TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

HEIKKI SARKANEN

SULAUTETTU LINUX AJONEUVO-PC:N OHJELMISTOALUSTANA

Diplomityö

Tarkastaja: Prof. Kari Systä
Tarkastaja ja aihe hyväksytty
Teknisten tieteiden tiedekunnan tiedekuntaneuvoston
kokouksessa 14.01.2015

TIIVISTELMÄ

HEIKKI SARKANEN: Sulautettu Linux ajoneuvo-PC:n ohjelmistoalustana
Tampereen teknillinen yliopisto
Diplomityö, 59 sivua, 14 liitesivua
Toukokuu 2016
Automaatiotekniikan koulutusohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: Prof. Kari Systä
Avainsanat: Linux, Qt, Yocto, U-boot, X11

Työn tavoitteena oli toteuttaa Linux-pohjainen ohjelmistoalusta mittaussovellukselle, joka sijoitetaan tablettia muistuttavaan, seitsemäntuumaisella kosketusnäytöllä ohjattavaan ajoneuvo-PC:hen. Työn lähtökohtana oli laitteistovalmistajan toimittama Freescalen i.MX6-järjestelmäpiiriin pohjautuva moduuli ja moduulin kanssa käytettävä evaluointiemolevy. Tälle laitteistolle oli valmiina laitteistovalmistajan toimittama Yocto Linux -työkalun käyttöön perustuva laitteistotukipaketti.

Tässä työssä otettiin käyttöön evaluointiemolevystä poikkeava tämän laitteen vaatimukset toteuttava lopullinen emolevy. Valmiiseen laitteistotukipaketin komponentteihin käynnistyslataajaan, Linuxiin, laitteistopuuhun ja Yocto-resepteihin tehtiin tarvittavat muutokset, joilla lopullisen emolevyn laitteiston ominaisuudet saatiin otettua käyttöön.

Laitteiston ominaisuuksien lisäksi ohjelmistoalustaan toteutettiin alustavasti ajoneuvo-PC:ssä tarvittavat Linux-kirjastot ja ohjelmistokomponentit valmiina saatavilla olleiden Yocto-reseptien pohjalta. Näistä tärkeimmät olivat etädiagnostiikka internetin välityksellä, alustan ohjelmiston sekä graafisen mittaussovelluksen päivittäminen internetin välityksellä, virtuaalinen näppäimistö sekä mahdollisuus mittaussovelluksen rinnalle asennettaviin apusovelluksiin.

Käytetyt Freescalen järjestelmäpiiri, Kontronin moduuli sekä Yocto Linux osoittautuivat hyvin dokumentoiduiksi ja käyttökelpoisiksi komponenteiksi ohjelmistoalustan toteuttamiseen. Alustan toteutuksessa yhtäkään ajuria tai Yocto-reseptiä ei tarvinnut toteuttaa tyhjästä, vaan kaikki pystyttiin muokkaamaan valmiiden mallien pohjalta. Freescalen järjestelmäpiiri ja Yocto ovat kuitenkin erittäin monipuolisia ja laajasti konfiguroitavia, joten työmäärä työkalujen käytön opetteluun osoittautui suureksi. Lopputuloksena oli vakaa, ylläpidettävä, ominaisuuksiltaan riittävä ja käyttötarkoitukseen muokattu Linux-jakelu. Ohjelmistoalustan tuotantokäyttöön saamiseksi vaaditaan kuitenkin vielä jatkokehitystä.

ABSTRACT

HEIKKI SARKANEN: Embedded Linux as a software platform for in-vehicle PC
Tampere University of Technology
Master of Science Thesis, 59 pages, 14 Appendix pages
May 2016
Master's Degree Programme in Automation Technology
Major: Software Engineering
Examiner: Prof. Kari Systä
Keywords: Linux, Qt, Yocto, U-boot, X11

The objective of the thesis was to implement a Linux based software platform for a measurement application, which is going to be used on a tablet-like seven inch touch screen device on an in-vehicle PC. In the beginning of the project, available hardware consisted of a Freescale i.MX6 System-on-Chip based Computer-on-Module and an evaluation carrier board provided by the manufacturer. A Yocto Linux board support package was provided for said hardware.

The final custom carrier board, which fulfills the requirements of this device, was taken into use for this project. Changes were made to the provided board support package components, including bootloader, Linux, device tree and Yocto recipes, so that the hardware features of the final carrier board could be used.

In addition to hardware features, initial implementation of Linux libraries and software components, based on the provided Yocto recipes, were carried out to fulfill the requirements of the in-vehicle PC system. The most important requirements were remote diagnostics via the Internet, updating platform software and a graphical measurement application via the Internet, virtual keyboard and an option for installing utility applications alongside the measurement application.

The Freescale System-on-Chip on Kontron Computer-on-Module and Yocto Linux turned out to be well documented and suitable components for the implementation of the software platform. In the implementation of the platform, neither drivers nor Yocto recipes needed to be written from scratch, as they could just be modified from the existing templates. The Freescale System-on-Chip and Yocto are very comprehensive and configurable, which increased the amount of research required to utilize all of the necessary tools. The result of the project was a stable and maintainable modified Linux distribution, which included all the essential features for the purposes of the project. However, further development is required for taking the software platform into production.

ALKUSANAT

Tämä tekniikan diplomityö on tehty yrityksen toimeksiannosta uuden Linux-pohjaisen tuotealustan kehittämiseksi. Haluan kiittää yritystä mielenkiintoisesta diplomityön aiheesta ja työn rahoittamisesta. Työ liittyy Tampereen teknillisen yliopiston Tietotekniikan laitoksen Ohjelmistotuotanto-pääainekokonaisuuteen. Tämä työ on tehty marraskuun 2014 ja toukokuun 2016 välisenä aikana.

Haluan kiittää professori Kari Systää erinomaisista vinkeistä työn sisältöön liittyen sekä kollegoita neuvoista elektroniikkaan liittyen. Lisäksi haluan kiittää läheisiäni ja ystäviäni kannustuksesta opiskelujeni valmiiksi saattamiseksi.

Tampereella, toukokuun 24. päivänä 2016

Heikki Sarkanen

SISÄLLYS

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

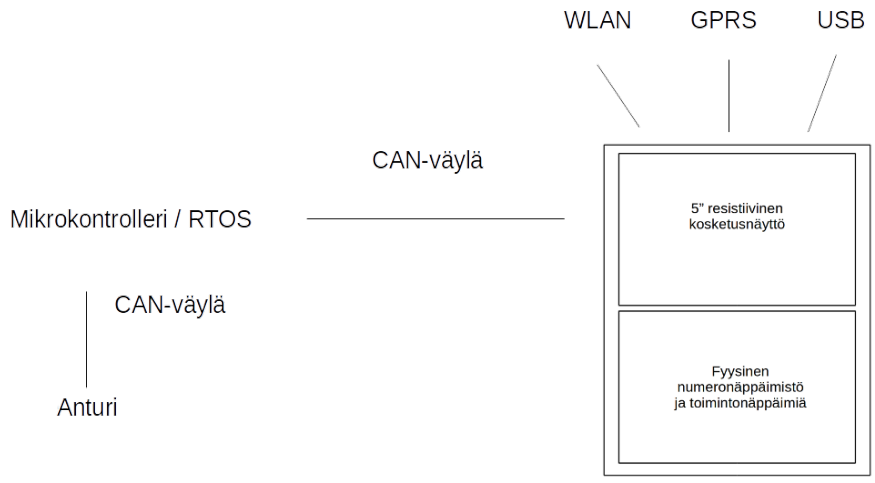
API	Application Programming Interface, rajapinta, joka määrittää eri ohjelmistokomponenttien vuorovaikutusmahdollisuudet
BIOS	Basic Input/Output System, yksinkertainen sisääntulojen ja ulosmenojen järjestelmä, joka vastaa tietokoneen käynnistämisestä ja matalan tason laitteistorajapinnasta
BOOTP	Bootstrap Protocol, automaattisen IP-osoitteen asettamisen mahdollistava protokolla
C++	C-kielestä kehitetty olio-ohjelmointia tukeva ohjelmointikieli
CAN	Controller Area Network, automaatioväylä, käytetään laajasti ajoneuvoissa ja teollisuudessa
DHCP	Dynamic Host Configuration Protocol, dynaaminen verkkoasetusten jakamiseen käytettävä protokolla
FTP	File Transfer Protocol, protokolla tiedostojen siirtoon TCP-protokollan avulla
GDB	GNU Debugger, yleiskäyttöinen testipenkki, jota käytetään laajasti esimerkiksi Linux-ohjelmistokehityksessä
GNU	Unixin kaltainen käyttöjärjestelmä, joka koostuu pelkästään vapaista ohjelmistoista
HTML	HyperText Markup Language, standardoitu kuvauskieli, jolla hyperlinkkejä sisältäviä hypertekstejä kuvataan
JavaScript	Alun perin Netscape Communicationsilla kehitetty Web-käyttöön tarkoitettu komentosarjakieli
Java	Alun perin Sun Microsystemsillä kehitetty olio-ohjelmointia tukeva ohjelmointikieli
Kernel	Käyttöjärjestelmän ydin
Kirjasto	Sisältää valmiita, useasti tarvittavia ja uudelleen käytettäviä ohjelmanosia
Levykuva	Levykuvalla Linuxin yhteydessä tarkoitetaan kokonaisen Linux-käyttöjärjestelmän sisältävää pakattua tiedostoa, joka voidaan purkaa laitteen tiedostojärjestelmään
LVDS	Low-voltage differential signaling, matalajännitteinen ja differentiaalinen nopeiden signaalien siirtotapa elektroniikan logiikkapiireissä, käytetään esimerkiksi DisplayPort ja Serial ATA -liitännöissä
MMU	Memory Management Unit, keskusmuistin hallintayksikkö
NFS	Network File System, verkkopohjainen tiedostojärjestelmä, joka mahdollistaa verkkopalvelimella sijaitsevien tiedostojen käytön paikallisen tiedostojärjestelmän osana

PCI	Peripheral Component Interconnect, tietokoneen sisäinen väylä, jonka välityksellä liitetään lisälaitteista järjestelmään
POSIX	Portable Operating System Interface, on joukko standardeja, jotka määrittelevät käyttöjärjestelmien yhteensopivuutta; POSIX:illa on myös oma API
Qt	Alun perin Trolltechin kehittämä ohjelmointiympäristö, joka tukee useita eri käyttöjärjestelmiä
SDK	Software Development Kit, yhdistelmä kehitystyökaluja, joilla voidaan kehittää tietyntyyppisiä ohjelmia
SMARC	Smart Mobility ARChitecture, standardi pienikokoisille yhden piirilevyn sulautetuille tietokoneille
TCP	Transmission Control Protocol, protokolla tiedonsiirtoon Internet-verkossa
TFTP	Trivial File Transfer Protocol, yksinkertainen ja tietoturvaton versio FTP-protokollasta
USB OTG	Universal Serial Bus On The Go, mahdollistaa USB-laitteen esiintymisen isäntälaitteena, jolloin siihen voidaan liittää muita USB-laitteita
USB	Universal Serial Bus, sarjamuotoisen tiedonsiirron arkkitehtuuri tietokoneen lisälaitteiden kytkemiseen

1. JOHDANTO

1.1 Tausta

Työn kohteena olevaan mittausjärjestelmään sisältyy mittausanturit, mittausanturei den dataa käsittelevä mikrokontrolleri ja näyttöpäätelaite. Mittausjärjestelmä sisältää elektroniikan lisäksi ohjelmistot sekä mikrokontrollerille että näyttöpäätelaitteeseen. Kosketusnäytöllä ja näppäimistöllä varustettu ajoneuvo-PC on näyttöpäätelaite, joka huolehtii mittaustietojen esittämisestä käyttäjälle ja niiden välittämisestä muihin järjestelmiin, kuten internetissä tarjottavaan raportointikäyttöliittymään. Tässä työssä keskitytään mittausjärjestelmän näyttöpäätelaitteen uuden version ohjelmistoalustaan. Kuvassa 1.1 on esitetty mittausjärjestelmän laitteistoarkkitehtuuri.



Kuva 1.1 *Mittausjärjestelmän laitteistoarkkitehtuuri.*

Näyttöpäätelaitteen nykyinen versio perustuu useamman vuoden ikäiseen Linux-versioon, jonka ylläpito on haasteellista. Myös nykyisen laitteen kosketusnäyttö,

prosessori ja muu laitteisto katsottiin tarpeellisiksi päivittää uudempiin. Muita olennaisia ominaisuuksia, kuten matkapuhelinverkossa tapahtuvaa tiedonsiirtoa, CAN-väylää ja GPS-paikannusta on tuettava myös uudessa näyttöpäätelaitteen versiossa.

Tässä diplomityössä kehitettävän näyttöpäätelaitteen Linux-pohjaisen ohjelmistoalustan kehityksen kanssa yhtä aikaa aloitettiin laitteiston suunnittelu. Laitteisto perustuu monimutkaisimmat toiminnot, kuten prosessorin, muistin ja näytönohjaimen, sisältävään moduuliin ja siihen liitettävään emolevyyn. Diplomityön alussa käytettävissä oli lopullinen moduuli, referenssialustan emolevy ja ohjelmistotuki referenssialustan emolevylle. Työn edetessä käyttöön saatiin lopullisen emolevyn ensimmäinen prototyyppi.

1.2 Tavoitteet

Tämän projektin tavoitteena on toteuttaa yhtenäinen yrityksen erilaisissa mittausjärjestelmissä käytettävä näyttöpäätelaite mittaustulosten graafiseen esittämiseen ja kommunikointiin mittauselektroniikan ja internet-palvelun välillä. Myöhemmin näyttöpäätelaitteen käyttöä voidaan laajentaa myös muun tyyppisiin mittausjärjestelmiin.

Tavoitteeseen liittyy kuusi keskeistä ohjelmistollista ominaisuutta:

- ohjelmistokehitykseen tarvittavien työkalujen käyttöönotto,
- lopullisen emolevyn laitteiston käyttöönotto ohjelmistossa,
- etädiagnostiikka internetin välityksellä,
- alustan ohjelmiston sekä uudemman Qt 5 -versioon pohjautuvan Qt-mittaussovelluksen päivittäminen internetin välityksellä,
- virtuaalinen näppäimistö ja
- mahdollisuus mittaussovelluksen rinnalle asennettaviin apusovelluksiin kuten internetselain tai laskin.

Alustan ohjelmiston tuottamiseen ja ylläpitoon käytettävät työkalut perustuvat pääosin laitteistovalmistajan tukemaan Yocto-projektiin, joka tarjoaa hyvän lähtökohdan erikoistetun Linux-jakelun kehittämiseen sulautetulle laitteelle. Yocto ei ole ollut aiemmin yrityksessä käytössä, joten sen käyttöönotto on ensimmäinen tavoite. Tämän jälkeen Yoctolla tuotetaan ristikäännös- ja virheenetsintätyökalut varsinaisen Qt-pohjaisen mittaussovelluksen sovelluskehityskäyttöön.

Projektin edetessä ja lopullisen emolevyn prototyypin valmistuessa sille kehitetään ohjelmistotuki matalan tason Linux-rajapintoihin asti käyttäen hyväksi laitteistovalmistajan referenssialustalle tarjoamaa ohjelmistotukea. Tässä vaiheessa tehdään laitteiston, kuten kosketusnäytön ja tiedonsiirtomoduulien, vaatimat muutokset Uboot-käynnistyslataajaan, Linux-käyttöjärjestelmän kerneliin sekä juuren tiedostojärjestelmään. Tavoitteena on siis luoda kaikki tarvittavat ohjelmistokomponentit, joilla lopullista emolevyä käyttävä laitteisto lisälaitteineen käynnistyy, ja sille voidaan kehittää itse mittaussovellusta.

Etädiagnostiikan käyttötarkoitus on se, että ensisijaisesti yrityksen huoltohenkilökunta pystyisi ottamaan graafisen etäyhteyden mittausjärjestelmän näyttöpäätteeseen ongelmatilanteissa ja näkemään saman käyttöliittymän, mikä näyttöpäätelaitteessa sillä hetkellä näkyy. Etädiagnostiikan mahdollisuus otetaan huomioon näytölle piirtämisestä huolehtivien ohjelmistokomponenttien valinnassa.

Mahdollisuus ohjelmistojen päivitykseen internetin välityksellä on puuttunut nykyisistä mittausjärjestelmistä, vaikka tämä ominaisuus koetaan USB-muistien avulla päivityksien jakamista nopeammaksi ja ketterämmäksi tavaksi jakaa useammin pienempiä päivityksiä. Alustan ohjelmistokomponenttien versiointi toteutetaan siten, että alustaa voidaan päivittää pakettienhallintaa ja omaa pakettivarastoa käyttäen.

Tavoite virtuaalisen näppäimistön toteuttamiseksi pohjautuu siihen, että samassa projektissa modernisoidusta laitteistosta jää fyysinen näppäimistö kokonaan pois. Virtuaaliseksi näppäimistöksi pyritään valitsemaan ohjelmistokomponentti, jota ei tarvitse upottaa itse sovellukseen vaan joka toimii järjestelmän laajuisesti.

Mahdollisuus apusovelluksiin on puuttunut nykyisestä mittausjärjestelmästä ja nyt uuden alustan tavoitteena on tarkoitus mahdollistaa nämä. Apusovellusten idea on se, että näyttöpäätelaitteen toiminnallisuutta voidaan laajentaa helpommin upottamatta kaikkea yhteen prosessiin ja yhteen sovellukseen.

1.3 Tutkimuksen rakenne

Tässä työssä esitellään ensin nykyisin käytössä oleva sekä tuleva mittauslaitteisto luvussa 2. Luvussa 3 käsitellään Yocto-projektia, jonka tarjoamat työkalut ovat keskeisiä työlle. Luvussa 4 esitellään käynnistyslataaja U-boot ja siihen tehtävät muutokset. Luvussa 5 kerrotaan laitteistopuukuvauksesta ja siihen tehtävistä muutoksista. Luku 6 kertoo Linux-käyttöjärjestelmän ytimestä ja sen muokkaamisesta laitteistolle sopivaksi. Luvussa 7 vedetään yhteen laitteistokomponenttikohtaisesti, mitä muutoksia kuhunkin ohjelmistokomponenttiin tarvitaan kyseisen laitteisto-

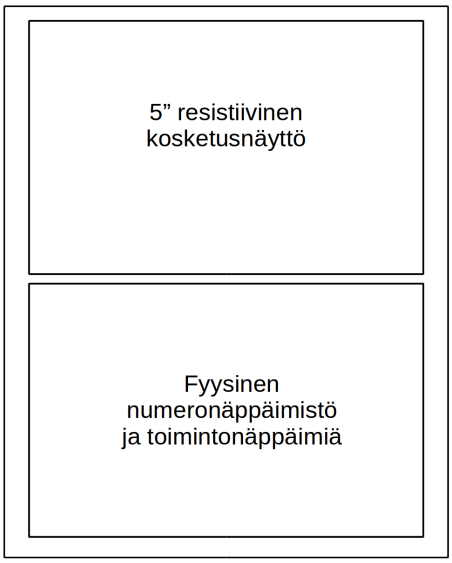
komponentin käyttöön ottamiseksi. Luvussa 8 esitetään johtopäätökset. Yhteenveto esitetään luvussa 9.

2. TUOTEALUSTAT

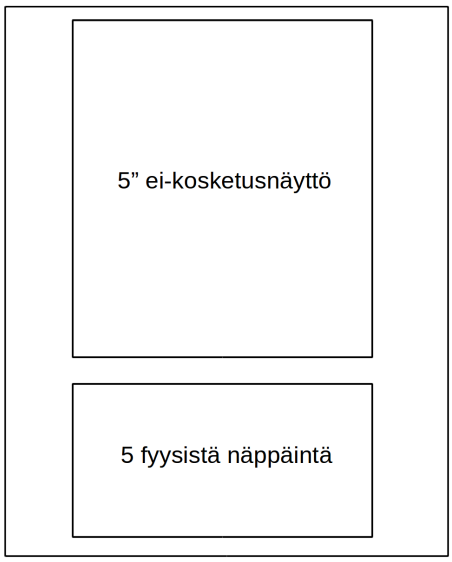
Nykyisten mittausjärjestelmien näyttöpäätelaitteiden laitteisto- ja ohjelmistoalustat ovat merkittävästi eriäviä. Nykyistä mittausjärjestelmä A:ta (kuvassa 2.1) voidaan pitää teknisesti edistyksellisimpänä. Tässä mittausjärjestelmässä käytetty v. 2010 julkaistu Qt versio 4.7 versio kaipaa kuitenkin jo päivitystä. Linux-buildroot -pohjainen ohjelmisto- ja laitteistoalusta on koettu myös ylläpidoltaan haasteelliseksi. Buildroot soveltuu yhden kehittäjänsä mukaan Yoctoa paremmin pienten ja yksinkertaisten muutaman ohjelmistokomponentin sisältävien Linux-järjestelmien rakentamiseen, mutta pakettienhallinnan puute ja suppea laitteistotuki rajoittavat käyttöä monimutkaisemmissa ympäristöissä [38]. Yocton wiki-sivujen mukaan Buildroot on vanhentunut Makefile-pohjainen järjestelmä, jonka arkkitehtuurin puutteet, kuten pakettienhallinnan puuttuminen, aiheuttivat sen, ettei Buildroot laajentunut koskaan monimutkaisempiin käyttötapauksiin [39]. Yleisesti Buildroot:ia kehitetään edelleen aktiivisesti, mutta Yoctoon verrattuna sillä ei ole yhdenkään suuremman organisaation tai yrityksen tukea verrattuna Yocton tukijoihin, kuten Linux-säätiöön, Inteliin ja Freescaleen. Osittain Buildrootin puutteiden vuoksi uusien ominaisuuksien tuominen, vikojen korjaaminen ja tietoturvapäivitysten tekeminen on jäänyt lähes ainoastaan alustalla ajettavan Qt-mittaussovelluksen päivitysten varaan. Lisäksi Buildrootin pakettienhallinnan tuen puuttumisen johdosta Qt-mittaussovelluksen päivitys on hoidettu USB-muistia ja manuaalista päivityskomentosarjaa käyttäen, mikä hidastaa uusien ohjelmistoversioiden käyttöönottoa.

Kuvassa 2.2 on taas huomattavasti sekä laitteistoltaan että ohjelmistoltaan edellä esitetystä mittausjärjestelmästä poikkeavan mittausjärjestelmä B:n näyttöpäätelaitteen periaatekuva. Tämän mittausjärjestelmän näyttö on pieni, eikä se tue kosketusta. Lisäksi sen käyttöliittymä ja ohjelmistoalusta on toteutettu täysin eri tekniikalla kuin aiemmin esitetyssä mittausjärjestelmässä A:ssa.

Tässä luvussa kuvataan työn taustaa ja sitä, miten uudella yhtenäisellä tuotealustalla päädyttiin käyttämään valittua laitteistoa ja ohjelmistoa.



Kuva 2.1 *Nykyisen mittausjärjestelmä A:n näyttöpäätteen periaatekuva.*



Kuva 2.2 *Nykyisen mittausjärjestelmä B:n näyttöpäätteen periaatekuva.*

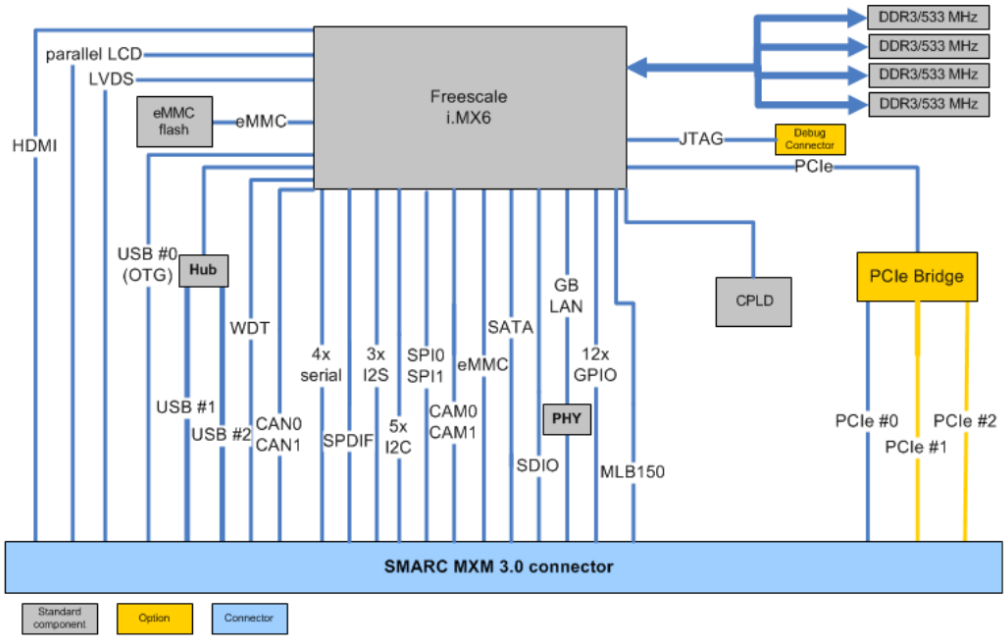
2.1 Laitteisto

Käytettävän laitteiston keskeisimmäksi osaksi valikoitui Kontron SMARC-sAMX6i Solo -moduuli. Valinnan perusteina olivat yleisesti käytetty ja laajasti tuettu Freescalen i.MX6-järjestelmäpiiri (engl. System-on-Chip), skaalautuva SMARC-arkkitehtuuri ja hyvä tuki Linux-käyttöjärjestelmälle sekä Qt-grafiikkakirjastolle.

