LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

SULAUTETUN JÄRJESTELMÄN TAKAISINMALLINNUS JA MODIFIOINTI

TADIL-A/LINK 11 -radiopuhelinsovellus

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Tietokone-elektroniikka
Opinnäytetyö AMK
Kevät 2012
Miika Ihanajärvi

Omistettu rakkaalle vaimolleni Realle sekä lapsilleni Daavidille ja Reetalle

Lahden ammattikorkeakoulu
Tietotekniikka

IHANAJÄRVI, MIIKA: Sulautetun järjestelmän takaisinmallinnus ja modifiointi
TADIL-A/Link 11 -radiopuhelinsovellus

Tietokone-elektroniikan opinnäytetyö 47 sivua, 9 liitesivua

Kevät 2012

TIIVISTELMÄ

Tämä opinnäytetyö on tehty Bandercom Oy Ltd:n tilauksesta kesällä 2011. Työn tavoitteena on modifioida Kenwood TS-480SAT -radiopuhelin tukemaan TADIL-A/Link 11 -sotilasstandardia HF-taajuuksilla tapahtuvaa tiedonsiirtoa varten.

Laitetta on modifioitu paljon jo valmiiksi. Modifikaatiot on testattu ja laitteen on todettu soveltuvan vastaanottoon, mutta sen kaistanleveys on osoittautunut hieman liian kapeaksi lähettämiseen nopeimmilla lähetteillä.

Lähetyksessä ja vastaanotossa on käytössä audiotaajuusalueella digitaalisella signaaliprosessorilla toteutettu digitaalinen suodin. Lisäksi laitteessa on välitaajuudella sijaitseva analoginen kidesuodin kaistanpäästösuotimena.

Digitaalinen suodin käyttää lähetyksessä ja vastaanotossa eri päästökaistanleveyksiä. Vastaanotossa suodin on säädettävissä melko rajattomasti PC-pohjaisella ohjelmistolla 6 kHz:iin asti. Lähetyksessä maksimikaistanleveys on rajoitettu enintään 2,4 kHz:iin.

Tämän opinnäytetyön kuluessa tutkin eri mahdollisuuksia muuttaa Kenwood TS-480SAT:n ominaisuuksia lähetyksen suhteen sekä vertailen ohjelmisto- ja laitteistopohjaisia toteutusvaihtoehtoja. Valitettavasti Suomen tekijänoikeuslaki asetti rajoituksia saadun tiedon julkaisemiseen. Sen sijaan, että pääsisin esittelemään tilanteita yksityiskohtaisesti, joudun paikoin esittelemään asioita yleisellä tasolla.

Projektina takaisinmallinnus sekä sulautetun ohjelman modifiointi toivat arvokasta tietotaitoa takaisinmallinnuksesta ja digitaalitekniikasta sekä minulle että Bandercom Oy Ltd:lle.

Avainsanat: Takaisinmallinnus, sulautetun ohjelman modifionti

Lahti University of Applied Scienes
Degree Programme in Information Technology

IHANAJÄRVI, MIIKA: Reverse engineering and modifying embedded systems
Transceiver for TADIL-A/Link 11 -application

Bachelor's Thesis in Computer Electronics 47 pages, 9 appendices

Kevät 2012

ABSTRACT

This Bachelor's Thesis was made for Bandercom Oy Ltd in summer 2011. The objective of the work was to modify the Kenwood TS-480SAT radio transceiver to support HF data transimission according to TADIL-A/Link 11 military standards.

The radio had some modifications already. The modifications had been tested and the radio is working well on receiving, but the bandwidth is too narrow for transmitting with faster modes.

On receive and transmit there is a digital filter at an audio stage, which is made using a digital signal processor. There is also an analog bandpass filter at the intermediate frequency.

The digital filter has different bandwidths while transmitting and receiving. On receive mode the digital filter bandwidth can be altered all the way up to 6 kHz with a PC based application. While transmitting the maximum bandwidth was limited to 2.4 kHz. The worst bottleneck of the system was the digital filter.

During this study the different alternatives were examined for how to change the features of transmitting of Kenwood TS-480SAT. Unfortunately Finnish copyright laws do not allow publishing of information gathered by reverse-engineering. Thats why the results can only be presented in a very general way in this Thesis.

The project was successful. The reverse engineering and the firmware modification brought valuable knowledge about digital systems for the writer and for Bandercom Oy Ltd.

Key words: reverse engineering, firmware modification

SISÄLLYS

SANASTO

AM	Amplitudimodulaatio
Assembleri	Ohjelma, joka muuttaa symbolisen, assembly-kielisen koodin binäärimuotoiseksi ohjelmaksi, joka voidaan suorittaa kohdeprosessorilla.
Assembly	Symbolinen konekieli
CPLD	(Complex Programmable Logic Device) eräs ohjelmoitava logiikkapiirityyppi
CPU	(Central Processing Unit) Prosessorin ydin
DARAM	(Dual Access RAM) RAM-muisti, jota kaksi laitetta voi käyttää yhtäaikaisesti
Debuggeri	Ohjelma, jossa voidaan ajaa ja analysoida toisia ohjelmia. Mahdollistaa monesti ohjelman dissassembloinnin ja suorittamisen käsky kerrallaan sekä muuttujien tietojen tarkastelun.
Dekompileri	Ohjelma, joka muuttaa konekielisen koodin korkeamman tason ohjelmointikelelle.
Dissassembleri	Ohjelma, joka muuttaa konekielisen koodin luettavaan muotoon esimerkiksi assemblyksi
DMA	(Direct Memory Access) Tiedon kopiointia paikasta toiseen ilman suorittimen käyttöä
DPSK	(Differential Phase Shift Keying) Differentiaalinen vaiheensiirtoavainnus.
DSP	(Digital Signal Processor) Digitaalinen signaaliprosessori on mikroprosessori, jossa erityisiä käskyjä esimerkiksi suodatusta vaativiin operaatioihin.
FIR	(Finite Impulse Response) Äärellinen impulssivaste
Firmware	Sulautettu ohjelma. Sulautetun järjestelmän ohjelmisto.
FM	(Frequency Modulation) Taajuusmodulaatio
FPGA	(Field Programmable Gate Array) eräs ohjelmoitava logiikkapiirityyppi
half-duplex	Vuorosuuntainen yhteys. Lähettäjiä voi olla vain yksi kerrallaan, mutta lähettäjä voi vaihtua.
HF	(High Frequency) Sähkömagneettisen spektrin 3MHz - 30MHz alue.

IDA	(Interactive Dissassembler) Hex-Raysin valmistama dissassembleriohjelma
ISI	(Intersymbol Interference) Symbolien välinen keskinäisvaikutus.
MCU	(Micro Controller Unit) Mikrokontrolleri
MIL-STD	(Military Standard) USA:n armeijan standardi.
RAM	(Random Access Memory) Suorasaantimuisti
ROM	(Read Only Memory) Lukumuisti.
SDK	(Software Development Kit) Kokoelma ohjelmistokehitystyökaluja.
Social engineering	Tiedustelun muoto, jossa manipuloidaan henkilö paljastamaan luottamuksellisia tietoja tai tekemään toimia sen sijaan, että käytettäisiin murtautumista tai teknistä keinoa.
SSB	(Single Sideband) Yhden sivukaistan lähete. (LSB = lower sideband = alempi sivukaista, USB = upper sideband = ylempi sivukaista)
STANAG	(Standardization agreement) NATO standardisointisopimus
TADIL	(Tactical Digital Information Link) Taktinen digitaalinen tiedonsiirtolinkki
TX	(Transmit) Lähetys
UHF	(Ultra High Frequency) Sähkömagneettisen spektrin 300 MHz - 3GHz taajuusalue

1 JOHDANTO

Tämä opinnäytetyö on tehty Bandercom Oy Ltd:n tilauksesta kesällä 2011. Työn tavoitteena on modifioida Kenwood TS-480SAT -radiopuhelin tukemaan TADIL-A/Link 11 -sotilasstandardia HF-taajuuksilla tapahtuvaa tiedonsiirtoa varten.

Laitetta on modifioitu paljon jo valmiiksi. Modifikaatiot on testattu ja laitteen on todettu soveltuvan vastaanottoon, mutta sen kaistanleveys on osoittautunut hieman liian kapeaksi lähettämiseen nopeimmilla lähetteillä.

Lähetyksessä ja vastaanotossa on käytössä audiotaajuusalueella digitaalisella signaaliprosessorilla toteutettu digitaalinen suodin. Lisäksi laitteessa on välitaajuudella sijaitseva analoginen kidesuodin kaistanpäästösuotimena.

Digitaalinen suodin käyttää lähetyksessä ja vastaanotossa eri päästökaistanleveyksiä. Vastaanotossa suodin on säädettävissä melko rajattomasti PC-pohjaisella ohjelmistolla 6 kHz:iin asti. Lähetyksessä maksimikaistanleveys on tuntemattomasta syystä rajoitettu enintään 2,4 kHz:iin.

Tämän opinnäytetyön kuluessa tutkin eri mahdollisuuksia muuttaa Kenwood TS-480SAT:n ominaisuuksia lähetyksen suhteen sekä vertailen ohjelmisto- ja laitteistopohjaisia toteutusvaihtoehtoja. Valitettavasti Suomen tekijänoikeuslaki asetti rajoituksia saadun tiedon julkaisemiseen. Sen sijaan, että pääsisin esittelemään tilanteita yksityiskohtaisesti, joudun paikoin esittelemään asioita yleisellä tasolla.

Projektina takaisinmallinnus sekä sulautetun ohjelman modifiointi toivat arvokasta tietotaitoa takaisinmallinnuksesta ja digitaalitekniikasta sekä minulle että Bandercom Oy Ltd:lle.

2 TUTKIMUKSEN LÄHTÖKOHDAT

2.1 Yritys

Bandercom Oy Ltd on vuonna 1992 perustettu suomalainen yritys, joka myy ja vuokraa langattomia viestintävälineitä sekä tuottaa langattomaan viestintään liittyviä palvelukokonaisuuksia loppukäyttäjille, kuten kuluttajille, yrityksille ja julkishallinnolle (Bandercom 2011). Suurin asiakasryhmä on julkisen sektorin asiakkaat, kuten puolustusvoimat, poliisi ja pelastuslaitos. Bandercom suorittaa valtakunnallisen viranomaisverkko VIRVE:n tukiasema-asennuksia sekä järjestelmäintegrointia VHF-verkkojen ja VIRVE:n välillä. Bandercom vastaa myös järjestelmien toiminnasta, suunnittelusta, tuotekehityksestä ja käyttöönotosta. Ylläpitopalvelut ja järjestelmien kehittäminen kuuluvat myös Bandercomin toimenkuvaan (Bandercom 2011).

Bandercom on alansa johtavia toimijoita Suomessa. Se oli ensimmäinen alan yritys internetissä, ja nykyään suurin osa laitemyynnistä tapahtuu verkossa. Bandercom toimittaa keskimäärin 35 radiolaitetta päivässä. Lisäksi Bandercom toimittaa myös pelkkiä varusteita. (Bandercom 2011.)

Bandercom on aloittanut vientitoimintansa vuonna 1994, ja se on kerännyt jo yli 3000 asiakasta yli 60 maassa (Bandercom 2011). Etenkin parannellut versiot yleisesti saatavilla olevista radiopuhelimista ovat tehneet Bandercomista tunnetun myös kansainvälisesti. Bandercomin liikevaihto 2011/03 oli 767 000 euroa ja tilikauden tulos 50 000 euroa (Finder 2012).

2.2 Tutkimuksen tausta

Vuonna 2007 tehty projekti Kenwood TS-480SAT -radiopuhelimen muokkaamiseksi oli onnistunut Bandercom Oy Ltd:llä toiveiden mukaisesti. Vuosia on kulunut, ja laitteita on toimitettu paljon. Ajatus radiopuhelimen kehittämiseksi entistä paremmaksi oli kuitenkin kytenyt. Keskustellessani aikaisemman modifikaation suunnitelleen Mika Niemelän kanssa, hän sanoi: ”tarkoitus on tehdä parhaista laitteista vieläkin parempia”.

Olin ollut töissä Bandercomilla muissa tehtävissä opintojeni ohella noin kaksi vuotta, kun opinnäytetyön aiheen hankkiminen tuli ajankohtaiseksi. Petri Peltolalla, Bandercom Oy Ltd:n toimitusjohtajalla, oli nopeasti tarjolla useitakin eri opinnäytetyöaihevaihtoehtoja, joista mielenkiintoisimpana ja haastavimpana pidin tätä projektia.

2.3 Tutkimuksen tavoitteet

Tutkimukseni tavoitteeksi asetettiin löytää keino, jolla saadaan Kenwood TS-480SAT:n lähetyskaistanleveys laajemmaksi. Silloin se tukisi paremmin TADIL-A/Link 11:n mukaisia suurempia datanopeuksia.

Ensimmäinen välitavoitteeni oli kartoittaa asiat, jotka vaikuttavat kaistanleveyteen. Sen jälkeen tuli mahdolliseksi selvittää, millä keinoilla radion kaistanleveyden voisi muuttaa leveämmäksi. Kun erilaiset keinot olivat selvinneet ja paras vaihtoehto valittu, oli mahdollista alkaa tehdä varsinaista toteutusta.

Lopullinen tavoite oli saada aikaan radio, joka vastaisi TADIL-A/Link 11:n vaatimuksia HF-taajuuksilla.

3 LAITTEEN ESITTELY

Tutkimuksen kohteena oleva radiopuhelin on Kenwood TS-480SAT (Kuva 1). Se on julkaistu vuonna 2004, ja se on suunniteltu alun perin radioamatöörikäyttöön.

Kuva 1: Kenwood TS-480SAT/HX (Kenwood, 2011)

Radiopuhelimeen on tehty suuria muutoksia, mutta muutosten yksityiskohtiin en tässä opinnäytetyössäni syvenny. Käyn vain läpi joitain oleellisimpia asioita.

Peruskokoonpanossa Kenwood TS-480SAT:n taajuusalue kattaa radioamatööritaajuudet 160-6 m ja sen lähetysteho on säädettävissä välillä 5W - 100 W. Lähetelajeina AM, SSB ja FM. Tästä laitteesta on poistettu radioamatööritaajuuksien ulkopuolisen lähetyksen rajoitus.

SSB:llä vastaanotossa on Bandercomin omaa tuotantoa oleva 3,5 kHz:n kidesuodin, joka on asennettu lisävarusteena hankittavalle suotimelle varattuun paikkaan. Kuitenkin lähetyksen aikana signaali kulkee aina tehdasasenteisen 2,4 kHz kaistanlevyisen kidesuotimen läpi. Sekä lähetyksessä että vastaanotossa signaali kulkee audiotaajuisena Texas Instruments TMS320VC5402:lla toteutetun digitaalisen suotimen kautta.

4 TAKAISINMALLINNUS

Sana takaisinmallinnus (engl. reverse engineering) tarkoittaa prosessia, jossa hankitaan tuntemusta tai suunnittelumalleja mistä tahansa ihmisen tekemästä. Takaisinmallinnuksella hankitaan usein sellaista tietoa, ideoita tai suunnittelutapoja, joita ei ole muuten saatavilla. Haluttu tieto saattaa olla jollain toisella, joka ei halua jakaa sitä. Tieto voi olla myös hävinnyt tai tuhoutunut. (Eilam 2005, 3-4.)

Useissa teknologiayrityksissä kilpailijoiden tuotteiden takaisinmallinnus on normaali keino hankkia tietoa kilpailijoiden innovaatioista. USA:n korkein oikeus onkin kutsunut takaisinmallinnusta tärkeäksi osaksi keksintöjen tekemisessä. (Huang 2003b, 180.)

Monilla teollisuuden aloilla takaisinmallinnus sisältää tuotteen tutkimista mikroskoopilla tai sen purkamista ja päättelyä mitä mikäkin osa tekee. Tavallisesti siinä selvitetään, millä tekniikoilla laite on tehty ja kuinka voisi itse tehdä paremman laitteen. (Eilam 2005, 3-4.)

Laillista takaisinmallinnusta on Huangin mukaan kolmen tyyppistä: Kilpaileva takaisinmallinnus, joka pyrkii vastaavan tai paremman tuotteen kehittämiseen. Yhteensopivuuteen tähtäävä takaisinmallinnus pyrkii selvittämään miten voisi tehdä tuotteen, joka toimii yhteen tutkittavan tuotteen kanssa. Kolmannessa takaisinmallinnuksen muodossa tutkijat pyrkivät hankkimaan tietoa ilman kaupallista päämäärää. (Huang 2003b, 180.)

