksinkertaiseksi, jotta säästetään mahdollisimman paljon muistia.

Lopullinen prototyyppi oli modernin tietoverkon kanssa yhteensopiva VoA-protokollaa käyttävä sulautettu VoIP-puhelin, jonka äänenlaatu oli kohtalainen ja viive sekä vakaus

erinomainen. Prototyyppiä voi käyttää hyväksi yritysten sisäisessä kommunikaatiossa, osana kaiutinjärjestelmää, ja mikä tärkeintä, inspiraationa Metropolia Ammattikorkeakoulun eri kursseilla.

Opin suunnattomasti uusia asioita projektin aikana ja vastoinikäymisistä huolimatta sain rakennettua toimivan prototyypin sulautetusta VoIP-puhelimesta. Tavoitteet toisin sanoen kohdattiin ja olen erittäin tyytyväinen lopputulokseen. Tunnen olevani etuoikeutettu, että olen saanut työskennellä tällaisen projektin parissa.

Lähteet

- 1 What is VOIP. 2013. Verkkodokumentti. VOIP-Info.org LLC. <<http://www.voip-info.org/wiki/view/What+is+VOIP>>. Luettu 13.4.2013.
- 2 Protocol (computer science). 2013. Verkkodokumentti. Encyclopedia Britannica. <<http://global.britannica.com/EBchecked/topic/410357/protocol>>. Luettu 13.4.2013.
- 3 Atmel megaAVR Microcontroller. 2013. Verkkodokumentti. Atmel Corporation. <<http://www.atmel.fi/products/microcontrollers/avr/megaAVR.aspx>>. Luettu 7.4.2013.
- 4 Arduino products. 2013. Verkkodokumentti. Arduino. <<http://arduino.cc/en/Main/Products>>. Luettu 7.4.2013.
- 5 PSoC kits. 2013. Verkkodokumentti. Cypress Semiconductor Corporation. <<http://www.cypress.com/?rID=63754>>. Luettu 7.4.2013.
- 6 PSoC instrucion set. Verkkodokumentti. Cypress Semiconductor Corporation. <<http://www.cypress.com/?docID=3101>>. Luettu 7.4.2013.
- 7 Arduino libraries. Verkkodokumentti. Arduino. <<http://arduino.cc/en/Reference/Libraries>>. Luettu 7.4.2013.
- 8 Wiznet W5100 datasheet. Verkkodokumentti. Wiznet Corporation. <http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5100_Datasheet_v1.2.2.pdf>. Luettu 7.4.2013.
- 9 Seven Layers of Open Systems Interconnection (OSI) Model. 2010. Verkkodokumentti. Omnisecu.com. <<http://www.omnisecu.com/tcpip/osi-model.htm>>. Luettu 7.4.2013.
- 10 VoIP sampling and quantizing. 2013. Verkkodokumentti. LWC Training Corporation. <<http://www.lwctraining.com/VoIP/VoIP03/voip03c.htm>>. Luettu 13.4.2013.
- 11 Ethernet Technologies. 2012. Verkkodokumentti. Cisco Systems, Inc. <http://docwiki.cisco.com/wiki/Ethernet_Technologies>. Luettu 7.4.2013.
- 12 Internetin BGP reitityksen osuudet. 2013. Verkkodokumentti. Geoff Huston. <<http://bgp.potaroo.net/index-bgp.html>>. Luettu 13.4.2013.
- 13 ITU-T recommendation H.323. 2009. Verkkodokumentti. ITU. <http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.323-200912-!!PDF-E&type=items>. Luettu 13.4.2013.

