VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Jani Perkiö

MEDIASOITIN

Tekniikka ja liikenne
2010

VAASAN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Jani Perkiö
Opinnäytetyön nimi	Mediasoitin
Vuosi	2010
Kieli	Suomi
Sivumäärä	59 + 37 liitettä
Ohjaaja	Jukka Matila

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa mediasoitin.

Opinnäytetyö käsitti koko soittimen suunnittelun, toteutuksen ja testauksen. Soitin koostuu Atmel NGW100 –kehitysalustasta, laajennuslevystä, 4,3” kosketusnäytöstä, Linux-käyttöjärjestelmästä ja graafisesta käyttöliittymästä.

Laitteeseen toteutettiin Qt:lla graafinen käyttöliittymä, millä ohjataan soittimen toimintaa. Mediatiedostot voidaan siirtää laitteeseen käyttäen USB- tai ethernet porttia. Lähiverkossa laite näkyy Samba-palvelimena.

Asiasanat Musiikki, Soitin, Toistin, Qt, Kosketusnäyttö, Linux

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Jani Perkiö
Title	Media Player
Year	2010
Language	Finnish
Pages	59 + 37 Appendices
Name of Supervisor	Jukka Matila

The purpose of this thesis was to design and implement a media player.

The device consists of Atmel NGW100 evaluation board, expander board, 4,3" touch screen, Linux operating system and graphical user interface.

Graphical user interface was implemented by using Qt software. Files can be transferred to the system via USB or ethernet. At local area network the device can be seen as a Samba server.

Keywords Music, Player, Qt, Touch Screen, Linux

KÄYTETYT MERKINNÄT JA LYHENTEET

Seuraavia lyhenteitä ja vieraskielisiä sanoja on käytetty tässä insinöörityössä:

ADC	Analog to Digital Converter	Analogia-digitaalimuunnin
DAC	Digital to Analog Converter	Digitaali-analogiamuunnin
DIL	Dual In Line-package	Komponentin kotelotyyppi
DMA	Direct Memory Access	Oikosiirto
DSP	Digital Signal Processing	Digit. signaalin käsittely
GPIO	General Purpose Input Output	Yleiskäyttöinen portti
I/O	Input / Output	Sisääntulo / Lähtö
LCD	Liquid Crystal Display	Nestekidenäyttö
LED	Light Emitting Diode	Loistediodi, hohtodiodi
LGPL	GNU Lesser General Public License	GNU-projektin lisenssi
RAM	Random Access Memory	Keskusmuisti, käyttömuisti
RISC	Reduced Instruction Set Computer	Suoritinarkkitehtuuri
ROM	Read Only Memory	Lukumuisti
SPI	Serial Peripheral Interface Bus	Sarjaliikenneväylä
TFT	Thin Film Transistor	Ohutkalvotransistori

SISÄLLYS

1 JOHDANTO

Musiikin toisto kotitalouksissa on yleistynyt merkittävästi digitaalisen median tultua markkinoille. Lisääntyneen kysynnän seurauksena markkinat ovatkin pullollaan toinen toistaan erikoisempia musiikinsoittimia, joiden ominaisuudet ovat keskivertokuluttajalle enemmän tai vähemmän hyödyllisiä. Suurimpana haasteena kaikissa näissä soittimissa on kuitenkin se, että ne tukevat yleensä vain yhdentyyppistä tallennusmediaa. Tilannetta hankaloittaa entisestään se, että vaikka tallennusmedia olisikin eri soitinten välillä sama, voi tallennusformaatti olla eri, kuten esimerkiksi CD-levy. CD-levylle on mahdollista tallentaa musiikkia pakkaustavasta riippuen eri määriä, ja tämä ajaa markkinoilla olevat soittimet yhteensopivuusongelmiin. Käytännön esimerkkinä voidaan tarkastella esimerkiksi tavallisen CD-levyn ja MP3-levyn eroa: molemmissa tallennuspohjana käytetään samaa levytyyppiä. Formaattiongelmaan ei ole suoranaista ratkaisua, koska musiikin pakkaustapoja on ennestäänkin jo liikaa, ja lisää on tulossa. Ainoa tapa lähestyä tätä ongelmaa onkin luoda soitin, joka tukee mahdollisimman montaa eri mediatyyppiä.

Lähtökohtanani on päättötyössäni lähestyä tätä ongelmaa luomalla alkeellinen prototyyppi mediasoittimesta, joka ratkaisisi mahdollisimman monta nykypäivän musiikintoistoon liittyvistä ongelmista.

2 KONSEPTI

2.1 Soitintyyppien esittely

Markkinoilla on lukuisia erilaisia laitteita, joissa on tavalla tai toisella yritetty ratkaista kuvaamaani ongelmaa, mutta ei mielestäni riittävän hyvin lopputuloksin. CD-soittimet tukevat erilaisia pakkaustapoja, mutta tallennustapa on jäänyt alkeelliseksi. Jos levy menee fyysisesti rikki, on musiikki menetetty. Varmuuskopioiden ottaminen CD-levyistä on laitonta, mikä hankaloittaa asioita entisestään.