Laitteistoille ja ohjelmistoille suoritettava takaisinmallinnus on tavallista yritysten tuotekehityksessä.

4.1 Tietokoneohjelmien takaisinmallinnuksen laillisuus

Ohjelmiston takaisinmallinnusta kohdellaan Suomen laissa eri tavalla kuin muuta takaisinmallinnusta. Tärkeimmät seikat, jotka rajoittavat tietokoneohjelmien takaisinmallinnusta on mainittu tekijänoikeuslaissa erityisesti 25 j ja k §.

Tekijänoikeuslain tarkoituksena on suojata ohjelmistoja ja muita immateriaalioikeuksia luvattomalta kopioinnilta. Eilamin (2005, 19) mukaan, takaisinmallinnuksen vastustajat ovat väittäneet, että takaisinmallinnusprosessi loukkaa tekijänoikeuslakia, koska prosessin aikana luodaan "välillisiä kopioita".

Esimerkiksi ohjelman dekompiloinnissa ohjelma täytyy kopioida muistiin vähintään kerran. Ajatuksena on, että vaikka dekompilointi itsessään olisi laillista, välillinen kopiointi loukkaa tekijänoikeuslakeja. Väite ei ole toiminut oikeudessa. Useissa oikeustapauksissa Yhdysvalloissa välillistä kopiointia on pidetty kohtuullisena käyttönä (engl. fair use), koska lopullinen tuote ei ole sisältänyt mitään alkuperäisestä tuotteesta kopioitua. (Eilam 2005, 19.)

Teknisessä mielessä edellinen on täysin ymmärrettävää, koska välillisiä kopioita luodaan aina ohjelmistoja käytettäessä riippumatta, onko kyseessä takaisinmallinnus vai ei. Ensimmäinen kopio luodaan, kun ohjelma kirjoitetaan asennettaessa kiintolevylle DVD-ROM-levyltä. Toinen välillinen kopio luodaan käynnistettäessä asennettu ohjelma kiintolevyltä. Silloin ohjelmakoodi kopioidaan keskusmuistiin, että sitä voitaisiin suorittaa prosessorilla. (Eilam 2005, 19.)

Suomen tekijänoikeuslaissa ei ole mainintaa kohtuullisesta käytöstä, josta Yhdysvaltojen tekijänoikeuslaki puhuu. Suomen tekijänoikeuslaissa on maininnat esimerkiksi sitaattioikeudesta ja yksityiseen käyttöön kopioinnista.

4.2 Sallittu muunteleminen ja virheiden korjaus

Tekijänoikeuslain mukaan ”ohjelman laillisesti hankkinut saa tehdä ohjelmaan muutoksia, jotka ovat tarpeen ohjelman käyttämiseksi aiottuun tarkoitukseen. Tämä koskee myös virheiden korjaamista” (Tekijänoikeuslaki, 25 j § 1). Tätä voidaan kuitenkin rajoittaa, koska tekijänoikeuslain 25 j § momentti 5 jättää mahdollisuuden kieltää muokkaukset sopimusehdolla.

Tavallisesti sulautettujen laitteiden firmwareille eli ohjelmistolle ei ole kirjoitettu sopimuksia tai käyttöehtoja, joihin käyttäjän tulisi sitoutua. Näkisin, että mahdollisuus muokkaukseen on olemassa suurimmassa osassa sulautettuja laitteita. Korkeimman oikeuden ennakkopäätöksessä KKO:2008:45 selvennetään tarkemmin, miten tekijänoikeuslakia tulee tulkita.

KKO:2008:45:n mukaan tarkasteltaessa sitä, mikä on tietokoneohjelman "aiottu käyttötarkoitus" on perusteltua kiinnittää huomiota muun muassa siihen, millaiseen toiminnalliseen tarpeeseen ohjelma on hankittu ja millaista sen tavanomainen käyttö kyseisessä yhteydessä on. Ohjelma voi toisaalta muodostaa yhdessä esimerkiksi tuotantolaitteiston tai -koneiston kanssa kokonaisuuden, jossa kumpikaan ei ole käyttökelpoinen ilman toista. Tällöin puhutaan niin sanotusta sulautetusta järjestelmästä. Mitä yksinkertaisempi sulautetun järjestelmän sisältämä ohjelma on, sitä suurempi merkitys ohjelman tarkoitusta määriteltäessä yleensä on laitteiston käyttötarkoituksella.

Korkeimman oikeuden ennakkopäätös siis painottaa sitä, mihin tarpeeseen ohjelma on hankittu ja millaista sen tavanomainen käyttö on. Sulautetuissa järjestelmissä usein laitteiston käyttötarkoitus määrittää ohjelman käyttötarkoituksen (KKO:2008:45). Laitteistomuutokset pakottavat usein myös ohjelmistomuutoksiin.

Kyseisessä ennakkopäätöksessä on kysymys sulautetusta järjestelmästä, johon tehty laitteistomuutos pakotti ohjelmistomuutoksen tekemiseen. Se salli ohjelmiston muutoksen, koska ohjelman käyttötarkoitus ei muuttunut eikä muuttamista ollut kielletty sopimuksella. (KKO:2008:45.)

4.3 Tutkiminen ja takaisinmallinnus

Määrittelen takaisinmallinnuksen ja tutkimisen hyvin läheiseksi toisilleen. Ne sisältävät useita yhtäläisyyksiä, eikä niiden välinen raja ole mitenkään selkeä. Molempien lähtökohta on ohjelmiston periaatteiden ja ideoiden selvittäminen. Takaisinmallinnuksessa tutkiminen on välttämätöntä ja ainoa keino saada haluttua tietoa.

Tekijänoikeuslaki sallii sen henkilön, joka saa lain mukaan käyttää tietokoneohjelmaa, tarkastella, tutkia ja kokeilla tietokoneohjelman toimintaa niiden ideoitten ja periaatteiden selvittämiseksi, jotka ovat ohjelman osan perustana. Tarkastelu, tutkiminen ja kokeilu tulee tekijänoikeuslain mukaan tehdä ohjelman tietokoneen muistiin lukemisen tai ohjelman näyttämisen, ajamisen, siirtämisen tai tallentamisen yhteydessä. Tätä oikeutta ei voida kumota edes ohjelman sopimusehdoissa. (Tekijänoikeuslaki, 25 j §.)

Tekijänoikeuslaki ei siis suojaa ideoita eikä algoritmeja. Se suojaa koodin suoralta kopioimiselta. Mielestäni lakikohta ei edes rajoita tutkimista tavanomaisen käytön yhteyteen vaan sisältää paljon tulkinnan varaa.

Tietokoneohjelmien yhteensopivuuteen tähtäävässä takaisinmallinnuksessa pätee hieman eri säännöt. Tekijänoikeuslaki 25 k § 1 sanoo:

> *Ohjelman koodin kopioiminen ja sen muodon kääntäminen on sallittua, jos nämä toimenpiteet ovat välttämättömiä sellaisten tietojen hankkimiseksi, joiden avulla voidaan saavuttaa yhteentoimivuus itsenäisesti luodun tietokoneohjelman ja muiden ohjelmien välillä.*

Tekijänoikeuslaki määrittää myös ehtoja, joiden tulee täyttyä, ennen kuin yhteensopivuuteen tähtäävää takaisinmallinnusta voidaan suorittaa. Tekijänoikeuslaki rajoittaa myös saatujen tietojen käyttöä ja julkaisemista.

5 TAKAISINMALLINNUSTEKNIIKKAA SULAUTETUILLE JÄRJESTELMILLE

5.1 Takaisinmallinnuksen motiivit

Kävin läpi joitain motiiveja takaisinmallinnukselle jo aiemmin takaisinmallinnusta yleisesti käsittelevässä luvussa. Niiden lisäksi Grand esitteli Black Hat DC 2011 Workshopissa joitakin muita motiiveja takaisinmallinukselle.

Grandin (2011) mukaan seuraavat motiivit ajavat takaisinmallinnukseen. Laitteiston turvallisuusjärjestelmien testaus vioilta ja heikkouksilta. Hän mainitsee myös kuluttajansuojan, josta Huang tarkentaa esitelmässään (2003a, 4) esittämällä aiheeseen liittyvät kysymykset: ”sainko sitä mistä maksoin? Tekeekö laite jotain, mitä en haluaisi? Rikkooko se oikeuksiani?”.

Sotilastiedustelu pyrkii selvittämään minkälainen laitteisto on, kuinka laite on suunniteltu ja ketkä sen ovat suunnitelleet. Lisäksi koulutuksellinen tutkiminen ja uteliaisuus johtavat selvittämään, kuinka asiat toimivat. Tutkimiseen saattaa liittyä myös halu tehdä jotain uutta ja sellaista, mitä muut eivät ole vielä tehneet. (Grand 2011, 3.)

Grand mainitsee mahdollisiksi motiiveiksi myös palvelun varastamisen, eli jonkin palvelun hankkimisen ilmaiseksi, joka tavanomaisesti maksaisi. Hän mainitsee myös tiedon hankkimisen tuotteen kloonaamiseksi, että saataisiin markkinaetua sekä käyttäjän identiteetin peukaloimisen järjestelmään pääsyn saamiseksi. Lisäksi motiiveina voi olla suojausominaisuuksien ohittaminen tai käyttöoikeuksien korottaminen, että saataisiin parempi hallinta järjestelmään. (Grand 2011, 4.)

Takaisinmallinnuksen motiivina voi olla myös jonkin tuotteen parantelu vieläkin paremmaksi. Se voi tapahtua lisäämällä, poistamalla tai muokkaamalla ominaisuuksia. Tämä parantelu on tässä opinnäytetyössäni tavoitteena.

5.2 Top-down -metodi

Top-down -metodi tarkoittaa ylhäältä alaspäin suuntautuvaa lähestymistapaa. Siinä hankitaan ensin yleiskäsitys tutkittavasta asiasta, jonka jälkeen syvennytään tarvittaviin yksityiskohtiin.

Tutkimus voi alkaa toiminnallisten lohkojen ja pääkomponenttien identifioimisella. Lohkokaavioesitykset helpottavat toiminnallisten lohkojen välisen vuorovaikutuksen ymmärtämisessä. Tutkittavasta aiheesta tehdyt referaatit ja tutkielmat auttavat sisäistämään laitteen rakennetta ja toimintaa. (Hyytiäinen 2011.)

Kun päätason lohkot on tehty, voidaan siirtyä tutkimaan yhtä lohkoa kerrallaan. Näitä lohkoja käsitellään kuten edellisellä tutkimuskierroksella. Tehdään tarvittaessa uusi lohkokaavio tämän lohkon sisäisistä vuorovaikutuksista sekä mahdollisesti tutkielmia aiheista, jotka tarvitsevat lisätietoa. Kun lohkon toimintaa on tarpeellisella tarkkuudella tutkittu, siirrytään seuraavan lohkon tarkasteluun. Tätä tutkimusprosessia voidaan jatkaa tarpeen mukaan pureutumalla niin syvälle laitteen toimintaa, kuin tarpeellista. (Hyytiäinen 2011.)

Top-down -metodia käyttäessä on helppoa jakaa takaisinmallinnustehtäviä useille henkilöille. Tarvittaessa tutkittavan laitteen toiminnasta voidaan koostaa yhtenäinen dokumentaatio.

5.3 Informaation kerääminen

Sulautettujen järjestelmien takaisinmallinnuksessa ovat suurena apuna kaikki mahdolliset dokumentit, jotka käsittelevät laitteen ominaisuuksia ja toimintaa.

Grandin (2011, 6) mukaan suurin osa suunnitteluinsinööreistä ei tunne turvallisuusnäkökulmia, ja suuri osa tuotteista perustuu komponenttivalmistajien toimittamiin, yleisesti saatavilla oleviin referenssikytkentöihin. Komponentteihin on myös helppo päästä käsiksi, ja ne on helppo identifioida ja tutkia.

Tarvittavaa tietoa komponenteista löytyy myös komponenttikohtaisista datalehdistä. Lisäksi komponenttien käytöstä on monesti olemassa sovellusohjeita (engl. application note). Sovellusohjeista löytyy referenssikytkentöjä eli tavallisia kytkentämalleja, joita laitesuunnittelijat voivat käyttää omissa suunnitelmissaan vapaasti. Sovellusohjeet sisältävät usein myös kirjallisen toimintakuvauksen.

Suunnittelijat ja valmistajat haluavat tuotteen testauksen ja debuggauksen olevan vaivatonta (Grand 2011, 8). Siksi valmistajat tekevät testauspisteitä ja -liitäntöjä tuotteisiin. Nämä päätyvät usein myös loppukäyttäjille, jolloin niitä voidaan hyödyntää myös takaisinmallinnuksessa.

Valmistajat kierrättävät samoja hyviksi todettuja suunnittelumalleja tuotteesta toiseen. Jos tutkittavaan laitteeseen ei ole materiaalia saatavilla, voidaan usein käyttää apuna materiaalia saman valmistajan lähes vastaavasta tuotteesta. Tarvittaessa voidaan käyttää muun valmistajan materiaaleja saman tyyppisestä tuotteesta.

Luulen, että paras yksittäinen lähde on toiminnallinen kuvaus, joka on laitteen suunnitelleen yrityksen sisäinen dokumentti laitteen toiminnasta. Se sisältää tiedon, mitä laitteen suunnittelijat ovat nähneet tarpeelliseksi dokumentoida siltä varalta, että laitteeseen joutuisi joskus tekemään muutoksia. Toiminnallinen kuvaus saattaa olla vaikea hankkia. Niitä pidetään yrityssalaisuuksina, ja yrityksillä on usein tiukka politiikka, että niiden ei anneta vuotaa tuotekehitysosaston ulkopuolelle.

Huolto-oppaissa on myös paljon arvokasta tietoa. Ne sisältävät usein suhteellisen tarkan toiminnallisen kuvauksen laitteen huollettavista osista. Huolto-oppaassa on tavallisesti laitteen piirikaaviot, komponenttilistat, kalibrointiarvoja, osasijoittelukuvat, räjäytyskuvia sekä puolijohdekomponenttien kuvaukset.

Piirikaaviot säästävät aikaa komponenttien välisten yhteyksien tutkimisessa. Käyttöohjeet sisältävät laitteen käyttämiseen tarvittavan ohjeistuksen lisäksi teknisiä tietoja. Jotkut käyttöohjekirjat sisältävät myös piirikaavion. Myyntimateriaaleista saattaa löytää teknisiä tietoja ja toiminnallista selostusta.

Käyttäjiltä saatava informaatiota ei voi väheksyä. Internetissä on runsaasti postituslistoja, sivustoja ja foorumeita, joilta voi etsiä tai pyytää apua laitteen tutkimisessa. Internetistä on hyvä etsiä viitteitä, onko joku muu jo tehnyt aikaisemmin saman, mihin itse pyrkii.

Social engineering on keino hankkia luottamuksellista tai muuten vaikeasti saatavissa olevaa tietoa. Social engineering tarkoittaa esimerkiksi keskustelemista laitteen suunnitelleen henkilön kanssa ja taivutella tai saada hänet muuten epähuomiossa paljastamaan haluttua informaatiota.

Kaikkien saatujen tietojen kanssa tulee huomioida, että niiden sisältämä informaatio ei ole välttämättä täysin oikeaa. Laitteen valmistaja on saattanut tarkoituksella lisätä dokumentteihinsa virheellistä tietoa takaisinmallinnuksen hankaloittamiseksi.

Tarpeellisilta osilta kytkennät ja ominaisuudet pitää siis tarkistaa empiirisesti. Seuraava takaisinmallinnuksen vaihe on laitteiston kotelon avaaminen ja komponenttien sekä niiden välisten yhteyksien tutkiminen esimerkiksi edellisessä luvussa mainittua top-down-metodia hyödyntämällä.

6 SULAUTETUN OHJELMAN TAKAISINMALLINNUS

6.1 Sulautetun ohjelman hankkiminen

Firmwaren hankkimisessa on aluksi syytä tarkistaa, onko laitteen valmistajalta saatavilla lähdekoodia tai binääritiedostoa firmwareen. Jos valmistaja tarjoaa päivityksiä tuotteelleen, jossain sain tapauksessa yksinkertaisin vaihtoehto saattaa olla hankkia päivitysohjelma, jolla uusi firmware siirretään laitteelle.

Windows-käyttöjärjestelmälle Windows PE -tiedoston sisältämiä resursseja voidaan lukea ja editoida sitä varten kehitettyjen ohjelmien avulla. Jokin päivitysohjelman resursseista sisältää suurella todennäköisyydellä laitteen binäärikoodin. Windows PE-tiedostojen lukemiseen ja muokkaamiseen soveltuvia ilmaisia ohjelmia ovat esimerkiksi Angus Johnsonin Resource Hacker tai Colin Wilsonin XN Resource Editor.