- 14 RFC SIP standard. Verkkodokumentti. 2002. IETF.
<<http://www.ietf.org/rfc/rfc3261.txt>>. Luettu 13.4.2013.
- 15 RFC SDP standard. 1998. Verkkodokumentti. IETF.
<<http://tools.ietf.org/html/rfc2327>>. Luettu 13.4.2013.
- 16 Koistinen Tommi. 2013. Protocol overview: RTP and RTCP. Verkkodokumentti. Aalto-yliopisto. <<http://www.netlab.tkk.fi/opetus/s38130/k99/presentations/4.pdf>>. Luettu 13.4.2013.
- 17 Meet Ubuntu. 2013. Verkkodokumentti. Canonical Ltd.
<<http://www.ubuntu.com/ubuntu>>. Luettu 13.4.2013.
- 18 What is Asterisk. 2013. Verkkodokumentti. Digium Inc.
<<http://www.asterisk.org/get-started>>. Luettu 13.4.2013.
- 19 Apple - OS X Mountain Lion. 2013. Verkkodokumentti. Apple Inc.
<<http://www.apple.com/osx/>>. Luettu 13.4.2013.
- 20 CounterPath X-Lite overview. 2013. Verkkodokumentti. CounterPath Corporation.
<<http://www.counterpath.com/x-lite.html>>. Luettu 13.4.2013.
- 21 About Wireshark. 2013. Verkkodokumentti. Wireshark Foundation.
<<http://www.wireshark.org/about.html>>. Luettu 13.4.2013.
- 22 ITU-T recommendation G.711. Verkkodokumentti. ITU.
<http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.711-198811-!!!PDF-E&type=items>. Luettu 13.4.2013.
- 23 Bruno A. Olshausen. Aliasing. 2000. Verkkodokumentti. Redwood Center for Theoretical Neuroscience.
<<http://redwood.berkeley.edu/bruno/npb261/aliasing.pdf>>. Luettu 13.4.2013.
- 24 Proto-DAC shield for Arduino. 2008 Verkkodokumentti. Maker Media Inc.
<<http://blog.makezine.com/2008/05/29/makeit-protodac-shield-fo/>>. Luettu 13.4.2013.
- 25 A-Law Compressed Sound Format. 2008. Verkkodokumentti. Library of congress.
<<http://www.digitalpreservation.gov/formats/fdd/fdd000038.shtml>>. Luettu 13.4.2013.
- 26 Lossless and lossy data compression. 2011. Verkkodokumentti.
<http://www.maximumcompression.com/lossless_vs_lossy.php>. Luettu 13.4.2013.
- 27 Arduino release notes. 2013. Verkkodokumentti. Arduino.
<<http://arduino.cc/en/Main/ReleaseNotes>>. Luettu 13.4.2013.

- 28 A-Law and mu-Law Companding Implementations using TMS320C54x. 1997. Verkkodokumentti. Texas Instruments Inc. <<http://www.ti.com/lit/an/spra163a/spra163a.pdf>>. Luettu 13.4.2013
- 29 CY3215-DK In-Circuit Emulation Development Kit. 2013. Verkkodokumentti. Cypress Semiconductor Corporation. <<http://www.cypress.com/?rID=3411>>. Luettu 13.4.2013
- 30 Communication Overhead. 2006. Verkkodokumentti. Carnegie Mellon's School of Computer Science. <http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume28/clement07a-html/7_3Communication_Overhead.html>. Luettu 13.4.2013
- 31 ITU-T recommendation G.114. 2003 Verkkodokumentti. ITU. <http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.114-200305-!!!PDF-E&type=items ITU-T G.114>. Luettu 14.3.2013
- 32 Arduino Analogread. 2013. Verkkodokumentti. Arduino. <<http://arduino.cc/en/Reference/AnalogRead>>. Luettu 13.4.2013
- 33 Man-in-the-middle attack. 2009. Verkkodokumentti. Open Web Application Security Project. <https://www.owasp.org/index.php/Man-in-the-middle_attack>. Luettu 13.4.2013
- 34 Koskinen J. Kryptologiaa. 2002. Verkkodokumentti. Tampereen teknillinen yliopisto. <<http://www.cs.tut.fi/~8306000/kr.html>>. Luettu 13.4.2013