Tietokoneella on mahdollista ladata ja toistaa erilaisiin formaatteihin pakattua musiikkia. Käytön helppous on kuitenkin unohtunut; musiikin toisto ei tapahdu enää play-nappia painamalla. Laitteiston käynnistyksessä kestää kauan ja ohjelmat täytyy käynnistää erikseen toistoa varten. Toistinohjelmia on saatavilla todella paljon ja tästä syystä peruskäyttäjällä voi olla ohjelman valitsemisessa liikaakin valinnanvaraa. Lisäksi tietokoneen pitäisi käytännössä sijaita samassa huoneessa kaiuttimien ja vahvistimen kanssa. Tietokoneen yhdistäminen vahvistimeen käy helposti, mutta ylimääräiset johdot huoneen lattialla eivät ilahduta ketään.

Lähimmäksi oikeaa ratkaisua ovat mielestäni päässeet verkkosoittimet, joiden tarkoituksena on toistaa musiikki tietokoneelta, omasta muistista tai verkkolevyltä. Monet tämäntyyppiset laitteet ovat silti valitettavan riippuvaisia tietokoneesta tai ylimääräisestä tallennusasemasta, ja laitteen ohjaus tapahtuu yleensä television kautta. Jos laitetta ohjataan television kautta, muodostuu liittimien yhteensopivuus usein ongelmaksi, koska liittimiä voi olla useita erilaisia (esim. SCART-, HDMI-, VGA-liitin jne.). Yleensä soitin on lisäksi riippuvainen tietokoneella sijaitsevasta ohjelmasta, jonka avulla musiikki voidaan siirtää laitteeseen. Tällaiset ohjelmat ovat lisäksi yleensä sidottu vain yleisimpiin käyttöjärjestelmiin, joten yhteensopivuusongelmiin törmätään hyvin helposti.

2.2 Soittimen tyypin valinta

Soitintyypiksi valittiin verkkosoitin, jonka toimintaperiaatteina olisivat:

- Eri medioiden soittomahdollisuus

- Helppokäyttöisyys
- Laajennettavuus
- Päivitettävyys

Edellä mainittujen kriteerien perusteella projektin pohjaksi valittiin sulautettu Linux. Linuxin vahvuutena on valmiiden ajureiden ja ohjelmiston saatavuus ja niiden helpohko päivitettävyys. Jos ja kun, uusia toistettavia tiedostomuotoja tulee, on laite helposti päivitettävissä toistamaan uusimpia tiedostoja. Graafisen käyttöliittymän pohjaksi valittiin Qt-kehitysympäristö. Piirilevytasolta valitsin pohjaksi Atmel NGW100 –kehitysalustan (Network Gateway Kit), koska levyn hinta oli kohtuullinen ja kyseisestä kehitysalustasta löytyi USB, Ethernet, SD MMC, RS-232 sekä prosessorin I/O-liittimet. Kehitysalustan päälle täytyi silti rakentaa rajapinnat äänelle, näytölle sekä resistiiviselle kosketuskalvolle. Lisäoptioiksi kyseiselle lisälevylle tuli myös reaaliaikakello ja I/O expander –piirit I^2C-väylään. Kaikista valinnoista lisää myöhemmissä kappaleissa.

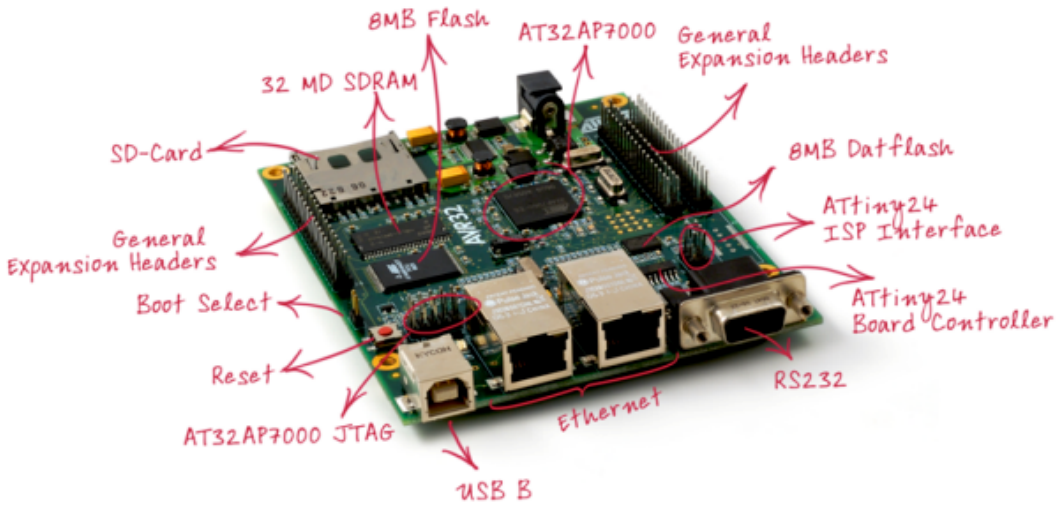
Työ toteutettiin luomalla ensin elektroniikka, jonka jälkeen toteutettiin Linux-ympäristö ja graafinen käyttöliittymä.