Aina valmistajalta ei ole firmwarea saatavilla. Seuraavissa kappaleissa esiteltyjä keinoja voidaan käyttää, jos firmwarea ei ole muuten saatavilla.

Prosessori saattaa hakea ohjelmakoodin ulkoisesta muistista. Tiedonsiirtoväylillä siirrettävä data voidaan kerätä ja analysoida logiikka-analysaattorin avulla. Joskus väylällä liikkuva data voi olla salattua, jolloin salausalgoritmi täytyy selvittää datan muuttamiseksi takaisin sellaiseksi, että sen sisältöä voitaisiin analysoida.

Ohjelmakoodin sisältävä muisti voidaan lukea irrottamalla muistipiiri ja lukemalla se tarkoitukseen soveltuvalla laitteella. Se voidaan lukea myös suoraan kytkennässä, kun varmistetaan, että muille komponenteille ei anneta lukemisen aikana käyttöjännitteitä.

Edellä mainittua lukemista voidaan vaikeuttaa monella tapaa. Dataväylän signaaleita on voitu esimerkiksi sekoittaa vaihtamalla niiden järjestystä. Osoiteväylällä voidaan signaaleita invertoida tai sekoittaa samalla tavoin kuin dataväylälläkin. Myös väylällä liikkuvan datan salaus on mahdollista.

Tutkittava prosessori saattaa sisältää ROM-muistia, joka sisältää ohjelmakoodia. Joissain prosessoreissa on useita eri käynnistystiloja, jotka voidaan valita käyttöön ulkoisten kytkentöjen avulla. Joissakin tapauksissa prosessori voidaan pakottaa käynnistymään siten, että käynnistyksen yhteydessä ensimmäinen komento haetaankin eri osoitteesta kuin tavallisesti. Sinne voidaan laittaa hyppykäsky omaan koodiin, joka esimerkiksi siirtää ohjelmakoodin helpommin luettavissa olevaan paikkaan.

6.2 Sulautetun ohjelman analysointi

Kun ohjelmakoodi on hankittu, riippuen hankintatavasta se on tavallisesti joko binääri tai hex dump -muodossa. Hex dump tarkoittaa muotoa, joka tavallisesti sisältää rivillä osoitteen ja osoitteessa olevan datan heksadesimaalimuodossa. Hex dump esityksiä erilaisia, mutta perusperiaate on sama.

Firmwaren tutkimisessa alkuun pääsemisessä ovat tutkittavan prosessorin tuntemus ja oikeanlaiset työkalut ratkaisevassa asemassa. Myös prosessoriin kytkettyjen oheiskomponenttien toiminnasta on syytä olla tietoinen. Tästä syystä firmwaren analysointi vaatii myös huolellista laitteiston takaisinmallinnusta.

Jos firmwaren saa dissassembloitua, saatua assemblykielistä koodia voi yrittää simuloida kohdeprosessorille tarkoitetulla debuggerilla. Osa mikroprosessori- ja -kontrollerivalmistajista tarjoaa niitä ilmaiseksi rekisteröitymistä vastaan. Jos debuggeria ei ole saatavilla, niin se hidastaa tutkimista.

6.2.1 Lähtökohdat

Ohjelmien takaisinmallinnukseen on Eilamin (2005, 140) mukaan kaksi perus lähestymistapaa: offlineanalyysi ja liveanalyysi. Edellä mainituista offlineanalyysi tarkoittaa koodilistauksen tutkimista ilman debuggeria tai koodin suorittamista muulla tavoin. Liveanalyysissä käytetään koodin suorittamiseen tarvittavaa välineistöä.

Eagle (2011, 6 ja 536) kutsuu liveanalyysiä dynaamiseksi analyysiksi ja offline-analyysiä staattiseksi analyysiksi. Termistö ei siis ole täysin vakiintunut.

Offlineanalyysissa binäärikoodi voidaan kääntää ensin helpommin luettavaan muotoon dissassembloimalla tai dekompiloimalla. Halutut tiedot etsitään tätä luettavampaa muotoa hyväksi käyttäen. (Eilam 2005, 140.)

Eilamin (2005) mukaan offlineanalyysi vaatii liveanalyysiin verrattuna parempaa koodin tuntemusta, koska siitä ei näe, mitä dataa kulloinkin käsitellään ja miten se kulkee. Datatyypit täytyy päätellä käytön mukaan.

Eilamin (2005, 140) mukaan offlineanalyysi tulee mahdottomaksi, jos koodin pakkaamista tai salausta on käytetty. Silloin vain liveanalyysi on mahdollinen. Mielestäni offlineanalyysi ei ole silloinkaan mahdotonta, koska pakattu tai salattu data voidaan muuttaa luettavaan muotoon selvittämällä purkualgoritmin ja ohjelmoimalla purkukoodin.

Liveanalyysin etuna on se, että sillä nähdään esimerkiksi rekisterien tilat kunkin komennon suorituksen aikana. Siinä näkyvät myös muuttujien arvot koodin suorituksen eri vaiheissa.

6.2.2 Prosessorin toimintatilat

Useissa prosessoreissa on ulkoisia tuloja, joiden tila prosessorin käynnistyshetkellä määrittää prosessorin toimintatilan. Toimintatilasta riippuen prosessorin ominaisuudet saattavat muuttua paljonkin.

Renesas H8S/2338-mikrokontrollerissa on kolme mode-pinniä (MD0, MD1, MD2), joilla valitaan käynnistyksen yhteydessä mikrokontrollerin toimintatila. Näillä pinneillä voidaan valita esimerkiksi, onko käytössä 8 vai 16-bittinen ulkoinen dataväylä vai kokonaan pois käytöstä ja onko MCU:n sisäinen ROM-käytössä. Niiden avulla voidaan valita myös ohjelmointitila, jolloin ohjelmamuisti voidaan kirjoittaa uudestaan.

Myös Texas Instrumentsin TMS320VC5402:ssa on mode-pinnit, joilla valitaan mistä osoitteesta ohjelmakoodin suorittaminen aloitetaan. Jossain tilassa sisäi-

nen ROM, RAM tai keskeytysvektorit voivat olla myös kartoitettuna (mapped) eri kohtaan osoiteavaruutta. Jossain tilassa sisäinen muisti voi olla korvattavissa ulkoisella.

6.2.3 Muistiavaruus

Prosessorin muistiavaruudesta tärkeimmät tiedot ovat yleiskäyttöisten rekisterien, kontrollirekisterien, keskeytysvektoritaulun, datamuistin, ohjelmamuistin ja ulkoisten I/O:n sijainnit. Ne voidaan selvittää lukemalla prosessorin datalehtiä ja sovellusoppaita sekä tutkimalla laitteen kytkentää. Esimerkiksi Texas Instrumentsin TMS320VC54xx-sarjan DSP:issä keskeytysvektoritaulun sijainti on konfiguroitavissa ohjelmallisesti.

Keskeytysvektorit ovat tavallisesti muistiosoitteita, joista prosessori lataa keskeytyksen tullen ohjelmalaskuriinsa uuden arvon. Keskeytysvektoritaulussa voi olla myös varattu useampi muistipaikka kutakin keskeytystä varten, jolloin siellä voidaan suorittaa useita käskyjä ja lopuksi suorittaa hyppykäsky tai paluu aliohjelmasta.

6.2.4 Käskykanta

Käskykanta tarkoittaa prosessorin tuntemien käskyjen kokoelmaa. Assemblymuotoista koodia tutkittaessa käskykannan tunteminen on välttämätöntä. Käskykanta on prosessorikohtainen, ja sen laajuus ja käskyt riippuvat pitkälti siitä, mihin käyttöön prosessori on suunniteltu.

DSP:t sisältävät enemmän signaalinkäsittelyyn liittyviä käskyjä kuin tavalliset mikroprosessorit. Esimerkiksi C54xx-sarjan signaaliprosessoreissa on MAC- ja FIR-käskyt sekä rengaspuskureideiden käytön mahdollistavia toimintoja, joita ei tavallisista prosessoreista löydy.

6.3 Työkalut

Olen käyttänyt firmwaren takaisinmallinnuksessa Ubuntu 10.04 LTS -käyttöjärjestelmällä varustettua tietokonetta. Siihen on asennettu muun muassa Bless Hex Editor, Anywhere PE Viewer, GNU Octave ja strings. Lisäksi olen käyttänyt Winen avulla IDA Pro Advanced 6.1:ä ja XN Resource Editoria sekä ohjelmoinut tarpeen mukaan myös omia ohjelmia. Osasta tätä projektia varten ohjelmoimistani ohjelmista on lähdekoodit saatavilla liiteosiossa.

Bless Hex Editor on binääritiedostojen tai lohkolaitteiden (engl. block device) sisältämän datan tarkasteluun ja muokkaamiseen soveltuva ohjelma, joka näyttää datan heksamuodossa. Bless näyttää myös tulostettavissa olevat merkit. Sen ominaisuuksiin kuuluu myös datamuunnostaulukko. (Bless 2011.)

Anywhere PE Viewer on Javalla ohjelmoitu ilmainen työkalu, jolla voidaan tarkastella PE-tiedostoja. PE-tiedostojen sisältämien resurssien tallentaminen tiedostoon on mahdollista. Sitä voidaan käyttää suoraan UCware.com -internetsivustolta Java Web Startin avulla. (UCware 2011.)

Strings on yksinkertainen komentorivityökalu, joka tulee GNU Binutils -binäärityökalukokoelman mukana. Sillä voidaan näyttää tiedoston sisältämät tulostettavissa olevat merkit. Stringsin tulosteen voi putkittaa esimerkiksi less-ohjelmaan tai tiedostoon lukemisen helpottamiseksi.

Wine on avoimen lähdekoodin ohjelmisto, jonka avulla monia Windowsille ohjelmoituja ohjelmia voidaan suorittaa muissa käyttöjärjestelmissä. Se mahdollistaa useiden Windowsille ohjelmoitujen ohjelmien käyttämisen ilman, että tarvitsee asentaa Windows-käyttöjärjestelmää ja täten maksaa lisenssiä. (Wine 2011.)

Resource Hacker on freeware apuohjelma, jolla voidaan näyttää, muokata, nimetä uudelleen, tallentaa tiedostoon, lisätä ja poistaa resursseja 32-bittisen Windows suoritus- tai resurssitiedoston sisältä. Resource Hacker sisältää myös muita hyviä ominaisuuksia, kuten kääntäjän ja dekompilerin. Resource Hackeria ei kehitetä enää. Vastaava, ilmainen, avoimen lähdekoodin ohjelma on XN Resource Editor. (Johnson 2011.)

6.3.1 GNU Octave

GNU Octave on avoimeen lähdekoodiin perustuva korkean tason kuvauskieli sekä ohjelmisto, joka on tarkoitettu numeeriseen laskentaan. Se on hyvin samanlainen kuin Matlab. Suuri osa Matlabiin ohjelmoiduista ohjelmista on siirrettävissä suoraan Octaveen. (Octave 2011.)

Octaveen on saatavissa monipuolisesti lisäosia eri tyyppisten laskentatehtävien suorittamiseksi sekä muihin toimintoihin, kuten eri muotoisten tiedostojen lukemiseen ja kirjoittamiseen. Ainoa Octaven lisäosa, jonka tarvitsin, oli Ubuntun ohjelmistovarastostakin löytyvä octave-signal.

Octave-signal sisältää signaalinkäsittelyyn, kuten suodatukseen ja ikkunointiin soveltuvia funktioita. Muut tarvitsemani funktiot löytyivät Octaven mukana tulleiden perusfunktioiden joukosta.

6.3.2 IDA Pro

IDA Pro on belgialaisen Hex-Rays SA:n interaktiivinen dissassembleri ja debuggeri. Sillä voidaan muuttaa binäärikoodia assemblyksi ja suorittaa koodia käsky kerrallaan toiminnan selvittämiseksi. IDA Pro on saatavilla Advanced ja Standard -versioina, joista ensin mainittu tukee useampaa prosessoria. Versiosta 6.0 lähtien IDA Pro on ollut saatavilla graafisella käyttöliittymällä varustettuna Windowsille, Linuxille ja Mac OS X:lle, kun ennen versiota 6.0 graafinen käyttöliittymä on ollut saatavilla vain Windowsille.

Vaikka IDA Pro onkin suhteellisen kallis (Standard 419€, Advanced 809€), niin se helpottaa takaisinmallinnusta niin paljon, että siitä aiheutuvat kustannukset on mahdollista saada nopeasti takaisin töiden nopeutumisen ansiosta. Hex-Raysin internetsivuilla www.hex-rays.com/idapro on tarjolla myös IDA Prosta kokeiluversioita.

IDA Pro:hon on olemassa SDK, joka mahdollistaa lisäosien eli pluginien ja prosessorimoduulien kehittämisen. Lisäksi IDA:n mukana tulee IDAPython, joka mahdollistaa nopean lisäosien ja skriptien kehittämisen tulkatulla Python-kielellä.

6.3.3 Dissassemblointi IDA Prolla

Binääritiedosto avataan IDA Prossa. IDA:lle on mahdollista kertoa, mille kohdeprosessorille koodi on käännetty. Lisäksi voidaan antaa monia muita parametreja esimerkiksi koodin analysointiin liittyen. Seuraavaksi määritetään muistialueet ja syötetyn koodin alkuosoite kohdejärjestelmässä.

Jos avattu tiedosto ei ole IDAn tukemaa muotoa, IDA pyytää käyttäjää ilmoittamaan mistä osoitteesta automaattinen koodin analysointi aloitetaan. Käyttäjän määriteltyä alkukohdan, IDA suorittaa rekursiivisen analyysin, joka dissassembloi binäärikoodin menemällä haarautumis- ja hyppykäskyjen mukaan dissassembloimaan myös alirutiineihin.

IDA Pro jättää analysoimatta alueet, joille ei ole viitteitä analysoitavaksi valitussa koodissa. Siitä seuraa etu, että data-alueita ei yritetä vahingossa dissassembloida.

Analysoituja koodialueita voidaan tarkastella joko tavallisena koodilistauksena tai lohkokaaviomuodossa. IDA osaa myös tehdä puu-mallisia kuvaajia, joista selviää, miten ohjelmassa voidaan päätyä tiettyyn alirutiiniin tai missä alirutiineissa tiettyä dataa käsitellään.

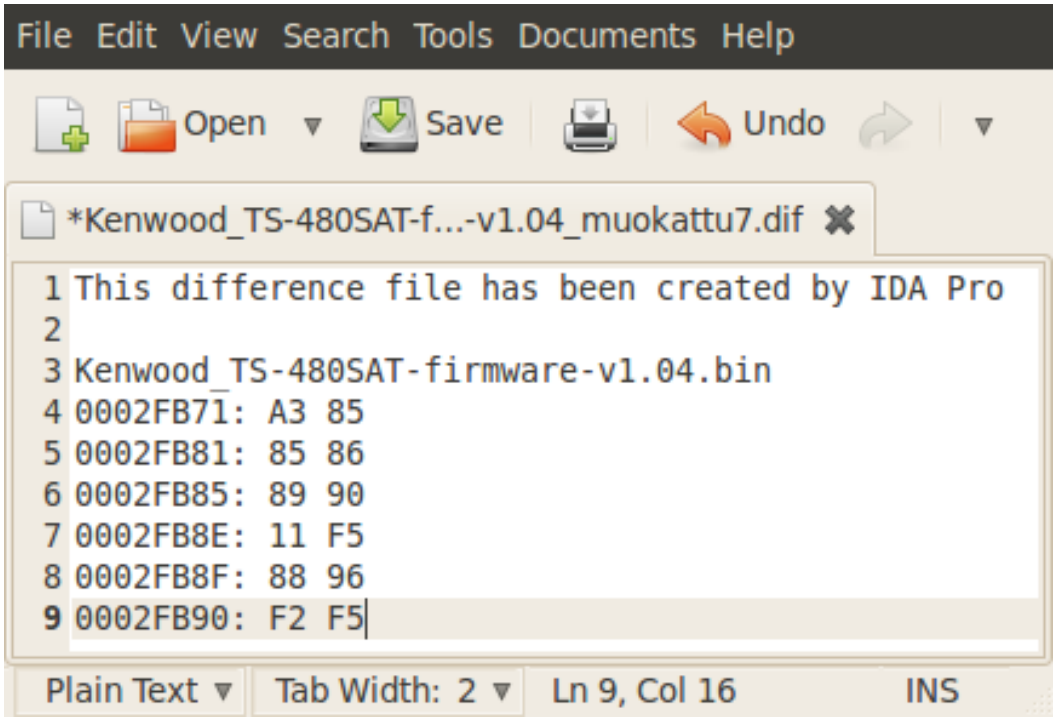
Hyppykäskyissä olevat osoitteet sekä muut osoitteet toimivat linkkeinä, joista pääsee kaksoisklikkauksella tarkastelemaan kohdetta. Osoitteita on myös mahdollista lisätä kommentteihin, joiden sisältöä pääsee tarkastelemaan viemällä hiiren osoitteen päälle tai kaksoisklikkauksella.