2.1.1 SMARC-standardi

SMARC-arkkitehtuuri on vuonna 2013 julkaistu ja Standardization Group for Embedded Technologies -ryhmän ylläpitämä avoin standardi tietokonemoduuleille. Pie-

nikokoiset SMARC-moduulit ovat mitoiltaan joko 82x50 tai 82x80 mm, ja ne liittyvät sovelluskohtaisesti räätälöitävään emolevyyn 314-pinnisellä liittimellä. Tyypillisesti tehonkulutus on alle 6 wattia, mutta myös suurempaa tehoa vaativia moduuleita noin 15 watin luokkaan asti on mahdollista suunnitella. SMARC-standardi määrittelee moduulin rajapinnat, joista osa on pakollisia ja osa vapaaehtoisia toteuttaa. Pakollisia ovat yksi näyttöliitäntä, muistikorttiliitäntä, useammat tiedonsiirtoväylät (kuten I2C ja USB), järjestelmänhallinta sekä käynnistysmedian valinta. Freescalen i.MX6-järjestelmäpiiriin perustuvan Kontronin SMARC-moduulin lohkokaavio on esitetty kuvassa 2.3. [23], [24]

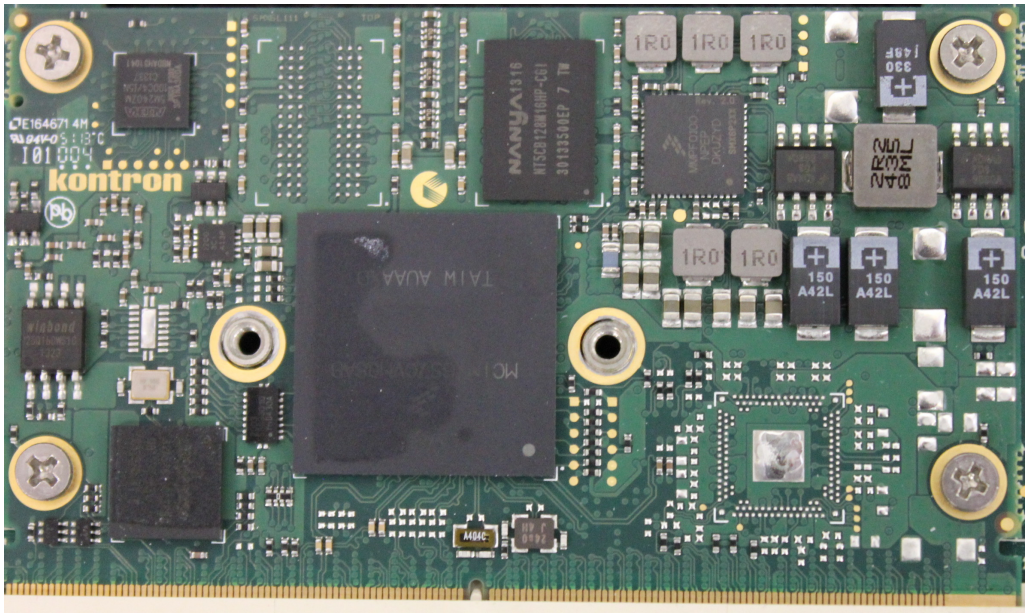


Kuva 2.3 *i.MX6-järjestelmäpiiriin perustuvan SMARC-moduulin komponentit [4].*

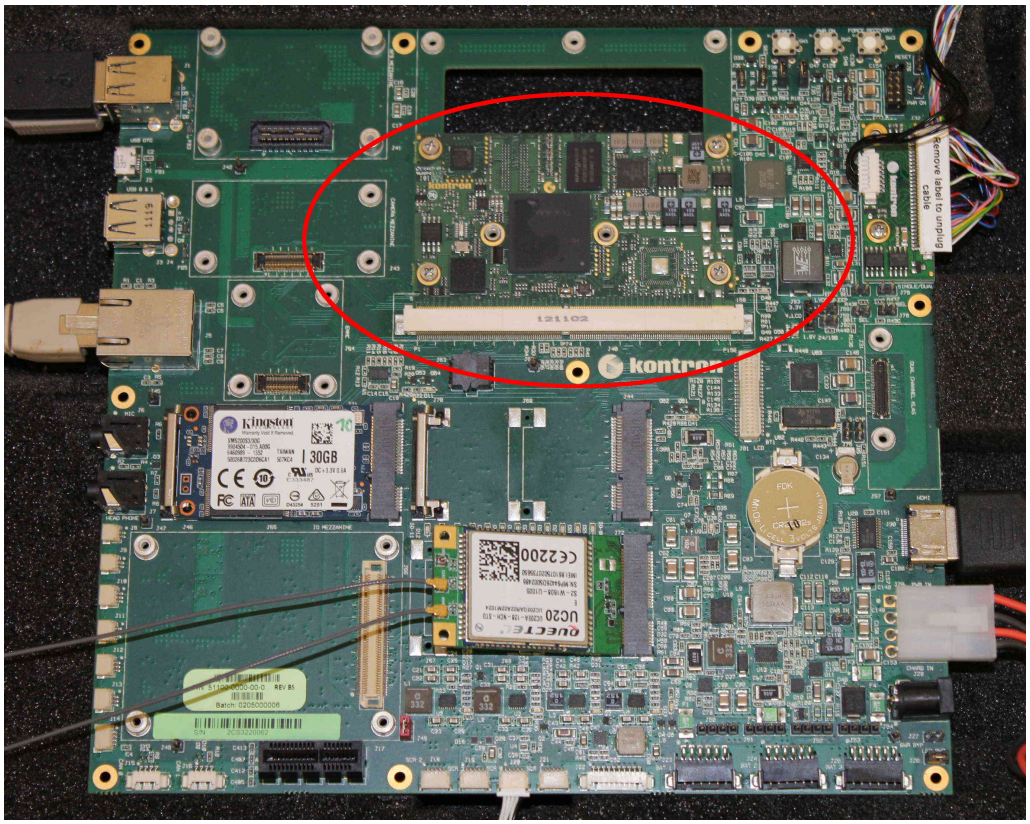
2.1.2 Kontron SMARC-sAMX6i

Kontron SMARC-sAMX6i -moduuli on esitetty kuvassa 2.4. Tämän diplomityöprojektin aikana ohjelmistoalustan kehitysvaiheessa SMARC-sAMX6i-moduuli on ollut kiinnitettynä Kontronin SMARC-evaluointiemolevyyn, joka tarjoaa fyysiset liitännät moduulin oheislaitteisiin. SMARC-evaluointiemolevy on esitetty kuvassa 2.5. Lopputuotteessa käytetään samaa SMARC-sAMX6i Solo -moduulia, mutta SMARC-evaluointiemolevy vaihdetaan samassa projektissa ohjelmistoalustakehityksen kanssa yhtä aikaa suunniteltuun piirilevyyn. Tälle itse suunnitellulle piirilevylle on toteutettu kaikki käytössä tarvittavat lisäominaisuudet ja liitännät, kuten esimerkiksi 3G/GLONASS-moduuli ja fyysinen LVDS-näyttöliitäntä. Kehitysvaiheessa on ollut mahdollista kehittää ohjelmistoa tulevalle kosketusnäytölle ja

3G/GLONASS-moduulille, koska ne on saatu liitettyä SMARC-moduuliin SMARC-evaluointiemolevyn kautta. Itse suunniteltu piirilevy on rajattu tämän diplomityön ulkopuolelle.



Kuva 2.4 *Kontron SMARC-sAMX6i Solo -moduuli.*



Kuva 2.5 *Kontron SMARC-evaluaatioemolevy, johon on kiinnitetty SMARC-moduuli.*

Kontronin SMARC-sAMX6i Solo -moduulin Freescale i.MX6 -järjestelmäpiirin prosessori perustuu ARM Cortex A9 -arkkitehtuuriin. Kyseessä on teollisuuskäyttöön tarkoitettu versio i.MX6-järjestelmäpiiristä, jonka prosessorin maksimikellotaajuus on 800 megahertsiä. Teollisuuskäyttöön suunnitellun version selkeä etu on laajempi käyttölämpötila-alue ($-40°C \ldots 105°C$). Samalle järjestelmäpiirille on integroitu myös Vivanten 2D- ja 3D-kiihdytystä tukeva näytönohjain. Freescale i.MX6 -järjestelmäpiiri julkaistiin tammikuussa 2011. [17], [5]

Kontron on integroinut SMARC-sAMX6i Solo -moduulille Freescalen i.MX6-piirin lisäksi 512 megatavua DRAM-muistia ja 4 gigatavua flash-muistia. Moduuli tukee monipuolisesti liitäntöjä – kuten I2C:tä, LVDS:ää ja USB:tä – lisälaitteiden liittämiseksi. SMARC-arkkitehtuuri on skaalautuva ja tukee useampien valmistajien moduuleja ja eri prosessoriarkkitehtuureja. Skaalautuva arkkitehtuuri mahdollistaa sen, että alun perin valittu yksiytiminen moduuli voidaan jatkossa vaihtaa kaksi- tai neliytimisen prosessorin sisältävään moduuliin hyvin pienin muutoksin laitteistopuuhun, joka on esitetty tarkemmin luvussa 5. Myös vaihto esimerkiksi X86-arkkitehtuuriin perustuvaan moduuliin on teoriassa mahdollista, mutta se edellyttää hieman suurempia muutoksia ainakin ohjelmistoon. Myös standardissa vapaaehtoisesti toteutettavien, mutta tässä projektissa välttämättömien rajapintojen, kuten CAN-väylän, mahdollinen puute rajoittaa toisenlaiseen järjestelmäpiiriin perustuvaan moduuliin vaihtamista. [3]

2.1.3 SMARC-emolevy

SMARC-emolevy suunniteltiin ohjelmistoalustan projektin rinnalla samanaikaisesti kulkeneessa projektissa. Ohjelmistoprojektissa käyttöjärjestelmän valinta oli jo suoritettu, kun laitteistokomponentteja alettiin valita, joten komponentteja valittaessa voitiin kiinnittää huomio siihen, että niistä löytyy tukea valitulle käyttöjärjestelmälle.

Tärkeimmäksi lisälaitteeksi eli itse näyttöyksiköksi valikoitui seitsemäntuumainen kapasitiivinen kosketusnäyttö. Mittausjärjestelmien paikannuksesta ja tiedonsiirrosta mobiiliverkossa huolehtii yhdistetty globaaleja taajuuksia tukeva 3G/GLONASS-moduuli. Lisäksi valittiin yhdistetty Bluetooth ja WLAN -moduuli, joka huolehtii tiedonsiirrosta lyhyemmillä etäisyyksillä laitteen ympäristössä. Nämä lisälaitemoduulit piirilevyantenneineen juotetaan tähän itse suunniteltuun lopulliseen SMARC-emolevyyn. Lisäksi piirilevylle tulee huomattava määrä muita pienempiä komponentteja, kuten äänimerkin tuottamiseen sekä analogisen kameran ja tulostimen liittämiseen vaadittavat komponentit, mutta niiden ominaisuuksia ei tarkastella tarkemmin tässä työssä.

2.2 Ohjelmisto

Projektin alussa laitteiston valitsemisen lisäksi yksi tärkeimmistä valinnoista on käyttöjärjestelmä. Jo nykyisen näyttöpäätteen toiminnallisuudet ovat niin monipuoliset, että puhtaasti ilman käyttöjärjestelmää kehittäminen ei ollut vaihtoehtona. Luvussa 2.1 esitetyn laitteiston valmistaja Kontron tarjoaa ohjelmistotuen Linuxille ja Windows Embedded Compact 7:lle. Lisäksi Kontronilla oli tarjota vain epävakaaksi havaittu testiversio ilman mitään tuotetukea Android 4.4:stä, joka osoittautui liian raskaaksi SMARC-moduulin yksiytimiselle 800 megahertsin prosessorille ja 512 megatavun muistimäärälle. Käytännössä Linux ja Windows olivat vaihtoehdot käytettävälle käyttöjärjestelmälle. [3]

Linux ja Windows ovat olleet käytössä sulautetuissa laitteissa jo pitkään. Siirtymistä aiemman näyttöpäätteen Linux-ympäristöstä kokonaan erilaiseen Windows-ympäristöön pidettiin kalliina ja riskialttiina alustavaihtoehtojen erilaisuudesta, Windows-kehityskokemuksen puutteesta ja lisensointikuluista johtuen.

Linux ei itsessään vielä ole yksikäsitteinen käyttöjärjestelmä, vaan siitä on saatavilla useita valmiita eri jakeluita ja lisäksi ensisijaisesti sulautettuihin järjestelmiin tarkoitettuja työkaluja oman jakelun rakentamiseen. Järjestelmäpiirin valmistaja Freescale julkaisee säännöllisesti uusia versioita laitteistotukipaketistaan, jossa on virallinen tuki vain Freescalen itsensä valmistamille referenssipiirilevyille. Nykyiset ja tulevat versiot näistä laitteistotukipaketeista julkaistaan Linux-alustalla ainoastaan Yocto-työkalun metadatana. Freescalen järjestelmäpiiriä käyttävä laitteistovalmistaja Kontron tekee Freescalen julkaisun pohjalle tarvittavat lisäykset, jotta ohjelmisto tukee kaikkia Kontronin laitteiston ominaisuuksia ja julkaisee nämä lisäykset Yocto-metadataan omana virallisena laitteistotukipakettina. Tämän laitteistovalmistajan tekemän valinnan johdosta Yocto valittiin tässä projektissa käytettäväksi työkaluksi oman sulautetun Linux-jakelun rakentamiseen. Yocto-työkalu on kuvattu tarkemmin luvussa 3. [25]

Käyttöjärjestelmän lisäksi olennainen valinta on käytettävä grafiikkakirjasto. Aiemman näyttöpäätteen toteutuksessa on käytetty Qt-grafiikkakirjastoa. Samaa Qt:lla toteutettua mittaussovellusta on myös asennettu Android-tabletteihin ja Windows-kannettaviin. Näissä käyttötapauksissa asiakkaalla on käytössä vain yrityksen mittauselektroniikka ja ohjelmistolisenssi sovellukseen, mutta ei yrityksen näyttöpäätettä. Tästä johtuen käytettävän grafiikkakirjaston vaatimus on tukea myös työpöytä-Windowsia ja tabletti-Androidia yrityksen Linux-näyttöpäätteen lisäksi. Vanhan Qt-koodin uudelleenkäyttömahdollisuuksia uudessa mittaussovelluksessa pidettiin kuitenkin lähes merkityksettöminä, koska sovellus on tarkoitus kirjoittaa kokonaan

uudestaan. Qt:n lisäksi vaihtoehtoja olisi mahdollisesti ollut C#- tai JavaScript-pohjaisissa grafiikkakirjastoissa, mutta käytännössä Qt valittiin grafiikkakirjastoksi uudelle näyttöpäätteelle ilman laajempaa tutkimusta. Tähän vaikutti tuotekehittäjien aikaisempi kokemus Qt:n käytöstä ja Qt:n erinomainen tuki käytettävälle laitteistolle. Qt:n tuki i.MX6-järjestelmäpiiriin perustuville laitteille on seurausta siitä, että Freescalen samaan järjestelmäpiiriin perustuva referenssialusta SABRE SD i.MX6 kuuluu korkeimmalla prioriteetilla Qt:n tukemiin laitteisiin. Käyttöjärjestelmän valintaan liittyen huomionarvoista on, että Qt:n tuki i.MX6-järjestelmäpiirille koskee vain käyttöä sulautetulla Linux-alustalla eikä Androidia tai Windows Embedded Compact 7:ää. Qt-versioksi valikoitui käyttöön kesäkuussa 2015 julkaistu 5.4.2. Mittausjärjestelmän useammille laitteille skaalautuvan käyttöliittymän toteutukseen käytetään QML-kuvauskieltä. [18]

3. YOCTO-PROJEKTI

Yocto-projekti on Linux-säätiön alainen työryhmä, jonka tarkoitus on vapaasti suomennettuna määritetty seuraavasti:

> "Yocto-projekti tarjoaa avoimen lähdekoodin, korkealaatuisen infrastruktuurin ja työkalut auttamaan ohjelmistokehittäjiä tekemään oman erikoistetun Linux-jakelun mille tahansa laitteistoarkkitehtuurille monilla eri tuotealueilla. Yocto-projekti pyrkii tarjoamaan helpon lähtökohdan ohjelmistokehittäjille." [1, s.288]

Yocton työkaluilla siis voidaan luoda Linux-jakelu, jossa on halutulla tavalla konfiguroitu käynnistyslataaja (engl. bootloader), käyttöjärjestelmän ydin, kirjastot ja sovellukset. Yleensä konkreettisia osia ovat käyttöjärjestelmän ydin, käynnistyslataaja ja juuren tiedostojärjestelmä (engl. root filesystem). Yoctossa on työkalut, joilla voi tuottaa nämä kaikki alkaen lähdekoodien lataamisesta ja päättyen komponenttien oikeanlaiseen paketointiin. Käännöksessä käytetään myös tehokkaasti uudelleen lopputuotteita, joiden käännösohjeisiin ei edellisen kerran jälkeen ole tullut muutoksia tai jotka on käännetty jo aiemmin jotain muuta pakettia varten. Yocto valmistelee oman ristikäännösympäristönsä, mutta sillä on edelleen joitakin vaatimuksia isäntätietokoneelle, kuten vain tietyt Linux-jakelut ovat tuettuja, ja lisäksi joitakin kirjastoja ja ohjelmia pitää olla valmiiksi asennettuna. Yocto tarjoaa kuitenkin mahdollisuuden määrittää kohdejärjestelmään käytetyt ohjelmistokomponentit versioineen niin tarkasti, että tuotettua järjestelmää voidaan pitää deterministisenä. Näin ollen samanlaiset komponentit on mahdollista tuottaa Yoctoa käyttäen myös toisella isäntäkoneella. [1, s.288-318]

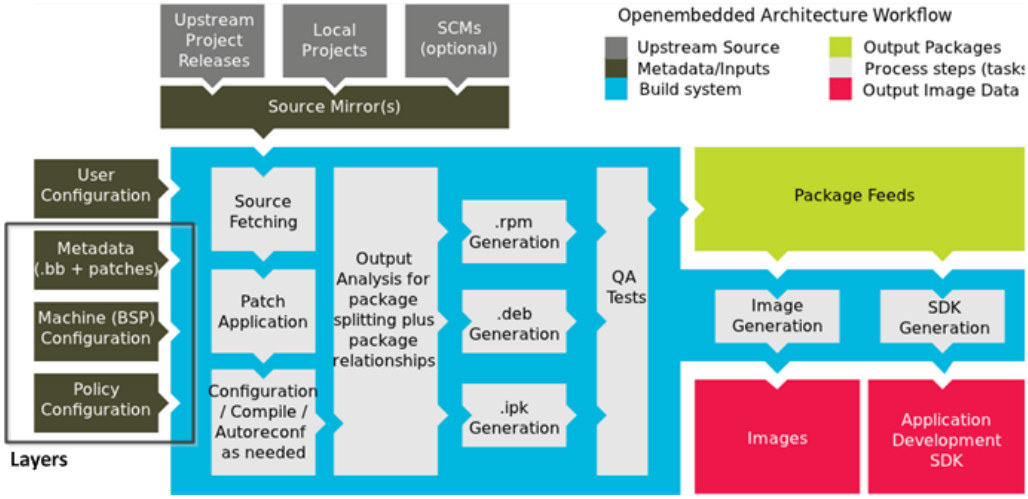
Yoctosta julkaistaan uusia versioita yhteisön kanssa laaditun suunnitelman mukaisesti yleensä kuuden kuukauden välein [10]. Tässä työssä käytetty Yocton versio on huhtikuussa 2015 julkaistu 1.8, koodinimeltään *Fido*.

3.1 Yocton historiaa

Yocto-projekti käynnistettiin Linux-säätiön toimesta vuonna 2010. Maaliskuussa 2011 projektiin liitettiin vuodesta 2003 kehitetty OpenEmbedded-projekti. Samassa yhteydessä Linux-säätiö julkisti useita uusia projektia tukevia yhtiöitä, joista tunnetuimpia ovat Dell, Freescale, Intel, Mentor Graphics, Texas Instruments ja Wind River. OpenEmbedded pohjautuu Gentoo Portagen pakettienhallinnan tehtävien ajastamiseen tarkoitettuun työkaluun nimeltään BitBake. Vuonna 2004 OpenEmbedded-projekti hajautettiin kahteen erilliseen osaan, jotka olivat: BitBake eli Unix-järjestelmissä yleisesti käytetyn *Make*-työkalun kaltainen yleinen tehtävien suorittaja ja OpenEmbedded eli käännösohjeet, joita BitBake hyödyntää[9]. [8] [1, s.346]

3.2 Yocton ominaisuuksista yleisesti

Kuvassa 3.1 esitetään korkealla tasolla, mitä työvaiheita liittyy Yocto:n käännösprosessiin. Prosessiin menee sisään lähdekoodia ja metadataa, ja siitä välivaiheiden kautta syntyy ohjelmistopaketteja ja lopulta kokonainen Linux-jakelu kehitystyökaluineen.



Kuva 3.1 *Korkean tason vuokaavio Yocto-ympäristön työnkulusta [6].*

BitBake on Yocton tärkein työkalu. Se huolehtii käännösohjeiden eli luvussa 3.3.1 kuvatun metadatan parsimisesta, tehtävien listaamisesta metadatan pohjalta ja lopulta listattujen tehtävien suorittamisesta. BitBake perustuu Python- ja Bash-komentosarjakielillä toteutettujen tehtävien suorittamiseen tehokkaasti rinnakkaistettuina ja samanaikaisesti monimutkaiset tehtävien väliset riippuvuudet huomioi-

den. Konseptiltaan BitBake muistuttaa paremmin tunnettua Unix-pohjaisissa järjestelmissä yleisesti käytössä olevaa *Make*-nimistä käännösautomaatiotyökalua, mutta eroaa siitä myös joiltain osin huomattavasti. *Make*-työkalun Linux-versio on GNU:n kehittämä ja tunnetaan nimellä *GNU Make*. Kuten *Make* myös BitBake kontrolloi, miten ohjelmisto käännetään. Yksinkertaisten ohjelmistojen lisäksi BitBake pystyy huolehtimaan myös monimutkaisemmistakin tehtävistä, kuten kokonaisen käyttöjärjestelmän tuottamisesta. *Maken Makefile*-tiedostojen sijaan BitBakessa käytetään reseptejä, jotka nimetään paremmin kuvaamaan kyseisen reseptin tuottamaa ohjelmistoa. BitBake:n metadata on jaettu erityyppisiin tiedostoihin, joilla voi olla erilaisia suhteita keskenään. Metadatan tyypit on kuvattu luvussa 3.3.1. Toisin kuin *Make* BitBake sisältää myös kirjaston lähdekoodin noutamiseen paikallisista tiedostoista, versionhallintajärjestelmistä ja muualta internetistä. Lisäksi BitBake noudattaa asiakaspalvelin-abstraktiota, ja sitä voidaan käyttää joko komentoriviltä, XML-RPC-palvelun avulla tai joidenkin graafisten käyttöliittymien kautta. [9]

BitBakea lähdettiin alun perin kehittämään, koska vastaavaa työkalua sulautettujen Linux-alustojen tuottamiseen ei ollut olemassa. Työpöydälle tarkoitettujen Linux-jakeluiden käyttämistä työkaluista puuttui joitain tärkeitä sulautetuissa järjestelmissä tarvittavia ominaisuuksia, ja sulautettuihin järjestelmiin suunnitellut luvussa 2 esitellyn Buildrootin kaltaiset järjestelmät eivät olleet skaalautuvia eivätkä ylläpidettäviä. Yleisesti BitBaken suunnittelun lähtökohtina ovat olleet joustavuus, yleiskäyttöisyys ja riippumattomuus. Tarkemmin eritellen BitBaken tärkeimpiä ominaisuuksia ovat hallita ristikäännökset, hallita pakettien väliset riippuvuudet, tukea rajatonta määrää erilaisia tehtäviä pakettia kohti ja olla yksittäisistä Linux-jakeluista riippumaton sekä käännösympäristön että kohdeympäristön osalta. Muita tärkeitä tavoitteita ovat olleet pysyä arkkitehtuurista riippumattomana ja olla itsenäinen käännösympäristön tiedostojärjestelmään nähden. Metadataan liittyen on ollut tärkeää, että siinä on voinut olla ehdollisia arkkitehtuurikohtaisia, käyttöjärjestelmäkohtaisia, jakelukohtaisia ja laitekohtaisia osia sekä yhteisten osien periytymismekanismin vaativia yhteisiä osia. Lisäksi paikallisen metadatan lisäämisen mahdollistaminen on ollut olennaista. Myöhemmin projektin laajentuessa on huomattu, että metadataa on ollut hyvä jakaa useampiin toisiaan korvaaviin ja täydentäviin kerroksiin, joita on tarkemmin kuvattu luvussa 3.3.2. [9]

3.3 Käsitteitä

Yoctoon liittyy monia erilaisia käsitteitä. Metadatan esittämis- ja käyttötavat sekä metadatan jakaantuminen erilaisiin kerroksiin ovat näistä olennaisimpia. Tässä luvussa luodaan katsaus näihin Yocton käsitteisiin ja niiden välisiin suhteisiin.