3 ELEKTRONIIKKA

3.1 Atmel NGW100 –kehitysalustan ominaisuudet

NGW100 on edullinen ja helposti laajennettavissa oleva kehitysalusta Atmelilta. Alusta sisältää Atmelin 32-bittisen AT32AP7000 RISC –prosessorin, joka toimii 150MHz kellotaajuudella. Alustan tärkeimmät komponentit ovat:

- AT32AP7000-prosessori
- ATtiny24-mikroprosessori
- 32Mb SDRAM-muisti
- 8Mb Flash-muisti
- SD-muistikortin lukija
- 2 kpl Ethernet liitintä
- RS232-liitin
- USB-liitin
- GPIO-liitinrimat



Kuva 1: ATNGW100 Kehitysalusta [1]

3.1.1 Tietoja AT32AP7000 prosessorista

AT32AP7000 on täydellinen System-on-chip sovellus, joka sisältää AVR32 RISC –prosessorin, jolla saavutetaan 210 DMIPS 150 MHz:n kellotaajuudella. Prosessori on suunniteltu kustannusherkkiin sulautettuihin sovelluksiin, ja sen ominaisuuksissa on keskitytty erityisesti alhaiseen virrankulutukseen, kooditiheyteen ja hyvään suorituskykyyn. [2]

Piiri sisältää MMU-yksikön, ja sen joustava keskeytysohjain tukee nykyaikaisia ja reaaliaikaisia käyttöjärjestelmiä. Suorittimessa on myös paljon DSP- ja SIMD-ohjaimia, jotka on suunniteltu erityisesti multimedia- ja telealan sovelluksiin. AT32AP7000 sisältää 32KB SRAM-muistia, ja SDRAM-ohjaimen ansiosta piiriin voidaan liittää Compact Flash, MultiMediaCard (MMC), Secure Digital (SD), SmartCard, NAND ja Atmelin DataFlash-muisteja. Direct Memory Access (DMA) –ohjain mahdollistaa tiedonsiirron muistien välillä ilman prosessorin väliintuloa. [2]

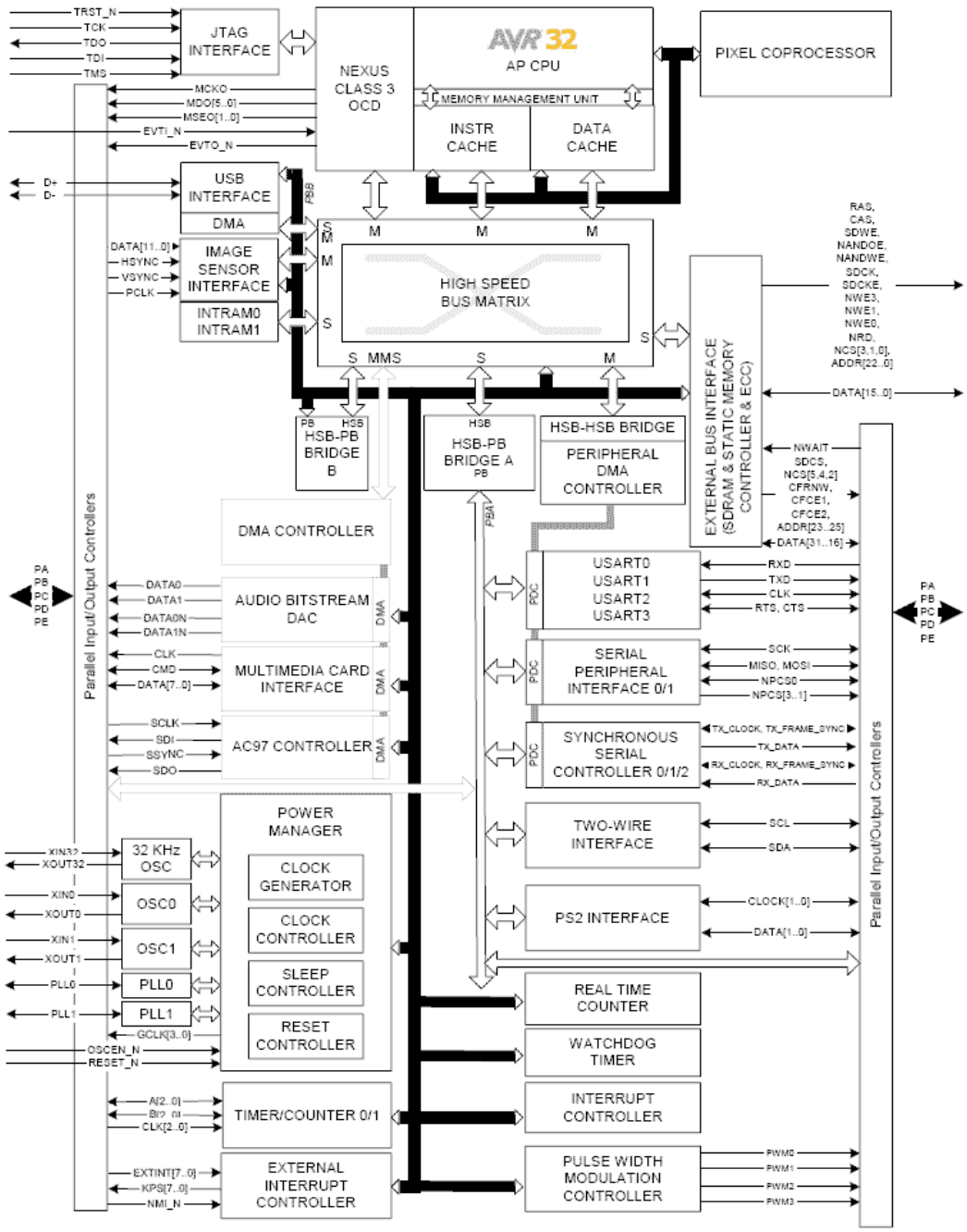
Piirissä on myös ajastin/laskuri, joka sisältää kolme samanlaista 16-bittistä ajastin/laskuri kanavaa. Jokainen kanava voidaan ohjelmoida suorittamaan erilaisia toimintoja, kuten esimerkiksi taajuuden tai tapahtumien mittaus, pulssigeneraattori, viiveistys ja PWM-pulssin luominen.