6.3.4 Binäärin muokkaaminen IDA Prolla

Varsinaiseen tutkittavan binäärikoodin muokkaamiseen IDA ei pysty, mutta siinä se on erittäin hyvänä apuna. Patch-valikko ei ole oletuksena näkyvissä graafisessa käyttöliittymässä, mutta tarvittaessa sen saa esille muokkaamalla idagui.cfg-tiedostoa. Patch-valikossa on kolme eri muokkausvaihtoehtoa: Change byte, Change word ja Assemble. Assemble-vaihtoehto ei toimi kaikille prosessoreille, koska kaikille prosessoreille ei tule IDA:n mukana assembleria. (Eagle 2011, 237.)

Jos assemble-vaihtoehto ei toimi, voidaan käyttää byten ja sanan muokkaamiseen tarkoitettuja toimintoja. Niillä muokatessa on käytettävä käskyjen heksakoodimuotoja. Tehdyn koodimuokkauksen tulokset tulevat näkyviin dissassembloidussa listauksessa.

Patch-toiminnot eivät muokkaa alkuperäistä koodia vaan IDA:n omaa tietokantaa, joka on muodostettu alkuperäisen binäärin pohjalta. Muutosten tekemisen jälkeen IDA:lla voidaan generoida DIF-tiedosto, joka sisältää tehdyt muutokset tekstinä.



Kuva 2: Esimerkki IDA Pron generoimasta DIF-tiedostosta

IDA Pro:n mukana ei tule apuohjelmaa, jolla voisi toteuttaa DIF-tiedoston mukaiset muutokset binääritiedostoon. DIF-tiedoston syntaksi on todella selkeä (Kuva 2). Olisi siis helppo ohjelmoida oma ohjelma suorittamaan muutosten tekemisen. Internetissä on saatavilla lukuisia ohjelmia, joilla kyseinen tehtävä onnistuu.

6.3.5 Omat ohjelmat

Kaikkiin takaisinmallinnnustehtäviin ei ole valmiita ohjelmia saatavilla, joten ne täytyy kirjoittaa itse. Niissä auttaa jonkin korkean tason tai tulkatun ohjelmointikielen taitaminen. Tulkatuista ohjelmointikielistä esimerkiksi Pythonilla voi tehdä nopeasti ohjelmia suorittamaan pieniä usein toistuvia tehtäviä.

Itse olen käyttänyt tässä projektissa C/C++-kieltä esimerkiksi binääritiedostojen käsittelyä vaativissa operaatioissa ja Octave-ohjelmointikieltä matemaattisiin operaatioihin. Lisäksi olen hyödyntänyt Bash-skriptikieltä edellä mainituilla kielillä ohjelmoimieni ohjelmien ja muiden ohjelmien toiminnallisuuksien yhdistämiseksi tarpeen mukaan. Tätä opinnäytetyötä varten itse tekemieni ohjelmien lähdekoodeja löytyy liiteosiosta.

7 KÄYTÄNNÖN TOTEUTUS

Työn tavoitteena on modifioida Kenwood TS-480SAT radiopuhelin tukemaan TADIL-A/Link 11 -sotilasstandardia HF-taajuuksilla tapahtuvaa tiedonsiirtoa varten. HF-taajuuksilla tarkoitettu tiedonsiirto on määritelty seuraavalla tavalla USA:n armeijan kenttäohjeessa FM3-52 (2002, 73) :

> *TADIL-A/Link 11 on salattua dataa siirtävä digitaalinen linkki, joka toimii half-duplexina. Se siis lähettää ja vastaanottaa, mutta ei tee molempia yhtä aikaa. Se siirtää digitaalista informaatiota ilma-, maa- ja merivoimien taktisien datajärjestelmien kesken. Se on tarkoitettu siirtämään dataa pääasiallisesti näköalueen ulkopuolelle. TADIL-A:ta voidaan käyttää HF tai UHF taajuuksilla, mutta armeija käyttää sitä vain HF:llä. (Suomennos kirjoittajan)*

TADIL-A/Link 11 on tarkemmin standardoitu STANAG 5511:ssä sekä MIL-STD 6011:ssä.

Minimi kaistanleveys TADIL-A:ssa on periaatteessa noin 2,4 kHz. Ensimmäinen datääänі 39-äänisessä DPSK-tiedonsiirrossa on 393,75 Hz:n ja viimeinen 2812,5 kHz:n kohdalla (MIL-STD-188-110B, 69 - 70). Aldrichin (2003, 98) mukaan 300-3500 Hz:n kaistan täytyy pysyä $\pm$ 1.5 dB rajoissa.

Kenwood TS-480SAT:ia on testattu ja se on todettu soveltuvan vastaanottoon, mutta sen kaistanleveys on osoittautunut liian kapeaksi lähettämiseen. Sekä lähetyksessä että vastaanotossa käytetään digitaalisella signaaliprosessorilla toteutettua digitaalista suodinta.

7.1 Vaatimusten määrittely

MIL-STD-188-203-1A yhteistoiminta- ja suorituskykystandardit taktiselle digitaaliselle tiedonsiirtolinkille sekä NATO-standardi STANAG 5511 Annex B muodostavat minimistandardit tekniikalle ja suunnittelukriteereille TADIL-A/Link 11:a käyttäville laitteistoille (Aldrich 2003, 1). Aldrichin(2003, 89-93) materiaalissa kerrotaan, miten TADIL-A/Link 11 soveltuvuuden mittaukset audion osalta tehdään.

Aldrichin mukaan 300 - 3500 Hz:n kaistalla audiovasteen tulee olla ±1,5dB:n rajojen sisällä. Audiovaste mitataan siten, että asetetaan signaaligeneraattorin amplitudiksi 0dBm ja säädetään se ensin taajuudelle 300Hz. Samalla spektrianalysaattoriin asetetaan maksimiarvon tallennus. Sen jälkeen säädetään spektrianalysaattoriin keskitaajuudeksi 2kHz ja taajuusalueeksi 4kHz. Spektrianalysaattorin säädön jälkeen mitataan lähdön amplitudit 3500Hz:n asti säätämällä signaaligeneraattorin taajuutta 100Hz:n askelin-. Lopuksi tarkistetaan, että amplitudit ovat ±1,5dB:n rajoissa. (Aldrich 2003, 89-93.)

Digitaaliset lähetelajit vaativat lineaarisen vaihevasteen välttyäkseen säröltä ja symbolien väliseltä keskinäisvaikutukselta (engl. lyh. ISI) (Bloom 2009, 15.15). Lineaarinen vaihevaste tuottaa myös luonnollisemman kuuloisen audion (Bloom 2009, 15.15). Digitaalisista suotimista lineaarisen vaihevasteen vaatimuksen toteuttaa vain symmetrinen FIR-suodin.

7.2 Ratkaisuvaihtoehtojen tutkiminen

Käytännössä TX-suodatusongelman ratkaisuun on olemassa kaksi vaihtoehtoa: Radion kytkentöihin voidaan tehdä muutos esimerkiksi siten, että korvataan signaalitiellä oleva DSP jollain omalla signaalinkäsittelyä suorittavalla laitteella. Toinen vaihtoehto on muokata nykyisen DSP:n ohjelmistoa siten, että se sallii suuremman kaistanleveyden käyttämisen lähetyksessä.

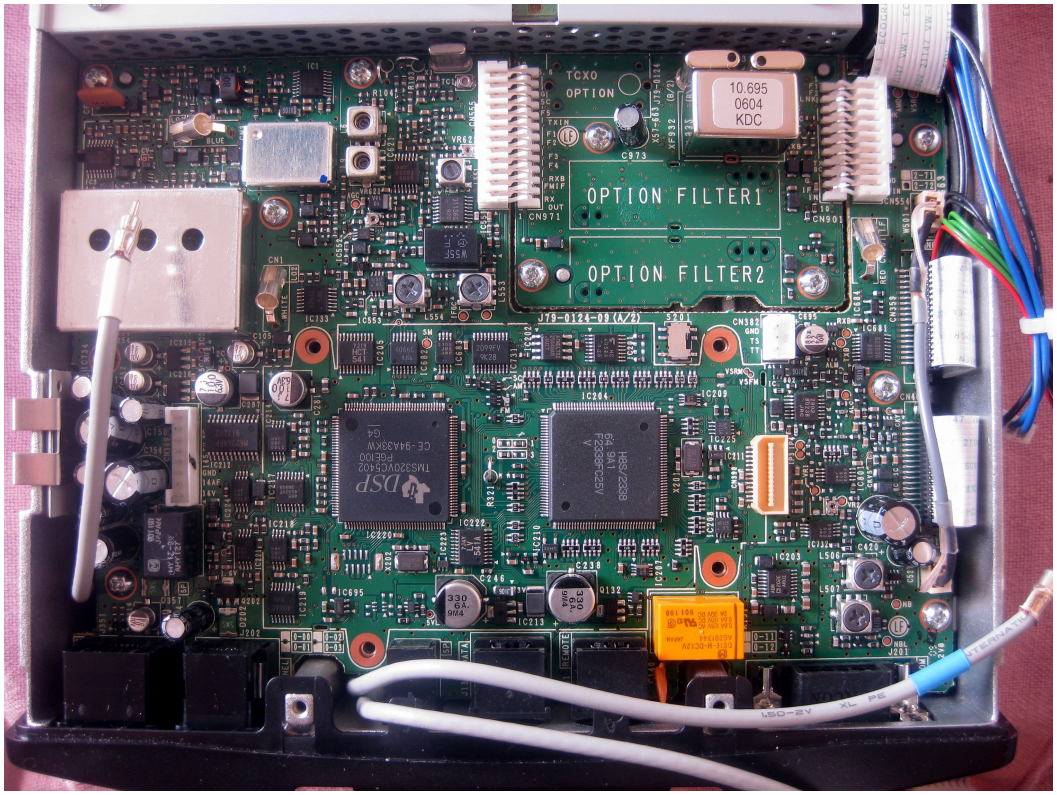
Tarkoitukseni oli alun perin etsiä keino, miten korvata olemassa olevan DSP-suodattimen leveämmän päästökaistan omaavalla omalla suotimella. Lopputulokseen pääsemiseksi oli kaksi lähestymistapaa: laitteisto- tai ohjelmistomodifikaatio.

Projektin alkuvaiheessa laitteistomodifikaatio näytti väistämättömältä. Tutkin tarkkaan radion kytkentöjä ja selvitin, miten lähetystä varten saisi laitettua oman signaaliprosessointiyksikön tai ohitettua olemassa olevan signaaliprosessorin muulla keinolla. Vaihtoehtoina olisi ollut signaaliprosessori tai -kontrolleri tai CPLD- tai FPGA-piiri. Käytännössä aloitin tutkimuksen laitteiston takaisinmallinnuksella.

Tutkimuksen edetessä löysin mahdollisuuden hyödyntää radion firmwaren päivitysohjelmaa firmwaren hankkimiseksi. Lopulta päädyin firmwaren muokkaamiseen, koska laitteistomuutokset olisivat voineet aiheuttaa sähkömagneettisten häiriöiden lisääntymistä ja kohteena oli sähkömagneettisille häiriöille herkkä radiolähetinvastaanotin.

Laitteistomuutokset aiheuttavat myös riskin laitteen rikkoutumiselle, kun joudutaan irrottamaan komponentteja. Komponenttien lisääminen aiheuttaa myös suuremman riskin laitteen rikkoutumiselle käytössä. Virrankulutus olisi myös kasvanut.

7.3 Takaisinmallinnus



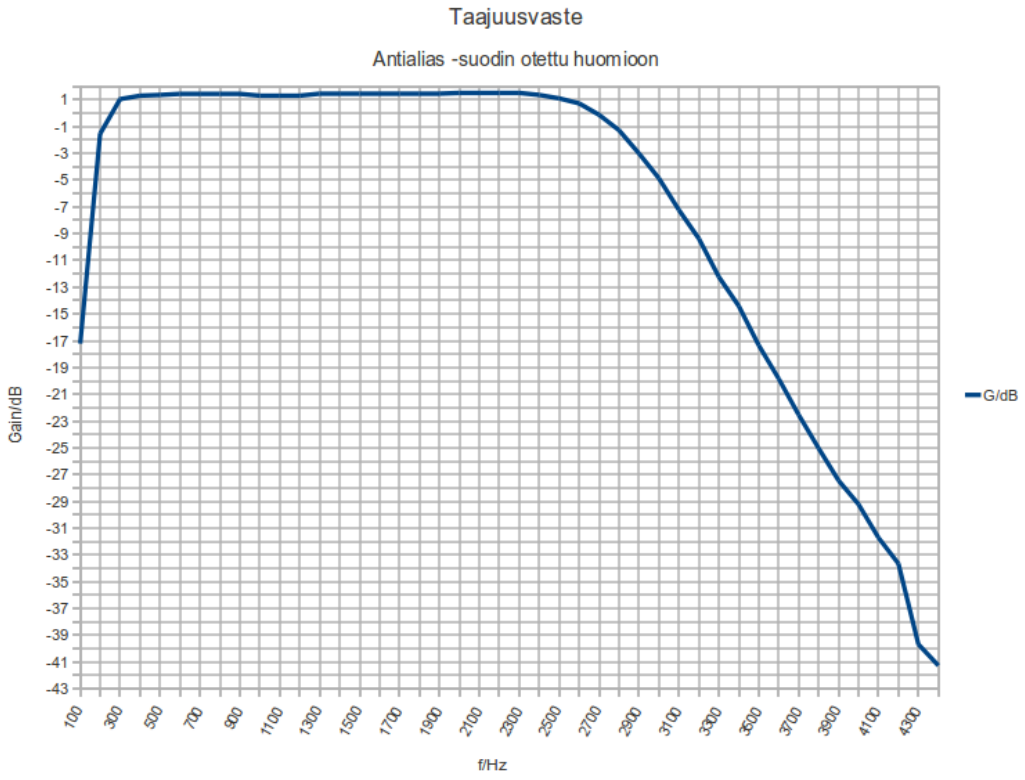
Kuva 3: Prosessorilevy suojakuoret poistettuna

Takaisinmallinnus alkoi hakemalla tietoa Kenwood TS-480SAT:n toiminnasta kirjallisista lähteistä. Sain toimeksiantajani Bandercom Oy Ltd:ltä kautta kyseisen radion huolto-oppaan (Kenwood 2003), jossa oli kytkentäkaaviot ja hyvä toimintaselostus yleisellä tasolla. Lisäksi hankin tarvittavien komponenttien datalehtiä ja sovellusoppaita. Paikallistin tarvittavat komponentit (Kuva 3).

Olin kriittinen huolto-oppaan (Kenwood 2003) materiaalin oikeellisuuden suhteen, erityisesti kytkentäkaavion suhteen. Tarkistin tarvittavat kytkennät uudestaan muutoksien varalta. Piirilevyrevisio oli vaihtunut ja aiheuttanut muutoksia komponenttien sijoittelussa. Osa signaaleista oli myös merkitty virheellisesti tai jätetty merkitsemättä.

7.4 DSP:n mittaukset

Mittasin radion audioasteen suodatuksen toteuttavan DSP:n taajuusvasteen. Mittaus toteutettu siten, että lähetyksen kaistanleveydeksi valittu laajin eli 2,4 kHz ja equalisaattori pois päältä.



Kuva 4: DSP:n taajuusvaste lähetyksellä

Taajuusvastekuvaajasta (Kuva 4) voidaan havaita, että TADIL-A/Link 11 standardin vaatimat ±1,5 dB:n pisteet sijoittuvat noin 200 Hz ja 2800 Hz:n kohdille. Käyttöohjekirjan mukaan lähetyksen kaistanleveys on 2400 Hz ja taajuusalue 300-2700 Hz. Kuvan 4 mukaisen suotimen lisäksi käyttöohjekirjan arvossa on mukana välitaajuusasteessa oleva 2400 Hz (-6dB) kidesuodin.