3.3.1 Metadata

Yoctossa metadata koostuu konfiguraatiotiedostoista, resepteistä ja luokista. Konfiguraatiotiedostot sisältävät järjestelmässä globaalisti hyödynnettävää tietoa siitä, miten luokat ja reseptit toimivat. Liitteessä E on esimerkki laitteistokonfiguraatiotiedostosta, joka kuvaa tässä työssä käytössä olleen laitteiston. Resepti sisältää tiedon, miten yksi ohjelmistopaketti rakennetaan. Resepti kertoo, miten BitBake voi noutaa lähdekoodin, mahdollisesti purkaa lähdekoodin arkistosta, kääntää ja asentaa paketin sisältämän yksittäisen sovelluksen. Yocto-resepteissä konfiguraatiota hallitaan pakettikohtaisesti *PACKAGECONFIG*-muuttujan avulla. Erilaiset konfiguraatiovaihtoehdot on listattu Yocto-reseptissä. Tietyn ominaisuuden lisääminen globaalisti koko käyttöjärjestelmän tasolla konfiguraatiotiedostossa tai reseptikohtaisesti muuttaa käytettäviä konfiguraatioparametreja sekä paketin käännöksenaikaisia että ajonaikaisia riippuvuuksia muihin paketteihin. Näin esimerkiksi ikkunointijärjestelmän valinta, joka vaikuttaa moniin paketteihin konfiguraatioineen ja riippuvuuksineen, voidaan vain listata käännettävän käyttöjärjestelmän käyttöjärjestelmäkohtaisessa konfiguraatiotiedostossa, ja tämän jälkeen kaikki paketit toimivat käyttöön valitussa ikkunointijärjestelmässä. Liitteessä A on esimerkki reseptistä, jossa on kuvattu virtuaalinäppäimistön tuottaminen. Luokat sisältävät yleisiä tietoja työvaiheista, joita voidaan hyödyntää useammassa reseptissä. Liitteessä A olevassa reseptissä peritään luokka *qmake5*, joka sisältää yleisesti käytettäviä osia Qt5:n *Qmake*-käännöstyökalun käyttämisestä Yocto ympäristössä. [6]; [1, s.1066]; [1, s.1572]

3.3.2 Kerros

Yocton metadata jaotellaan kerroksiin (engl. layer). Kerros on siis kokoelma konfiguraatiotiedostoja, reseptejä ja luokkia, jotka liittyvät jotenkin toisiinsa. Ainoa pakollinen tiedosto kerroksessa on sen konfiguraatio, joka kertoo muun muassa kerroksen riippuvuuksista muihin kerroksiin ja miten tiedostot sijaitsevat kansiorakenteessa. Tyypillisesti kerrokset nimetään *meta*-etuliitteellä, kuten Freescalen ARM-alustan reseptit on koottu *Meta-fsl-arm*-kerroksen alle. Kerros voi sijaita vapaasti tiedostojärjestelmässä, mutta niiden sijainnit pitää määrittää erillisessä käännöskansiokohtaisessa *bblayers.conf*-konfiguraatio tiedostossa. Kerroksilla on prioriteetti, joka määrittää, mitä reseptiä käytetään, jos samasta paketista on resepti useammassa eri kerroksessa. Luvussa 3.3.1 esitetyllä tavalla resepti voi myös muokata jossain toisessa kerroksessa olemassa olevaa reseptiä, jolloin prioriteetti määrittää, missä järjestyksessä reseptiä muokataan. [1, s.1436]

Kerrokset voidaan jakaa kolmeen pääluokkaan: jakelukerroksiin (engl. distribution

layer), laitteistotukikerroksiin (engl. board support package layer) ja ohjelmistokerroksiin (engl. software layer). Jakelukerros määrittää korkean tason säännöstöjä, kuten sen, että käytetäänkö ohjelmistopaketeissa Opkg-, Rpm- vai Debian-paketointia tai mitä ikkunointi-järjestelmää käytetään. Laitteistotukikerros tarjoaa tietylle laitteistolle tarkoitettuja reseptejä ja muutoksia muiden kerrosten resepteihin. Kun alustalle ominaiset muutokset on eriytetty omaan kerrokseensa, vaihto esimerkiksi Freescalen ARM-arkkitehtuurista Intelin X86-arkkitehtuuriin pitäisi sujua ilman suurempaa työmäärää. Ohjelmistokerrokset taas sisältävät ohjelmistojen tuottamiseen tarvittavaa metadataa, jota voidaan käyttää käyttöjärjestelmän ominaisuuksista tai laitteiston ominaisuuksista riippumatta. Tällaisia kerroksia Yoctoon, joita on listattuna OpenEmbeddedin hakemistossa [49], on saatavissa huomattavia määriä, ja näistä esimerkkejä ovat esimerkiksi *Meta-java*, *Meta-qt5* ja *Meta-browser*. [6]

Kuvassa 3.2 on esitetty, mitä kerroksia tyypillisesti on yhtä aikaa käytössä. Alimmainen *yleinen metadatakerros* on ohjelmistokerros, joka sisältää paljon yleisen tason reseptejä, luokkia ja QEMU-emulaattorin konfiguraatiotiedostot. Tämän kerroksen metadata pohjautuu alun perin Yoctoon yhdistetyn OpenEmbedded-projektin metadataan. *Käyttöjärjestelmäkerros* on jakelukerros, joka on alun perin Poky-projektissa toteutettu referenssitoteutus Linux-käyttöjärjestelmälle. *Laitteistotuenkerros* on useimmiten laitteistovalmistajan toteuttama, ja se lisää tuen jollekin tietylle laitteistolle. *Käyttöliittymäkirjastokerros* on ohjelmistokerros, joka lisää tuen jollekin käyttöliittymäkirjastolle, kuten GTK:lle, Qt:lle tai Java:lle. *Kaupallisen toimittajan kerros* mahdollistaa kaupallista tukea sulautetulle Linuxille tarjoavien yhtiöiden, kuten Windriverin tai Mentor Graphicsin kaupallisten komponenttien lisäämisen. *Kehittäjäkohtainen kerros* sisältää projektissa syntyneet muokkaukset ja lisäykset alempien kerrosten metadataan. [1, s.1436]

Liitteessä H esitetään kansiorakenne yritykselle spesifisen kerroksen sisältämistä resepteistä, konfiguraatiotiedostosta ja muutostiedostoista. Toistaiseksi on katsottu riittäväksi pitää vain ohjelmistoalustaan liittyvät muutokset tässä kerroksessa. Jatkossa jos laitteistoja tulee useampia tai muutosten määrä kasvaa huomattavasti, voisi olla syytä jakaa yritysspesifiset muutokset erillisiin jakelu-, laitteistotuki- ja ohjelmistokerroksiin.

3.3.3 Jakelu

Oman jakelun (engl. distribution) luominen Poky-referenssijakelun pohjalta on yksinkertaista. Poky on Yocton referenssijakelu, joka pitää sisällään suuren määrän metadataa eli reseptejä, laitemäärittelyitä ja luokkia jaoteltuna useaan eri kerrokseen. Jakelukohtaisissa konfiguraatiotiedostoissa määritellään kaikkiin resepteihin

Kuva 3.2 *Metadatakerrosten jakautuminen, sovellettu lähteestä [19].*

vaikuttavia ominaisuuksia, kuten käytetäänkö X11:tä, Waylandia vai näyttöpuskuria (engl. framebuffer) grafiikan esittämiseen. Yksinkertaisimmillaan omassa jakelussa peritään Pokyn oletuskonfiguraatio, mutta muutetaan vain esimerkiksi Wayland käytetyksi ikkunointiympäristöksi. Oman jakelun käyttöön ottamiseksi se pitää määrittää käännöshakemistokohtaisessa konfiguraatiotiedostossa, joka on tarkemmin kuvattu luvussa 3.4.2. [1, s.1620]

3.3.4 Laitteistotukipaketti

Laitteistotuki on toteutettu omissa laitteistotukipaketeissa (engl. Board Support Package). Freescale tarjoaa laitteistotuen kaikille valmistamilleen ARM-pohjaisille referenssilaitteistoillensa *Meta-fsl-arm*-kerroksessa. Freescalen referenssilaitteistoissa on tavallisesti Freescalen oman järjestelmäpiirin lisäksi muut toiminnan kannalta välttämättömät osat, kuten muisti ja tallennusmedia, sekä joitain optionaalisia ominaisuuksia, kuten näyttö tai USB-portteja [15]. Kolmansien osapuolien suunnittelemat laitteistot, jotka käyttävät Freescalen ARM-järjestelmäpiiriä laitteistotukeen, tarvitsevat *Meta-fsl-arm-extra*-kerroksen, jossa niille on lisätty laitteistotuki. *Meta-fsl-arm-extra*-kerros riippuu *Meta-fsl-arm*-kerroksesta ja käyttää järjestelmäpiirien konfiguraatiotiedostoja sieltä. *Meta-fsl-arm-extra*-kerros on Yocto-yhteisön ylläpitämä ja läheskään kaikki laitteistovalmistajat, kuten tässä työssä esitelty Kontron, eivät ole tukea sinne lisänneet vaan laitteistotukipaketti jaetaan jotain muuta kautta. Kontronin tapauksessa laitteistotukipaketti on saatavissa omana erillisenä kerroksena rekisteröitymistä vaativalta internet-sivustolta. [1, s.1935]

Laitteistotukipaketti muistuttaa luvussa 3.3.3 kuvattua jakelua siinä mielessä, että

siinä asetetaan koko tuotettavaan Linux-järjestelmään vaikuttavia muuttujia. Esimerkiksi WLAN-tuen tuominen lopulliseen asennukseen vaatii että ominaisuus on konfiguroitu sekä laitteistotukipaketin laitteistokonfiguraatiossa että jakelun konfiguraatiotiedostossa. Jos laitteistotukipaketissa WLAN-ominaisuus on konfiguroitu, mutta jakelussa ei, on tuloksena järjestelmä, jossa ei ole WLAN-ominaisuuden käyttöön tarvittavia ohjelmistoja, kuten yhteyksien hallintaa hoitavaa taustaprosessia, ja ominaisuus ei toimi. Myös toisinpäin: jos jakelun konfiguraatiossa on WLAN-ominaisuus konfiguroitu ja laitteistotukipaketin konfiguraatiossa ei, on tuloksena järjestelmä, jossa WLAN-ominaisuuden käyttöön tarvittavat ohjelmistot, kuten yhteyksien hallinta, löytyy mutta ominaisuus ei toimi. [1, s.1654]

Laitteistotukikerroksia löytyy myös muilta valmistajilta, kuten Inteliltä ja Texas Instrumentsilta. Periaatteessa saman Linux-järjestelmän siirtäminen toiselle alustalle ja jopa eri prosessori-arkkitehtuurille pitäisi olla yksinkertaista.

3.4 BitBaken käyttö

Käännösprosessi hoidetaan käyttäen BitBake-työkalua. Ohjelmistojen lähdekoodi muuttuu Yocto-reseptien mukaisesti kääntämisen jälkeen useiksi erilaisiksi tiedostoiksi, jotka siirretään kehitettävän laitteen pysyväismuistille. Tässä kappaleessa käsitellään, millaisia asioita käännösprosessissa ja sen tuotosten hallinnassa pitää ottaa huomioon.

3.4.1 Tarvittavien tiedostojen noutaminen

Freescale tarjoaa yksinkertaisen tavan noutaa Freescalen alustalla tarvittavat Yocto-reseptit eli Freescalen laitteistotukipaketti Git-versionhallintajärjestelmästä yhdellä komennolla käyttäen Googlen kehittämää Repo-työkalua [25]. Repo-työkalu osaa hakea yhdellä komennolla kaikista tarvittavista repositorioista tietyn ennalta määrätyn Git-haaran. Sama on mahdollista tehdä myös hakemalla manuaalisesti tarvittavat kerrokset eri Git-repositorioista ja saattamalla ne haluttuun tilaan, kuten tiettyyn Yocton julkaisuversioon. Tässä projektissa tarvittavat kerrokset on esitelty tarkemmin luvussa 3.5.1.

3.4.2 Ympäristön valmistelu

Metadatan noutamisen jälkeen luodaan käännöshakemisto käyttäen Freescalen laitteistotukipaketin mukana tulevaa komentosarjaa, jolle annetaan ympäristömuut-

tujalla käytettävä laite. On käytännöllistä käyttää useita eri käännöshakemistoja, jolloin voi tehdä kokeiluja vaikuttamatta toisiin käännöskonfiguraatioihin [1, s.399]. Lisäksi käytäntö osoitti, että käytettäessä koko jakeluun vaikuttavia muuttujia, kuten ikkunoinninhallintajärjestelmän valitsevaa ominaisuutta *DISTRO_FEATURES*-muuttujassa tai eri laitteisto ominaisuuksia *MACHINE_FEATURES*-muuttujassa, on syytä valmistella molemmille konfiguraatioille erilliset käännöshakemistot ongelmien välttämiseksi. Jossain harvinaisissa käännökseen liittyvissä ongelmatilanteissa ratkaisu voi olla luoda uusi käännöshakemisto ja aloittaa puhtaalta pöydältä.

Käännöshakemistokohtainen konfiguraatio jakautuu kahteen tiedostoon: *local.conf* ja *bblayers.conf*. *local.conf*-tiedostossa määritetään koko käännökseen vaikuttavia parametreja, kuten käytettävä laitteisto, käytettävä jakelu ja rinnakkaisten säikeiden määrä käännösprosessissa. *local.conf*-tiedostossa voidaan myös ylikirjoittaa kaikki resepteissä asetetut muuttujat, mutta tätä ominaisuutta ei kannata käyttää muuten kuin kokeilutarkoituksessa ja pysyvät muutokset kannattaa tehdä itse resepteihin, jotka pidetään versionhallintajärjestelmässä [1, s.423]. *bblayers.conf*-tiedostossa määritetään BitBaken käyttämät metadatakerrokset ja niiden sijainnit. [2, s.1402]

3.4.3 Kääntäminen

Käytettäessä useampia rinnakkaisia säikeitä voidaan käännösprosessia nopeuttaa huomattavasti, jos käytössä on moniydinprosessori. Laitteistosta riippuen ensimmäinen täyden jakelun käännös voi kestää yli 10 tuntia, mutta seuraavat käännökset – olettaen, että tehdään muutoksia vain osaan resepteistä – ovat huomattavasti nopeampia. Kiintolevytilan tarve käännöksille on huomattava, ja tilaa kannattaa useampia kokeiluja varten varata moninkertaisesti dokumentaatiossa suositeltuun minimimäärään (50 gigatavua) nähden [40].

Ympäristön valmistelun jälkeen itse käännösprosessin käynnistäminen käy yhdellä komennolla, jossa BitBakelle annetaan käännettävän reseptin nimi. Resepti voi koostaa kokonaisen Linux-jakelun levykuvan tai vain osan siitä, kuten yhden sovelluksen. Lisäksi BitBake tarjoaa laajat mahdollisuudet suorittaa vain osa reseptissä olevista tehtävistä. Esimerkiksi on mahdollista hakea kaikista resepteistä lähdekoodit valmiiksi yhdellä komennolla ja myöhemmin tehdä itse käännös ilman tarvetta verkkoyhteydelle.

BitBake tukee lähdekoodin lataamista eri versionhallintajärjestelmistä, kuten Git, Subversion ja Mercurial. Lisäksi tuettuna on lähdekoodin hakeminen internet-protokollia, kuten http:tä tai ftp:tä, käyttäen ja purkaminen tiedostopaketeista. Usein

resepteihin liittyy myös muutostiedostoja (engl. patch), joita voidaan tarvita siihen, että ladattu lähdekoodi saadaan toimimaan reseptin kohdeympäristössä, kuten tämän alustan tapauksessa Freescalen ARM-prosessorilla. Useissa tapauksissa sulautettua laitetta kehitettäessä on tarve myös itse lisätä resepteihin laitteistokohtaisia muutoksia. Yleisesti muutostiedostojen ei ole tarkoitus olla pysyvä ratkaisu, vaan muutokset pyritään myös saamaan mukaan itse käännettävän sovelluksen versionhallintaan. [6]

BitBake-työkalu käyttää oletuksena luomaansa *Gcc*-ristikääntäjää ja käännösympäristöä käännöksen tekemiseen isäntäkoneella. BitBake käyttää useita eri kansioita käännösprosessin aikana, joista osa on reseptikohtaisia ja osa liittyy koko käännökseen. Kansioissa on useita välituotteita, kuten ladattu lähdekoodi ja ensimmäisenä tuotetut ristikäännöstyökalut. Kääntämisen lisäksi BitBake huolehtii ohjelmien asentamisesta tuotettavaan juuren tiedostojärjestelmään.

3.4.4 Pakettienhallinta

Kääntämisen jälkeen BitBake tukee myös ohjelmien paketointia ja paketoi sovellukset luvussa 3.4.2 esitetyssä *local.conf*-tiedostossa valitulla paketointityökalulla. Yocton tukemia paketointiformaatteja ovat Debian-paketointi, Rpm-paketointi ja Opkg-paketointi. Paketoinnissa tärkeää on pakettien väliset suhteet, joita määritellään käännösaikaisilla ja ajonaikaisilla riippuvuuksilla. Liitteessä A on esimerkki reseptistä, jossa *DEPENDS*- ja *RDEPENDS*-kentät määrittävät vastaavasti reseptin käännösaikaiset ja ajon aikaiset riippuvuudet. Lisäksi Yocto-reseptin sisältämään paketin metadataan sisältyy komentosarjoja, jotka on tarkoitettu ajettavaksi eri vaiheissa paketin elinkaaren aikana: asennettaessa, poistettaessa ja päivitettäessä [14, s.76-81]. Liitteen A reseptissä näitä komentosarjoja ei ole näkyvissä, koska ne ovat määritettyinä perityssä *qmake5*-luokassa.

Pakettienhallintaan olennaisena osana liittyy pakettien versiointi. Yocto-reseptit sisältävät version, jota voidaan käyttää pakettienversiointiin ja näin ollen BitBaken (automaattisesti levykuvan lisäksi) tuottamat ohjelmistopaketit ovat suoraan soveltuvia laitettavaksi pakettivarastoon. Pakettivarasto (engl. package repository) on paikka, johon on tallennettu paketteja. Pakettivarasto voi sijaita millä tahansa medialla esimerkiksi internetissä saatavilla olevalla palvelimella, CD-levyllä tai paikallisessa kansiossa [14, s.102].

3.4.5 Lisenssienhallinta

Linux-pohjainen järjestelmä koostuu tyypillisesti ohjelmistoista, joissa on erityyppisiä lisenssejä käytössä. Useiden avoimen lähdekoodin lisenssien lisäksi käytössä voi olla kaupallisiin lisensseihin perustuvia ohjelmistoja. Kaikissa Yocto-resepteissä on oltava dokumentoituna se, mitä lisenssiä sovellus noudattaa. BitBake ei oletuksena sisällytä käännökseen mitään kaupalliseen lisenssiin pohjautuvaa sovellusta, mutta tätä voidaan kontrolloida luvussa 3.4.2 esitellyssä käännöshakemistokohtaisessa konfiguraatiotiedostossa sallimalla yksittäinen tai kaikki kaupalliset lisenssit. [2, s.2741]; [1, s.1828]

Usein avoimen lähdekoodin projektit määrittelevät lisenssin erillisessä tiedostossa tai lähdekooditiedoston alussa kommenttina. Tästä lisenssin määrittelevästä tekstistä lasketaan Md5-summa, joka sisällytetään reseptiin. Tämän jälkeen, jos erillinen tiedosto tai lähdekooditiedoston alun kommentti muuttuvat, antaa Yocto tästä käännösvirheen, jolloin lisenssin muuttuminen ei jää huomaamatta. [1, s.1814]

Osa avoimen lähdekoodin lisensseistä sisältää vaatimuksen, että ohjelmistoon tehdyt muutokset on julkaistava samalla lisenssillä kuin ohjelmisto alun perin oli tarjolla. Tätä kutsutaan nimellä käyttäjänoikeus (engl. copyleft). Esimerkiksi Linux-kernelissä käytetty GPL-lisenssin versio 2 sisältää vaatimuksen käyttäjänoikeudesta. Yocto tarjoaa mahdollisuuden generoida arkistot lähdekoodimuutoksista, jotka voidaan julkaista tuotteen julkaisun yhteydessä ja näin täyttää käyttäjänoikeuden sisältävien lisenssien vaatimukset. [1, s.1848-1878]

3.5 Käyttö tässä projektissa

Käytettäessä Kontronin laitteistotukipakettia ympäristön valmistelu tehdään samaan tapaan kuin luvussa 3.4.2 kuvatun Freescalen yleisen laitteistotukipaketin kanssa. Tämän lisäksi otetaan käyttöön Kontronin kerros, joka lisää luvussa 3.3.1 kuvatun Kontronin moduulin konfiguraatiotiedoston ja kernel-reseptin sekä joitakin esimerkkireseptejä. Kernel-resepti pohjautuu Freescalen i.MX-prosessoreita varten ylläpidettyyn kernel-haaraan ja lisää siihen luvussa 6.3.1 kuvatuttuja laitteistokohtaisia muutoksia käännöskonfiguraatioon, luvussa 6.3.2 kuvattuja laitteistokohtaisia muutoksia itse kerneliin ja luvussa 5 kuvatut laitteistopuukuvaukset moduulille ja evaluointiemolevylle. Kontronin kernel-resepti on esitetty liitteessä C. Kontronin lisäysten jälkeen laitteistolle voidaan tuottaa levykuva Yoctон esimerkkijakelusta käyttäen BitBakea.

3.5.1 Yocto muutokset

Tuotealustan vaatimukset on kuvattu luvussa 2. Tässä luvussa kuvataan, miten Yocto-resepteissä otetaan käyttöön juuri tällä tuotealustalla käytössä olevat laitteistoresurssit, ja toisaalta kaikki mittaussovelluksen ja sitä tukevien sovellusten vaatimukset. Laitteistosta on olemassa vain yksi versio, joten todennäköisesti laitteistoon liittyvien kernel-reseptin ja laitteistokonfiguraatiotiedoston päivitystarve jäävät pienemmiksi jatkossa, mutta muuhun ohjelmistoalustaan varmasti tulee jatkossakin lisää vaatimuksia ja näin ollen resepteissä on hyvä huomioida laajennettavuus. Kuvassa 3.3 on esitetty tässä projektissa käyttöön valikoituneet metadatakerrokset. Verrattaessa kuvan 3.2 kerroksiin *Meta-yritys*-kerros on kehittäjäkohtainen kerros, kaupallisen toimittajan kerrosta ei ole, *Meta-qt5* ja *Meta-ruby* ovat käyttöliittymätuen kerroksia, *Meta-fsl-arm* on laitteistotuen kerros, *Meta-yocto* on jakelu- eli käyttöjärjestelmäkerros ja *Meta* on yleinen metadata kerros.

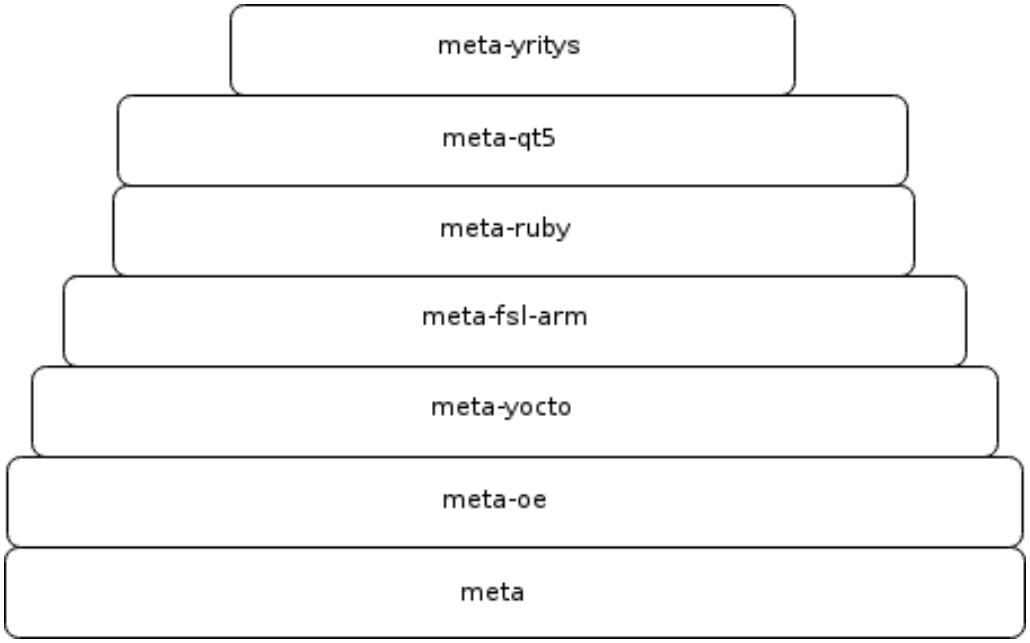
Meta-kerros sisältää OpenEmbedded-projektiin pohjautuvan metadata-pohjan, jonka reseptit ovat käytännössä välttämättömiä Linux-käyttöjärjestelmän toteuttamiseksi. Laitteistotuen osalta tästä kerroksesta käytetään liitteen D reseptissä *linux-dtb.inc*-tiedostoa, joka sisältää yleisen tuen laitteistopuun käytölle Linuxissa. Lisäksi liitteen E laitteistokonfiguraatiotiedostossa käytetään *tune-cortexa9.inc*-tiedostoa, joka sisältää ARM Cortex A9 -arkkitehtuuriin perustuvien prosessorien Linux-tuen. Tämän lisäksi muun kuin kernelin osalta tästä kerroksesta käytetään useita reseptejä, kuten C-standardikirjaston (joka tässä projektissa on *glibc*) tuottavaa reseptiä ja käyttöjärjestelmän käynnistämisestä vastuussa olevan komponentin (joka tässä projektissa on *SystemV*) tuottavaa reseptiä.

Meta-oe lisää yleistä täydentävää, ei käyttöjärjestelmän toiminnan kannalta välttämätöntä metadataa *meta*-kerrokseen. Tässä projektissa tästä kerroksesta on käytössä muun muassa *libsocketcan*-resepti, joka lisää tuen CAN-väylän käytölle unix-socketin avulla. *Meta-yocto* on jakelukerros ja sen sisältämän referenssijakelun runkoa käytetään myös tässä projektissa luodun oman Linux-jakelun pohjana.

Meta-fsl-arm lisää tuen Freescalen referenssialustoille. Tätä kerrosta Freescale ylläpitää yhdessä avoimen lähdekoodin yhteisön kanssa. Laitteistotuen osalta tästä kerroksesta käytetään liitteen D reseptissä *linux-imx.inc*-tiedostoa, joka sisältää yleiset osat Freescalen kernel-haaran konfiguroinnista. Lisäksi liitteen E laitteistokonfiguraatiotiedostossa käytetään *imx-base.inc*-tiedostoa, joka sisältää i.MX-prosessorien yleiset laitteiston ominaisuudet. Tämän lisäksi muun kuin kernelin osalta ohjelmistoalustalla käytetään *Meta-fsl-arm*-kerroksesta useita paketteja, kuten Freescalen X11-ikkunointijärjestelmään soveltuvia näytönohjaimen ajureita.

Meta-ruby-kerros, joka lisää tuen Ruby-ohjelmointikielelle, on mukana, koska se on määritetty riippuvuudeksi *Meta-qt5*-kerrokselle. *Meta-ruby*-kerrosta ei tämän projektin resepteissä mihinkään tarvita, mutta se on mukana koska jotkin reseptit, kuten *QtWebkit*-selainkirjaston resepti *Meta-qt5*-kerroksessa riippuu siitä ja BitBake kaikkia reseptejä parsiessaan vaatii kaikki riippuvuudet, vaikkei reseptejä käytettäisikään. *Meta-qt5* kerros sisältää reseptit Qt5-käyttöliittymäkirjaston käyttämiseen. Tästä kerroksesta on käytössä muun muassa *qtbase*-resepti, joka sisältää ohjeet Qt:n peruskirjaston tuottamiseen.

Meta-yritys-kerros on luotu tässä projektissa Kontronin tarjoaman kerroksen pohjalta pelkästään tätä laitteistoalustaa varten. Tässä projektissa liitteen D kernel-resepti ja liitteen E laitteistokonfiguraatiotiedosto siis sijaitsetvat tässä kerroksessa. Lisäksi tämä kerros sisältää muun muassa *tmr1443-smx6-image-sato*-reseptin, joka listaa kaikki käytettävät paketit ja mahdollistaa näin yhdellä BitBake-komennolla koko käyttöjärjestelmän sisältävän levykuvan tuottamisen. *tmr1443-smx6-image-sato*-resepti on esitetty liitteessä F. Reseptissä on toistaiseksi mukana monia paketteja testausta varten, ja tällaista levykuvaa ei tulla käyttämään tuotantojärjestelmässä.