AT32AP7000 sisältää myös LCD-ohjaimen, joka tukee sekä mustavalkoisia että värillisiä, passiivisia STN LCD –moduuleja ja aktiivisia TFT LCD –moduuleja. Ohjainta käytettäessä mustavalkoisena päästään jopa 16:een eri harmaan sävyyn, ja värillisellä STN-näytöillä päästään huikeaan 4096:een väriin. LCD-ohjain on ohjelmoitavissa maksimissaan 2048 x 2048 resoluutioon, ja pikselin syvyys on ohjelmoitavissa alueille 1 - 24 bittiä per pikseli. [2]

Näiden ominaisuuksien lisäksi piiri sisältää:

- 4 kpl USART-porttia
- 3 kpl synkronista sarjaporttiohjainta mm. I2S- ja SPI-väylille
- Tuen 12-bittiselle CMOS-kameralle

- 2 kpl 10/100Mbps Ethernet-porttia
- 16-bittinen D/A-muunnin audiokäyttöön
- PS2-tuki näppäimistölle ja hiirelle



Kuva 2: AT32AP7000 Lohkokaavio [2]

3.2 Laajennuslevy kehitysalustalle

Koska kehitysalustasta puuttui tärkeitä ominaisuuksia, kuten esimerkiksi liittimet näytölle sekä kunnollinen DAC, täytyi nämä komponentit lisätä erilliselle levylle. Piirilevy sijoitetaan alustan ”general expansion headers” –liittimiin.

Levyn suunnittelu aloitettiin tarvittavien ominaisuuksien listaamisesta, jonka jälkeen voitiin valita käytettävät komponentit ja kytkennät.

Valitut ominaisuudet:

- Jännitelähde
- LCD
    - Liittimet näytölle
    - Taustavalon virtalähde
- Liitin resistiiviselle kosketuskalvolle
- AD-muunnin resistiiviselle kosketuskalvolle
- DA-muunnin äänelle
- I/O expander –piiri
- Reaaliaikakello

3.2.1 Levylle valitut komponentit

Seuraavissa kappaleissa on esitelty pääasialliset komponenttivalinnat, joita laajennuslevyssä käytettiin.