Estokaistan vaimennuksesta voidaan päätellä, että suotimen suunnittelussa on mahdollisesti käytetty Hamming tai Hanning-ikkunaa. Hanning-ikkunalla toteutetun suotimen estokaistan vaimennus on 42 dB, kun Hamming-ikkunalla se on 53dB (Huttunen 2005, 84).

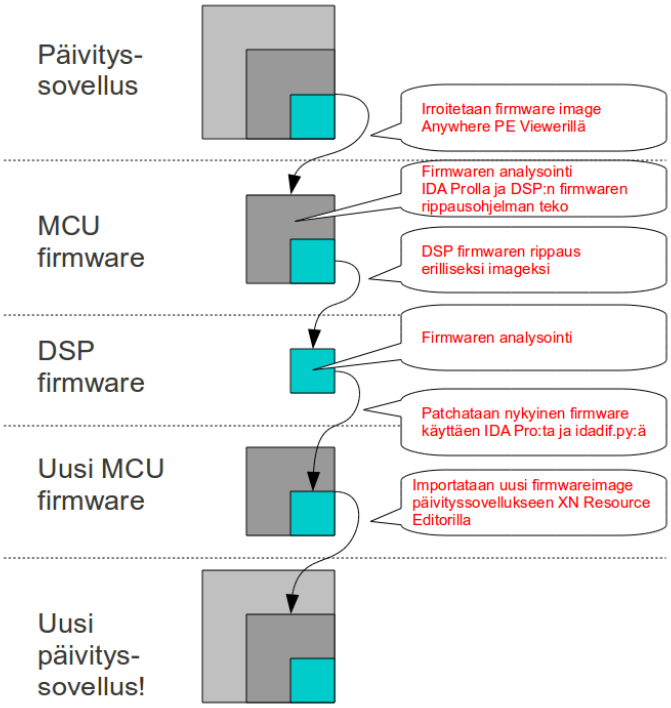
7.5 Ohjelmiston takaisinmallinnus

Aloitin firmwaren takaisinmallinnuksen tutkimalla mahdollisuuksia hankkia firmwaren lähde- tai binäärikoodi. Lähdekoodia ei ollut mahdollista saada. Mikrokontrollerin firmwaren binäärikoodin olisi voinut hankkia kaappaamalla sen sarjaväylältä firmwaren päivityksen yhteydessä. DSP:n binäärikoodin olisi voinut kaapata esimerkiksi HPI-väylältä logiikka-analysaattorin avulla. Havaitsin yksinkertaisimmaksi tavaksi mikrokontrollerin binäärin ottamisen uusimmasta saatavissa olevasta Kenwood TS-480:n päivitysohjelmasta.

Päivitysohjelmaa pystyi myöhemmin käyttämään myös uuden, muokatun firmwaren siirtämiseksi radioon. Käytin mikrokontrollerin firmwaren saamiseksi Anywhere PE Vieweriä, jolla PE-tiedostossa olevat resurssit voidaan tallentaa helposti erillisiin tiedostoihin.

Olettamuksena oli, että kyseinen Kenwood TS-480:n päivitysohjelma sisältää mikrokontrollerin ja DSP:n firmwaren. Olettamuksen syynä oli, että huolto-ohjeen(Kenwood 2003) mukaan DSP:n firmware ei säily DSP:n muistissa vaan mikrokontrolleri siirtää DSP:n firmwaren joka käynnistyksen yhteydessä DSP:lle. Siksi DSP:n firmwaren täytyy sijaita mikrokontrollerin firmwaren sisällä. Olettamus osoittautui myöhemmin oikeaksi.

Työn kulku meni kuvan 5 mukaisesti. Aluksi saatavilla oli vain Kenwood TS-480 -päivitysohjelma. Havaitsin päivitysohjelman sisältävän Renesas H8S/2338-mikrokontrollerin ROM-muistin kokoisen BINARY-nimisen resurssin, kun analysoin sitä Anywhere PE Viewerillä. Anywhere PE Viewerillä voidaan tallentaa PE-tiedoston sisältämiä resursseja tiedostoon, joten tallensin BINARY-resurssin omaksi tiedostokseen myöhempää analysointia varten.



Kuva 5: Työn kulku

7.5.1 Renesas H8S/2338 -mikrokontrolleri

Renesas H8S/2338 -mikrokontrolleri sisältää H8S/2000 CPU:n 32-bittisellä arkkitehtuurilla. H8S/2000 CPU:ssa on kuusitoista 16-bittistä yleiskäyttöistä rekisteriä, joita voidaan käyttää myös 8-bittisinä tai niistä saadaan kahdeksan 32-bittistä rekisteriä. Käskykanta tukee 8/16/32-bittisiä aritmeettisia ja loogisia käskyjä.

IDA Pro -dissassemblerissa on tuki Renesas H8S/2000 CPU:n binääreille, joten myös H8S/2338 mikrokontrollerin ohjelmat ovat dissassembloitavissa. IDA:sta valittava kohdeprosessori on silloin Hitachi H8S.

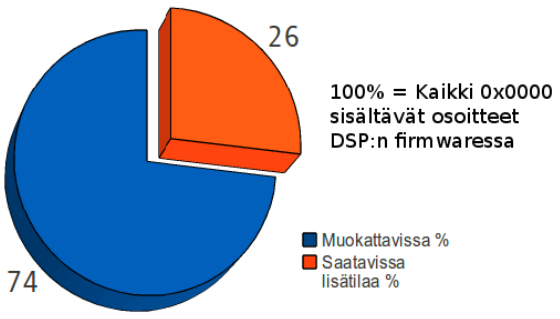
7.5.2 Mikrokontrollerin ohjelmiston takaisinmallinnus

Renesas H8S/2338 -mikrokontrollerin firmwaren tutkimisen tavoitteena oli löytää algoritmi, joka suorittaa DSP:n firmwaren siirtämisen DSP:lle. Koska tutkittava binääri ei ollut IDA Pro -dissassemblerin tunnistama tiedostomuoto, IDA Prolle piti kertoa ensin mistä osoitteesta ohjelmakoodi alkaa. Alkuosoite löytyi helposti Bless-heksaeditorilla. Tutkittavan MCU:n alkuosoite oli kerrottu firmwaren ensimmäisessä 32 bitissä, joka on datalehden (Renesas 2007, 89) mukaan reset-vektori.

Firmware siirretään DSP:n DARAM-alueelle 8-bittistä HPI-väylää pitkin. MCU hyödyntää samaa HPI-väylää rajapintana DSP:n ohjaamiseen firmwaren siirron jälkeen.

Löysin firmwaren siirtoon käytetyn algoritmin tutkimalla HPI-kirjoitusfunktiota käyttäviä aliohjelmia. Sen jälkeen tein C++:lla ohjelman, jolla DSP:n firmware voitiin tallentaa tiedostoon myöhempää analysointia varten. Kirjoittamani ohjelman lähdekoodi löytyy liittestä A.

DSP:n firmware oli pakattu erittäin yksinkertaisella tavalla, mutta se väheni noin 8% sille varatun muistin tarvetta MCU:ssa. MCU:n firmware sisälsi pääosin vain niitä osia DSP:n firmwaresta, jotka sisältävät ohjelmakoodia tai muuta tärkeää dataa. Ideana on luultavasti ollut, että tyhjiä alueita ei laitettaisi MCU:n muistia täyttämään.



Kuva 6: Tyhjän tilan sijoittuminen MCU:n firmwareen.

Kuvasta 6 voidaan havaita, että edellä mainittu idea ei toimi kunnolla. Sininen alue tarkoittaa suoraan muokattavissa olevia 0x0000:n sisältävien osoitteiden osuutta DSP:n pakatussa firmwaressa verrattuna siirretyssä firmwalessa olevaan lisätilaan. Lisätila on saatavissa käyttöön siirtoalgoritmin muutoksella. Vain noin neljännes tyhjästä tilasta on onnistuttu pitämään MCU:n muistin ulkopuolella. Radion käynnistys olisi nopeampi, jos toiminnan kannalta epäolennaista dataa siirrettäisiin vähemmän.

Tärkeät alueet oli jaettu lohkoihin, joiden ensimmäisessä osoitteessa oli lohkon koko. Jokaisen lohkon ensimmäisen datan osoite MCU:lla (lähdeosoite) ja sen kohdeosoite DSP:llä oli kovakoodattu firmwarensiirtoalgoritmin sisään. Siirto tapahtui silmukassa, jossa kasvatettiin lähde- ja kohdeosoitteita kunnes lohkon koko lohko oli siirretty. Sen jälkeen ohjelma siirtyi seuraavaan lohkoon tarkistaen lohkon koon ennen siirron aloittamista DSP:lle.

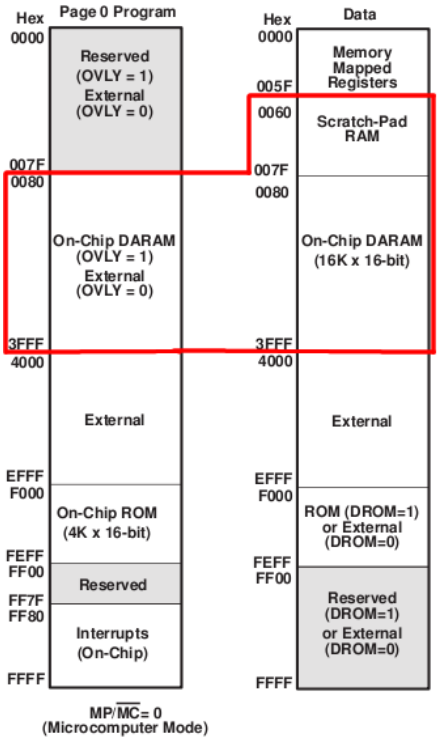
Koodin pakkaus aiheutti sen, että minun piti huomioida, mihin osoitteisiin kirjoitin uusia käskyjä DSP:n firmwaressa. Kaikkia osoiteita ei ollut mahdollista käyttää ilman muutosta DSP:n firmwarensiirtoalgoritmiin.

7.5.3 TMS320VC5402:n muistikartta ja arkkitehtuuri

Ohjelmoimani DSP:n firmwarensiirto-ohjelma teki binääri-imagen DSP:n datamuistista. Offset binääritiedoston alusta kuhunkin dataan oli sama kuin olisi oikeasti DSP:n datamuistissakin osoitteesta 0x0000.

Texas Instruments TMS320VC5402 perustuu modifioituun Harward-arkkitehtuuriin, jossa on yksi ohjelmaväylä ja kolme dataväylää. Kolmen dataväylän olemassaolo mahdollistaa kahden luku- ja yhden kirjoitusoperaation suorittamisen yhden kellojakson aikana. (Texas Instruments, 2005, 3)

Kuva 7:stä näkee muistialueen (merkitty punaisella), jota myös ulkoinen laite voi käyttää HPI-väylän kautta. Normaalisti Harward-arkkitehtuuria käyttävät prosessoreissa on täysin erillinen ohjelma- ja datamuisti.

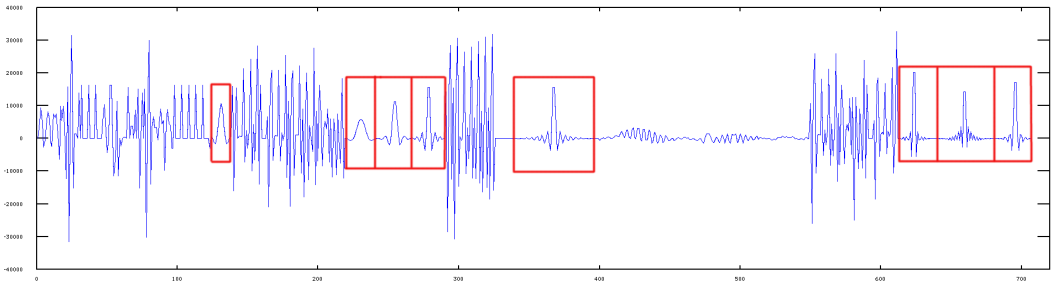


Kuva 7: TMS320VC5402 DSP:n muistikartta microcomputer-tilassa (Texas Instruments, 2005, 13)

Huomionarvoista kuvassa 7 on se, että DARAM-alueeseen päästään käsiksi ohjelma- ja datamuistin kautta. Koska tässä sovelluksessa ohjelmakoodia ajetaan DARAM-alueelta, tämä modifioitu Harward-arkkitehtuuri mahdollistaa esimerkiksi ohjelman ajonaikaisen muokkaamisen.

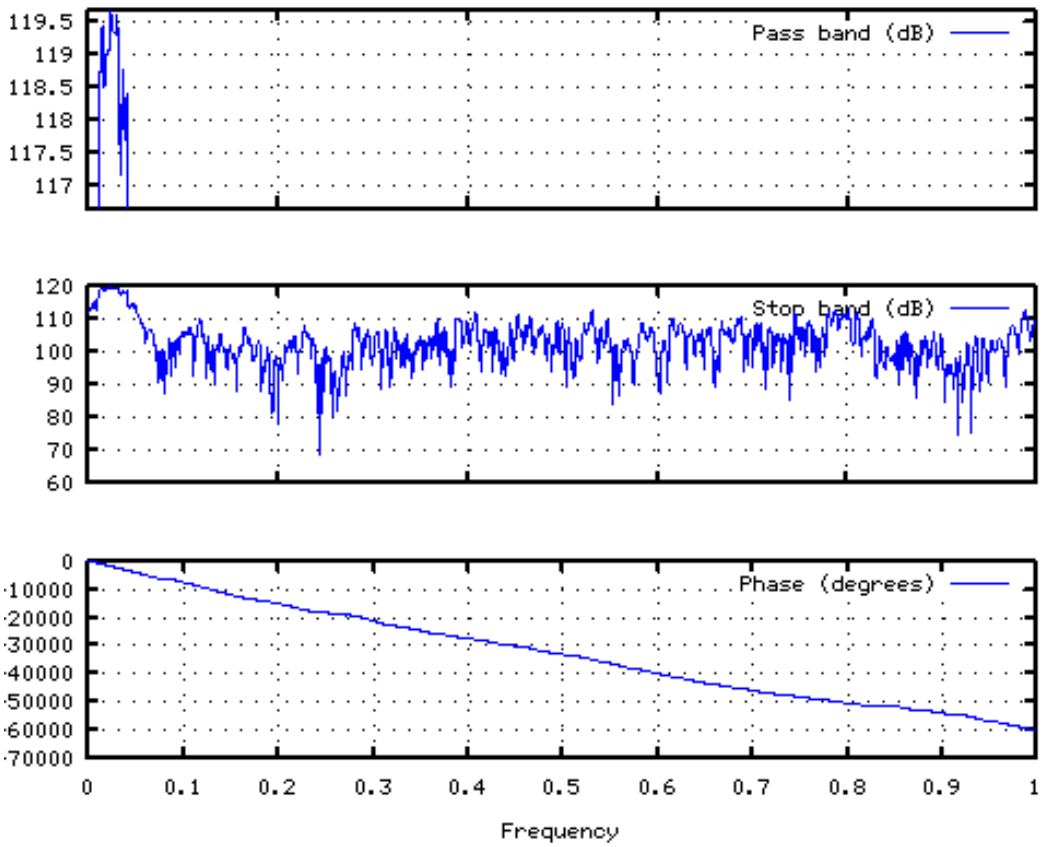
7.5.4 TX-suotimen kertoimien etsiminen

Irroitettuani DSP:n firmwaren mikrokontrollerin firmwaresta, sain idean miten ohittaa monimutkaisen dissassemblointivaiheen. Oli mahdollista alkaa etsimään suotimen kertoimia graafisesti. Mahdollisia FIR-suodinten kertoimia alkoi löytymään, kun ohjelmoin Bash-skriptausta, C++:aa ja Octaven laskenta- ja kuvaajienpiirtämisfunktioita hyödyntävän työkalun. Työkalun lähdekoodit löytyvät liitteistä B.1, B.2 ja B.3.



Kuva 8: Esimerkki graafisen työkalun tuottamasta käyrästä

Työkalulla voidaan etsiä FIR-suotimien kertoimia graafisesti. Olen merkinnyt punaisella mahdollisia kerroinalueita (Kuva 8). Ne tunnistaa usein meksikolaishattu-tyylisestä käyrämuodosta. Työkalu näyttää myös valitun impulssivasteen perusteella Nyquistin taajuuteen normalisoidun taajuus- ja vaihevasteen, kuten kuva 9 osoittaa.



Kuva 9: Kuvan 8 kuvaajan taajuus- ja vaihevaste graafisella työkalulla kuvattuna

Graafinen analysointi osoittautui todella tehokkaaksi tavaksi etsiä impulssivasteita ja ohjelmakoodialueita. Ohjelmakoodi näyttää graafisesti tarkasteltuna yleensä satunnaiselta kohinalta verrattuna impulssivasteen kaareviin ja

symmetrisiin muotoihin. Ongelmana oli selvittää mitkä kertoimet kuuluivat lähetykselle ja mitkä vastaanotolle.