Kuva 3.3 *Metadata-kerrosten jakautuminen tässä projektissa.*

3.5.2 Laitteistotuki

Laitteistotuen osalta Kontronin tarjoamaan Yocton laitteistotukikerrokseen tarvittavien muutosten osuus tässä projektissa jäi suhteellisen pieneksi. Liitteessä C on

esitetty Kontronin kernel-resepti, jonka avulla Yoctolla voi tuottaa toimivan kernelin Kontronin moduulille ja evaluointiemolevylle. Liitteessä D on esitetty kernel-resepti, jossa Kontronin reseptiä on muokattu niin, että sillä voidaan tuottaa kernel lopulliselle emolevylle. Suurin osa työstä on siis muokata itse kerneliä sekä laitteistopuuta luvuissa 6 ja 5 kuvatuilla tavoilla niin, että laitteistotuki saavutetaan. Yocto-reseptissä ainoastaan dokumentoidaan tarvittavat muutokset käytettävien muutostiedostojen muodossa. Tässä projektissa tuotetut tai muualta käyttöön otetut muutostiedostot on numeroitu 0049-0074 liitteessä D. Kernel-reseptin sisällyttämistä muutostiedostoista osa on täysin laitteistokohtaisia, joten jos esimerkiksi tulevaisuudessa vaihdetaan moduuli yhden i.MX6-ytimen versiosta neliytimiseen i.MX6-malliin, niin nämä muutostiedostot pitää käydä läpi ja varmistaa, että kernel toimii myös neliytimisellä prosessorilla. Jatkossa, jos pidetään yllä useampia laitteistoversioita, voi olla järkevää eriyttää laitteistokohtaiset muutokset laitteistokohtaisiin resepteihin.

Liitteessä E on Kontronin laitteistokonfiguraatiotiedoston pohjalta muokattu laitteistokonfiguraatiotiedosto lopulliselle emolevylle. Laitteistokonfiguraatiotiedoston muutoksia olivat käytetty sarjaportti, *MACHINE_FEATURES*-muuttujassa listatut ominaisuudet, käytettävä laitteistopuu ja käytettävän tiedostojärjestelmän tyyppi.

Kernel-tuen lisäksi laitteisto vaatii käyttöjärjestelmän käynnistyksessä joitakin alustustoimenpiteitä, joita ei tehdä luvussa 4 esitellyssä käynnistyslataajassa. Tällaisia muutoksia ovat 3G/GPS- ja Bluetooth/WLAN -moduulien alustukset. Taulukossa 3.1 on listattu laitteistotukeen liittyvät Yocto-reseptit ja konfiguraatiotiedostot kuvauksineen.

Taulukko 3.1 *Listaus laitteistotukea varten luoduista ja muokatuista Yocto-resepteistä ja konfiguraatiotiedostoista.*

Tiedosto	**Kuvaus**
smarc-samx6i.conf	Laitteistokonfiguraatiotiedosto, liite E
linux-smx6_3.14.28.bb	Kernel-resepti, liite D
tmr1443hwinit_0.1.bb	Lisää SystemV-käynnistyskomentosarjan juuren tiedostojärjestelmään, joka alustaa 3G/GPS- ja Bluetooth/WLAN-moduulit

3.5.3 Ohjelmistoalusta

Ohjelmistoalusta koostuu kirjastoista ja Linuxin käyttäjätilan sovelluksista, joita tarvitaan itse mittaussovelluksen lisäksi. Yocto-reseptejä on saatavilla todella laa-

jasti erilaisten sulautettuihin laitteisiin tarkoitettujen Linuxin kirjastojen ja käyttäjätilan sovellusten tuottamiseen, joten tässä projektissa yhtäkään reseptiä ei tarvinnut luoda tyhjästä. Tärkeimpiä näistä ovat Qt-grafiikkakirjasto ja yksinkertainen X11-ikkunointiin perustuva *Matchbox*-ikkunointijärjestelmä. *Matchbox*-ikkunointijärjestelmä mahdollistaa mittaussovelluksen rinnalle asennettavat sovellukset, mikä oli yksi luvussa 1.2 määritetyistä keskeisistä tavoitteista. Tässä projektissa testattiin myös Wayland-pohjaista ikkunoinnin hallintaa Weston-nimisellä referenssikompositoijalla, mutta se todettiin vielä ominaisuuksiltaan puutteelliseksi. Suurin yksittäinen puute oli internetin yli toimivan etädiagnostiikan mahdollistavan sovelluksen puuttuminen. X11-ympäristössä *X11vnc*-niminen sovellus mahdollisti etädiagnostiikan käyttämisen. Virtuaaliseksi näppäimistöksi valikoitui Qt-pohjaiseen *Maliit*-nimiseen sovelluskehykseen pohjautuva WebOS-käyttöjärjestelmän näppäimistö. *Maliit* on käytössä myös Ubuntu Phone- ja Jollan SailfishOS-käyttöjärjestelmissä, joten se soveltuu useampia merkistöjä tukevan virtuaalinäppäimistön toteuttamiseen. Taulukossa 3.2 on listattu ohjelmistoalustaan liittyvät Yocto-reseptit ja konfiguraatiotiedostot, jotka muodostavat *Meta-yritys*-kerroksen. *Meta-yritys*-kerroksen reseptit ovat ainoat, joihin on tehty muutoksia verrattuna Yocto-repositorioissa saatavilla oleviin resepteihin.

Pakettienhallinnassa Yocton oletuspaketointiformaatti on Rpm-paketointi. Tämän projektin puitteissa testattiin myös Debian-paketointia, mutta se johti Yocton sähköpostilistallakin useampaan kertaan esiintyneisiin virhetilanteisiin, joten nähtiin järkeväksi käyttää Rpm-paketointia [41]; [42]. Internetiin saataville asetettavan pakettivaraston käyttäminen BitBaken tuottamien Rpm-pakettien jakamiseen asiakkaiden laitteisiin täyttää vaatimuksen internetin yli tapahtuvasta alustan – ja myöhemmin myös mittaussovelluksen – päivityksestä, joka oli määritetty yhdeksi luvussa 1.2 esitetyistä keskeisistä tavoitteista.

3.5.4 Ristikäännös- ja etävirheenetsintäympäristö PC:lle

Yocton *Meta-qt5*-kerros tarjoaa valmiin yleiskäyttöisen *Meta-toolchain-qt5*-reseptin sovelluskehitys-työkalupaketin (engl. Software Development Kit) luomiseen Qt5-sovelluksen kehittämiseen ja etävirheenetsintään. Näin tarvittavat Qmake, ristikääntäjä, käännösympäristön tiedostojärjestelmä ja muut työkalut, kuten virheenetsintään käytettävä *Gdb*, voidaan asentaa kehitystyössä käytettävälle tietokoneelle ja lisätä *Qt-Creator*-sovelluskehitysympäristön asetuksiin. Tämän jälkeen varsinaisen alustalla käytettävän mittaussovelluksen kehittäminen on mahdollista hoitaa suoraan käyttäen pelkästään *Qt-Creator*-sovelluskehitysympäristöä. [1, s.1198-1214]

Meta-toolchain-qt5-reseptillä luotu ristikäännösympäristö, jonka käännösaikaisissa

Taulukko 3.2 *Listaus ohjelmistoalustaa varten luoduista ja muokatuista Yocto-resepteistä ja konfiguraatiotiedostoista.*

Tiedosto	**Kuvaus**
yritysyocto.conf	Periytyy Yocton oletusjakelusta Pokysta ja määrittelee käyttöjärjestelmän ominaisuuksia, kuten käytettävän ikkunointijärjestelmän ja grafiikkarajapinnan
layer.conf	Määrittelee Meta-yritys-kerroksen ominaisuudet, kuten prioriteetin, joka on tällä kerroksella korkein ja mahdollistaa muiden kerrosten määritysten ylikirjoituksen
tmr1443-smx6-image-sato.bb	Resepti X11-pohjaisen ikkunointiympäristön sisältävän käyttöjärjestelmän levykuvan tuottamiseen, liite F
ppp_2.4.7.bbappend	Lisää chat-komentosarjat 3G-yhteyden testaamiseen
psplash_git.bbappend	Ylikirjoittaa käynnistyksessä näytettävän Yocto-logon yrityksen logolla
maliit-framework-qt5_-0.99.1.bb	Konfiguroi virtuaalinäppäimistön sovelluskehyksen
qtbase_git.bbappend	Asettaa Qt-käännöskonfiguraation vastaamaan mittaussovelluksen vaatimuksia
matchbox-session-sato_-0.1.bbappend	Ylikirjoittaa ikkunointiympäristön käynnistyskonfiguraation: piilottaa hiiriosoittimen, kääntää näytön sekä kosketuspaneelin pystyasentoon ja asettaa ympäristömuuttujia
webos-keyboard_git.bb	Resepti WebOS-virtuaalinäppäimistön tuottamiseen, liite A

paketeissa on käytössä vain Qt:n kirjastot riippuvuuksineen ja työkaluineen, toimi projektin alussa riittävänä ristikäännösympäristönä Qt5-mittaussovelluksen kehityksessä. Projektin edetessä koko käyttöjärjestelmän sisältävän levykuvan tuottavaan (liitteessä F esitettyyn) reseptiin lisättiin periytyvyys Yocton yleisestä ja Qt:n käännösympäristön kuvaavista luokista. Tämän seurauksena levykuvaa tuotetteassa luodaan aina myös vastaava ristikäännösympäristö, joka sisältää täysin laitteelle asennettavaa ympäristöä vastaavan juuren tiedostojärjestelmän. Näin voidaan varmistua siitä, että mittaussovellusta käännettäessä on käytettävissä kaikki käytössä olevien reseptien *DEPENDS*-muuttujissa määritellyt käännösaikaiset riippuvuudet, jotka myös Yocto reseptejä käännettäessä ovat käytössä.

Ristikäännösympäristöä käytettäessä on huomattava, että ennen *Qt-Creatorin* käynnistämistä on suoritettava ristikäännösympäristöön sisältyvä komentosarja, joka asettaa tarvittavat ympäristömuuttujat. Tämän jälkeen *Qt-Creator* on käynnistettävä samasta terminaalista, jossa komentosarja ajettiin. [1, s.1214-1242]

4. KÄYNNISTYSLATAAJA U-BOOT

Käynnistyslataaja on ohjelmiston osa, joka käynnistyy ensimmäisenä, kun laitteistoon kytketään virta. Tällä laitteistolla laitteistovalmistaja tarjoaa *U-boot*-nimisen käynnistyslataajan. Intelin X86-arkkitehtuuriin perustuvissa PC-tietokoneissa käynnistyslataajana on yleisimmin *BIOS* ja Linux-käytössä lisäksi *GRUB*, joka vain lataa käyttöjärjestelmän levykuvan. *U-boot* eroaa kuitenkin *BIOS*:ista siinä, että se on toiminnassa ainoastaan laitteistoa käynnistettäessä, kun taas *BIOS* on toiminnassa myös käyttöjärjestelmän ollessa käynnissä tarjoten sille matalan tason laitteistorajapinnan. *GRUB*:iin verrattuna taas *U-boot* toteuttaa huomattavasti suuremman toiminnallisuuden laitteiston alustuksessa, koska *U-boot* ei saa tukea *BIOS*:ilta matalan tason laitteiston alustuksiin. Käynnistyslataajia on kaiken kaikkiaan olemassa useita, mutta *U-boot* on monipuolisin, joustavin ja aktiivisimmin kehitetty avoimen lähdekoodin käynnistyslataaja sulautetuille laitteille [22, s.6993]. [20, s.225]

4.1 Historia

U-boot on alun perin Magnus Dammin kehittämä käynnistyslataaja PowerPC-arkkitehtuurille, jonka alkuperäinen nimi oli *8xxROM*. Tämän jälkeen mukaan on tullut lisää kehittäjiä ja kehitys jatkui *PPCBoot*-nimellä. *PPCBoot*ista haarautui myös versio ARM-arkkitehtuurille nimeltään *ARMBoot*. Nimeksi vaihtui *U-boot*, kun *PPCBoot* ja *ARMBoot* yhdistyivät yhdeksi useampaa arkkitehtuuria tukevaksi käynnistyslataajaksi. Nykyään *U-boot* tukee 15:tä eri prosessoriarkkitehtuuria ja yli 400:ä erilaista kehitysalustaa. *U-boot*:in kehitys on läheisesti sidoksissa Linuxin kernelin kehitykseen, ja osa käytetyistä ohjelmakoodin otsikkotiedostoista on jaettu Linuxin kanssa. *U-boot* käy myös muiden käyttöjärjestelmien tai pelkkien laitteistolla suoraan ajettavien sovellusten käynnistämiseen, mutta kehityksessä on kiinnitetty erityistä huomiota Linux-käyttöjärjestelmien käynnistämiseen. [20, s.225]; [21]

4.2 Ominaisuudet

U-boot:in komennot kirjoitetaan sarjaporttiin ja vastaukset luetaan oletuksena samasta sarjaportista. *U-boot* on myös mahdollista asettaa USB-sarjaportti tilaan,

jolloin kytkettäessä USB-kaapeli se näkyy isäntäkoneelle USB-sarjaporttilaitteena. *U-boot*:in hallinta perustuu ympäristömuuttujien ja komentosarjakielen käyttöön. Ympäristömuuttujat ja komentosarjat tallennetaan erillisellä komennolla pysyväismuistiin esimerkiksi samalle medialle, jossa *U-boot*:in binäärinen konekoodi sijaitsee. Muuttamalla ympäristömuuttujia tai ympäristömuuttujiin tallennettuja komentosarjoja voidaan vaikuttaa *U-boot*:in toimintaan hyvin monipuolisesti. Kontronin toimittaman *U-boot*:in ympäristömuuttujien oletusarvot on listattu liitteessä G. Esimerkiksi liitteen G *bootcmd*-komentosarja kokeilee, että löytyykö Linux-kerneliä, laitteistopuuta ja juuren tiedostojärjestelmää USB-muistilta, sisäiseltä flash-muistilta tai muistikortilta, ja löydettäessä käynnistää käyttöjärjestelmän. Joidenkin suurempien muutosten, kuten laitteistotason muutosten, tekemiseen vaaditaan kuitenkin muutoksia *U-boot*:in lähdekoodiin.

U-boot tukee oletuksena laitteistopuun (engl. device tree) käyttöä. Laitteistopuu voi sijaita samalla tai eri medialla kuin varsinainen Linux-kernel. *U-boot* sisältää tuen myös laitteistopuun lukemiseen ja muokkaamiseen. *U-boot* tekee oletuksena joitakin muokkauksia laitteistopuuhun ennen sen antamista Linux-kernelin käyttöön. Tällaisia muokkauksia voivat olla esimerkiksi ympäristömuuttujasta saadun laitteen sarjanumeron ja *U-boot*-version lisääminen. Laitteistopuun käyttöä on selitetty tarkemmin luvussa 5. [29, s.90-91]

Laitteiston käynnistäminen siten, että Linux-kernel, laitteistopuu ja juuren tiedostojärjestelmä ovat kehittäjän PC:llä ja että ne ladataan lähiverkon kautta, on ideaali alkuvaiheen kehitysympäristö [22, s.1449]. Tällä keinolla iteraatiot ovat nopeita johtuen siitä, että Linux-kerneliä, laitteistopuuta ja juuren tiedostojärjestelmää ei tarvitse kirjoittaa muille medioille kuin kehityksessä käytettävän Linux-isäntäkoneen kiintolevylle. *U-boot* tukee Linux-kernelin ja laitteistopuun lataamista lähiverkon kautta käyttäen Tftp-tiedonsiirtoa ja juuren tiedostojärjestelmän lataamista lähiverkon kautta NFS-protokollaa käyttäen. Lisäksi on mahdollista käyttää DHCP- tai BOOTP-protokollaa Tftp- ja NFS-palvelinten verkko-osoitteiden selvittämiseen. Verkon yli käynnistäminen ei yleensä ole mahdollista tuotannossa olevissa järjestelmissä. [22, s.6993]

Verkon yli käynnistämisen lisäksi *U-boot* tukee monen tyyppisiä eri medioita, joissa Linux-kernel, laitteistopuu ja juuren tiedostojärjestelmä voivat sijaita. Juuren tiedostojärjestelmä, Linux-kernel ja laitteistopuu voivat sijaita samalla medialla tai eri medioilla. Medialla käytettävät tiedostot sijaitsevat tiedostojärjestelmässä ja *U-boot* tukee laajasti eri tyyppisiä tiedostojärjestelmiä. [22, s.7004]; [21]

Käynnistysmedioista erillisenä ominaisuutena *U-boot* tukee USB OTG -tilaa, jossa

apuohjelman avulla itse käynnistyslataaja sijaitsee USB-väylän päässä isäntäkoneen tiedostojärjestelmässä. Tässä tilassa *U-boot* ladataan suoraan USB-väylän kautta käytettävän laitteen RAM-muistiin, josta se käynnistetään. Kun *U-boot* ensin käynnistetään RAM-muistista, niin sitä käyttäen voidaan ladata uusi *U-boot*-binääri lähiverkon yli Tftp-palvelimelta ja kirjoittaa se laitteen sisäiseen muistiin edellisen version päälle. Tämä mahdollistaa toipumisen itse käynnistyslataajan vioittumisesta. Aiemmin sulautetuissa järjestelmissä tähän on käytetty erillistä BDM- tai JTAG-rajapintaa hyödyntävää erillistä laitetta, jolla voidaan kirjoittaa suoraan isäntälaitteelta kehitettävän laitteen muistiin [22, s.7227]. Myös jotkin lisälaitteet, kuten I2C-liitäntää hyödyntävät yksinkertaiset piirit, USB-muistit, USB-näppäimistöt, USB-Ethernet muuntimet ja näytöt, ovat tuettuina. USB-Ethernet muunnin mahdollistaa verkon kautta käynnistämisen, vaikka laitteella ei ole fyysistä Ethernet-porttia. Näytöllä ei esitetä käyttöliittymää, mutta tyypillisesti näyttötukea voidaan käyttää esittämään yrityksen tai tuotteen logo näytöllä laitetta käynnistettäessä. Mielestäni näytön käyttö myös parantaa laitteen käytettävyyttä loppukäyttäjän näkökulmasta, koska välittömästi virran kytkemisen jälkeen *U-boot* esittää grafiikkaa näytöllä, josta käyttäjä voi päätellä, että laitteeseen on kytkeytynyt virta. [21]

4.3 Käyttö tässä projektissa

U-boot on laitteistovalmistaja Kontronin toimesta asennettuna uusissa SMARC-sAMX6i Solo moduuleissa SPI-väylällä yhdistetyllä flash-muistilla, ja näin ollen uuden moduulin käyttöönottoon evaluointiemolevyn kanssa ei välttämättä vaadita mitään ympäristömuuttuja- tai lähdekoodimuutoksia *U-boot*:iin. Kontron on toteuttanut *U-boot*:in yleisesti saatavilla olevan version 2015.04 pohjalta, johon on lisätty tuki Kontronin laitteistoille.

Verkon yli käynnistäminen käyttäen kehityskoneelle asennettuja Tftp- ja NFS-palvelinta osoittautui erittäin käyttökelpoiseksi tavaksi iteroida Linux-kernelin, laitteistopuun ja juuren tiedostojärjestelmän kehitystä. Tftp- ja NFS-palvelinten käynnistäminen ohjelmistokehityksessä käytetylle Ubuntu 14.04 LTS Linux -jakelulle oli hyvin dokumentoitua ja suoraviivaista toteuttaa. DHCP tai BOOTP protokollien käyttöönotto jäi tekemättä, koska ne olisivat vaatineet muutoksia yrityksen sisäverkkoon. Tämän sijaan moduuleille ja kehitysympäristön sisältäneelle Tftp- ja NFS-palvelinkoneelle käytettiin kiinteitä IP-osoitteita, jotka asetettiin manuaalisesti *U-boot*:in ympäristömuuttujiin.

Ympäristömuuttujien arvot ovat tallennettuina saman SPI-flash-muistin osion loppuun, missä itse käynnistyslataajankin suoritettava binääri-tiedosto on. Tämä mahdollistaa myös *U-boot*:in päivittämisen ilman että tallennettujen ympäristömuuttu-

jien arvot palautuvat oletusarvoihin. Tyypillisessä käyttötapauksessa käyttäjä muuttaa kerran *U-boot*:in *bootcmd*-muuttujaa ja tallentaa muutetun arvon pysyvästi SPI-flash-muistiin siten, että käyttöjärjestelmä käynnistyy automaattisesti. [4]

U-boot:in ympäristömuuttujat voidaan myös lukea tekstitiedostosta ja alustaa kaikki kerralla itse *U-boot*:ia käyttäen. Tekstitiedosto voidaan lukea verkon yli Tftp-palvelimelta tai USB-muistilta. *U-boot*:in ympäristömuuttujien sijaitessa *U-boot*:in binäärin kanssa samalla SPI-flash-muistilla voidaan myös ympäristömuuttujien oletusarvoista generoida binääri, joka kirjoitetaan SPI-flash-muistille. Tähän on olemassa *U-boot*:in mukana tuleva *mkenvimage*-työkalu, joka luo binäärin tekstimuotoisen konfiguraatiotiedoston pohjalta [21]. Jos tuotannossa käytettävä *U-boot*-ympäristö kirjoitetaan jo moduulin valmistusvaiheessa SPI-flash-muistille, niin kokoonpanovaiheessa moduuleita lopulliseen emolevyyn yhdistettäessä voidaan välttyä ylimääräiseltä työvaiheelta, jossa sarjaportin kautta konfiguroitaisiin ympäristömuuttujat oikeisiin arvoihinsa.

U-boot-binäärin ja ympäristömuuttujien arvojen lisäksi myös käynnistyksessä näytettävä logo on mahdollista kirjoittaa SPI-flash-muistiin, mistä *U-boot* osaa sen lukea. Tuettuja formaatteja on rajallisesti ja myös tiedoston muoto on tärkeää olla oikein, että kuva saadaan näkymään värillisenä ja oikein. Myös kuva voidaan ladata Ttft-palvelimelta tai USB-muistilta.

4.4 U-bootin muokkaaminen laitteistolle sopivaksi

Tässä projektissa samanaikaisesti suunniteltu lopullinen emolevy, joka esiteltiin tarkemmin luvussa 2.1 sisältää myös sellaisia laitteistotason muutoksia Kontronin evaluointiemolevyyn verrattuna, että Kontronin toimittaman *U-boot*:in käyttö sellaisenaan ei ole mahdollista. Lopullisella emolevyllä *U-boot*:in kommunikointiin käyttämä sarjaportti poikkeaa evaluointiemolevyllä käytetystä sarjaportista. Sarjaportin vaihto vaati uusien pinnien konfiguroinnin ja alustamisen i.MX6-manuaalin [33] mukaisesti ja konfiguraation muokkaamisen. Lisäksi lopullinen emolevy ei sisällä Ethernet-porttia, mutta *U-boot* tukee USB-Ethernet-adaptereita. USB-Ethernet-adaptereiden tuki ei ole kuitenkaan oletuksena käytössä. USB-Ethernet-adapteri saatiin käyttöön yksinkertaisesti lisäämällä konfiguraation määritykset USB-Ethernet-adapterille. USB-Ethernet-adapterin valinnassa oli olennaista, että sen laitetunnus oli valmiiksi tuettuna *U-boot*:issa. USB-Ethernet-adapteriksi valikoitui hyvän saatavuuden ja tuen vuoksi Asixin AX88772-piirisarjaan pohjautuva Apple MC704ZM/A.

Laitteiston liittämiseen tehtävien muutosten lisäksi *U-boot* mahdollistaa oheislaitteiden yleiskäyttöisten (engl. general purpose input/output) pinnien alustamisen

oikeisiin tiloihinsa. Tässä projektissa Kontronin toteuttamien oletusalustusten lisäksi *U-boot*:in toiminnallisuuteen lisättiin näytön taustavalon ohjaamiseen käytettävän jännitteenmuuntajan ohjaus päälle. Tämän johdosta näyttö saadaan heti päälle, kun laitteen virtapainiketta painetaan. Lisäksi lisälaitteiden yksinkertaiset alustukset, kuten kosketusnäytön kosketussensorin, 3G/GLONASS-moduulin ja WLAN/Bluetooth-moduulin päälle kytkeminen, toteutettiin *U-boot*:in alustuksessa. Näin voidaan olla varmoja siitä, että kun Linux-kernel käynnistyessään lataa laitteistoajureita, niin moduuleissa on virrat kytkettynä.

Näiden tarvittavien ominaisuuksien johdosta *U-boot*:in lähdekoodiin täytyy tehdä muutoksia. Kontronin *U-boot*:in Yocto-tuen puuttuessa myös erillinen ristikäännösympäristö tarvitaan *U-boot*:in kääntämiseen. Ristikäännösympäristöksi soveltui Kontronin suosittelema Ubuntun pakettivarastosta saatavilla oleva *gcc*-kääntäjä. Jatkossa, kun projektissa siirrytään tuotantovaiheeseen, valmiiksi muokattu *U-boot*-binääri tullaan kirjoittamaan jo laitteistovalmistaja Kontronin tuotantolinjalla moduulin muistiin, jolloin vältytään ylimääräiseltä työltä kokoonpanovaiheessa.

4.5 U-boot ympäristön muokkaaminen laitteistolle sopivaksi

U-boot:in binäärin lisäksi myös liitteessä G esitettyyn *U-boot*-ympäristöön tarvittiin muutoksia laitteistoa varten. Tärkeimmät muutokset liittyvät Linux-kernelin saamiin komentoriviparametreihin. Nämä komentoriviparametrit määrittävät muun muassa käytettävän sarjaportin, massamuistin sekä Linux-kernelin ja laitteistopuun tiedostonimet. Lähiverkon yli tai USB-muistilta käynnistäminen vaatii vastaavat muutokset *U-boot*-ympäristömuuttujiin. Vaikka *U-boot* tukee myös komentosarjoja, joilla voidaan esimerkiksi käynnistää ehdollisesti joko verkon yli tai flash-muistilta, niin tämän projektin puitteissa osoittautui yksinkertaisemmaksi vaihtaa muuttujien arvoja manuaalisesti käynnistystavan vaihtuessa melko harvoin. Yhtenä huomiona USB-muistia käytettäessä on tärkeää asettaa kernelin komentoriviparametreihin sarjaportin ja juuren tiedostojärjestelmän lisäksi myös optio *rootwait*, joka saa kernelin odottamaan tiedostojärjestelmän lukemista hitaasta muistista. Kernelin komentoriviparametreilla voidaan myös ylikirjoittaa joitakin laitteistopuussa tai kernelin käännöksessä määritettyjä parametreja. Tämä on nopea tapa testata esimerkiksi erilaisia HDMI- ja LVDS-näyttöjä ilman että tarvitsee tehdä muutoksia kernel- tai laitteistopuu-binääreihin.