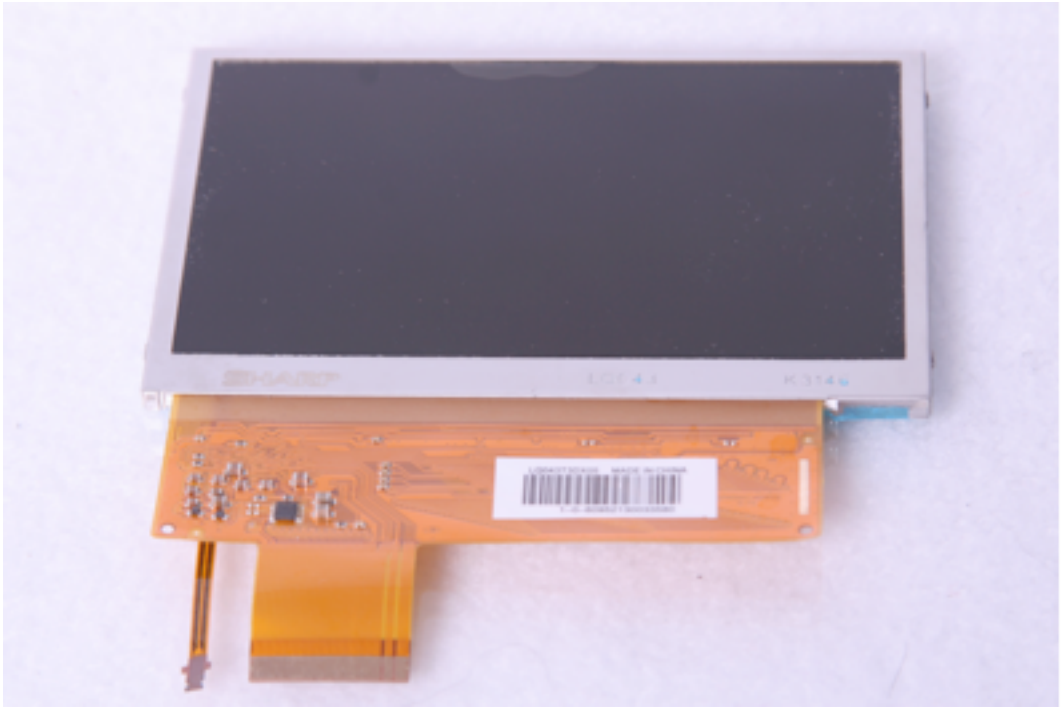
3.2.1.1 Näyttö

Projektin näytöksi valittiin 4.3” Sharp LQ043T3DX02, jota käytetään mm. Portable Playstationissa. Näytön valintaan vaikutti merkittävästi se, että kyseistä näyttöä on käytetty samassa alustassa ennemminkin, mikä lisää luotettavuutta sen

toimivuuteen. Koska näyttö on yleinen, se oli myös helposti ja edullisesti saatavilla eBay:sta.

Näytön liittäminen kehitysalustaan oli suhteellisen yksinkertaisesti toteutettavissa, koska kaikki datalinjat oli mahdollista liittää suoraan prosessorilta näytön FPC-liittimille. Lisäksi näytössä on enable-signaali, joka ohjaa näytön päälle. Em. signaali täytyy, joko kytkeä kiinteästi päälle, tai erikseen aktivoida. Työssä signaali kytkettiin prosessorin ylimääräiseen porttiin, josta se ohjattiin aktiiviseksi.

Näytön osalta ainoa komponentteja vaativa osio oli näytön taustavalaistuksen virransyöttöyksikkö. Tähän tarkoitukseen valittiin piiri LT3593. Piiri on pieni hakkurivirtalähde, joka syöttää tasaisella virralla näytön sarjassa olevia LED-valoja. Piiri säätää lähtöjännitteen LEDien paluuvirran perusteella. Piirissä oli myös mahdollista ohjata LEDien kirkkautta erillisellä ohjauspinnillä, mutta tätä ominaisuutta ei lisätty alustaan. Kirkkauden ohjauspinni onkin kiinteästi kytketty 0R-vastuksen läpi +5V käyttöjännitteisiin, joka tarkoittaa sitä, että taustavalo on jatkuvasti kirkkaimmillaan. Jos 0R-vastus otetaan irti ja tilalle viedään esim. I^2C I/O expanderilta yksi signaalijohdin, voidaan näytön kirkkautta ohjelmallisesti ohjata.