Kirjoitin ohjelman, jolla pystyin nollaamaan halutut kertoimet DSP:n firmwaressa MCU:n binääritiedostosta selvittääkseni mitkä kertoimet kuuluvat lähetykselle. Onnistuin kokeilemaan kertoimien nollamisen vaikutusta korvaamalla XN Resource Editorin avulla päivitysohjelmassa olevan vanhan firmwaren muokatulla versiolla ja siirtämällä muokatun firmwaren radioon.

Nollaamalla kerralla kaikki löytämäni kertoimet sain selville, että löytämäni suotimien kertoimet vaikuttivat vain vastaanoton suodatukseen. Tästä syystä minun piti alkaa analysoimaan DSP:n firmwarea tarkemmin. Mahdollisesti TX-suotimen kertoimet olisivat olleet mikrokontrolleriin kytketyssä EEPROM:issa, mutta minulla ei ollut lukijaa sitä varten, niin se mahdollisuus jäi tutkimatta.

Jos lähetyksen kertoimet olisivat olleet helposti saatavilla, yksikköimpulssin tasaiseen taajuus- ja vaihevasteeseen perustuvilla uusilla kertoimilla olisin saanut ohitettua lähetyksen suodatuksen. Nyt jouduin turvautumaan vaihtoehtoiseen keinoon, josta lisää seuraavissa luvuissa.

7.5.5 TMS320VC5402:n dissassemblointi ja analysointi

DSP:n ohjelmakoodin alkuosoite oli tiedossa, koska mikrokontrollerin suorittamassa firmwarensiirtofunktiossa viimeisenä muistiin kirjoitettiin DSP:n ohjelmakoodin alkuosoite. DSP:n boot mode -pinnit sekä INT2 oli kytketty siten, että TMS320VC5402 käynnistyy HPI-boot-tilaan. Osoitteen kirjoittaminen osoitteeseen 0x007F aiheuttaa Taterin (2002, 5) mukaan silloin hyppykäskyn DSP:n bootloaderista sinne kirjoitettuun osoitteeseen.

Ohjelmakoodin alussa kerrottiin esimerkiksi mille sivulle (engl. page) DSP:n keskeytysvektorit on kartoitettu (engl. mapped). Alussa suoritetaan myös muut DSP:n alustukset, kuten pinomuistin sijainti sekä oheislaitteiden alustukset. Keskeytysvektoritaulusta löytyi haarautumiskäskyt keskeytyspalveluihin ja sitä kautta myös keskeytyspalveluiden sijainnit osoiteavaruudessa.

Koodia tarkastellessa selvisi muun muassa, että kodekin aiheuttama keskeytys liipaisee DMA-siirrot, jotka siirtävät näytteet kodekilta DSP:lle ja DSP:ltä kodekille. Kyseinen kodekki on kaksikanavainen 16-bittinen AD- ja DA-muunnin, joka käyttää yhtä sarjamuotoista McBSP-väylää tiedon siirtoon. Toinen McBSP-väylä on konfiguroitu 12kHz:n kellosignaaliksi ja se on kytketty kodekin näytteenottotaajuuden määrittelevään MCLK-kellotuloon.

7.5.6 TMS320VC5402:n firmwaren modifiointi

Tein modifikaation DMA-siirron keskeytyspalveluun. Sen ansiosta kodekilta tuleva TX-näyte siirretään suoraan kodekille menevän datan lähetyspuskuriin. Lähetettävälle signaalille ei siis suoriteta suodatusta.

Minulla ei ollut TMS320VC5402:lle assembleria saatavilla, joten modifikaation toteuttaminen tapahtui IDA Pron patch-toiminnon avulla. Muutenkin koin yksinkertaisemmaksi tehdä modifikaation suoraan olemassa olevaan binääriin, kuin alkaa generoida uutta firmwarea sopivaan muotoon.

Muokkasin IDA:n heksaeditorilla tarvittavat bitit ja tarkistin, että koodi näyttää dissassembloituna oikealta. Kun tulos näytti hyvältä, kopioin muutetun alueen DSP:n change byte -näkymässä ja liitin MCU:n change byte -näkymään valittuani ensin oikean offsetin.

Toinen vaihtoehto modifikaation suorittamiseksi olisi voinut olla määrittää DMA-siirrot siten, että ne siirtävät sarjaväylän RX-rekisteristä datan suoraan TX-puskuriin, jos kodekin kytkentä olisi ollut looginen. Tässä tapauksessa se ei olisi kuitenkaan toiminut, koska vastaanoton ja lähetyksen datat vaihtavat järjestystä siten, että lähetettävä signaali tulee kodekin vasemman puolen tuloon (AINL) ja lähtee oikean puolen lähdöstä (AOUTR).

7.5.7 Renesas H8S/2338 firmwaren modifiointi

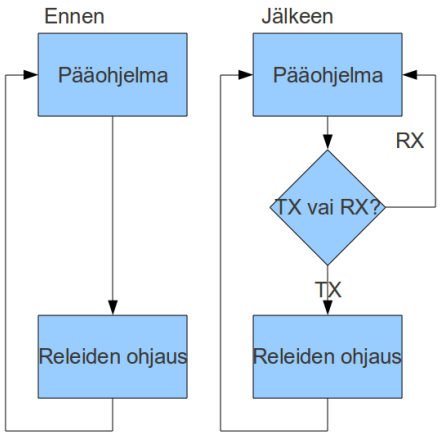
Tein myös toisen modifikaation, jossa muokattiin Renesas H8S/2338 -mikrokontrollerilla suoritettavaa koodia. Kyseinen muokkaus liittyi lähetyssignaalin

alipäästösuotimien valinnassa käytettävien releiden käyttöön. Releitä ohjataan alkuperäisessä ohjelmistossa silloin, kun radiolla vastaanotetaan.

Normaalikäytössä radiolla vastaanotetaan paljon enemmän kuin lähetetään. Sillä skannataan eri taajuusalueita ja aina taajuusalueen vaihdon yhteydessä myös lähetyspuolen releitä käytetään täysin turhaan. Koska releet ovat mekaanisia komponentteja, niille luvataan tietty määrä avaus/sulku-kertoja. Turhat avaamiset ja sulkemiset lyhentävät laitteiden käyttöikää. Kyseessä on siis käytännössä suunnitteluvirhe.

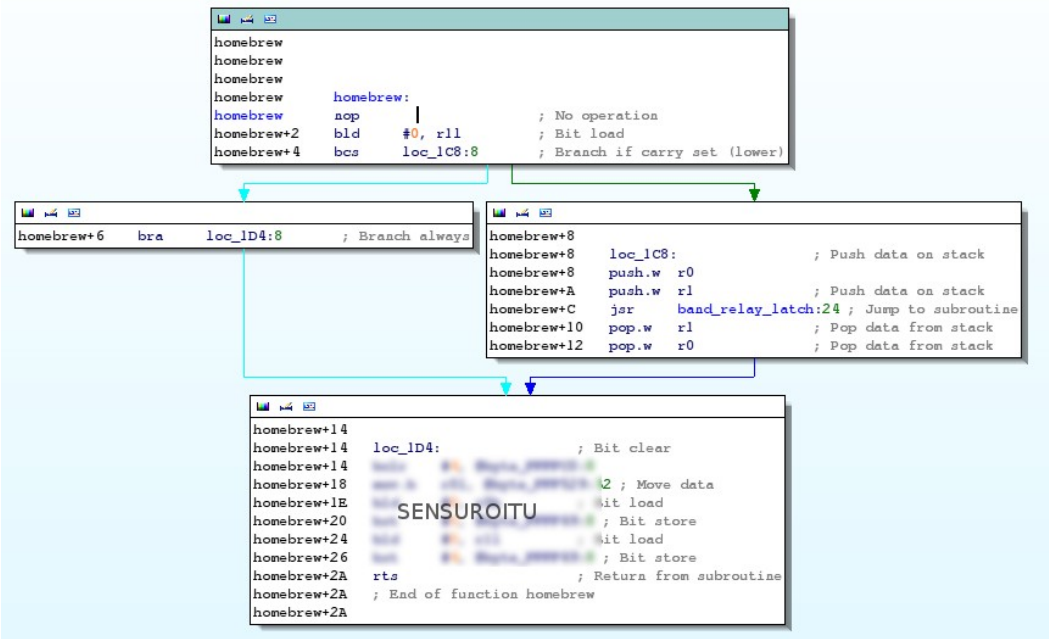
Jouduin lisäämään koodiin ehdon, jossa tarkastellaan lähetyksen aiheuttavaa bittilippua. Koodin lisäämisessä oli ongelmana se, että vertailua ja ehtoa ei pystynyt lisäämään suoraan sinne, missä releitä ohjaavan piirin ohjauksesta huolehtivaa aliohjelmaa kutsutaan.

En voinut lisätä omaa lähetyslipun tarkastelun suorittavaa koodia siten, että olisin siirtänyt myöhempiä käskyjä edemmäksi, koska silloin olisin joutunut muokkaamaan myös kaikkia kyseisen muutoksen jälkeiseen ohjelmakoodiin viittaavia osoitteita. Ongelmaa ei olisi ollut, jos koodiin tehtävään muutokseen ei olisi tarvittu enempää tilaa kuin alkuperäiset käskyt tarvitsivat.



Kuva 10: Vuokaavio ohjelman muokkauksesta lähetykseen liittyvien releiden ohjauksessa

Ensin etsin käytettävissä olevan vapaan alueen muistista. Kirjoitin sinne vertailun suorittavan aliohjelman (kuva 10 ja kuva 11). Vertailussa tarkastellaan, onko lähetys vai vastaanotto käytössä. Vastaanoton tapauksessa releitä ei ohja-



Kuva 11: Omaa koodia IDA Prolla tarkasteltuna

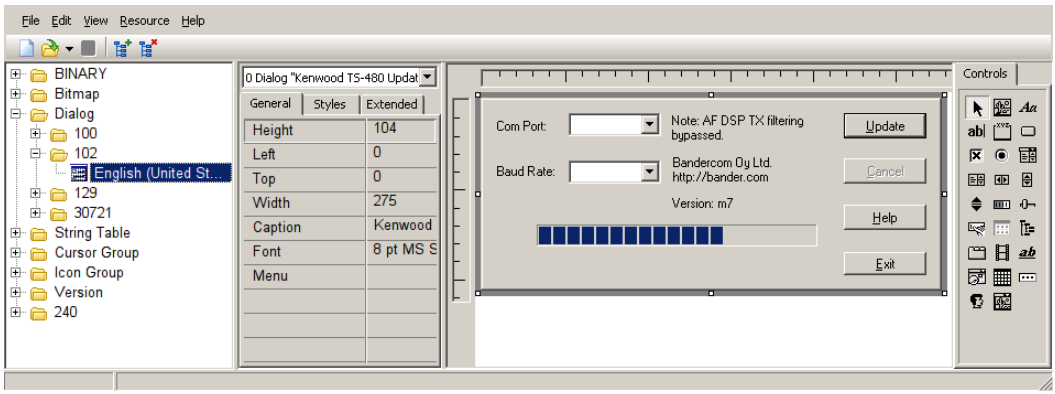
ta. Lähetyksen tapauksessa mennään ehdollisen haarautumisen myötä suorittamaan releiden ohjaukseen liittyvät toiminnot.

Seuraavaksi vaihdoin releille kirjoittavan aliohjelman kutsukäskyn osoitteen oman aliohjelmani alkuosoitteeksi, jolloin hypätäänkin tekemääni aliohjelmaan. Aliohjelmakutsulle piti tehdä tilaa siirtämällä osa alkuperäisistä käskyistä uuteen tekemääni aliohjelmaan. Siirron aiheuttaman aukon täytin nop-käskyillä.

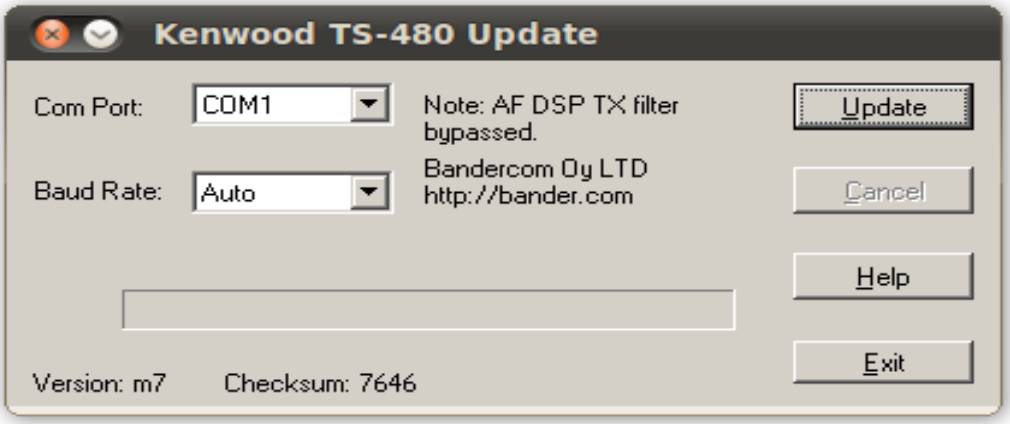
7.5.8 Uuden firmwaren luominen mikrokontrollerille

Tehtyäni muutokset MCU:n firmwaren IDA-tietokantaan, tein IDA:lla DIF-tiedoston, joka sisältää tekemäni muutokset. Latasin Internetistä idadif.py-nimisen Python-ohjelman, jolla pystyi suorittamaan muutokset MCU:n firmwaren binäärikoodiin DIF-tiedostossa olevien tietojen mukaan.

Nyt piti enää siirtää uuden firmwaren binääri vanhan tilalle. Siirto onnistui helposti XN Resource Editorilla (Kuva 12). Päivitysohjelma ei tarkastanut firmwarea muutoksen varalta ennen siirtoa radioon.



Kuva 12: XN Resource Editor



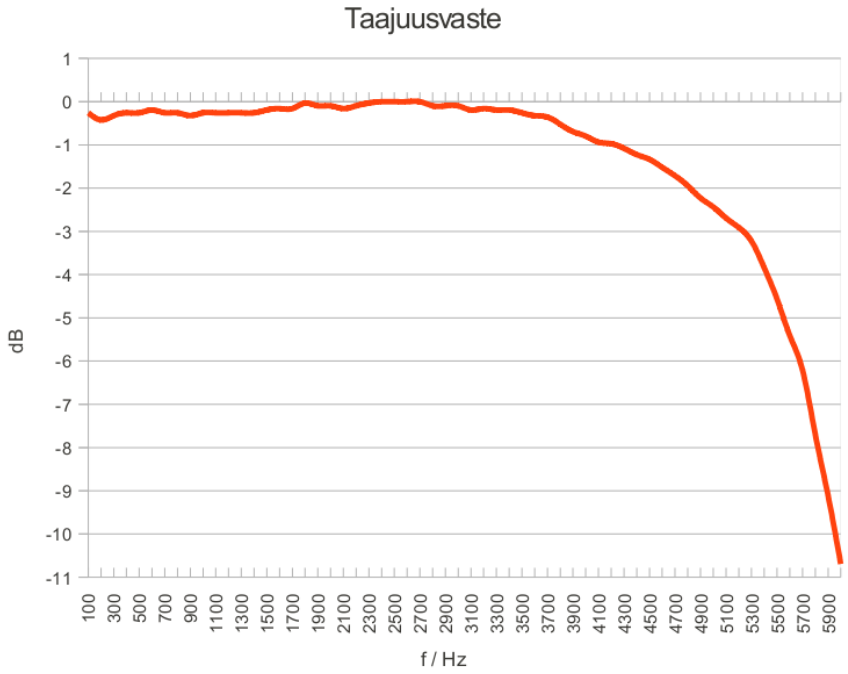
Kuva 13: Uusi Kenwood TS-480:n päivitysohjelma

7.6 Testaus

Päivitin Kenwood TS-480SAT:iin uuden firmwaren muokatulla päivitysohjelmalla (Kuva 13). Firmware siirrettiin kytkemällä suora sarjakaapeli tietokoneen ja radion välille. Pävittäminen sujui ongelmitta.

Mittasin modifioidulla firmwarella päivitetyn laitteen taajuusvasteen. Mitattu -3dB:n raja on 5200Hz, kuten kuva 14:sta voidaan havaita. Aldrichin (2003, 98) mukainen 300-3500 Hz:n ±1,5dB:n kaistan vaatimus ylitetään selvästi.