Tässä projektissa ympäristömuuttujien oletusarvojen tuonti testattiin käyttäen tekstitiedostoa. Tuotantovaiheeseen siirryttäessä on tarkoitus, että muokatun *U-boot*-binäärin lisäksi myös *U-boot*-ympäristö asetetaan oikeaksi jo laitteistovalmistaja Kontronin tuotantolinjalla. Toisin kuin itse *U-boot*-binääristä, ympäristöstä ei voida

käyttää yhtä versiota kaikille moduuleille, koska ympäristö sisältää myös moduulikohtaisen sarjanumeron ja MAC-osoitteen.

5. LAITTEISTOPUU

Laitteistopuu (engl. device tree) on tietorakenne, joka kuvaa käytettävän järjestelmän fyysisen laitteiston ominaisuudet. Laitteistopuun käyttö mahdollistaa sen ettei laitteiston ominaisuuksia tarvitse kovakoodata kernelin lähdekoodiin. Laitteistopuu on ajonaikainen tietorakenne, jota käynnistyslataaja ja laite-ohjelmistot voivat muokata ja jota Linux-kernel lukee. Laitteistopuusta käytetään useita eri nimiä ja lyhenteitä eri lähteissä, jotka kaikki viittaavat saman tyyppiseen tietorakenteeseen. Yleisimmin käytettyjä nimiä ovat U-bootin dokumentaatiossa käytetty litistetty laitteistopuu möykky (engl. flattened device tree blob) ja Linuxin dokumentaatiossa käytetty avoimen laiteohjelmiston laitteistopuu (engl. Open Firmware Device Tree). Näistä myös lyhyemmät versiot, joissa esiintyy vain osa sanoista ja kirjainlyhennelmät, kuten FDT, OF, OFW ja DT, ovat myös käytössä. [29, s.90-91]; [16]

5.1 Historia

Laitteistopuu on alun perin osa OpenBoot-laiteohjelmistoa (engl. firmware), jonka kehitys alkoi Sun Microsystemsillä vuonna 1988. Sun kehitti samaan aikaan tietokoneita, jotka perustuivat kolmeen eri prosessorityyppiin, ja tästä syntyi tarve kehittää prosessorin käskykannasta riippumaton laiteohjelmisto. OpenBoot-versio 1 esiteltiin ensimmäistä kertaa Sun SPARCstation 1 tietokoneen julkaisun yhteydessä [35, s.1]. SPARCstation 1 tuli myyntiin vuonna 1989, ja siinä oli käytössä 20 megahertsin kellotaajuudella toiminut RISC-arkkitehtuurin perustuva prosessori [36]. OpenBoot-versio 2 julkaistiin SPARCstation 2 tietokoneen julkaisun yhteydessä. IEEE:n standardi 1275 käynnistyksen laiteohjelmistojen tärkeimmistä vaatimuksista ja käytännöistä perustuu OpenBootin versioon 2. Standardi on laaja ja suunnattu yleiskäyttöisiin tietokoneisiin. Tämän johdosta se määrittelee myös monia asioita, jotka eivät sulautetussa järjestelmässä ole relevantteja, kuten erilliseltä ulkoiselta laitteelta ladattavat FCode-kieliset laitteistoajurit ja Forth-kieleen pohjautuva ohjelmoitava käyttöliittymä. Tämän johdosta esimerkiksi vuonna 2011 julkaistu Power.org:in standardin versio 1.1 sulautetun PowerPC-arkkitehtuurin alustavaatimuksista [30] hyödyntääkin IEEE-1275 standardista ainoastaan laitteistopuun konseptin. [30, s.10]; [43]

SPARC-arkkitehtuurin jälkeen laitteistopuu on siis valikoitunut käyttöön myöskin RISC-prosessoriarkkitehtuuriin perustuvissa PowerPC-prosessoreissa ja Linux-kernelin osalta vuodesta 2005 eteenpäin on vaadittu, että kaikkien PowerPC-prosessoriin perustuvien alustojen pitää käyttää laitteistopuuta [16]. PowerPC:n jälkeen kaikki muutkin Linuxin tukemat arkkitehtuurit, kuten tässä työssä käytetty PowerPC:n kanssa myöskin RISC-prosessoriarkkitehtuuriin perustuva ARM-arkkitehtuuri, ovat siirtyneet käyttämään oletuksena laitteistopuuta [2, s.2337]. Ainoan poikkeuksen tekee X86-arkkitehtuuri. Intel huomasi vuonna 1997 uutta Itanium-arkkitehtuuria kehittäessään, että BIOS on liian vanha käynnistyslataaja uudelle arkkitehtuurille. Intel arvioi IEEE-1275-standardin käynnistyslataajan teknisesti edistykselliseksi, mutta tiukassa aikataulussa sen käyttöönotto ei kaupallisista syistä ollut järkevää, vaan päädyttiin kehittämään oma UEFI-käynnistyslataaja, joka nykyisin on käytössä myös yleisesti X86-arkkitehtuurilla [31]. Myös X86-arkkitehtuurille löytyy jonkinasteinen laitteistopuutuki Linuxista [16], mutta arvioisin sen olevan vähemmän käytetty ja huonommin testattu kuin muilla arkkitehtuureilla – Intelin tukiessa oletuksena vain UEFI:a myös sulautetuilla kehitysalustoilla, kuten MinnowBoard MAX:lla [44]. [30]

5.2 Ominaisuudet

Laitteistopuu (engl. device tree) määrittely kuvaa laitteiden ominaisuudet kernelistä erillään sijaitsevassa binääritiedostossa. Nimensä mukaisesti kyseessä on asyklinen puumainen tietorakenne, jossa listataan solmuja (engl. node) ja niiden nimettyjä ominaisuuksia. Solmuilla voi olla mielivaltainen määrä nimettyjä ominaisuuksia, jotka sisältävät mielivaltaista dataa. Tämän lisäksi voidaan tehdä mielivaltaisia viittauksia, jotka eivät noudata puurakennetta. [16]; [30, s.14]

Linux-kernel lukee laitteiston fyysiset ominaisuudet laitteistopuusta, ja näin se osaa ohjata oikein kyseistä laitteistoversiota. Tällaisia ovat esimerkiksi käytettävä I2C-väylän nopeus ja USB-väylä, johon 3G-moduuli on kytketty. [2, s.2337]

Laitteistopuun käyttö mahdollistaa sen, että samaa Linux-kerneliä voidaan käyttää sekä lopputuotteessa että kehitysalustalla vaihtamalla ainoastaan käytettävää laitteistopuuta. Ennen laitteistopuun yleistymistä samat alustakohtaiset asiat on määritelty kovakoodattuna kernelin laitetiedostossa. Tämän lisäksi käynnistyslataaja, joka sulautetuissa laitteissa on tyypillisesti U-boot, on ollut vastuussa siitä, että Linux-kernenille välitetään oikea laitteiston tunnistenumero. Laitteistopuun tapauksessa käynnistyslataaja antaa Linux-kernelille parametrina ainoastaan käytettävän laitteistopuun. [2, s.2337]

5.3 Käyttö tässä projektissa

Laitteistovalmistaja Kontron tarjoaa laitteistopuun, jossa on määritetty SMARC-moduulilla olevat komponentit ominaisuuksineen sekä luvussa 2.1.2 esitellyn kehitysalustan komponentit. Lopputuotteessa evaluointiemolevy vaihdetaan itse suunniteltuun emolevyyn, minkä vuoksi laitteistopuuhun määriteltiin kaikki itse suunnitellun emolevyn komponentit. Evaluointiemolevyllä oli myös sellaisia komponentteja, joita ei ole itse suunnitellulla emolevyllä, joten osittain laitteistopuusta jouduttiin myös poistamaan solmuja. Yhteensopivuuserot olivat lisälaitteissa, joten molemmilla emolevyillä käyttöjärjestelmän käynnistäminen onnistuu myös toisen emolevyn laitteistopuuta käyttäen. Väärän laitteistopuun käytön havaitsee helpoiten käyttöjärjestelmän käynnistyksessä kernelin lokiin tulevista laiteajureiden virheilmoituksista. Tässä projektissa luvussa 6.3.1 kuvatussa kernelin käännöskonfiguraatiossa ei otettu käyttöön ajonaikaista laitteistopuun muokkausmahdollisuutta, joten laitteistopuu on Linuxin käynnistymisen jälkeen täysin staattinen tietorakenne.

5.4 Muutokset laitteistopuuhun

Liitteessä B on esimerkkinä laitteistopuukuvauksista kosketusnäytön kosketussensorin laitteistopuukuvaus. Lisäksi Linuxin lähdekoodissa on paljon esimerkkejä erilaisilla alustoilla käytetyistä oletuslaitteistopuista. Laitteiston määrittelyn voi jakaa useisiin tiedostoihin, ja niitä voi liittää toisiinsa tyypillisillä C-kielen esikääntäjän sisällytysdirektiiveillä (engl. include). Lisäksi muutkin C-kielen esikääntäjän direktiivit, kuten määritys (engl. define), ovat käytettävissä. Tässä työssä Kontronin SMARC-moduuli on määritetty erillisessä tiedostossa ja siihen liitetty emolevy toisessa. Tällä erottelulla on yksinkertaista ylläpitää eri laitteistopuuta evaluointiemolevylle ja varsinaiselle tuotteessa käytettävälle emolevylle.

Laitteistopuun juurisolmua kuvataan kauttaviivalla. Juurisolmuja on yhden laitteen määrittelyssä vain yksi. Esimerkin laitteistopuu (liitteessä B) ei sellaisenaan ole toimiva, koska standardin [30, s.37] mukaan laitteistopuussa pitää aina olla vähintään yksi prosessori- ja yksi muistisolmu. Laitteistopuukuvauksessa solmun kuvaus avataan ja suljetaan aaltosulkeilla ja solmun nimi annetaan ennen avaavaa aaltosulkua. Solmujen nimien täytyy olla uniikkeja koko laitteistopuuhierarkiassa ja tätä voidaan *reg*-solun sisältävien laitteiden osalta edistää lisäämällä solun määrittämä laitteen osoite solmun nimeen. Solmuun voidaan mistä tahansa muualta puusta viitata käyttäen *&*-merkkiä solmun nimen edessä. Myös makrojen käyttö laitteistopuumäärittelyissä on yleistä kuten liitteen B I2C-väylän määrittelyssä on tehty. Makrojen perässä olevat heksaluvut määrittävät pinnin sähköteknisen konfiguraation, kuten

esimerkiksi pinniin kytketyt prosessorin sisäiset ylösveto- ja alasvetovastukset ja niiden arvot. Tämän työn aikana ei ollut tarvetta muuttaa sähköteknisiä arvoja järjestelmäpiirivalmistajan ja laitevalmistajan oletusarvoista. [30]; [33, s.2179,2186,2550-2551,2563-2564]; [32]

Solmujen solujen arvot esitetään kulmasulkeiden sisällä ja solu voi koostua useasta välilyönnillä erotetusta 32-bittisestä etumerkittömästä kokonaisluvusta. Merkkijonot esitetään lainausmerkeissä. Liitteen B esimerkissä kosketusnäytön yhteensopivuus merkkijono (engl. compatible) määrittää käytettävän kernel-ajurin, joka tässä tapauksessa on EETI:n eGalax-sarjan monikosketusohjaimen ajuri. [30]; [16]

6. LINUX-KERNEL

Linux kuuluu Unix-tyyppisiin käyttöjärjestelmiin, ja se poikkeaa monista muista käyttöjärjestelmistä siinä, että sen lähdekoodi on saatavilla avoimella lisenssillä. Linux-kernel on käyttöjärjestelmän ydin, joka on käytössä kaikissa Linux-järjestelmissä. Kernel on vastuussa laitteistoresurssien jakamisesta eri ohjelmistokomponenttien kesken. Tällaisia resursseja ovat prosessorin suoritinaika, käytössä oleva RAM-muisti ja epäsuora pääsy kaikkiin muihin laitteistoresursseihin. Käytettäessä Linux-kerneliä sovellusten ei tarvitse kommunikoida suoraan laitteiston osien kanssa vaan ne voivat käyttää kernelin tarjoamia korkeamman tason rajapintoja. [26, s.1]; [22, s.3790]

Kuten useimpien ohjelmistokomponenttien toimintaa, myös kernelin ominaisuuksia voidaan ottaa käyttöön ja poistaa käytöstä ennen käännöstä tehtävällä konfiguraatiolla. Tämä mahdollistaa sulautetuissa laitteissa tuen karsimisen ominaisuuksilta, joita ei koskaan tarvita, ja näin pienentää kernelin kokoa ja monimutkaisuutta. Toisaalta myös monet juuri tällä sulautetulla laitteella tarvittavat kernelin ominaisuudet voidaan ottaa helposti käyttöön. Tästä syystä johtuen sulautetuissa laitteissa harvoin käytetään X86-arkkitehtuuriin perustuvien työpöytä- ja palvelintietokoneiden usein käyttämiä Linux-jakelun tekijän kääntämiä suurikokoisia, yleiskäyttöisiä ja laajan laitteistotuen sisältäviä kerneleitä. [22, s.3790]

6.1 Historia

Linuxin edeltäjän Unixin kehitys alkoi vuonna 1969 epäonnistuneen Multics-käyttöjärjestelmäprojektin pohjalta. Bell-laboratorioiden tutkimuskeskuksessa Dennis Ritchie ja Ken Thompson suunnittelivat tiedostojärjestelmän, jonka ympärille Unix rakentui. Vuonna 1973 Unix kirjoitettiin uudestaan ja silloin uudella C-kielellä. Sittemmin laajasti yleistyneen C-kielen käyttö on myöhemmin auttanut tekemään Unix-versioita monille eri prosessoriarkkitehtuureille. Unixin yksinkertainen arkkitehtuuri ja lähdekoodin saatavuus aiheutti sen, että Unixista kehitettiin useita eri versioita monissa eri yliopistoissa ja yrityksissä. Unixin menestykseen johti yksinkertainen rakenne, jonka rajapinnassa on monista muista järjestelmistä poiketen

tuhansien järjestelmäkutsujen sijaan vain satoja järjestelmäkutsuja. Unixissa kaikki mahdollinen kuvataan tiedostoina. Tämän johdosta dataa ja laitteita voidaan käsitellä muutamalla järjestelmäkutsulla. Tämän lisäksi Unixissa on nopeaa luoda uusia prosesseja ja kommunikointimekanismi prosessien välillä toimii robustisti. Tämä on johtanut siihen, että Unixiin on paljon yhden käyttötarkoituksen yksinkertaisia ohjelmia, jotka hoitavat yhden tehtävän hyvin. Näitä ohjelmia voidaan ketjuttaa kompleksisempien ongelmien ratkaisuun. [37, s.626-822]

Linuxin kernelin kehittämisen aloitti vuonna 1991 suomalainen Linus Torvalds, joka on edelleen vastuussa kehityksestä. Kehityksen aloittamiseen motivoi tehokkaan ja ilmaisen Unix-käyttöjärjestelmän puuttuminen markkinoilta. Tuohon aikaan hallitsevaa Microsoftin DOS-käyttöjärjestelmää Torvalds ei kokenut hyödylliseksi, ja akateemisen opetuskäyttöön suunniteltu edullinen Unix versio *Minix* oli suunnittelultaan ongelmallinen ja lisenssiltään hänen mielestään liian rajoittava. Linus Torvalds aloitti Linuxin kehittämisen tyhjästä, joten Linux ei sisällä koodia mistään Unix-järjestelmästä. Sen sijaan se lainaa monia ideoita Unixista ja toteuttaa Unix-rajapinnan, minkä johdosta Unix-ohjelmat toimivat myös Linuxissa. Alun perin käyttöjärjestelmä tuki vain IBM-yhteensopivia Intelin 80386 mikroprosessoriin perustuvia järjestelmiä. Tämän jälkeen Linux on sovitettu toimimaan lukuisilla eri prosessoriarkkitehtuureilla, kuten ARM, PowerPC ja AMD64. Linuxin kehittämiseen on osallistunut vuoden 2005 jälkeen noin 11800 kehittäjää, joista suurimmalle osalle yritykset ovat maksaneet palkkaa Linuxin kehitykseen osallistumisesta. Tällaisia yrityksiä on lähes 1200, ja osa niistä on toistensa kilpailijoita muilla alueilla. Lähdekoodirivejä 7.12.2014 julkaistussa versiossa 3.18 on lähes 19 miljoonaa [28]. [37, s.626-822]

6.2 Käyttö tässä projektissa

Tässä projektissa on alusta lähtien ollut käytössä järjestelmäpiirin valmistajan Freescalen ylläpitämä kernel-haara. Freescale ylläpitää omia muutoksiaan virallisen Linux-kernelin päällä päivittäen muutokset uudempien virallisten kernel-versioiden pohjalle. Lisäksi Freescale työskentelee yhdessä Linux-kernel-yhteisön kanssa saadakseen oman laitteistotukensa tarvitsemia muutoksia myös viralliseen kerneliin. Tämän päälle laitteistovalmistaja Kontron tarjoaa omat muutoksensa Freescale-kernelin sovittamiseksi Kontron SMARC-sAMX6i -moduulien eri versioille. Laitteistovalmistaja Kontronin muutostiedostot ovat listattuina liitteen C Yocto-reseptissä. Tässä projektissa Kontronin muutosten lisäksi kerneliin jouduttiin tekemään useita muita muutoksia laitteistotuen tuomiseksi omalle lopulliselle SMARC-emolevylle. Tässä työssä Kontronin muutokset ja itse lisätyt kernel-muutokset ovat listattuina liitteen

D Yocto-reseptissä. Osa itse lisätyistä kernel-muutoksista on saatu laitevalmistajalta, kuten WLAN-moduulin vaatimat muutokset. Osa taas on toteutettu osana tätä projektia kuten kosketusnäytön monikosketustuki.

Tässä projektissa aloitin laitteistovalmistaja Kontronin tarjoamalla kernel-versiolla 3.10.17, mutta laitteistovalmistajan päivitysten puuttuessa päädyin itse päivittämään laitteistovalmistajan kernel-muutokset Freescalen 3.10.53 kernel-version päälle. Tämä toi käyttöön uudemmat grafiikka-ajurit ja tuen Texas Instrumentsin yhdistetylle Wlan-Bluetooth-moduulille. Uudemmat grafiikka-ajurit mahdollistivat paremmin Wayland- ja X11-ikkunointijärjestelmien vertailun. Texas Instruments tarjoaa Freescalen referenssialustalle valmiit muutostiedostot, joilla Wlan-Bluetooth-moduuli saadaan käyttöön vain uudemmalle kernel-versiolle, joten aiemman kernel-version käyttö olisi tuottanut lisätyötä. Projektin edetessä Kontron toimitti päivityksen kernel-versioon 3.14.28, jossa tuki edellä mainituille ominaisuuksille oli vieläkin parempi ja se otettiin heti käyttöön. Jokainen kernel-version vaihto tuottaa huomattavan lisätyön, koska projektissa on sen verran projektikohtaisia muutoksia, jotka pitää siirtää uuden kernel-version päälle ja testata. Onkin melko epätodennäköistä, että kernel-versiota enää päivitetään kun laite on viety tuotantoon.

Laitteistopuun ja kernelin kehityksen nopeuttamiseksi päädyin tekemään laitteistopuu- ja kernel-käännöksiä ilman Yocton BitBake-työkalua käyttäen ainoastaan Yocton tuottamaa ristikäännösympäristöä. Tämä on myös yleisesti käytetty työtapa kernel-kehittäjien keskuudessa [2, s.1904]. BitBake-työkalu soveltuu paremmin ennalta testattujen ohjelmistokomponenttien kääntämiseen eikä muutosten nopeaan kokeiluun. Lisäksi kernelin ja siihen liittyvän laitteistopuun siirtäminen moduulille oli erittäin nopeaa käyttäen luvussa 4.3 esitettyä Tftp-tiedonsiirtoa.

6.3 Kernel-muutokset

Tässä projektissa kerneliin tehtiin useita muutoksia, jotka kaikki liittyivät laitteistoajureihin. Osana kerneliä ovat ajurit, jotka kommunikoivat laitteiston kanssa. Otettaessa käyttöön uutta laitetta kernelin kanssa ideaalitapauksessa laitteen ajurituki on ollut mukana jo oletuskäännöskonfiguraatiossa eikä laitteen ajurikoodi tarvitse mitään erillisiä konfiguraatioarvoja laitteistopuusta. Tämän seurauksena laite toimii suoraan ilman mitään muutoksia. Näin tapahtuu usein yleiskäyttöisten työpöytäkäyttöön suunnattujen kernelien kanssa. Tässä projektissa ainoastaan Kontron SMARC-sAMX6i -moduulien ominaisuudet kuten CAN-väylä ja I2C-väylät toimivat näin. Eli laitteistovalmistajan tarjoaman laitteistotukipaketin avulla käännetty kernel mahdollisti näiden väylien käytön Linuxin käyttäjätilan apuohjelmilla. Myös yleisimmät USB-laitteet, kuten näppäimistöt ja USB-muistit, toimivat suoraan. Kernel-

muutokset jakautuvat kolmeen eri osa-alueeseen: käännöskonfiguraation, laitteistopuun ja kernelin lähdekoodin muutoksiin. Laitteistopuun muutoksista on kerrottu tarkemmin kappaleessa 5.4.

6.3.1 Muutokset kernelin käännöskonfiguraatioon

Kernelin käännöskonfiguraatio on tekstitiedosto, mutta sen syntaksi sisältää riippuvuuksia, joten sen muuttamiseen on suositeltavaa käyttää tarkoitukseen suunniteltua ohjelmistoa. Lisäksi erillisen ohjelmiston hakutoiminnolla on mahdollista löytää valintoja, jotka eivät ole näkyvissä tekstitiedostossa, johtuen jonkun riippuvuuden puuttumisesta. Tässä projektissa käytettiin komentorivillä toimivaa, mutta graafista *menuconfig*-työkalua. Käytössä oleva käännöskonfiguraatio talletetaan juurihakemistoon *.config*-nimellä, josta kernelin GNU *Make* -pohjainen *kbuild*-käännösympäristö sen lukee. Kernelin lähdekoodissa on paljon esimerkkejä erilaisten alustojen oletuskäännöskonfiguraatioista. [2, s.1770]

Kernelin osia, kuten ajureita, voi sisällyttää kerneliin tai ottaa mukaan ajonaikaisesti ladattavina kernel-moduuleina. Suuri osa konfiguraatiovalinnoista on yksinkertaisia boolean-arvoja: osa joko on sisällytetty tai ei ole sisällytetty tuotettavaan kerneliin. Toinen yleisesti käytetty konfiguraatio lisää boolean-arvoon kolmannen vaihtoehdon *M*, joka tarkoittaa, että osa otetaan mukaan kernel-moduulina, mutta sitä ei välttämättä ladata käyttöön kernelin käynnistyksessä. Näin kernelin vaatima RAM-muistin määrä vähenee ja osa, kuten laitteistoajuri, voidaan ladata siinä vaiheessa, kun laitetta tarvitaan. Esimerkiksi USB-muistien ajurin kernel-moduuli voidaan ladata vasta, kun USB-muisti kiinnitetään laitteeseen. Näiden lisäksi parametreina voi olla harvinaisempia lukuarvoja ja merkkijonoja. Esimerkiksi käytettävän sarjaportin nimi määritetään merkkijonona. [27, s.93]

6.3.2 Muutokset kerneliin

Ideaalitilanteessa kernelin lähdekoodia ei tarvitsisi muuttaa lainkaan, vaan fyysisen laitteiston vaatimat muutokset pystyttäisiin tekemään vain ottamalla osia käyttöön kernelin käännöskonfiguraatiossa ja määrittelemällä niiden parametrit laitteistopuussa. Laitteistopuu on kuitenkin ollut käytössä vasta melko vähän aikaa, joten on monia ajureita, joita ei ole päivitetty käyttämään sitä. Lisäksi uskoisin, että aina tulee olemaan laitteistoajureita, joita kirjoitettaessa ei ole otettu huomioon kaikkia mahdollisia variantteja, joita laitteesta on saatavilla. Tämän johdosta lähdekoodimuutoksia varmasti joudutaan tekemään tulevaisuudessakin.

6.3.3 Kernelin kääntäminen

Tein kernel-käännökset käyttäen Yocton tuottamaa ristikäännösympäristöä, joka on tarkemmin esiteltynä luvussa 3.5.4. Kernelin kääntäminen tyhjästä kestää useita minuutteja käännöskonfiguraatiosta ja laitteistosta riippuen. Moniydinprosessorin hyödyntäminen rinnakkaistamalla käännösprosessia nopeuttaa sitä kuitenkin huomattavasti. Rinnakkaistaminen on mahdollista käyttäen *Maken -j*-parametria. [37, s.896-962]

Laitteistotoimittaja Kontronin U-boot-käynnistyslataaja tukee vain *uImage*-muotoista kernel binääriä Linux-kernelin käännösympäristön oletuksena tuottaman *zImage*-tyypin sijaan, joten lisäksi on annettava parametrit sen tuottamiseen. Lisäksi *uImage*-tyypin yhteydessä on annettava muistiosoite, johon U-boot-käynnistyslataaja sijoittaa kernelin laitteen RAM-muistissa. [2, s.1901]

6.3.4 Kernelin asentaminen

Kehitettävän ominaisuuden ollessa sisäänrakennettu kerneliin voidaan asennus suorittaa yksinkertaisesti kopioimalla kernel binääri U-bootin käyttämään tiedostojärjestelmään, kuten Tftp-palvelimelle tai laitteen eMMC-muistiin. Ajureita kehitettäessä osoittautui, että ajurin sisällyttäminen kerneliin oli nopein tapa testata muutoksia. Jos lisäksi tarvitaan kernel-moduuleina mukaan otettuja ominaisuuksia, niin on käännettävä erikseen vielä kernel-moduulit ja siirrettävä ne juuren tiedostojärjestelmään, kuten NFS-tiedostojenjakopalvelimelle tai laitteen eMMC-muistiin.