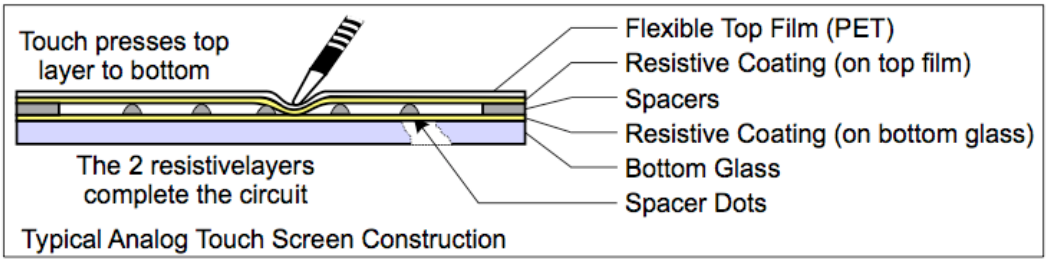


Kuva 3: Työssä käytetty Sharp LQ043T3DX02 –näyttö

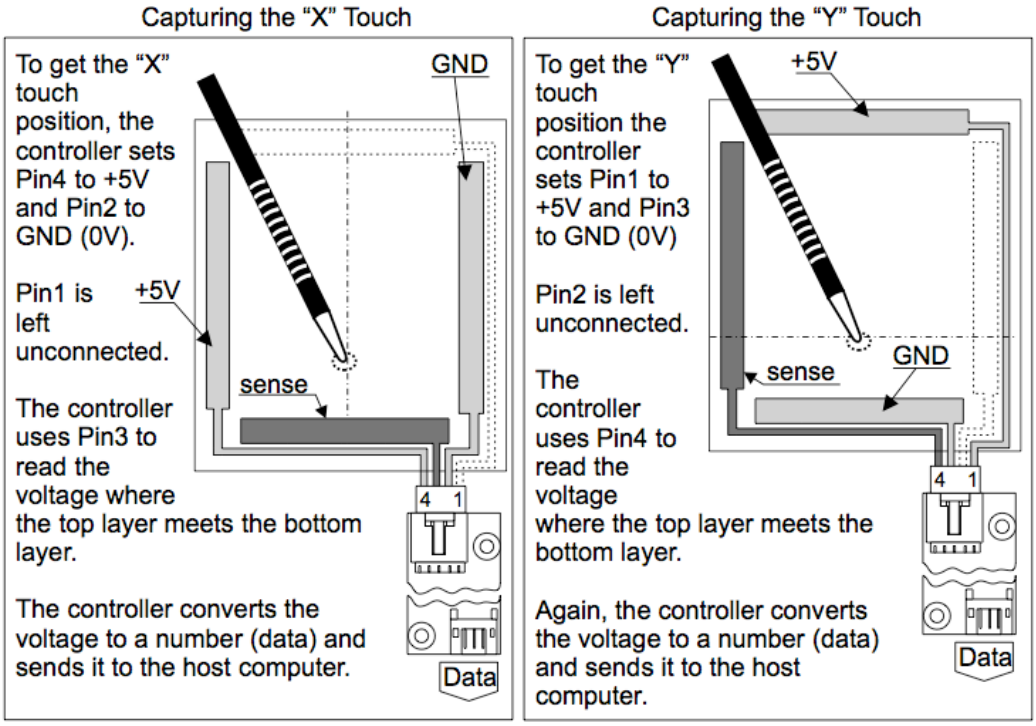
3.2.1.2 Kosketuskalvo

4,3” PSP:n näytön päälle valittiin saman kokoinen resistiivinen kalvo Hantronix HDA430T, joka tilattiin Sparkfun –verkkokaupasta (www.sparkfun.com). Kalvon toiminta perustuu kahden hyvin ohuen, erillään olevan, sähköä johtavan kerroksen yhteisvaikutukseen. Kun näyttöä koskettaa, koskettavat nämä kaksi kerrosta toisiaan. Kosketuskohta aiheuttaa suljetun virtapiirin näiden kahden kerroksen välille. Tämän risteyskohdan X- ja Y-koordinaatit saadaan selville mittaamalla resistanssi näytöstä lähtevien neljän johtimen avulla.

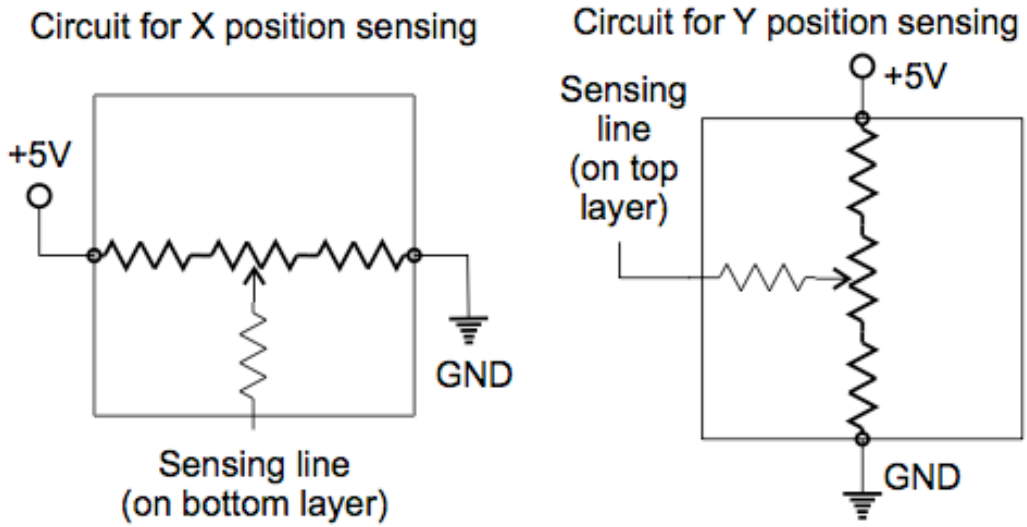
Toiminnan kannalta 470Ω-vastus oli lisättävä YM-linjan kanssa sarjaan resistanssisovituksen takia.



Kuva 4: Kerrosten rakenne [3]



Kuva 5: Kosketuspaikan löytäminen [3]