Testausvaiheessa havaittiin ongelma vastaanotossa. Audio tuli vastaanotolla normaalisti radion kaiuttimesta, mutta erillinen audioliitäntä, johon datamodeemi piti kytkeä, oli mykkä. En ollut huomannut, että vastaanotossa DSP laittaa kodekin molempiin lähtöihin signaalia, kun lähetyksellä laitetaan vain toiseen. Vika oli korjattavissa uudella koodimuutoksella.



Kuva 14: Lähetyksen taajuusvaste modifioidulla firmwarella

8 POHDINTAA FIRMWAREN TAKAISINMALLINNUKSEN ESTOTEKNIIKOISTA

Seuraavaksi pohdin firmwaren takaisinmallinnuksen estotekniikkoja. Voidaan havaita, että ei ole täysin mahdollista estää firmwaren takaisinmallinnusta, mutta sitä on helppoa hidastaa yksinkertaisilla keinoilla.

Osa estotekniikoista perustuu siihen, että hyödynnetään dissassembleriohjelman ominaisuuksia tietyissä erityistilanteissa. Toinen osa perustuu siihen, että takaisinmallinnus vaatii aina paljon vuorovaikutusta. Koodin seurattavuutta voidaan heikentää helposti monella keinolla, joista lisää myöhemmin. Kolmas keino on lisätä jokin tekninen suojauskeino, kuten pakkaminen tai salaus.

Siiretyn koodin lukemista tulisi myös vaikeuttaa. Väylillä liikkuvan datan tulisi olla salattua. Erityisesti silloin, kun väylillä siirretään ohjelmakoodia.

Seuraavissa luvuissa otan esimerkkejä erityisesti tutkimassani radiossa olleista Texas Instrumentsin TMS320VC5402 -digitaalisesta signaaliprosessorista ja Renesas H8S/2338 -mikrokontrollerista ja kuinka analysoimista voidaan vaikeuttaa erityisesti IDA Pro:n tapauksessa. Osan keinoista olen löytänyt itse tämän projektin aikana, loput aiheeseen liittyvästä kirjallisuudesta.

8.1 Epäolennaisuuksien lisääminen

Ohjelmakoodin tutkimista hankaloittamaan voidaan lisätä epäolennaisia, ohjelmiston varsinaiseen toimintaan liittymättömiä käskyjä. Takaisinmallintajan täytyy tutkia epäolennaistenkin käskyjen tekemät toiminnot, ennen kuin ne voidaan todeta turhiksi. Tämä hidastaa tutkimista ja kohdistaa takasinmallintajan huomion pois olennaisesta.

Muistipaikojen sisältöä voidaan vaihdella keskenään. Koodin toiminnan ymmärtäminen vaikeutuu, jos samoja muuttujia käytetään koodin eri kohdissa eri tarkoituksiin, koska takaisinmallinnuksessa kullekin muuttujalle pyritään määärittämään tarkoitus.

8.2 Hypyt ja haarautumiset

Ehdollisen haarautuminen (conditional branch) voidaan laittaa osoittamaan dataosoitteeseen. Haarautumisen ehdon pitäisi olla sellainen, että data-alueelle ei kuitenkaan koskaan jouduttaisi. Edellä mainittua tehostaisi vielä se, että data olisi ohjelmakoodin seassa mahdollisimman pieninä segmentteinä. IDA Pro tulkitsisi datan koodiksi ja yrittäisi dissassembloida sitä.

Koodin lukemista voidaan vaikeuttaa myös lisäämällä ehdottomia haarautumisia siten, että haaraudutaan aina parin koodirivin jälkeen jonnekin muualle. Koodin seuraaminen on tulee todella hitaaksi. IDA Pron graafinen näkymä auttaa kyllä jäsentämään koodin helpommin, mutta pelkän koodilistauksen lukeminen hidastuu todella paljon.

Texas Instrumentsin C54x-sarjan käskykannan Branch to accumulator (bacc) -käskyssä, akkumulaattorin arvo määrää haarautumisosoitteen. IDA Pro ei osaa laskea automaattisesti mahdollisia haarautumisosoitteita eikä jatkaa dissassembloimista oikeissa osoitteissa. Takaisinmallintajan on siis etsittävä mahdolliset haarutumisosoitteet koodia tutkimalla.

8.3 Muuttuva koodi

Kenwood TS-480SAT:n DSP:n firmwarea suoritetaan DARAM-muistialueelta. DARAM-alue on käytettävissä sekä ohjelma että datamuistina. Koodialueiden keskelle oli jätetty monin paikoin nop-käskyjä (no operation). Koodia tutkittaessa on havaittavissa, että joidenkin nop-käskyjen päälle kirjoitetaan haarautumis- tai hyppykäskyjä.

IDA Pro:n automaattinen analysointi, joka dissassembloi rekursiivisesti, ei löydä näitä aliohjelmia, joihin viittaavat hyppykäskyt tai haarautumiset kirjoitetaan koodin suorittamisen aikana.

8.4 Koodin pakkaus ja salaus

Merkittävin vaikutus pakkauksella on tiedoston muokkaamisen kannalta, koska muokkauksen jälkeen koodi on kyettävä pakkaamaan. Suoritettavan koodin alussa on vain purkualgoritmi. (Hirvonen 2008, 8.)

Koodin salaus tuo samoja etuja kuin pakkauskin, mutta salaus on vaikeampi purkaa ja muokatun koodin uudelleen salaaminen on vaikeampaa. Käytännössä kuitenkin ohjelman täytyy sisältää purkukoodin ja purettuna koodin on oltava suoritettavissa prosessorilla.

9 YHTEENVETO

Aloitin opinnäytetyön tekemisen kolmannen lukuvuoden jälkeen kesällä. Sihen mennessä ei oltu pidetty vielä juuri lainkaan signaalinkäsittelykursseja, joten jouduin opiskelemaan lähes kaiken itse. Ainoastaan yksi signaaliprosessorikurssi oli käyty, josta sain pohjan signaaliprosessorikytkentöihin.

Hyvien lähteiden ansiosta onnistuin luomaan itselleni käsityksen signaalinkäsittelystä sellaisella tasolla, että pystyn itsenäisesti suunnittelemaan ja toteuttamaan digitaalisia signaalinprosessointijärjestelmiä.

Sulautettujen järjestelmien ohjelmistoon ja rautaan kohdistuvaan takaisinmallinnukseen ja modifiointiin kertyi runsaasti kokemusta. Tein projektin itsenäisesti alusta loppuun lukuun ottamatta alussa speksien määrittämistä, jolloin Bandercomin Petri Peltola antoi työnkuvauksen. Lopputestauksessa auttoi Bandercomin Mika Niemelä.

9.1 Teoreettinen osio

Lähteiden oikeellisuuden arvioinnissa käytin pääasiallisesti mittarina sitä, kuinka paljon muut tutkijat käyttävät kirjoittajan aineistoa referenssinä omissa teksteissään. Käytin lähteiden oikeellisuuden arviointiin myös käyttämällä useita lähteitä samasta aiheesta.

En löytänyt aluksi kuin yhden teoksen, joka käsitteli laitteiston takaisinmallinusta, Hacking Xbox: An Introduction to Reverse Engineering (Huang 2003b). Olihan niitä lähteitä lopulta muitakin, mutta kirjallisuutta on saatavilla niukasti. Suurin osa ohjelmiston takaisinmallinnusta käsittelevistä lähteistä käsitteli PC-arkkitehtuurilla toteutettuja ohjelmia.

Bloomin tekstissä oli joitakin asiavirheitä, jotka havaitsin vertaamalla hänen aineistoaan muihin teksteihin. Esimerkiksi puhuttaessa noise shapingista Bloomin (2009, 15.9) mukaan kvantisointikohina jakaantuu tapahtuu 0 Hz:stä

näytteenottotaajuuteen asti. Mielestäni hänen tekstinsä oli hyvä nopeaan pintapuoliseen perehtymiseen, mutta syvällisempään tutkimiseen tarvitaan muita lähteitä.

Kenwood TS-480 huolto-ohje sisälsi myöskin lukuisia virheitä. Virheet johtuvat mahdollisesti siitä, että huolto-ohje oli aika vanha, ja muutoksia oli tullut laitteen jatkuvan kehityksen seurauksena. Luulen kuitenkin, että osa virheistä oli tehty tahallisesti.

9.2 Käytännön toteutus

Pääsin kehittämään takaisinmallinnustekniikkaani ohjelmistopuolella, kun onnistuin analysoimaan Kenwood TS-480 radionpuhelimen ohjelmistopäivitysohjelman. Sain erotettua radion mikrokontrollerille ja DSP:lle tarkoitetut binäärikoodit päivitysohjelmasta ja dissassembloitua ne. Dissassembloinnin jälkeen koodeja oli suhteellisen helppoa analysoida, vaikka se oli melko hidasta. Debuggerista olisi ollut paljon hyötyä.

Valitettavasti Suomen laki estää takaisinmallinnuksen avulla saatujen tulosten julkaisemisen. Olisin mielelläni kertonut yksityiskohtaisemmin ja näyttänyt koodiesimerkkejä DSP:n firmwaresta. Takaisinmallinnuksen tutkiminen ja toteuttaminen laajassa mittakaavassa antoi ymmärrystä, kuinka omassa tuotekehityksessä voisi hyödyntää takaisinmallinnusta sekä miten itse voisi suojautua takaisinmallinnukselta.

Itselläni ei ollut kokemusta Matlabista, mutta havaitsin Octaven käyttäminen helpoksi keinoksi suorittaa signaalinprosessointiin liittyviä laskutoimituksia. Löysin internetistä joitakin ohjeita, joiden mukaan tekemällä pääsin alkuun Octaven käytössä.

9.3 Oppimistavoitteiden täyttyminen

Opinnäytetyöprosessin alkuvaiheessa minnulla ei ollut juurikaan kokemusta takaisinmallinnuksesta eikä lainkaan kokemusta sulautettujen järjestelmien

ohjelmistojen modifioinnista. Kaikki oppi piti itse etsiä ja löytää. Kaiken kaikkiaan prosessi antoi erinomaiset lähtökohdat sulautettujen järjestelmien takaisinmallinnukseen, suunnitteluun ja modifiointiin tulevaisuudessa.

Opiskelin itsenäisesti myös digitaalisia suodattimia siten, että olisin pystynyt toteuttamaan suodatuksen rautamodifikaation avulla esimerkiksi DSP:tä tai FPGA:ta käyttäen. Octaven käytön opettelu avarsi myös käsitystä kunnollisen laskentatyökalun tarpeellisuudesta signaaliprosessorien kehityksessä.

IDA Pro:n käyttöä opin siinä määrin, että tavallisten toimintojen lisäksi myös patchien tekeminen onnistuu.

9.4 Kehitysideoita

DSP:n firmwaressa suotimen ohittamisen olisi voinut toteuttaa hienostuneemminkin. Suodatuksen lisäksi joitain muitakin lähetettävälle signaalille sovellettavia ominaisuuksia lähti pois.

Yksi lähtenyt ominaisuus on audiotaajuuksilla tapahtuva lähetyssignaalin voimakkuuden säätö. Koska radio on vain datakäytössä, lähetyksen audioasteen voimakkuutta ei tarvitse säätää ensimmäisen kalibroinnin jälkeen. Voimakkuutta voidaan kuitenkin säätää laitteeseen kytkettävästä datamodeemista.

LÄHTEET

Aldrich, S., 2003. MILITARY STANDARD-188-203-1A TADIL A and standadization agreement 5511 Annex B Link 11 conformance test procedures. Saatavissa: http://jitc.fhu.disa.mil/jtrs/procedures/link11.pdf.

Bandercom, 2011. Bandercom. [viitattu 29.06.2011]. Saatavissa: http://www.bandercom.fi.

Bless, 2011. Bless Hex Editor. [viitattu 29.06.2011]. Saatavissa: http://home.gna.org/bless/.

Bloom, A., 2009. DSP and Software Radio Design. Teoksessa Silver, H.W. (toim) 2009, ARRL Handbook for Radio Communications. ARRL.

Eagle, C., 2011. The IDA Pro Book, The unofficial guide to the world's most popular dissassembler. No Starch Press, San Fransisco.

Eilam, E., 2005. Reversing: Secrets of Reverse Engineering. Wiley Publishing Inc, Indianapolis, Indiana.

Finder, 2012. [viitattu 10.3.2012]. Saatavissa: http://www.finder.fi/Langatonta%20tiedonsiirtoa/Bandercom%20Oy%20Ltd%20/LAHTI/taloustiedot/195781.

FM3-52. Army Airspace Command and Control in a Combat Zone. 2002, [viitattu 12.05.2011]. Saatavissa: http://permanent.access.gpo.gov/lps24911/FM%203-52%2020020801.pdf.

Grand, J., 2011. Hardware Reverse Engineering: Access, Analyze, & Defeat. Black Hat DC 2011 Workshop . Grand Idea Studio. [viitattu 22.06.2011]. Saatavissa: https://media.blackhat.com/bh-dc-11/Grand/BlackHat_DC_2011_Grand-Workshop.pdf.

Hirvonen, T., 2008. Takaisinmallinnuksen estotekniikat. Kandidaatintyö, Tietotekniikan koulutusohjelma, Ohjelmistotuotanto. Tampereen teknillinen yliopisto. [viitattu 22.06.2011]. Saatavissa: http://www.students.tut.fi/~hirvon25/Takaisinmallinnuksen_estotekniikat.pdf.

Huang, A., 2003a. Power to the People: Hardware Hacking for the Masses. [viitattu 23.06.2011]. Saatavissa: http://conferences.oreillynet.com/presentations/et2004/huang.pdf.

Huang, A., 2003b. Hacking the Xbox, An Introduction to Reverse Engineering. No Starch Press Inc, San Fransisco, CA.

Huttunen, H., 2005. Signaalinkäsittelyn menetelmät. Tampere. [viitattu 12.05.2011]. Saatavissa: www.cs.tut.fi/kurssit/SGN-1200/Moniste.pdf.

Hyytiäinen, J., 2011. Luennot mikroprosessorijärjestelmien kurssilla Lahden Ammattikorkeakoulussa. Tekniikan laitoksella 15.02.2011 ja 22.02.2011.

Johnson, A., 2011. Resource Hacker. [viitattu 29.06.2011]. Saatavissa: http://www.angusj.com/resourcehacker/.

Kenwood, 2003. HF / 50MHz ALL MODE TRANSCEIVER TS-480HX/480SAT Service Manual.

Kenwood, 2011. [viitattu 13.07.2011]. Saatavissa: http://www.kenwood.com/i/products/info/amateur/ts_480/.

Korkeimman oikeuden päätös. KKO:2008:45, [viitattu 28.06.2011]. Saatavissa: http://www.kko.fi/43219.htm.

MIL-STD-188-110B. Interoperability and performance standards for data modems. 2000, [viitattu 11.5.2011]. Saatavissa: http://hflink.com/standards/MIL_STD_188_110B.pdf.

Octave, 2011. [viitattu 4.2.2012]. Saatavissa: www.gnu.org/s/octave/.

Renesas, 2007. H8S/2339 Group Hardware Manual, Renesas 16-bit Single Chip Microcomputer H8S Family/H8S/2300 Series, Rev 4.00. [viitattu 22.08.2011]. Saatavissa: http://documentation.renesas.com/eng/products/mpumcu/rej09b0245_2339.pdf.

Tater, S., 2002. Bootloading the TMS320VC5402 in HPI Mode. SPRA382.

Tekijänoikeuslaki, 1961. Tekijänoikeuslaki 8.7.1961/404. [viitattu 23.06.2011]. Saatavissa: http://www.finlex.fi/fi/laki/ajantasa/1961/19610404.

UCware, 2011. Anywhere PE Viewer. [viitattu 29.06.2011]. Saatavissa: http://www.ucware.com/apev/index.htm.

Wine, 2011. Wine. [viitattu 29.06.2011]. Saatavissa: http://wiki.winehq.org/.