Kernel-moduulien kääntäminen ja asentaminen onnistuu *Maken* avulla. Oletuksena moduulit asennetaan isäntäkoneeseen, mutta tässä ristikäännöksen tapauksessa täytyy määrittää erillinen polku, johon moduulit asennetaan. Kernel-moduulien asentamisen jälkeen voi olla tarpeellista suorittaa kehitettävällä ARM-laitteella komento *depmod -a*, joka lataa uudet kernel-moduulit käytettäviksi. Käytettävissä olevien kernel-moduulien ajonaikaiseen listaamiseen, käyttöön ottoon ja käytöstä poistoon on olemassa kattavat apuohjelmat. On tärkeää, että kernel-moduulit on käännetty samasta versiosta kuin itse kernel, jolloin ne toimivat yhteen. Tämä varmistetaan käyttämällä Git-versiohallinnan SHA-1-tarkistussummaa kernel-moduulien asennuskansion nimessä. Itse kernel-binääriin on tallennettuna vastaavat versiotiedot, jolloin se tunnistaa oikeassa hakemistossa olevat kernel-moduulit. Tämä mahdollistaa myös sen, että yhtä aikaa voi olla asennettuina useampien eri kernel-versioiden kernel-moduulit. [37, s.896-962]

7. ALUSTAN KÄYTTÖÖNOTTO

Sulautetun Linux-alustan käyttöön tarvitaan neljä erillistä ohjelmistokomponenttia. Nämä ovat käynnistyslataaja, laitteistopuu, Linux-kernel ja juuren tiedostojärjestelmä. Yocto-työkalu tarjoaa mahdollisuuden luoda nämä kaikki yhdellä kertaa yhdelle laitteistolle. Tässä projektissa käytössä olevan laitteiston valmistaja Kontron tarjoaa Yocto-tuen vain laitteistopuun, kernelin ja juuren tiedostojärjestelmän tuottamiseen. Tästä johtuen käynnistyslataaja pitää tuottaa erikseen luvussa 4 kuvatulla tavalla.

Lisäksi tehtäessä ohjelmistokehitystä alustalle useimmiten erilliset ristikäännöstyökalut ovat tarpeellisia. Erilliset ristikäännöstyökalut kuvataan tarkemmin luvussa 3.5.4. [2, s.547-563]

7.1 LVDS-näyttö

Tässä projektissa Kontronin evaluointiemolevyn ympärille rakennetulla kehitysalustalla oli valmiiksi käytössä LG:n valmistama kymmenentuumainen LVDS-liitäntää käyttävä näyttö. Itse suunnitellun lopullisen emolevyn kanssa käytetään Chefreen valmistamaa seitsemäntuumaista LVDS-näyttöpaneelia, joka otettiin käyttöön sekä *U-bootissa* että Linuxissa.

Laitteistovalmistaja Kontronin toimittaman *U-bootin* lähdekoodissa on konfiguroituna useampien LVDS-näyttöjen ajoitusparametreja resoluutioineen, joista käytössä oleva voidaan valita käyttäen ympäristömuuttujaa *panel*. Tällä konfiguraatiolla sekä LG:n että Chefreen näytöille saatiin kuva käynnistyksessä. Tämän lisäksi *U-boot* kytkee oletuksena näytön taustavalon päälle kirkkaimmalle tasolleen ilman säätömahdollisuutta. Tavallisessa käyttötapauksessa *U-bootin* näyttämä yrityksen logo näkyy ruudulla vain muutaman sekunnin ajan, ennen kuin Linux-kernel ottaa vastuulleen grafiikan esittämisen, joten kirkkauden säätöä ei pidetty olennaisena ominaisuutena. Tämän lisäksi evaluointiemolevyllä ja lopullisella emolevyllä taustavalon jännitteensyöttö on toteutettu eri tavoilla. Lopullisella emolevyllä on erillinen, erikseen päälle ohjattava jännitteenmuunnin, joka muuttaa 1,8 voltin jännitteen 3,3 voltiksi. *U-bootin* laitekohtaiseen alustustiedostoon lisättiin jännitteenmuuntimen

päälle kytkeminen, jolloin näytön taustavalo saatiin toimimaan heti käynnistyksessä. Taustavalon toiminta heti käynnistyksessä on tärkeää käytettävyyden kannalta, jotta käyttäjä näkee heti käynnistysnapin painamisen jälkeen, että näytölle tulee kuva ja laite käynnistyy.

Kehitysalustan LVDS-näyttö oli valmiiksi konfiguroitu käytettäväksi Linuxissa, jolloin sen kernelin käännöskonfiguraatio ja Freescalen LVDS-ajurit olivat käytettävissä ilman muutoksia. Laitteistopuuhun oli määritelty kehitysalustalla käytetyn näytön ominaisuudet. LVDS-paneeli määritellään LVDS-näyttösiltasolmussa (engl. LVDS Display Bridge). Ominaisuuksia ovat käytettävä kuvan prosessointiyksikkö (engl. Image Prosessing Unit), LVDS-signaalien järjestys, LVDS-näytön värien määrä, näytön kellon taajuus, resoluutio, näytön reunusten mittasuhteet ja horisontaalinen sekä vertikaalinen synkronointitaajuus. Chefreen LVDS-näytön käyttöön ottamiseksi riitti, että laitteistopuussa näytön ominaisuudet vaihdettiin Chefreen näytön ominaisuuksia vastaaviksi. Tässä ongelmia aiheutti näyttövalmistajan vajavainen dokumentaatio, ja kaikkia tarvittavia arvoja ei ollut suoraan saatavissa. Tämän johdosta osa arvoista on haettu kokeilemalla.

Kehitysalustalla oli lisäksi toteutettuna näytön LED-taustavalon kirkkauden säätö käyttäen Linux-kernelin taustavaloajuria. Taustavalon LED-polttimot saavat järjestelmäpiirin pulssinleveysmoduloijalta signaalin, jonka mukaan niiden kirkkautta voidaan säätää. Näytön taustavalo määritellään laitteistopuun taustavalo-solmussa. Ominaisuuksia ovat käytettävän pulssinleveysmodulaation pinni sisältäen modulaation taajuuden, käytettävät kirkkauden tasot, vakiokirkkaustaso, päälle kytkemiseen käytettävä pinni ja käytetty kuvalähde. Taustavaloajuri saatiin toimimaan Chefreen näytön kanssa vaihtamalla laitteistopuukuvaukseen oikea pulssinleveyden modulointitaajuus.

7.2 I2C-kosketuspaneeli

Kosketusnäytön ohjaus on toteutettu I2C-liitäntää käyttävällä Eetin valmistamalla Egalax EXC3132 -ohjainkortilla. *U-boot*-käynnistyslataajassa ei ole tukea kosketuspaneeleille, ja sen kanssa kommunikoidaan ainoastaan sarjaporttiterminaalin kautta. Tämän vuoksi kosketuspaneelin ajurit tarvitaan vain Linux-kerneliin. Ajureiden lisäksi käytettäessä kosketuspaneelia X11-ikkunointijärjestelmän kanssa tarvitaan paneelin kalibrointiarvot ikkunointijärjestelmälle. Kontronin kehitysalustalla oli käytössä USB-liitäntään kytketty vanhemman sukupolven Eetin Egalax-kosketuspaneeli, joten projektissa valittuun I2C-liitäntäiseen paneeliin oli valmiina ainoastaan X11-ikkunointiympäristön kalibrointiarvot, jotka toimivat myös I2C-kosketuspaneelissa.

Kosketuspaneelin saamiseksi käyttöön Linux-kernelissä täytyi paneelinohjaimen nollauspinni käydä vapauttamassa ennen Linux-kernelin ajureiden lataamista. Ilman nollauspinnin vapautusta kosketuspaneeli ei tunnistu käynnistyksessä oikein. Tämä toteutettiin lisäämällä *U-bootin* laitekohtaiseen alustustiedostoon nollauspinnin vapautus, jolloin Linux-ajuri tunnistaa I2C-laitteen.

Yleisesti ottaen I2C-kosketuspaneelin ajuri käyttää valmista kernelin I2C-laitteistoajuria kommunikointiin kosketuspaneelin ohjaimen kanssa. Tämän johdosta täytyy varmistaa, että I2C-väylän ajuri on toiminnassa ja oikein konfiguroitu. Linux-ajuri rekisteröi keskeytyksen kernelille, ja kosketuspaneelin mikrokontrolleri aiheuttaa keskeytyksen jokaisesta kosketuksesta. Linux-ajuri käy keskeytyskäsittelijässä lukemassa I2C-väylältä kosketuspisteet ja välittää ne eteenpäin käyttöjärjestelmälle. Tiedonsiirto kosketuspaneelin kanssa jää täysin kapseloiduksi ajurin sisäiseen toteutukseen.

Laitteistovalmistaja Eeti toimitti ajurikoodin, joka lisättiin omana käännösyksikkönään kerneliin. Ajurissa oli tuki useampien virallisten Linux-kernel-versioiden lisäksi myös Android-kernelille. Eetin toimittama ajuri osoittautui testeissä hitaasti kosketukseen reagoivaksi, täysin kernelin tyyliopasta noudattamattomaksi ja vaikealukuiseksi lähdekoodiksi. Eetin toimittaman ajurin lisäksi Linux-kernelissä oli myös valmiina Freescalen toteuttama kosketuspaneeliajuri, joka oli kirjoitettu käyttämään vanhemman sukupolven Eetin Egalax -kosketuspaneeleja. Vanhemmissa kosketuspaneeleissa I2C-väylän protokolla oli kuitenkin yksinkertaisempi. Protokollan eroavaisuudesta johtuen kernelin ajuri ei toiminut projektissa käytetyn uudemman kosketuspaneelin ohjaimen kanssa suoraan. Uudemman kosketuspaneelin dokumentaatiota lukemalla ja tekemällä muutoksia Freescalen kernel-ajuriin saatiin käyttöjärjestelmä reagoimaan kosketuksiin nopeasti. Lisäksi ajuri otettiin käyttöön kernelin käännöskonfiguraatiossa ja sen tarvitsemat parametrit, kuten keskeytyspinni ja I2C-väylän osoite, lisättiin laitteistopuuhun.

7.3 Yhdistetty mobiiliverkon tiedonsiirto- ja satelliittipaikannusmoduuli

Laitteessa käytetään Quectelin valmistamaa UC20-G-mallista yhdistettyä 2/3G-tiedonsiirto- ja GPS/GLONASS-paikannusmoduulia. Käynnistyslataaja *U-boot* ei tue mobiiliverkon tiedonsiirtoa, eikä moduulin tarvitse olla käytettävissä heti käynnistyksessä, joten sen käyttöönotto tehtiin vain Linux-kernelissä. Moduulin tarvitsema ajuri, joka on alun perin kehitetty toisen valmistajan, Optionin, laitteita varten, mutta se toimii myös useiden muiden valmistajien laitteiden kanssa, oli valmiiksi

kernelissä. Kernelin käännöskonfiguraatiossa kyseinen ajuri otettiin käyttöön. Lisäksi ajuri ei oletuksena tunnistanut tiedonsiirto- ja paikannusmoduulin yksilöivää USB-laitteen valmistajatunnistetta ja laitetunnistetta. Tämän vuoksi kernel-ajuriin lisättiin Quectelin valmistajatunniste ja UC20-G-laitteen laitetunniste. Ajurin lisäksi UC20-G-moduulin nollauspinni pitää vapauttaa, jotta laite toimii oikein. Tämä toteutettiin lisäämällä käynnistyskomentosarja, joka ajetaan Linuxin käynnistyessä. Laitteistopuuhun ei tarvinnut tehdä muutoksia Quectelin moduulia varten, vaan se riitti, että USB-portti, johon moduuli oli fyysisesti kytketty, oli konfiguroitu oikein USB-portiksi ja tämä oli valmiiksi tehtynä Kontronin tarjoamassa laitteistotukipaketissa.

Kun yhdistetty tiedonsiirto- ja paikannusmoduuli on alustettu oikein, niin se näkyy käyttöjärjestelmälle viitenä eri USB-sarjaporttina. USB-sarjaporteista ensimmäinen on tarkoitettu moduulin diagnostiikkaan, ja toisesta sarjaportista on luettavissa moduulin tuottama paikannusdata. Kolmanteen sarjaporttiin kirjoitetaan AT-komennot, joilla moduulin toimintaa ohjataan. Neljännessä sarjaportissa kulkee mobiilin tiedonsiirtoverkon välityksellä siirretty data. Viides sarjaportti toteuttaa QMI-rajapinnan, joka on piirivalmistajakohtainen USB-sarjaliikenteiselle tiedonsiirrolle valinnainen tiedonsiirtotapa. QMI tulee sanoista Qualcomm MSM Interface ja sen käyttöön on olemassa tuki kernelissa qmi_wwan-ajurin muodossa ja käyttäjätilaan on saatavilla kirjastoja kuten libqmi, mutta ne jätettiin tässä työssä tutkimatta, koska moduulivalmistaja Quectel suositteli USB-sarjaportin käyttöä.

7.3.1 Mobiiliverkon tiedonsiirto

Tiedonsiirron aloittamiseen tarvitaan pisteestä pisteeseen protokollaa (engl. point-to-point protocol) käyttävää Linuxin käyttäjätilan taustaohjelmistoa, joka ohjaa modeemia USB-sarjaportin AT-komennoilla. Lisäksi sama taustaohjelmisto huolehtii datan välittämisen käyttöjärjestelmän ppp0-rajapinnan ja sarjaportin välillä. Tiedonsiirron hallintaan on olemassa myös Linuxin käyttäjätilan ohjelmistot Ofono ja NetworkManager. Kyseiset ohjelmistot ovat AT-komentojen käyttämiseen tarkoitettuja välikerroksen taustaohjelmistoja, jotka tarjoavat ppp:n keskustelukomentosarjoja (engl. chat script) korkeamman tason rajapinnan modeemin käyttöön. Tämän diplomityön puitteissa näitä korkeamman tason ohjelmistoja ei testattu.

Tiedonsiirtonopeus 3G-verkossa testattiin sisätiloissa laitteen ollessa koteloitu ja käyttäen Taoglasin valmistamaa PA.25a-piirilevyantennia. Tiedonsiirtonopeus laskettiin käyttäen time- ja wget-apuohjelmia ladattaessa palvelimelta laitteelle dataa sekä time- ja scp-apuohjelmia käyttäen siirrettäessä dataa palvelimelle päin. Koe

toistettiin vain kerran, joten lukemia ei voida pitää kovin luotettavina. Tiedonsiirtonopeudeksi saatiin noin 6/3 megatavua sekunnissa, jota pidettiin riittävänä laitteen mittaustietokannan synkronointitarkoitukseen ja VNC-etätukeen.

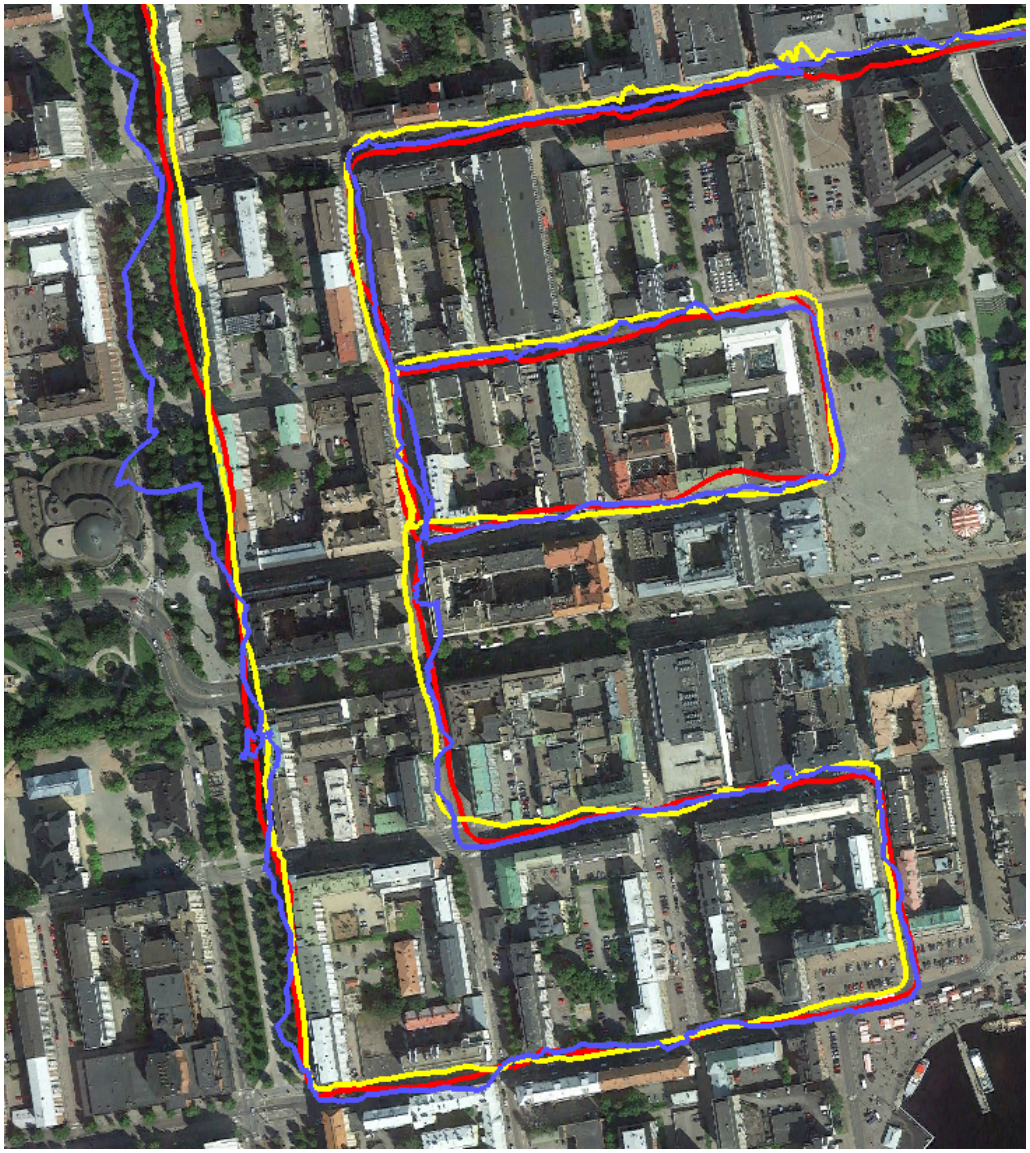
7.3.2 Satelliittipaikannus

Quectelin UC20-G-moduulin GPS/GLONASS-paikannustoimintoa ohjataan AT-komennoilla USB-sarjaportista kuten tiedonsiirtoakin. Moduulin tuottama NMEA-standardin mukainen paikannusdata on luettavissa toisesta sarjaportista. Paikannusdatan keräämiseen ja analysointiin on olemassa korkeamman tason rajapinnat tarjoavat taustaohjelmistot Gpsd ja Geoclue. Tämän diplomityön puitteissa näitä korkeamman tason ohjelmistoja ei testattu.

Paikannusmoduulin tarkkuus käyttäen Taoglasin valmistamaa GGBLA.01.A-piirilevyantennia testattiin ajamalla testikierros keskusta-alueella laitteen ollessa normaalisti koteloituna. Testi suoritettiin keräämällä sekunnin välein NMEA-paikannusdataa sarjaportista lokitiedostoon käyttäen Bash-komentosarjaa. Verrokkilaitteilla käytettiin sovelluskaupoista saatavilla olevia sovelluksia, jotka keräsivät paikannusdatan suoraan GPX-muodossa. Testidata on esitetty kuvassa 7.1. Verrokkeina testissä olivat pelkällä GPS-paikantimella varustettu Nokia N9-puhelin ja LG Nexus 5 -puhelin, jossa on tuki sekä GPS- että GLONASS-satelliiteille. Kuvassa tässä projektissa suunnitellun laitteen data on merkitty punaisella, Nokian data sinisellä ja LG:n data keltaisella. Koe toistettiin vain kerran, joten sitä ei voida pitää kovin luotettavana, mutta datan perusteella näyttäisi siltä, että laite on verrattavissa paikannustarkkuudessa LG:n puhelimeen. Pelkällä GPS:llä varustetun Nokian puhelimen tuottama paikannusdata on jonkin verran laitetta ja LG:n puhelinta epätarkempaa. Testin perusteella paikannustarkkuutta pidettiin riittävänä mittaussovelluksessa käytettävään ulkotiloissa tapahtuvaan paikannukseen.

7.4 Yhdistetty Bluetooth- ja WLAN-moduuli

Laitteessa käytetään Texas Instrumentsin valmistamaa WL1835MOD-mallista yhdistettyä Bluetooth- ja WLAN-moduulia. Texas Instruments tarjoaa valmiina tarvittavat: kernel-, kernelin käännöskonfiguraatio- ja laitteistopuumuutokset moduulin käyttöön ottamiseksi yhdellä Freescalen samaan prosessoriin perustuvalla kehitysalustalla [45]. Moduulin käyttöön saamiseksi erilaisen kehitysalustan laitteistopuu muutokset, kuten käytettävät pinnit, SD-laiteväylä ja USB-portti sovitettiin käytetylle lopulliselle emolevylle sopiviksi. Texas Instrumentsin moduulin vaatimat



Kuva 7.1 *Paikannusmoduulin vertailu kaupallisiin laitteisiin.*

lisäykset vaativat useiden kernel-moduulien asentamisen ja lisäksi joitakin laiteohjelmistojen binääritiedostoja. Näiden tuottamiseen on kuitenkin saatavilla hyvä dokumentaatio ja valmiit komentosarjat [45]. Tämän projektin puitteissa komentosarjoja ei integroitu osaksi Yocto-käännöstä vaan niiden suoritus on vielä erillinen työvaihe Yocto-käännöksen jälkeen. Lisäksi moduulin saamiseksi käyttöön Linux-kernelissä täytyi sekä Bluetooth- että WLAN-puolten nollauspinnit käydä vapauttamassa ennen Linux-kernelin ajureiden lataamista. Tämä toteutettiin lisäämällä käynnistyskomentosarja, joka ajetaan Linuxin käynnistyessä.

7.4.1 Bluetooth

Bluetooth-ohjain on fyysisesti kytketty i.MX6-järjestelmäpiirin sarjaporttiin, ja se näkyy käyttöjärjestelmälle tavallisena sarjaporttina. Bluetooth-laitteen saa käyttöön Linuxin käyttäjätilan isäntäohjainrajapintaa (engl. Host Controller Interface) käyttävillä matalan tason ohjelmistoilla. Näillä ohjelmistoilla voi esimerkiksi etsiä muita laitteita ja testata laitteiden välisen tiedonsiirron viivettä. Bluetoothin käyttöön on olemassa useita Bluetooth-protokollia tukeva Bluez-kirjasto, mutta sitä ei tämän diplomityön puitteissa testattu.

Bluetoothin kantamaa testattiin sisätiloissa käyttäen Texas Instrumentsin referenssitoteutuksen pohjalta suunniteltua piirilevyantennia. Testejä tehtiin käyttäen sekä tavallista Bluetooth-profiilia tukevaa puhelinta että vain Bluetooth 4.0 LE -profiilia tukevaa mittausanturia. Tavallista Bluetooth-profiilia tukevat laitteet MAC-osoitteineen on mahdollista löytää apuohjelmalla komennolla *hcitool scan* ja vastaavasti Bluetooth 4.0 LE profiilia tukevat laitteet *hcitool lescan*-komennolla. Tämän jälkeen komennolla *l2ping* siirrettiin Bluetooth-paketteja laitteiden välillä. Molemmilla laitteilla toimintasäteeksi muodostui noin 10-15 metriä.

7.4.2 WLAN

WLAN-ohjain näkyy käyttöjärjestelmälle Secure Digital -muistikorttistandardin mukaisena SDIO-laitteena. Kun WLAN-ohjain on konfiguroitu oikein, niin se näkyy käyttöjärjestelmälle tavallisena verkkorajapintana kuten langallisetkin lähiverkot. Matalan tason työkalu wpa_supplicant tarjoaa mahdollisuuden yhdistää myös salattuihin langattomiin verkkoihin. WLAN-verkkojen hallintaan on useampia korkeamman tason ohjelmistoja, kuten Connman ja NetworkManager, mutta tämän diplomityön puitteissa niitä ei testattu.

WLAN:in tiedonsiirtonopeutta testattiin sisätiloissa käyttäen Texas Instrumentsin referenssitoteutuksen pohjalta suunniteltua piirilevyantennia. Testeinä käytettiin Texas Instrumentsin WLAN-suorituskykytestejä [46] laitteen ja Linux-työaseman välillä. TCP-protokollaa käyttäen tiedonsiirtonopeudeksi saatiin noin 9/13 megatavua sekunnissa, jota pidettiin riittävänä laitteen mittaustietokannan synkronointitarkoitukseen ja VNC-etätukeen.

7.5 Analoginen videokameraliitäntä

Analogisen videokameran liittäminen on toteutettu I2C-väylän kautta ohjattavalla Analog Devices:in valmistamalla ADV7180-videodekooderilla, jonka digitaaliset

ulostulot on liitetty i.MX6-järjestelmäpiiriin integroituun kuvankäsittelypiirin rinnakkaisliitäntään. Kernelin konfiguraatiosta lisättiin tuki videodekooderin ajurille, ja laitteispuuhun lisättiin sen ominaisuudet, kuten käytetty videosisääntulo ja kellotaajuus. Ajurin tarjoaman puutteellisen laitteistopuukonfiguraation johdosta myös itse ajuriin tehtiin muutoksia, jotta se vastaa täysin käytettyä videodekooderin mallia. Lisäksi prosessoriin sisäänrakennettua kuvankäsittelypiiriä ohjataan prosessorin sisäisellä rekisterillä, johon laitteistopuutuen puuttuessa kokonaan täytyi erikseen vaihtaa rinnakkaisliitännän asetukset kernelin laitetiedostossa kovakoodatusti. Kahta kameraliitäntää voidaan tällä ohjainpiirillä käyttää vain yksi kerrallaan ja vaihto liitäntöjen välillä toteuttiin kirjoittamalla suoraan videodekooderin rekisteriin I2C-väylän ja käyttäjätilan apuohjelman avulla. Kameran kuva testattiin käyttäen i.MX-järjestelmäpiireille suunnattua versiota yleisesti multimedian käsittelyyn käytetystä Gstreamer-komponentista, joka yhden ympäristömuuttujan asettamalla syötti kuvan Qt:n mukana tulevalle QML-pohjaiselle kameran toimintaa esittelevälle esimerkkisovellukselle.

7.6 Kuittitulostin

Tulostimen käyttöönotto ajonaikaisesti vaatii jännitteen syötöstä vastaavan 24 voltin jännitteenmuuntimen ohjauksen päälle. Tämä toiminnallisuus lisätään Linuxin käyttäjätilaan kehitettävään mittaussovellukseen. Kommunikointi tulostimen kanssa onnistui suoraan käyttäen sarjaporttia, joten kernelin tai laitteistopuun muutoksille ei ollut tarvetta. Kuittitulostimet tukevat yleisesti alun perin Epsonin kehittämiä ESC/POS-ohjainkomentoja, joten kuittitulostimen käyttöön ei vaadita mitään ajureita. Sarjaportti, johon tulostin oli kytketty, oli valmiiksi konfiguroitu Kontronin laitteistotukipaketissa.