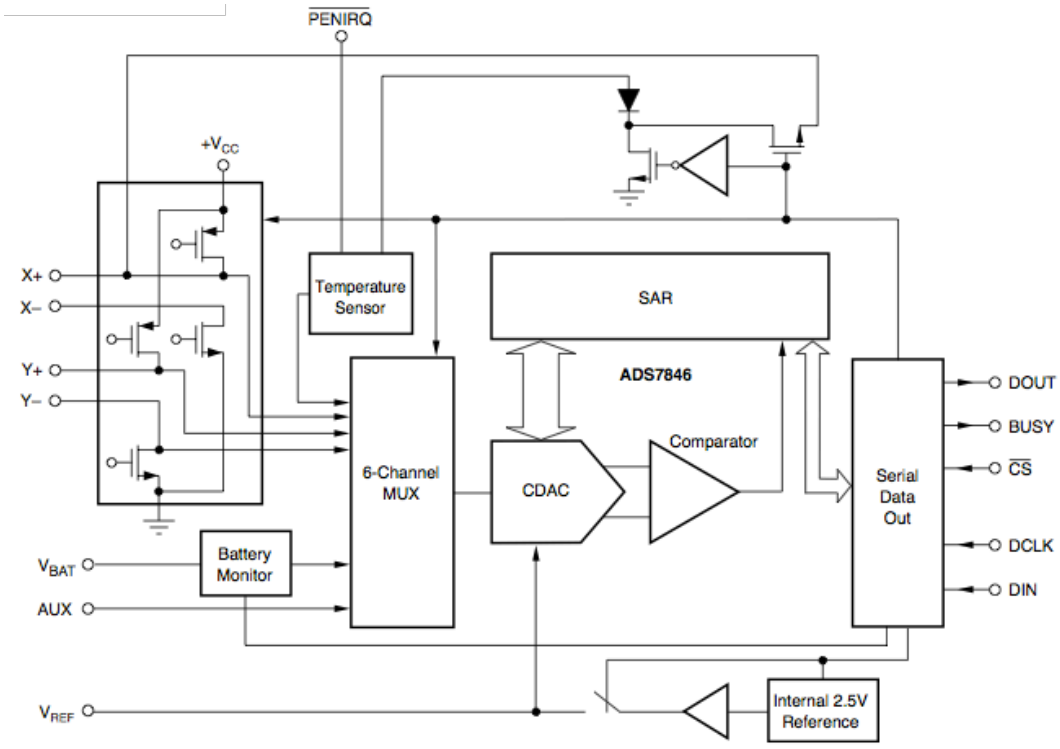
Kuva 6: Periaatteellinen kuva kosketuspaikan havainnoinnista [3]

3.2.1.3 AD-muunnin

Koska resistiivisen kalvon antama analoginen jännite täytyi mitata ja muuntaa digitaaliseen muotoon, tarvittiin tarkoitukseen AD-muunnin. Piiri muuntaa näytöstä saadut jännitetiedot digitaaliseksi ja lähettää ne prosessorille. Prosessori käsittelee saamansa datan ja laskee ne koordinaatit, mihin käyttäjä oli näyttöä koskettanut.

AD-muuntimen valintaa varten täytyi ottaa selvää, mitkä piirit kyseisellä kehitysalustalla olivat aiemmin olleet käytössä ja toimineet ilman merkittäviä ongelmia. Käyttäjien kokemuksia eri piireistä oli hyvin vähän, ja ainoaksi varmaksi vaihtoehdoksi todettiin ADS7846.

Piiri mittaa kalvolta saadun analogisen jännitteen ja prosessoi sen digitaaliseen muotoon, joka välitetään prosessorille SPI-väylän avulla. Kun piiri havaitsee kosketuksen kalvolla, lähetetään keskeytys prosessorille. Prosessorin on reagoitava keskeytykseen, koska muuten kosketusta ei havaita. Piirin AD-muunnin on 12-bittinen.



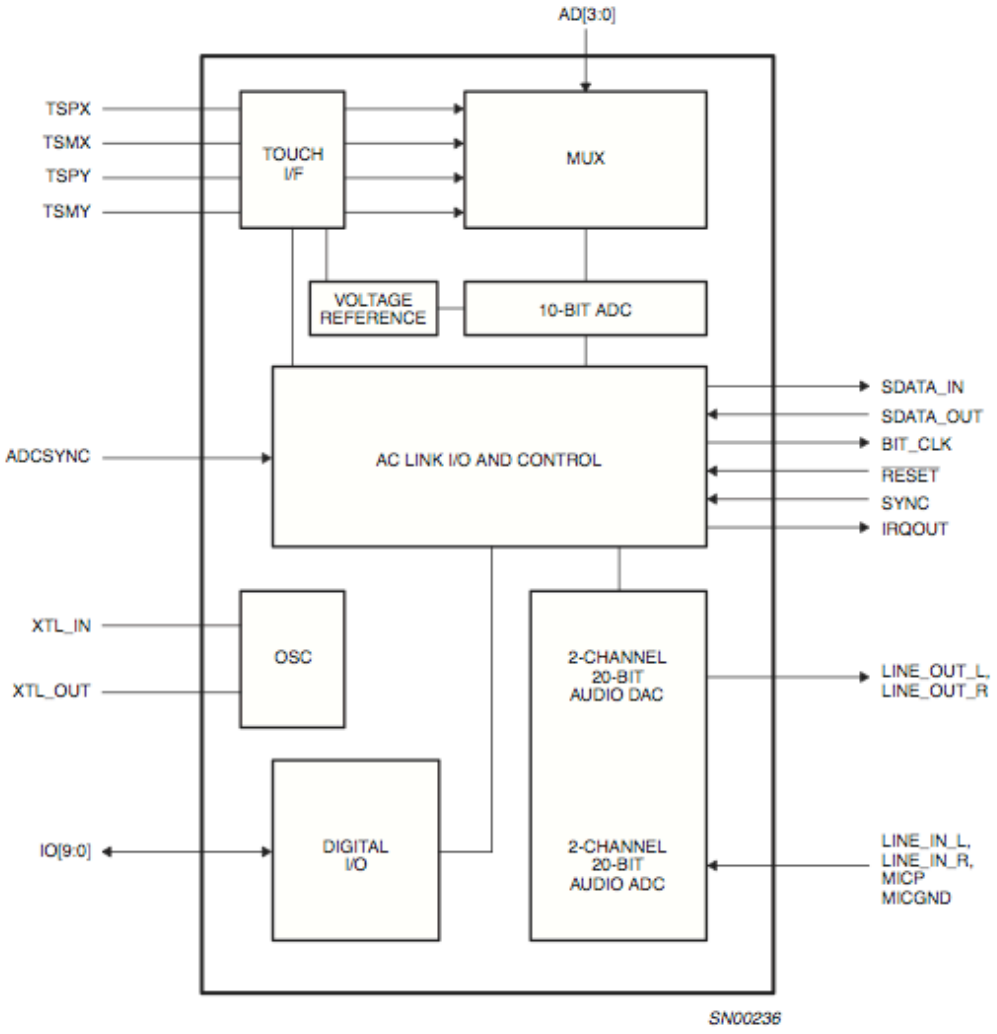
Kuva 7: ADS7846 lohkokaavio [4]