HAKEMISTO

A LIITE DSP-ripperin C++ -lähdekoodi

```
1  //TS480-DSP-ripper-1.cpp
2  /*******************************************
3  * ---------------------------------------------
4  * FILENAME        TS480-DSP-ripper-1.cpp
5  * DATE CREATED    Thu, 16 Jun 2011 08:18:27 +0300
6  * VERSION         1.0
7  * LAST MODIFIED   Fri, 08 Jul 2011 11:48:48 +0300
8  * AUTHOR          Miika Ihanajärvi (OH3FOB)
9  * EMAIL           miika.ihanajarvi [at] gmail.com
10 *---------------------------------------------
11 *
12 *
13 * Works only with the 1.04 version binaries.
14 * Can be used with other binaries as well if update startaddr table.
15 * There may be a solution for looking those values automagically from the
       binary. I have not check'd.
16 *
17 * Command-line arguments: infile outfile
18 *
19 * Example: TS480-DSP-ripper-1 MCU.bin DSP.bin
20 *
21 * Install libboost if your compiler doesn't recognize C99 integer types.
22 * Change paths to cstdint.hpp and cstdlib.hpp according your installation.
23 * I use C99 integer types to ensure that there is right amount of bits for
       every variable.
24 *
25 * Build with command:
26 * g++ TS480-DSP-ripper-1.cpp -Wall -oTS480-DSP-ripper
27 *
28 * Should build without any warnings. Works like a charm.
29 *
30 * Inform the author, please, if you have suggestions or found a bug or if you'
       re going to modify this
31 * piece of software.
32 * All rights reserved.
33 *******************************************/
34
35 #include <stdio.h>
36 #include </usr/include/boost/cstdlib.hpp> //stdlib.h can be used here as well
37 #include </usr/include/boost/cstdint.hpp> //boost library for c99 integer type
       support
38 #define DARAM 0x4000 //end of DARAM
39 uint8_t debug=0; //Debugging on(1)/off(0)
40                  //Note: Pipe output to less.. easier to read.
41
42 int main ( int argc,   // Number of strings in array argv
43   char *argv[] )       // Array of command-line argument strings
```

```
44 {
45   FILE * pInfile,
46     * pOutfile;
47   uint32_t llnfileSize;
48   uint8_t * buffer; //pointer for binary file buffer
49   uint16_t * dspBuffer; //pointer for dsp file buffer
50   uint16_t bufferUint16; //variable for making word out of two bytes
51   uint32_t i,k; //indexing
52   uint32_t startaddr[] = { 0x28F56, 0x28F98, 0x2909A, 0x29118, 0x2921A,
53     0x29E0E, 0x2BA94, 0x2C788, 0x2FC1E, 0x2FD20 }; //binary file address
         lookup table
54   uint32_t dspaddr[]= { 0x60, 0x80, 0x100, 0x180, 0x200,
55     0x800, 0x1800, 0x2000, 0x3B00, 0x3C00 }; //dsp address lookup table
56   uint32_t blockLenghts [ 10 ];
57   printf ( "TS-480␣DSP␣ripper␣version␣1.0␣\n" );
58   if ( argc == 1 ) { //no arguments
59     printf ( "Correct␣Syntax:␣TS480-DSP-ripper␣INFILE␣OUTFILE␣\n" );
60     exit ( 1 );
61   } else  if (( pInfile=fopen (argv[1], "r")) == NULL ){ //input file not
       found
62     printf ( "File␣could␣not␣be␣opened␣for␣reading.\n" );
63     fputs ( "File␣opening␣error",stderr );
64     exit ( 2 );
65   } else {
66       if (( pOutfile = fopen (argv[2], "w" )) == NULL ){ //can't create output
           file
67         printf ( "File␣could␣not␣be␣opened␣for␣writing.\n" );
68         fclose ( pInfile );
69         fputs ( "File␣error",stderr );
70         exit ( 3 );
71       }
72   }
73
74 /*********************************************
75 * Allocate memory for the input file
76 * 1. Get the file size
77 * 2. Allocate memory
78 * 3. Read the file to the allocated memory
79 * 4. Check everything is fine
80 * 5. Close the input file
81 *********************************************/
82
83   fseek ( pInfile , 0 , SEEK_END );
84   llnfileSize = ftell ( pInfile ); // get file size for malloc
85   rewind ( pInfile );
86   buffer = ( uint8_t* ) malloc ( sizeof ( uint8_t ) * llnfileSize ); //
       allocate memory
87
88   if ( buffer == NULL ) {
```

```
89      fputs ( "Memory error" , stderr ); exit ( 4 );
90    }
91
92    if ( fread ( buffer, 1, lInfileSize, pInfile ) != lInfileSize ) { //read
          binary file to the previously allocated memory
93      fputs ( "Reading error", stderr );
94      exit ( 5 ); // quick'n dirty check for corruption
95    }
96    fclose ( pInfile ); // Input file won't be needed anymore.
97
98  /*****************************************
99  * Every data block inform about their sizes on the first word.
100 * After the size information the actual data begins.
101 *****************************************/
102
103 // get block lenghts. As they are 16bits, I have to do a lil trick.
104   printf ( "Getting adresses and writing block lenghts.." );
105   if(debug) printf ( "\nAddress : DSPaddr: block lenght" );
106   for ( i=0; i<10; i++){
107     bufferUint16 = buffer [ startaddr [ i ] ];
108     bufferUint16 = bufferUint16 * 0x100; //move 8bits to tha righta
109     bufferUint16 = bufferUint16 + buffer [ startaddr [ i ] + 1 ];
110   if(debug) printf ( "\n0x%04X : 0x%04X : 0x%04X --> %d",
111                      startaddr [ i ],
112                      dspaddr [ i ],
113                      bufferUint16,
114                      bufferUint16 );
115     startaddr [ i ] = startaddr [ i ] + 2;    //increase address pointers to
          point the next word
116     blockLenghts [ i ] = bufferUint16;    //save block size to a table
117   }
118   printf ( " Done.\n" );
119
120 /*****************************************
121 * That's all folks. Let's make the DSP binary file.
122 * I'm gonna make a copy of the TI TMS3205402's memory map from the 0x0000 till
      the end of the DARAM area.
123 * DARAM ends at the 0x3FFF. This DSP has 16 bit data bus and unfortunately
      bytes of the words are in
124 * opposite order. At least in IDA Pro Advanced they looked like they were..
125 *
126 * First I will allocate a memory for whole program, so it's is easy to access
      to any address.
127 *
128 * Last but not least.. Mind to put 0x2000 to 0x7F.. The infamous start vector
      which will enable the DSP to
129 * break out from the bootloader.
130 *****************************************/
131   dspBuffer = ( uint16_t* ) malloc ( sizeof ( uint16_t ) * DARAM ); //
```

```
        allocate memory
132   printf ( "Moving␣data␣block␣to␣the␣memory␣buffer...␣" );
133   for(i=0; i<10; i++){
134     for( k=0; k < blockLenghts [ i ] ; k++ ){
135
136       // Do almost the same magic as earlier while reading the block lenghts
137       // this one will switch the bytes.
138
139       bufferUint16 = buffer [ startaddr [ i ] + (k * 2) + 1 ];
140       bufferUint16 = bufferUint16 * 0x100; //move 8bits to tha righta
141       bufferUint16 = bufferUint16 + buffer [ startaddr [ i ] + (k * 2) ];
142       dspBuffer [ dspaddr [ i ] ] = bufferUint16 ;
143       if(debug)     printf( "\naddr:␣0x%04X␣data:␣0x%04X",
144                             dspaddr[i],
145                             dspBuffer [ dspaddr [ i ] ]);
146       dspaddr[i]++;
147     }
148   }
149   dspBuffer [ 0x7F ] = 0x0020; //write the start address
150   printf ( "␣Done.␣\nWriting␣to␣file:␣%s␣...␣", argv [ 2 ] );
151 // memory image is now in the dspBuffer.  All I have to do is to write it to
      the output file.
152   fwrite ( dspBuffer,
153            sizeof(uint16_t),
154            sizeof(uint16_t)*DARAM,
155            pOutfile);
156   printf("Done.\n");
157   fclose ( pOutfile ); //closing file
158 /*********************************************
159 * Open output file for reading and check the size.
160 *********************************************/
161   printf("Check␣the␣file␣size...␣");
162   if (( pOutfile=fopen (argv[2], "r")) == NULL ){ //input file not found
163     printf ( "File␣could␣not␣be␣opened␣for␣reading.\n" );
164     fputs ( "File␣opening␣error",stderr );
165     exit ( 6 );
166   }
167   fseek ( pOutfile , 0 , SEEK_END );
168   if ( sizeof ( uint16_t ) * DARAM != ( ftell ( pOutfile ) / 2 )){
169     printf ( "Check␣error.␣Wrong␣file␣size.␣Is␣%d.␣Should␣be␣%d.\n",
170              (int)ftell ( pOutfile ),
171              (sizeof ( uint16_t )*DARAM) );
172   }
173   else {
174     printf("Done.␣Ok.\n");
175   }
176 /*********************************************
177 * Clean everything: close the file and release the memory.
178 * Shut the program.
```

```
179 ***********************************************/
180   fclose ( pOutfile );
181   free ( buffer );
182   free ( dspBuffer );
183   return 0;
184 }
```

B LIITE Coeff-finder

B.1 Bash-skripti

```
#!/bin/bash
#****************************************
#    Filename:    coff_finder.sh
#    Author:      Miika Ihanajärvi
#    E-mail:      miika.ihanajarvi()gmail.com
#    Created:     Wed, 13 Jul 2011 10:18:44 +0300
#    Edited:      Wed, 13 Jul 2011 14:30:48 +0300
#
# This script can be used for analyzing the data of the firmware of the DSP
#
# Command line arguments:
# ./coff_finder.sh ADDR LEN file
#
#   ADDR    -   an address where to start an analysis
#   LEN     - lenght of the block
#    file    -   Binary file of the firmware
#
#****************************************
MINPARAMS=3
FILE=temp_coff.txt #temporary file
if [ $# -lt "$MINPARAMS" ]
then
  echo "-----------------------------ERROR-----------------------------"
  echo "This script needs at least $MINPARAMS command-line arguments!"
    echo
    echo "./coff_finder.sh ADDR LEN file"
    echo "ADDR   -    an address where to start an analysis"
    echo "LEN        - lenght of the block"
    echo "file   -    Binary file of the firmware"
    echo "---------------------------------------------------------------"
    exit 0
fi
##Convert hex address to decimal
start_hex=`echo "$1" | sed -e 's:^0[bBxX]::' | tr '[a-f]' '[A-F]'`
dec=`echo "ibase=16; $start_hex" | bc`
let dec=dec*2
samples=$2
let samples=samples*2
./DSP-COFF-ripper $dec $samples $3 $FILE
echo "Click on or close Figure 2 window to continue..."
./coff.m $FILE
#rm $FILE
exit 0
```

B.2 C++-lähdekoodi

```
1  //DSP-COFF-ripper-1.cpp
2  /***********************************************
       ____________________________________________
3  * FILENAME        DSP-COFF-ripper-1.cpp
4  * DATE CREATED    Mon, 11 Jul 2011 15:32:29 +0300
5  * VERSION         1.1
6  * LAST MODIFIED   Wed, 13 Jul 2011 14:30:36 +0300
7  * AUTHOR          Miika Ihanajärvi (OH3FOB)
8  * EMAIL           miika.ihanajarvi [at] gmail.com
9  *--------------------------------------------------
10 * This program can be used for ripping 16bit hard coded coefficients from a
       firmware of a DSP.
11 * The number of coefficients is given..
12 *
13 * Command-line arguments: startaddress coefs infile outfile
14 *
15 *   startaddress    - where to start reading (bytes)
16 *   coefs          - how many bytes to read
17 *   infile         - input binary
18 *   outfile        - output textfile
19 *
20 * Example: DSP-COFF-ripper 0 1024 DSP.bin coefs.txt
21 *
22 * Install libboost if your compiler doesn't recognize C99 integer types.
23 * Change paths to cstdint.hpp and cstdlib.hpp according your installation.
24 * I use C99 integer types to ensure that there is right amount of bits for
       every variable.
25 *
26 * Build with command:
27 * g++ DSP-COFF-ripper-1.cpp -Wall -oDSP-COFF-ripper
28 *
29 * Should build without any warnings. Works like a charm.
30 *
31 * Inform the author, please, if you have suggestions or found a bug or if you'
       re going to modify this
32 * piece of software.
33 * All rights reserved.
34 ***********************************************/
35
36 #include <stdio.h>
37 #include </usr/include/boost/cstdlib.hpp> //stdlib.h can be used here as well
38 #include </usr/include/boost/cstdint.hpp> //boost library for c99 integer type
       support
39 uint8_t debug=0; //Debugging on(1)/off(0)
40                  //Note: Pipe output to less if used.. easier to read../
41 uint8_t negative=0; // 0xFFFF-data
42 int main ( int argc,  // Number of strings in array argv
```

```
43    char *argv[] )       // Array of command-line argument strings
44  {
45    FILE * pInfile,
46      * pOutfile;
47    uint32_t lInfileSize;
48    uint8_t * buffer; //pointer for binary file buffer
49    uint32_t i; //indexing
50    uint32_t startaddr=atoi(argv[1]); //where to start?
51    uint32_t endaddr=startaddr+atoi(argv[2]);      //the end
52  if(debug)  printf ( "TS-480 DSP COFF ripper version 1.0 \n" );
53
54    if ( argc == 1 ) { //no arguments
55      printf ( "Correct Syntax: DSP-COFF-ripper startAddress endAddress INFILE
          OUTFILE \n" );
56      exit ( 1 );
57    } else  if (( pInfile=fopen (argv[3], "r")) == NULL ){ //input file not
        found
58      printf ( "File could not be opened for reading.\n" );
59      fputs ( "File opening error",stderr );
60      exit ( 2 );
61    } else {
62        if (( pOutfile = fopen (argv[4], "w" )) == NULL ){ //can't create output
            file
63          printf ( "File could not be opened for writing.\n" );
64          fclose ( pInfile );
65          fputs ( "File error",stderr );
66          exit ( 3 );
67        }
68    }
69
70  /*********************************************
71  * Allocate memory for the input file
72  * 1. Get the file size
73  * 2. Allocate memory
74  * 3. Read the file to the allocated memory
75  * 4. Check everything is fine
76  * 5. Close the input file
77  *********************************************/
78
79    fseek ( pInfile , 0 , SEEK_END );
80    lInfileSize = ftell ( pInfile ); // get file size for malloc
81    rewind ( pInfile );
82    buffer = ( uint8_t* ) malloc ( sizeof ( uint8_t ) * lInfileSize ); //
        allocate memory
83
84    if ( buffer == NULL ) {
85      fputs ( "Memory error" , stderr ); exit ( 4 );
86    }
87
```

```
88    if ( fread ( buffer, 1, lInfileSize, pInfile ) != lInfileSize ) { //read
          binary file to the previously allocated memory
89      fputs ( "Reading error", stderr );
90      exit ( 5 ); // quick'n dirty check for corruption
91    }
92    fclose ( pInfile ); // Input file won't be needed anymore.
93
94  /*********************************************
95  * Read coefficents from the memory and write to the file
96  *********************************************/
97    if(debug) printf ( "Moving data to the text file.. " );
98    for(i=startaddr; i<endaddr; i=i+2){
99          if(negative){
100             if(debug) printf("Addr:%d\n",i);
101             fprintf( pOutfile, "%d\n", 0xFFFF-(buffer [ i ]*0x100 + buffer [ i
                    +1 ] )+1);
102             if(debug)    printf( "Data: 0x%04X\n", 0xFFFF-(buffer [ i ]*0x100
                    + buffer [ i+1 ])+1);
103         }
104         else{
105             if(debug) printf("Addr:%d\n",i);
106             fprintf( pOutfile, "%d\n", (int16_t)(buffer [ i ]*0x100 + buffer [
                     i+1 ] ));
107             if(debug)    printf( "Data: 0x%04X\n",(buffer [ i ]*0x100 + buffer
                     [ i+1 ]));
108         }
109   }
110 if(debug)  printf("Done.\n");
111 /*********************************************
112 * Clean everything: close the file and release the memory.
113 * Shut the program.
114 *********************************************/
115   fclose ( pOutfile );
116   free ( buffer );
117   return 0;
118 }
```

B.3 Octave-skripti

```
#!/usr/bin/octave -qf
%
% Filename: coff.m
% Author:   Miika Ihanajärvi
% Email:    miika.ihanajarvi()gmail.com
% Created:  12.07.2011
% Edited:   Wed, 13 Jul 2011 14:16:13 +0300
% Version   1.01: Added support for analyzing FIR coefficents with freqz()
%       and -qf flag for reducing printing
%       1.0: Inital version
% On Ubuntu install: apt-get install octave-signal
% HowTo:    1) Give permission to execute: $chmod +x coff.m
%       2) Run: $./coff.m inputfile
% inputfile:    Text file containing one 16 bit signed integer value on a row
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
arg_list = argv ();
b= load("-ascii", arg_list{1});
figure(1)
freqz(b) %FIR analyzing
figure(2)
plot(b), grid on; %plotting the data
waitforbuttonpress;
```