8. JOHTOPÄÄTÖKSET

8.1 Työn tulokset

Työn tavoitteena oli toteuttaa Linux-pohjainen ohjelmistoalusta seuraavan sukupolven ajoneuvo-PC:n käyttöön. Käytettäväksi työkaluksi valikoitui monipuolisuutensa ja laitteistovalmistajien tuen perusteella Yocto, joka mahdollistaa oman, käyttötarkoitukseen erikoistetun, Linux-jakelun rakentamisen sulautetulle laitteelle.

Työssä Freescalen i.MX6-järjestelmäpiiriin perustuvan Kontronin SMARC-sAMX6i Solo -moduulin kanssa käytetty laitteistovalmistajan evaluointiemolevy vaihdettiin muokattuun lopulliseen emolevyyn. Evaluointiemolevyllä käytettyihin valmiina saatuihin ohjelmistokomponentteihin U-boot-käynnistyslataajaan, Linux-kerneliin, laitteistopuuhun ja Yocto-resepteihin tehtiin tarvittavat muutokset, joilla kaikki muokatun lopullisen emolevyn ominaisuudet saatiin otettua käyttöön käyttöjärjestelmän tasolla.

Laitteiston ominaisuuksien käyttöön ottamisen lisäksi tavoitteena oli toteuttaa ohjelmistoalustaan alustavasti etädiagnostiikka internetin välityksellä, alustan ohjelmiston sekä Qt 5 -versioon pohjautuvan mittaussovelluksen päivittäminen internetin välityksellä, virtuaalinen näppäimistö ja mahdollisuus mittaussovelluksen rinnalle asennettaviin apusovelluksiin. Tavoitteessa onnistuttiin siten, että kaikki vaaditut ominaisuudet saatiin alustavasti toteutettua Yocto-resepteihin.

Laitteistolla käytetty Freescalen i.MX6-järjestelmäpiiri osoittautui erittäin hyvin tuetuksi ja konfiguroitavuudeltaan erinomaiseksi. Freescalen dokumentaatio on kattava, ja sen lisäksi Freescalen internetin keskusteluryhmien vertaistuki osoittautui hyödylliseksi konfiguraation muokkauksessa. Erittäin laajan konfiguroitavuuden johdosta piirin konfiguroinnissa alkuunpääsy oli haastavaa.

Yocto osoittautui työkaluksi, jolla pääsee nopeasti alkuun, kun on olemassa evaluointiemolevy ja siihen valmiit Yocto-reseptit. Yocto on erittäin monipuolinen työkalu, ja siihen on olemassa kattava dokumentaatio. Lisäksi Yocton käyttö muiden i.MX6-järjestelmäpiiriin perustuvien laitteistojen valmistajien keskuudessa on laajaa, ja esimerkkejä erilaisista lisälaitteista ja konfiguraatioista löytyy paljon. Virallisissa

Yocton repositorioissa näistä on osa, mutta tämän lisäksi erityisesti Gateworksin [47] ja Varisciten [48] kattavat wiki-sivut ja julkisesti saatavilla olevat metadatakerrokset osoittautuivat erittäin hyödyllisiksi. Ohjelmistoalustan osuudessa Yocto-reseptien saatavuus Linuxissa käytettäville kirjastoille ja ohjelmistokomponenteille todettiin myös laajaksi. Valmiiden reseptien ja metadatakerrosten löytämisen helpottamiseksi on olemassa kattava hakemisto [49]. Viime aikoina on julkaistu myös enenevissä määrin kirjallisuutta Yocton käyttöön ja Yocton kattavan dokumentaation lisäksi ainakin yksi käytännönläheinen kirja on erittäin hyödyllinen.

Ohjelmistoalustalle valituista ohjelmistokomponenteista X11-pohjainen Matchbox-ikkunointijärjestelmä osoittautui projektin vaatimuksiin riittäväksi pienin muutoksin. X11vnc etäyhteys toimi ilman muutoksia erittäin luotettavasti. Maliit-sovelluskehykseen perustuva WebOS-näppäimistö osoittautui suuritöiseksi ja keskeneräiseksi. Toisaalta taas yksinkertaisemmat virtuaalinäppäismistöt eivät tarjonneet riitävästi tukea eri merkistöille ja yhteentoimivuutta Qt-sovelluksen tekstikenttien kanssa. Rpm-pakettienhallinta Yocton tuottamien pakettien kanssa toimi luotettavasti, mutta todelliseen testiin internetin yli tapahtuvaan laitteiden päivitykseen se ei vielä kerinnyt.

Alun jälkeen Yocton muokkaaminen osoittautui haastavaksi. Mielestäni monipuolisuuden ja oppimisen haastavuuden perusteella Yocto on verrattavissa Git-versionhallintaan – sillä erotuksella, että Yoctoon on saatavilla huomattavasti vähemmän tukea graafisilta työkaluilta. Muutosten tekeminen sekä metatiedon jaottelu useisiin kerroksiin ja resepteihin vaatii siis asiaan perehtymistä. Yocton lisäksi alustan kehittäminen vaatii perehtymistä laitteistoon, käynnistyslataajaan, laitteistopuuhun ja Linux-kerneliin.

Alkuvaiheen jälkeen tässä projektissa, jossa käyttöön otettavaa omaa laitteistoa on runsaasti, jää Yocto hetkittäin turhaksikin keskityttäessä käynnistyslataajaan, laitteistopuuhun ja kerneliin. Myöhemmin kuitenkin alustaa ylläpidettäessä Yocto-reseptit osoittautuivat erinomaiseksi tavaksi dokumentoida ja versioida alustaan tehdyt muutokset. Yocton tarjoamaa mahdollisuutta jakaa reseptit kerroksiin kannattaa myös hyödyntää, ja tässä työssä luotiinkin erillinen *Meta-yritys*-kerros, jossa on dokumentoitu ja versioitu kaikki alustaan projektissa tehdyt muutokset. Yocton aktiivinen kehittäjäyhteisö päivittää reseptejä säännöllisesti, ja näin oman jakelun saa pidettyä ajan tasalla tietoturvan ja kriittisten ohjelmistovirheiden osalta.

Yoctolla tuotettu Linux-pohjainen käyttöjärjestelmäjakelu on testikäytössä osoittautunut erittäin vakaaksi ja kaikki vakausongelmat ovat osoittautuneet lopullisen emolevyn prototyypin laitteisto-ongelmista johtuviksi, ja ne ovat korjautuneet pie-

nin laitteistomuutoksin. Laitteiston suorituskyvylle, kuten paikannusmoduulin paikannustarkkuudelle ja WLAN-moduulin tiedonsiirtokyvylle, tehtiin yksinkertaisia testejä, ja ne osoittautuivat ominaisuuksiltaan käyttötarkoituksiinsa riittäviksi.

Työn tekeminen jakautui kahteen vaiheeseen. Projektin alussa samaan aikaan aloitettiin emolevyn suunnittelu, jolloin ominaisuuksia ei heti päästy testaamaan lopullisella laitteistolla. Evaluointiemolevy mahdollisti kuitenkin ohjelmistoalustan osuuden kehittämisen ja testaamisen ennen lopullisen emolevyn valmistumista. Tämän johdosta Yoctoon tutustuminen ja ohjelmistoalustan ominaisuuksien, kuten etädiagnostiikan, ikkunointiympäristön ja virtuaalinäppäimistön, kehitys tehtiin ensin. Tämän jälkeen lopullisen emolevyn valmistuttua aloitettiin käynnistyslataajan, laitteistopuun ja Linux-kernelin muokkaaminen laitteistolle sopivaksi. Evaluointiemolevyllä kehitettyjen ominaisuuksien käyttöön ottaminen lopullisella emolevyllä ei tuottanut ongelmia.

Koko ohjelmistoalustan toteutuksessa yhtäkään ajuria tai Yocto-reseptiä ei tarvinnut toteuttaa tyhjästä vaan kaikki pystyttiin muokkaamaan valmiiden mallien avulla. Tämän diplomityön pohjalta ajoneuvo-PC:n mittaussovellukselle saadaan kehitettyä tarvittava ohjelmistoalusta. Tämän ohjelmistoalustan komponentit vaativat kuitenkin vielä jatkokehitystä, jotta ne ovat valmiita tuotantokäyttöön.

8.2 Työn arviointi

Yrityksellä ei ollut aiempaa kokemusta vastaavan laitteen kehittämisestä, vaan edellisen sukupolven laitteen Linux-ohjelmistoalusta oli toteutettu alihankkijalla. Laitteiston käyttöönottoa olisi saattanut helpottaa, jos olisi ollut jotakin kautta saatavilla Linux-asiantuntemusta vastaavien laitteiden käyttöönotosta. Laitteisto-ongelmien ennaltaehkäisemiseen myös tiiviimpi yhteydenpito laitteiston suunnitelleeseen alihankkijaan olisi voinut vähentää prototyyppiemolevyn käyttöönotossa ilmenneitä ongelmia. Laitteisto-ongelmissa kuitenkin kollegoiden vahvasta elektroniikka osaamisesta oli huomattavan suuri apu. Lopulta käyttöönotto onnistui muutamista prototyyppi emolevyn laitteisto-ongelmista ja i.MX6-järjestelmäpiirin kompleksisuudesta huolimatta.

Työn kirjallinen osuus olisi ollut mielestäni helpompi tuottaa kokonaan vasta kun projekti olisi ehtinyt pidemmälle. Laajempi ymmärrys Yoctosta ja sen tarjoamista työkaluista muodostui vasta kun varsinaisen laitteen käyttöönotto oli suoritettu.

Sulautettuun Linux-projektiin valitsisin uudessakin projektissa käyttöön Yocton työkalut, koska niissä on huomioitu useat vastaavan projektin haasteet. X11 on vielä

parempi valinta uuteenkiin projektiin tänä vuonna, mutta jatkossa on mielenkiintoista seurata kuinka Wayland ja tuki sille esimerkiksi Qt-sovelluskehyksen, ikkunointijärjestelmien ja etädiagnostiikan osalta kehittyy. Näin ollen pidän projektissa tehtyjä teknologiavalintoja onnistuneina. Toteutus on vielä osittain keskeneräinen, mutta kokonaisuutena projektia voidaan mielestäni pitää onnistuneena.

8.3 Jatkotutkimuskohteet

Seuraavana emolevystä tehdään uusi laitteistoversio, jossa ensimmäisessä prototyypissä käyttöönoton yhteydessä havaitut ongelmat on korjattu. Uuden emolevyn käyttöönotto vaatii myös ohjelmiston muokkausta vastaamaan muuttunutta laitteistoa, mutta pääosin sen pitäisi onnistua poistamalla nykyiset laitteisto-ongelmat kiertäneet ohjelmistokorjaukset pois käytöstä. Ohjelmistoalustan kehitys jatkuu ja nyt tavoitteena on saada ensimmäinen demo-laitteisto asiakkaiden kokeiltavaksi puolen vuoden päästä.

9. YHTEENVETO

Tämä työ oli osa tuotekehitysprojektia, jossa kehitettiin yritykselle uusi laitteisto- ja ohjelmistoalusta mittaustulosten esittämiseen ja synkronointiin internet-palveluun. Lopullinen kohdelaitteisto oli tablettia muistuttava seitsemän tuumainen kosketusnäytöllä ohjattava ajoneuvo-PC. Moduuliin perustuva laitteistorakenne mahdollisti ohjelmistokehityksen aloituksen samaan aikaan laitteiston kehityksen kanssa. Projektin alussa tutkittiin kehitystyökaluja ja aloitettiin Yocto Linux alustan ohjelmistojen muokkaus vastaamaan vaatimuksia. Yocton käyttö monimutkaisemmissa käyttötapauksissa vaatii laajaa perehtymistä monipuolisen konfiguroitavuuden ja puuttellisten graafisten työkalujen vuoksi. Projektin puolivälissä otettaessa oma laitteisto käyttöön vaihdettiin alun ohjelmistokehityksessä käytetty evaluointiemolevy lopulliseen emolevyyn. Laitteistokehityksen osuus ei kuulunut tähän diplomityöhön.

Ohjelmistoalustan toteutuksen pohjana käytettiin laitteistovalmistaja Kontronin tarjoamaa laitteistotukipakettia Yocto Linuxille. Ohjelmistoalusta koostui Yocto-pohjaisesta muokatusta Linux-jakelusta, joka tarkemmin jakautui U-boot-käynnistyslataajaan, Linux-kerneliin, laitteistopuuhun ja Linuxin kirjastoihin ja sovellusohjelmiin. Projektissa kaikkiin komponentteihin tehtiin tarvittavat muutokset niin, että laitteiston ominaisuudet kuten kosketusnäyttö ja 3G/GLONASS-moduuli saatiin käyttöön.

Yksiytiminen Freescalen i.MX6 -järjestelmäpiiri jonka lisäksi Kontron on SMARC-sAMX6i Solo moduuliin asentanut 512 megatavua RAM-muistia ja 4 gigatavua eMMC-flashmuistia osoittautui erittäin monipuolisesti konfiguroitavissa olevaksi laitteeksi. Laaja konfiguroitavuus osoittautui myös haasteeksi, joka vaati perusteellista ominaisuuksien tutkimista manuaaleista.

Yocto-muutokset dokumentoitiin ja niitä ylläpidetään omassa Meta-yritys-kerroksessa. U-boot käynnistyslataajalle ei laadittu Yocto-reseptiä, koska U-bootille ei ollut tarjolla Yocto-tukea valmistajalta. Lisäksi yhdistetyn WLAN/Bluetooth-moduulin tarvitsemien kernel-moduulien ja apusovelluksien käännöstä ei sisällytetty Yocto-resepteihin, koska moduulin laitteistovalmistaja tarjosi tarkoitukseen tehdyt komentosarjat, joilla tarvittavat lisäykset tehdään Yocton tuottamaan juuren tiedostojär-

jestelmään suoraan.

QML-käyttöliittymään ja Qt 5.4.2 versioon pohjautuva mittaussovellus saatiin toimimaan alustalla sulavasti. X11-ikkunointiin perustuva Matchbox-ikkunointiympäristö ja X11vnc-etäkäyttö osoittautuivat toimiviksi ratkaisuiksi. Maliit-sovelluskehykseen pohjautuva virtuaalinäppäimistöratkaisu oli työläimpiä yksittäisiä ohjelmistokomponentteja alustalle sovitettavaksi ja tämä komponentti vaatii vielä jatkokehitystä. Yocton tuottamien Rpm-pakettien asennus ja päivittäminen testattiin paikallisesti, mutta internet-pohjaisen pakettivaraston toteuttaminen jäi jatkokehitykseen.

Lopputuloksena oli siis Yocto-reseptit ja niihin liittyvät muutostiedostot, joilla saa tuotettua vakaan ja ylläpidettävän ohjelmiston tälle laitteistoalustalle. Projekti onnistui pääosin hyvin ja tuotettu ohjelmistoalusta täyttää sille etukäteen asetetut tavoitteet. Kehitys kuitenkin jatkuu vielä ennen kuin laite on valmis tuotantoon.

LÄHTEET

[1] Salvador, O. & Angolini, D. 2014. Embedded Linux Development with Yocto Project, 1. painos. Packt Publishing Ltd. E-book.

[2] González, A. 2015. Embedded Linux Projects Using Yocto Project Cookbook, 1. painos. Packt Publishing Ltd. E-book.

[3] Kontron SMARC-sAMX6i specification. [WWW]. [viitattu 21.1.2015]. Saatavissa: http://www.kontron.com/products/computeronmodules/smarc/smarc-samx6i/specification

[4] Kontron User's guide SMARC-sAMX6i, Document revision 1.0. [WWW]. [viitattu 20.5.2015]. Saatavissa: http://www.kontron.com/downloads/manual/smx6m100.pdf?product=89810

[5] i.MX6 series. [WWW]. [viitattu 21.1.2015]. Saatavissa: http://en.wikipedia.org/wiki/I.MX#i.MX6x_series

[6] Yocto Project Reference Manual. [WWW]. [viitattu 22.3.2016]. Saatavissa: http://www.yoctoproject.org/docs/1.8/ref-manual/ref-manual.html

[7] Yocto Project. [WWW]. [viitattu 23.1.2015]. Saatavissa: http://en.wikipedia.org/wiki/Yocto_Project

[8] Yocto Project Aligns Technology with OpenEmbedded and Gains Corporate Collaborators. [WWW]. [viitattu 25.1.2015]. Saatavissa: http://www.linuxfoundation.org/news-media/announcements/2011/03/yocto-project-aligns-technology-openembedded-and-gains-corporate-co

[9] BitBake User Manual. [WWW]. [viitattu 22.3.2016]. Saatavissa: http://www.yoctoproject.org/docs/1.8/bitbake-user-manual/bitbake-user-manual.html

[10] Yocto Project Wiki - FAQ. [WWW]. [viitattu 27.1.2015]. Saatavissa: https://wiki.yoctoproject.org/wiki/FAQ

[11] Turnbull, J., Lieverdlink, P. & Matotek, D. 2009. Pro Linux System Administration, 1. painos. Apress. E-book.

[12] dpkg(1) - Linux man page. [WWW]. [viitattu 5.2.2015]. Saatavissa: http://linux.die.net/man/1/dpkg

[13] rpm(8) - Linux man page. [WWW]. [viitattu 5.2.2015]. Saatavissa: http://linux.die.net/man/8/rpm

[14] Hertzog, R. & Mas, R. 2013. The Debian Administrator's Handbook, 2. painos. Apress. E-book.

[15] Freescale Platform for Smart Devices Reference Design Based on the i.MX 6 Series. [WWW]. [viitattu 18.2.2015]. Saatavissa: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RDIMX6SABREPLAT

[16] Linux and the Device Tree [WWW]. [viitattu 2.3.2015]. Saatavissa: https://www.kernel.org/doc/Documentation/devicetree/usage-model.txt

[17] i.MX 6Solo/6DualLite Applications Processors for Industrial Products [WWW]. [viitattu 24.4.2015]. Saatavissa: http://cache.freescale.com/files/32bit/doc/data_sheet/IMX6SDLIEC.pdf

[18] Qt 5.4 for Device Creation - Supported Platforms [WWW]. [viitattu 24.4.2015]. Saatavissa: http://doc.qt.io/QtForDeviceCreation/qtee-supported-platforms.html

[19] Yocto Project - Openembedded Core [WWW]. [viitattu 10.5.2015]. Saatavissa: https://www.yoctoproject.org/tools-resources/projects/openembedded-core

[20] Abbott, D. 2013. Linux for Embedded and Real-Time Applications, 3. painos. Elsevier Inc. E-book.

[21] U-boot lähdekoodi [WWW]. [viitattu 20.5.2015]. Saatavissa: http://git.denx.de/?p=u-boot.git

[22] Yaghmour, K., Masters, J., Ben-Yossef, G. & Gerum, P. 2009. Building Embedded Linux Systems, 2. painos. O'Reilly Media. E-book.

[23] Smart Mobility Architecture Design Guide [WWW]. [viitattu 22.2.2016]. Saatavissa: http://www.sget.org/fileadmin/_migrated/content_uploads/SMARC_DG_V1p0.pdf

[24] Smart Mobility Architecture Hardware Specification [WWW]. [viitattu 22.2.2016]. Saatavissa: http://www.sget.org/fileadmin/_migrated/content_uploads/SMARC_Hardware_Specification_V1p1.pdf

[25] FSL Community BSP Release Notes 1.8 documentation [WWW]. [viitattu 19.7.2015]. Saatavissa: http://freescale.github.io/doc/release-notes/1.8/index.html

[26] Bovet, D. & Cesati, M. 2005. Understanding the Linux Kernel, 3. painos. O'Reilly Media. E-book.

[27] Vaduva, A. 2015. Learning Embedded Linux Using the Yocto Project, 1. painos. Packt Publishing Ltd.

[28] Corbet, J., Kroah-Hartman, G. & McPherson, A. Linux Kernel Development. A Linux Foundation publication. 2015. Saatavissa: http://www.linuxfoundation.org/publications/linux-foundation/who-writes-linux-2015

[29] The DENX U-boot and Linux Guide (DULG) for canyonlands [WWW]. [viitattu 21.9.2015]. Saatavissa: http://www.denx.de/wiki/publish/DULG/DULG-canyonlands.pdf

[30] Power.org Standard for Embedded Power Architecture Platform Requirements [WWW]. [viitattu 21.9.2015]. Saatavissa: https://www.power.org/wp-content/uploads/2012/06/Power_ePAPR_APPROVED_v1.1.pdf

[31] Doran, M., Rothman, M. & Zimmer, V. Beyond BIOS: Exploring the Many Dimensions of the Unified Extensible Firmware Interface [WWW]. [viitattu 21.9.2015]. Saatavissa: http://www.researchgate.net/publication/235258563

[32] Device Tree Usage [WWW]. [viitattu 22.9.2015]. Saatavissa: http://devicetree.org/Device_Tree_Usage

[33] i.MX 6Solo/6DualLite Applications Processor Reference Manual, Rev. 2 [WWW]. [viitattu 22.9.2015]. Saatavissa: http://cache.freescale.com/files/32bit/doc/ref_manual/IMX6SDLRM.pdf

[34] Specifying interrupt information for devices [WWW]. [viitattu 22.9.2015]. Saatavissa: https://www.kernel.org/doc/Documentation/devicetree/bindings/interrupt-controller/interrupts.txt

[35] Sun Microsystems - OpenBootCommand Reference [WWW]. [viitattu 23.9.2015]. Saatavissa: https://docs.oracle.com/cd/E19457-01/801-7042/801-7042.pdf

[36] Sun SPARCstation 1 (Sun 4/60) [WWW]. [viitattu 23.9.2015]. Saatavissa: http://www.computinghistory.org.uk/det/22017/Sun-SPARCstation-1-(Sun-4-60)/

[37] Love, R. 2010. Linux Kernel Development, 3. painos. Addison-Wesley. E-book.

[38] Experiment with Yocto [WWW]. [viitattu 27.11.2015]. Saatavissa: http://free-electrons.com/blog/experiment-with-yocto/

[39] Yocto Project Wiki - Yocto Architecture [WWW]. [viitattu 27.11.2015]. Saatavissa: https://wiki.yoctoproject.org/wiki/Yocto_Architecture

[40] Yocto Project Quick Start. [WWW]. [viitattu 22.3.2016]. Saatavissa: http://www.yoctoproject.org/docs/1.8/yocto-project-qs/yocto-project-qs.html

[41] [meta-freescale] packagegroup-core-x11-utils errors [WWW]. [viitattu 4.12.2015]. Saatavissa: https://lists.yoctoproject.org/pipermail/meta-freescale/2013-May/002682.html

[42] [meta-freescale] packagegroup-core-x11-utils errors still exists. [WWW]. [viitattu 4.12.2015]. Saatavissa: https://lists.yoctoproject.org/pipermail/meta-freescale/2014-August/010000.html

[43] IEEE, IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices. Tech. Rep. IEEE, 1994. Saatavissa: http://ieeexplore.ieee.org/servlet/opac?punumber=6199

[44] Intel Architecture Firmware Resource Center: MinnowBoard MAX. [WWW]. [viitattu 6.12.2015]. Saatavissa: http://firmware.intel.com/projects/minnowboard-max

[45] Texas Instruments Wiki - WL18xx First Time Getting Started Guide (IMX6). [WWW]. [viitattu 6.12.2015]. Saatavissa: http://processors.wiki.ti.com/index.php/WL18xx_First_Time_Getting_Started_Guide_%28IMX6%29

[46] Texas Instruments Wiki - WLAN Throughput Test. [WWW]. [viitattu 6.12.2015]. Saatavissa: http://processors.wiki.ti.com/index.php/WLAN_Throughput_Test

[47] Gateworks Wiki - Building Yocto & Installing for the Gateworks Ventana Family. [WWW]. [viitattu 6.12.2015]. Saatavissa: http://trac.gateworks.com/wiki/Yocto/Building

[48] Variscite Wiki - VAR-SOM-MX6 - Yocto fsl-3.14.38_6qp Fido R2. [WWW]. [viitattu 6.12.2015]. Saatavissa: http://processors.wiki.ti.com/index.php/WLAN_Throughput_Test

[49] OpenEmbedded Metadata Index. [WWW]. [viitattu 22.3.2016]. Saatavissa: http://layers.openembedded.org/layerindex/branch/fido/layers/

LIITE A. RESEPTI WEBOS NÄPPÄIMISTÖN ALUSTAAN SISÄLLYTTÄMISEEN

```
SUMMARY = "webOS on-screen keyboard based on the Ubuntu Touch \
keyboard"
HOMEPAGE = "https://launchpad.net/ubuntu-keyboard"
LICENSE = "GPL-3.0 & BSD & Apache-2.0 & CC-BY-1.0"
LIC_FILES_CHKSUM = " \
file://COPYING;md5=6a6a8e020838b23406c81b19c1d46df6 \
file://COPYING.BSD;md5=9b2310382ed07cfdae9c4953c8d29078 \
file://COPYING.Apache-2.0;beginline=82;endline=257;\
md5=e23fadd6ceef8c618fc1c65191d846fa \
file://COPYING.CC-BY;md5=c14dd4d440694f070fc6520d9c9a65eb \
"

inherit qmake5

DEPENDS = "maliit-framework-qt5 (= 0.99.1) hunspell presage \
libpinyin"

RDEPENDS_${PN} += "qtsvg-plugins qtmultimedia-qmlplugins"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

SRCREV = "982f27d4be4e8b2f5c5fa655d7d1719a32481844"
PV = "0.99.1+git${SRCPV}"

SRC_URI = "\
git://github.com/webOS-ports/webos-keyboard.git;branch=master \
file://0001-CPP-compilation-without-LuneOS-dependencies.patch \
file://0002-WIP-Get-rid-off-LunaNext.Common-QML-module-import.patch \
file://0003-Remove-dependency-of-Units.gu-in-LunaNext.Common.patch \
file://0004-Add-UI.qml-to-qml.pro.patch \
file://0005-Remove-LunaNext.Common-import-from-UI.qml.patch \
file://0006-Hardcode-formfactor-to-tablet.patch \
file://0007-Remove-LunaNext.Common-dependency-from-ExtendedListS.patch \
file://0008-Remove-dependency-of-Units.gu-in-Keyboard.qml.patch \
file://0009-Add-missing-files-to-qml.pro.patch \
file://0010-Get-rid-off-LunaNext.Common-QML-module-import-in-las.patch \
file://0011-TESTING-Remove-language-support-for-other-than-en-fi.patch \
file://0012-Remove-dependency-of-Units.gu-in-rest-of-the-files.patch \
file://0013-Remove-dependency-of-Units.dp-in-LunaNext.Common.patch \
file://0014-Get-rid-off-FontUtils-in-LunaNext.Common.patch \
file://0015-Default-to-phone-UI.patch \
file://0016-Set-keyboardSizeChoice-to-L.patch \
file://0017-Fix-L-size-to-take-half-of-the-screen-on-1024-screen.patch \
"

EXTRA_QMAKEVARS_PRE = "\
    PREFIX=${prefix} \
    MALIIT_INSTALL_PRF=${QMAKE_MKSPEC_PATH}/mkspecs/features \
    MALIIT_PLUGINS_DATA_DIR=${datadir} \
    LIBDIR=${libdir} \
    CONFIG+=nodoc \
    CONFIG+=notests \
```

```
    CONFIG+=enable-presage \
    CONFIG+=enable-hunspell \
    CONFIG+=enable-pinyin \
"

INSANE_SKIP_${PN} += "libdir staticdev"
INSANE_SKIP_${PN}-dbg += "libdir"

FILES_${PN} += "\
    ${libdir}/maliit \
    ${datadir} \
"

FILES_${PN}-dbg += "\
${libdir}/maliit/plugins/.debug \
${datadir}/maliit/plugins/org/luneos/lib/*/.debug \
"

S = "${WORKDIR}/git"

EXTRA_OEMAKE += "INSTALL_ROOT=${D}"
```

LIITE B. OTE LAITTEISTOPUUSTA JOKA KUVAA KOSKETUSSENSORIN

```
/ {
  i2c3 {
    pinctrl_smx6_i2c3: i2c3grp-smx6 {
      fsl,pins = <
        MX6QDL_PAD_GPIO_5__I2C3_SCL  0x4001f8b1
        MX6QDL_PAD_GPIO_16__I2C3_SDA 0x4001f8b1
      >;
    };
  };

  &i2c3 {
    clock-frequency = <400000>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_smx6_i2c3>;
    status = "okay";

    egalax_ts@2a {
      compatible = "eeti,egalax_ts";
      reg = <0x2a>;
      interrupt-parent = <&gpio3>;
      /* 9 = pin, 8 = active low level-sensitive */
      interrupts = <9 8>;
      wakeup-gpios = <&gpio3 9 0>;
    };
  };
};
```

LIITE C. KONTRONIN LINUX-KERNEL YOCTO RESEPTI

```
SUMMARY = "Linux Kernel provided and supported by Freescale with \
patches for Kontron sAMX6i"
DESCRIPTION = "Linux Kernel provided and supported by Freescale \
with focus on i.MX Family Reference Boards, patched for Kontron \
sAMX6i. It includes support for many IPs such as GPU, VPU and IPU."

require recipes-kernel/linux/linux-imx.inc
require recipes-kernel/linux/linux-dtb.inc

DEPENDS += "lzop-native bc-native"

SRCBRANCH = "imx_3.14.28_1.0.0_ga"
SRCREV = "91cf351a2afc17ac4a260e4d2ad1e32d00925a1b"
LOCALVERSION = ""

COMPATIBLE_MACHINE = "smarc-samx6i"

SRC_URI += " \
file://0001-ARM-imx6q-drop-unnecessary-semicolon.patch \
file://0002-ARM-clk-imx6q-fix-video-divider-for-rev-T0-1.0.patch \
file://0003-ARM-imx6sl-Disable-imx6sl-specific-code-when-imx6sl-.patch \
file://0004-mmc-sdhci-esdhc-imx-Fixup-runtime-PM-conditions-duri.patch \
file://0005-Revert-net-fec-fix-the-warning-found-by-dma-debug.patch \
file://0006-arch-arm-mach-imx-avoid-link-errors-without-SOC_IMX6.patch \
file://0007-mxc-config-make-MXC_GPU_VIV-to-select-RESET_CONTROLL.patch \
file://0008-spi-nor-add-winbond-w25q16dw-and-w25q46dw-to-chip-ta.patch \
file://0009-backlight-gpio-backlight-Add-DT-support.patch \
file://0010-backlight-gpio-backlight-Fix-warning-when-the-GPIO-i.patch \
file://0011-pwm-backlight-Disable-backlight-on-shutdown.patch \
file://0012-backlight-introduce-an-OF-property-to-match-framebuf.patch \
file://0013-mxc-lcdif-add-native-video-mode-for-NLT-1280x768-pan.patch \
file://0014-ethernet-fec-fix-crash-on-TX-queue-timeout.patch \
file://0015-mxc_v4l2_capture-rework.patch \
file://0016-asoc-support-for-imx6-wm8903-combo.patch \
file://0017-pci-imx6-do-not-access-registers-before-enabling-clo.patch \
file://0018-pci-imx6-make-it-possible-to-disable-gen2.patch \
file://0019-pci-imx6-allow-up-to-4-reset-GPIOs.patch \
file://0020-mxc_hdmi-add-DT-property-to-accept-all-modes-from-ED.patch \
file://0021-smx6-device-tree-files-for-Kontron-SMARC-sAMX6.patch \
file://0022-mxc_v4l2_capture-more-cleanup.patch \
file://0023-mxc-capture-Kconfig-cleanup.patch \
file://0024-imx6qdl.dtsi-cleanup-audio-definition.patch \
file://0025-ov5640_mipi-use-clk_get_rate-instead-of-device-tree-.patch \
file://0026-ov5640_mipi-do-not-mix-MIPI-CSI-virtual-channel-id-w.patch \
file://0027-mach-imx6q-MIPI-CSI-iomux-configuration-for-smx6-boa.patch \
file://0028-imx6qdl-smx6.dtsi-add-serial-camera-support.patch \
file://0029-imx-pci-make-link-up-timeout-configurable.patch \
file://0030-sdhci-recompute-timeout_clk-when-needed.patch \
file://0031-clk-imx6q-do-not-try-to-set-unsupported-gpu2d_core_s.patch \
file://0032-gpio-generic-clamp-retured-value-to-0-1.patch \
file://0033-arm-don-t-add-devtree-defined-CPUs-to-cpu_possible_m.patch \
file://0034-ARM-mx6-Only-check-for-1.2GHz-for-mx6quad.patch \
```

```
file://0035-ARM-dts-imx-drop-invalid-size-and-address-cells-prop.patch \
file://0036-spi-imx-use-pio-mode-for-i.mx6dl.patch \
file://0037-kontron-smarc-evaluation-carrier.dtsi-add-accelerome.patch \
file://0038-imx6qdl-smx6.dtsi-restore-hdmi-mode_str-setting.patch \
file://0039-imx6qdl-smx6.dtsi-support-swapping-primary-and-secon.patch \
file://0040-imx6qdl-smx6.dtsi-support-configurable-framebuffer-d.patch \
file://0041-can-flexcan-Deferred-on-Regulator-return-EPROBE_DEFE.patch \
file://0042-imx6qdl-smx6.dtsi-enable-wake-on-lan.patch \
file://0043-smx6-move-pm-eeprom-to-imx6qdl-smx6.dtsi.patch \
file://0044-smx6-device-tree-rework.patch \
file://0045-smx6-support-PTP.patch \
file://0046-smx6-restore-32bpp-framebuffer-for-LCD-LVDS-display.patch \
file://0047-mxc_v4l2_capture-allow-non-changing-VIDIOC_S_INPUT-w.patch \
file://0048-smx6-support-pcb_veraion-0-boards.patch \
"

  KERNEL_DEVICETREE_append += "${@' \
  '.join(os.path.splitext(os.path.basename(x))[0] \
   + '.dtb' for x in (d.getVar('SMX6_CUSTOM_DTS', True) or \
   '').split() if x.endswith('.dts'))}"

  do_import_dts() {
   cd ${TOPDIR}
   if [ -n "${SMX6_CUSTOM_DTS}" ]; then
   cp ${SMX6_CUSTOM_DTS} ${S}/arch/arm/boot/dts/
   fi
  }
  addtask import_dts after do_patch before do_configure"
```

LIITE D. KONTRONIN LINUX-KERNEL YOCTO RESEPTISTÄ MUOKATTU YRITYS KERNEL

```
SUMMARY = "Linux Kernel provided and supported by Freescale with \
patches for Kontron sAMX6i and TMR-1443"
DESCRIPTION = "Linux Kernel provided and supported by Freescale \
with focus on i.MX Family Reference Boards, patched for Kontron \
sAMX6i. It includes support for many IPs such as GPU, VPU and IPU."

require recipes-kernel/linux/linux-imx.inc
require recipes-kernel/linux/linux-dtb.inc

DEPENDS += "lzop-native bc-native"

SRCBRANCH = "imx_3.14.28_1.0.0_ga"
SRCREV = "91cf351a2afc17ac4a260e4d2ad1e32d00925a1b"
LOCALVERSION = ""

COMPATIBLE_MACHINE = "smarc-samx6i"

SRC_URI += " \
file://0001-ARM-imx6q-drop-unnecessary-semicolon.patch \
file://0002-ARM-clk-imx6q-fix-video-divider-for-rev-T0-1.0.patch \
file://0003-ARM-imx6sl-Disable-imx6sl-specific-code-when-imx6sl-.patch \
file://0004-mmc-sdhci-esdhc-imx-Fixup-runtime-PM-conditions-duri.patch \
file://0005-Revert-net-fec-fix-the-warning-found-by-dma-debug.patch \
file://0006-arch-arm-mach-imx-avoid-link-errors-without-SOC_IMX6.patch \
file://0007-mxc-config-make-MXC_GPU_VIV-to-select-RESET_CONTROLL.patch \
file://0008-spi-nor-add-winbond-w25q16dw-and-w25q46dw-to-chip-ta.patch \
file://0009-backlight-gpio-backlight-Add-DT-support.patch \
file://0010-backlight-gpio-backlight-Fix-warning-when-the-GPIO-i.patch \
file://0011-pwm-backlight-Disable-backlight-on-shutdown.patch \
file://0012-backlight-introduce-an-OF-property-to-match-framebuf.patch \
file://0013-mxc-lcdif-add-native-video-mode-for-NLT-1280x768-pan.patch \
file://0014-ethernet-fec-fix-crash-on-TX-queue-timeout.patch \
file://0015-mxc_v4l2_capture-rework.patch \
file://0016-asoc-support-for-imx6-wm8903-combo.patch \
file://0017-pci-imx6-do-not-access-registers-before-enabling-clo.patch \
file://0018-pci-imx6-make-it-possible-to-disable-gen2.patch \
file://0019-pci-imx6-allow-up-to-4-reset-GPIOs.patch \
file://0020-mxc_hdmi-add-DT-property-to-accept-all-modes-from-ED.patch \
file://0021-smx6-device-tree-files-for-Kontron-SMARC-sAMX6.patch \
file://0022-mxc_v4l2_capture-more-cleanup.patch \
file://0023-mxc-capture-Kconfig-cleanup.patch \
file://0024-imx6qdl.dtsi-cleanup-audio-definition.patch \
file://0025-ov5640_mipi-use-clk_get_rate-instead-of-device-tree-.patch \
file://0026-ov5640_mipi-do-not-mix-MIPI-CSI-virtual-channel-id-w.patch \
file://0027-mach-imx6q-MIPI-CSI-iomux-configuration-for-smx6-boa.patch \
file://0028-imx6qdl-smx6.dtsi-add-serial-camera-support.patch \
file://0029-imx-pci-make-link-up-timeout-configurable.patch \
file://0030-sdhci-recompute-timeout_clk-when-needed.patch \
file://0031-clk-imx6q-do-not-try-to-set-unsupported-gpu2d_core_s.patch \
file://0032-gpio-generic-clamp-retured-value-to-0-1.patch \
file://0033-arm-don-t-add-devtree-defined-CPUs-to-cpu_possible_m.patch \
file://0034-ARM-mx6-Only-check-for-1.2GHz-for-mx6quad.patch \
```

```
file://0035-ARM-dts-imx-drop-invalid-size-and-address-cells-prop.patch \
file://0036-spi-imx-use-pio-mode-for-i.mx6dl.patch \
file://0037-kontron-smarc-evaluation-carrier.dtsi-add-accelerome.patch \
file://0038-imx6qdl-smx6.dtsi-restore-hdmi-mode_str-setting.patch \
file://0039-imx6qdl-smx6.dtsi-support-swapping-primary-and-secon.patch \
file://0040-imx6qdl-smx6.dtsi-support-configurable-framebuffer-d.patch \
file://0041-can-flexcan-Deferred-on-Regulator-return-EPROBE_DEFE.patch \
file://0042-imx6qdl-smx6.dtsi-enable-wake-on-lan.patch \
file://0043-smx6-move-pm-eeprom-to-imx6qdl-smx6.dtsi.patch \
file://0044-smx6-device-tree-rework.patch \
file://0045-smx6-support-PTP.patch \
file://0046-smx6-restore-32bpp-framebuffer-for-LCD-LVDS-display.patch \
file://0047-mxc_v4l2_capture-allow-non-changing-VIDIOC_S_INPUT-w.patch \
file://0048-smx6-support-pcb_veraion-0-boards.patch \
file://0049-tmr1443-dt-add-initial-devicetree-based-on-imx6dl-sm.patch \
file://0050-tmr1443-dt-remove-unnecessary-audio-define-and-carri.patch \
file://0051-tmr1443-dt-Chefree-CH070DLDL-CTX-display-parameters.patch \
file://0052-Input-egalax_ts-add-support-for-exc3132-touch-contro.patch \
file://0053-tmr1443-dt-add-eeti-egalax-exc3132.patch \
file://0054-smx6-add-defconfig-received-from-Kontron.patch \
file://0055-tmr1443-config-add-eeti-egalax-driver-as-built-in.patch \
file://0056-tmr1443-dt-take-serial-ports-for-BT-printer-and-debu.patch \
file://0057-tmr1443-dt-add-USB-ports-to-be-enabled.patch \
file://0058-usb-option-add-Quectel-device-id-for-option-driver.patch \
file://0059-tmr1443-config-add-usb-serial-drivers-for-UC20.patch \
file://0060-tmr1443-config-add-PPP-connection-for-UC20.patch \
file://0061-mach-imx6q-Fix-parallel-interface-for-smx6-boards.patch \
file://0062-tmr1443-config-add-ADV7180-camera-input-support.patch \
file://0063-tmr1443-dt-add-support-for-ADV7180-camera-input.patch \
file://0064-adv7180-disable-POWER_DOWN-pin.patch \
file://0065-tmr1443-dt-add-support-for-wilink8-wlan-and-bt.patch \
file://0066-Bluetooth-Add-tty-HCI-driver.patch \
file://0067-tmr1443-config-smx6_defconfig-enable-Wilink8-related.patch \
file://0068-st_kim-do-not-use-debugfs-functions-if-not-enabled.patch \
file://0069-st_kim-allow-suspend-if-callback-is-not-registered.patch \
file://0070-btwilink-add-minimal-device-tree-support.patch \
file://0071-ti-st-add-device-tree-support.patch \
file://0072-tmr1443-dt-change-47k-pull-up-to-100k-pull-down-as-i.patch \
file://0073-tmr1443-dt-decrease-wilink8-sdio-pins-drive-strength.patch \
file://0074-mmc-Add-SDIO-function-devicetree-subnode-parsing.patch \
"

KERNEL_DEVICETREE_append += "${@' \
'.join(os.path.splitext(os.path.basename(x))[0] \
 + '.dtb' for x in (d.getVar('SMX6_CUSTOM_DTS', True) or \
 '').split() if x.endswith('.dts'))}"

do_import_dts() {
  cd ${TOPDIR}
  if [ -n "${SMX6_CUSTOM_DTS}" ]; then
    cp ${SMX6_CUSTOM_DTS} ${S}/arch/arm/boot/dts/
  fi
}
addtask import_dts after do_patch before do_configure"
  "
```

LIITE E. KONTRONIN LAITTEISTOKONFIGURAATIO MUOKATTU TMR-1443 LAITTEELLE

```
#@TYPE: Machine
#@NAME: Kontron SMARC-sAMX6i
#@SOC: i.MX6Q
#@DESCRIPTION: Machine configuration for Kontron SMARC-sAMX6i with \
changes for TMR-1443
#@MAINTAINER: Heikki Sarkanen <heikki.sarkanen@student.tut.fi>

require conf/machine/include/imx-base.inc
require conf/machine/include/tune-cortexa9.inc

SERIAL_CONSOLE = "115200 ttymxc4"

MACHINE_FEATURES += "serial pci usbhost usbgadget touchscreen wifi \
3g bluetooth"

SOC_FAMILY = "mx6:mx6q"

KERNEL_DEVICETREE = "imx6dl-smx6-lvds.dtb"

IMAGE_BOOTLOADER = ""
IMAGE_FSTYPES = "tar.bz2"

PREFERRED_PROVIDER_virtual/kernel = "linux-smx6"

KERNEL_IMAGETYPE = "uImage"
```

LIITE F. YOCTO RESEPTI LEVYKUVAN TUOTTAMISEEN TMR-1443 LAITTEELLE

```
SUMMARY = "Company specific Kontron SMARC sAMX6i X11 Sato image"

IMAGE_FEATURES += "splash package-management x11 x11-base x11-sato \
hwcodecs"

LICENSE = "MIT"

inherit core-image distro_features_check populate_sdk \
populate_sdk_qt5

REQUIRED_DISTRO_FEATURES = "x11"
CONFLICT_DISTRO_FEATURES = "wayland directfb"

QT_LIBRARIES = "\
qtbase-fonts qtbase-plugins qtbase-tools \
qtconnectivity-qmlplugins qtdeclarative qtdeclarative-plugins \
qtdeclarative-qmlplugins qtdeclarative-tools \
qtgraphicaleffects-qmlplugins qtimageformats-plugins \
qtlocation-plugins qtlocation-qmlplugins qtmultimedia \
qtmultimedia-plugins qtmultimedia-qmlplugins \
qtquickcontrols-qmlplugins qtsensors qtserialport qtsvg \
qtsvg-plugins qtsystems qtsystems-tools qtsystems-qmlplugins \
qtx11extras"

QT_EXAMPLES = "\
cinematicexperience qt5-demo-extrafiles qt5everywheredemo \
qt5nmapcarousedemo qt5nmapper qt5ledscreen qtsmarthome quitbattery \
quitindicators qtbase-examples qtdeclarative-examples \
qtlocation-examples qtmultimedia-examples qtquickcontrols-examples \
qtsystems-examples"

OTHER_LIBRARIES = "cairo fontconfig freetype libsocketcan pango \
mobile-broadband-provider-info"

DEV_TOOLS = "\
git packagegroup-core-sdk packagegroup-core-ssh-openssh \
packagegroup-core-standalone-sdk-target \
packagegroup-core-tools-debug packagegroup-qt5-qtcreator-debug vim"

TEST = "\
mesa-demos packagegroup-fsl-tools-benchmark \
packagegroup-fsl-tools-testapps"

UTILITIES = "can-utils i2c-tools sqlite3"

DAEMONS = "maliit-framework-qt5 ppp"

COMPANY = "tmr1443hwinit webos-keyboard"

IMAGE_INSTALL_append = "\
packagegroup-base packagegroup-core-boot \
packagegroup-core-full-cmdline packagegroup-fsl-gstreamer1.0-full \
```

```
packagegroup-core-x11-sato ${QT_LIBRARIES} ${QT_EXAMPLES} \
${OTHER_LIBRARIES} ${DEV_TOOLS} ${TEST} ${UTILITIES} ${DAEMONS} \
${COMPANY}"
```

LIITE G. KONTRONIN U-BOOT KÄYNNISTYSLATAAJAN OLETUSYMPÄRISTÖ

```
Muuttujien määrittelyä ei voi jakaa useammalle riville, tässä
käytetty '\'-merkkiä rivinvaihtoihin, jotta muuttujat mahtuvat
sivulle

autoload=no
baudrate=115200
bootcmd=run usbboot; run mmcboot; run sdboot; run bootfailed;
bootdelay=3
bootfailed=echo Could not determine where to boot from, exiting...
console=ttymxc0
consoledev=ttymxc0
ethact=FEC
ethaddr=00:e0:4b:4d:c9:38
ethprime=FEC
ethrotate=no
fdtaddr=11000000
fdtfile=smx6.dtb
filesize=40DB20
hostname=SMARC-sAMX6
initrdaddr=12700000
initrdfile=initrd
load_splash_img=sf probe 2; sf read $loadaddr $splash_img_addr \
$splash_img_size;
loadaddr=0x10800000
loadfdt=ext2load mmc ${mmcdev}:${mmcpart} ${fdtaddr} ${fdtfile}
loadinitrd=fatload usb 0:1 ${initrdaddr} ${initrdfile} && fatload \
usb 0:1 ${loadaddr} ${uimage}
loads_echo=1
loaduimage=ext2load mmc ${bootdev}:${bootpart} ${loadaddr} ${uimage}
machid=10e9
mmcargs=setenv bootargs console=${console},${baudrate} \
root=${mmcroot}
mmcboot=setenv bootdev ${mmcdev}; setenv bootpart ${mmcpart}; \
mmc dev ${mmcdev}; if mmc rescan ${bootdev}; then if run \
loaduimage; then echo Booting from MMC ; setenv bootargs \
console=${consoledev},${baudrate} root=${mmcroot}; bootm \
${loadaddr}; fi;fi;
mmcdev=2
mmcpart=1
mmcroot=/dev/mmcblk0p1 rw
netargs=setenv bootargs console=${console},${baudrate} \
root=/dev/nfs ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp
netboot=echo Booting from net ...; tftp ${fdtaddr} ${fdtfile}; \
tftp ${loadaddr} ${uimage};bootm ${loadaddr} - ${fdtaddr}
netdev=eth0
netupdate=if tftp $loadaddr $updfile; then setenv loader tftp; \
source $loadaddr; else run updFal; fi
print_splash_img=run load_splash_img; if bmp info $load_addr; \
then bmp display $loadaddr; else echo No Splash Image found; fi
sdboot= setenv bootdev ${sddev}; setenv bootpart ${sdpart}; mmc \
dev ${sddev}; if mmc rescan ${bootdev}; then if run loaduimage; \
then echo Booting from SD Card ...; setenv bootargs \
```

```
console=${consoledev},${baudrate} root=${sdroot}; bootm \
${loadaddr};fi;fi;
sddev=1
sdpart=1
sdroot=/dev/mmcblk1p1 rootwait rw
serial#=UHD0X0035
splash_img_addr=100000
splash_img_size=80000
uimage=uImage
updFal=echo update failed
updMmcAddExt=mmc dev 0 && ext2load mmc 0 $loadaddr $updfile && \
setenv loader ext2load mmc 0 && source $loadaddr && true;
updMmcAddFat=mmc dev 0 && fatload mmc 0 $loadaddr $updfile && \
setenv loader fatload mmc 0 && source $loadaddr && true;
updMmcOnbExt=mmc dev 2 && ext2load mmc 2 $loadaddr $updfile && \
setenv loader ext2load mmc 2 && source $loadaddr && true;
updMmcOnbFat=mmc dev 2 && fatload mmc 2 $loadaddr $updfile && \
setenv loader fatload mmc 2 && source $loadaddr && true;
updSdExt=mmc dev 1 && ext2load mmc 1 $loadaddr $updfile && setenv \
loader ext2load mmc 1 && source $loadaddr && true;
updSdFat=mmc dev 1 && fatload mmc 1 $loadaddr $updfile && setenv \
loader fatload mmc 1 && source $loadaddr && true;
updUsbExt=usb start && usb dev 0 && ext2load usb 0:1 $loadaddr \
$updfile && setenv loader ext2load usb 0:1 && source $loadaddr \
&& true;
updUsbFat=usb start && usb dev 0 && fatload usb 0:1 $loadaddr
$updfile && setenv loader fatload usb 0:1 && source $loadaddr \
&& true;
update=run updUsbExt || run updUsbFat || run updSdExt || run \
updSdFat || run updMmcAddExt || run pdMmcAddFat || run \
updMmcOnbExt || run updMmcOnbFat || run updFal;
updfile=update_smx6/update
usbboot=usb start;if usb dev 0; then if run loadinitrd; then bootm \
${loadaddr} ${initrdaddr};fi;fi;
wince_kitl=disabled
```

LIITE H. *TREE*-TYÖKALUN TUOTTAMA TIEDOSTORAKENNELISTAUS META-YRITYS KERROKSESTA

```
|-- conf
|   |-- distro
|   |   `-- yritysyocto.conf
|   |-- layer.conf
|   `-- machine
|       `-- smarc-samx6i.conf
|-- recipes-bsp
|   `-- tmr1443-smx6-image-sato.bb
|-- recipes-connectivity
|   `-- ppp
|       |-- ppp
|       |   |-- Chat-Module-UC20-connect
|       |   |-- Chat-Module-UC20-disconnect
|       |   `-- Module-UC20
|       `-- ppp_%.bbappend
|-- recipes-core
|   |-- psplash
|   |   |-- psplash
|   |   |   |-- rotation
|   |   |   `-- yritys-logo.png
|   |   `-- psplash_%.bbappend
|   `-- tmr1443hwinit
|       |-- tmr1443hwinit
|       |   `-- tmr1443hwinit
|       `-- tmr1443hwinit_0.1.bb
|-- recipes-kernel
|   |-- linux-smx6-3.14.28
|   |   |-- 0001-ARM-imx6q-drop-unnecessary-semicolon.patch
...
|   |   |-- 0074-mmc-Add-SDIO-function-devicetree-subnode-parsing.patch
|   |   `-- defconfig
|   `-- linux-smx6_3.14.28.bb
|-- recipes-qt
|   |-- maliit
|   |   |-- maliit-framework-qt5
|   |   |   |-- 0001-Fix-MALIIT_INSTALL_PRF-to-allow-the-build-with-opene.patch
|   |   |   `-- maliit-server.desktop
|   |   `-- maliit-framework-qt5_0.99.1.bb
|   `-- qtbase_%.bbappend
|-- recipes-sato
|   |-- matchbox-sato
|   |   |-- matchbox-session-sato
|   |   |   |-- defaultenvironment.sh
|   |   |   `-- session
|   |   `-- matchbox-session-sato_%.bbappend
|   `-- packagegroups
|       `-- packagegroup-core-x11-sato.bbappend
|-- recipes-virtualkeyboard
|   |-- hunspell
|   |   `-- hunspell_1.3.2.bb
```

```
|   |-- libpinyin
|   |   |-- libpinyin-extraconf.inc
|   |   |-- libpinyin_git.bb
|   |   `-- libpinyin.inc
|   |-- presage
|   |   |-- presage
|   |   |   |-- disable-cygwin-related-automake-error.patch
|   |   |   |-- disable-help2man.patch
|   |   |   `-- no-automake-werror.patch
|   |   `-- presage_0.8.9.bb
|   `-- webos-keyboard
|       |-- webos-keyboard
|       |   |-- 0001-CPP-compilation-without-LuneOS-dependencies.patch
...
|       |   `-- 0017-Fix-L-size-to-take-half-of-the-screen-on-800-screen-.patch
|       `-- webos-keyboard_git.bb
`-- setup-environment
```