3.2.1.4 DA-muunnin

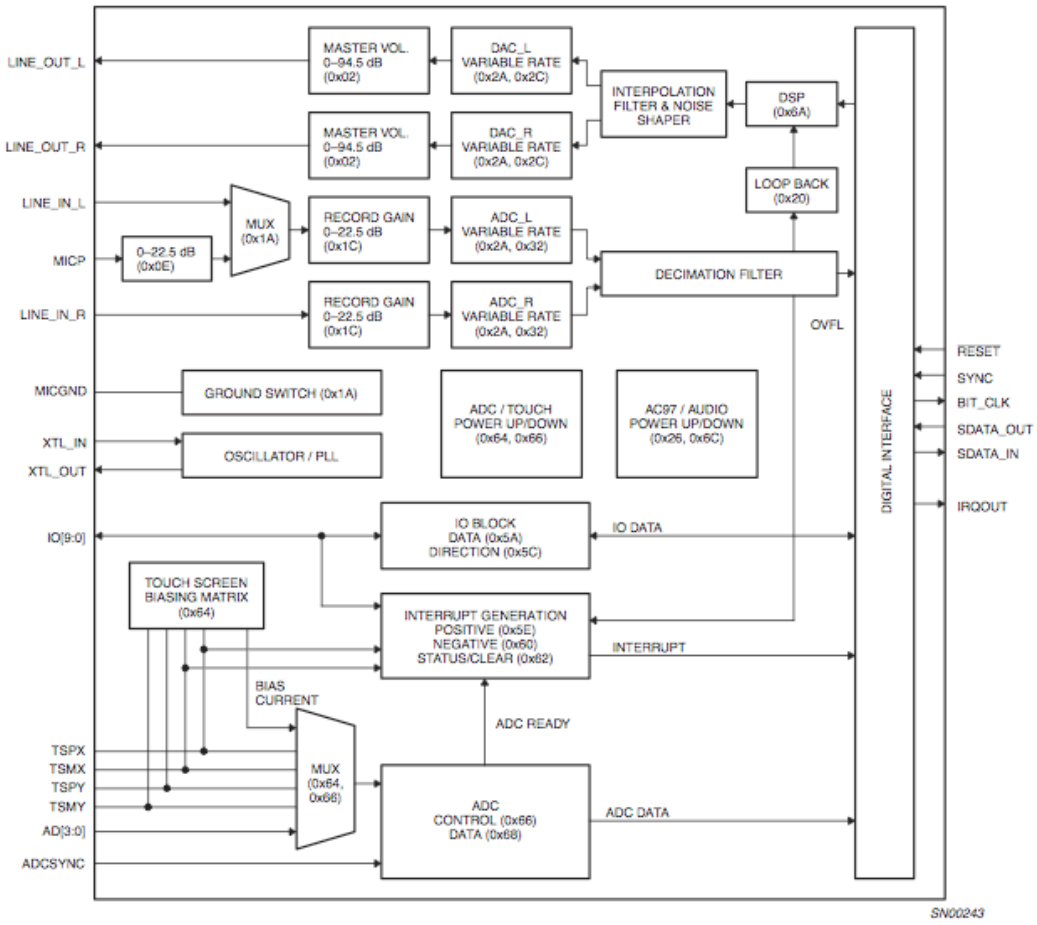
Jotta laitteesta saataisiin ääntä, tarvitaan DA-muunnin. Prosessorin lähettämä digitaalinen signaali on muunnettava analogiseen muotoon, ja edelleen signaali on vahvistettava esim. kuulokeliitäntään. Tähän tarkoitukseen valittiin Philips UCB1400:n, koska sen ajurit löytyivät jo sulautettuun Linuxiin. Vaikka prosessori sisältää DA-muuntimen, on sen antama lähtöteho mitätön, joka tarkoitti sitä, että lähtöä olisi täytynyt vahvistaa.

Piirin ainoa tarkoitus työssä on antaa ääntä Line-Out/kuulokeliitäntään, vaikka kyseinen piiri sisältää muitakin laajennusmahdollisuuksia. Kommunikointi piirin ja prosessorin välillä tapahtuu AC'97 (Audio Codec '97) väylää pitkin. Väylä muodostuu viidestä signaalista: kellosignaalista (12.288 MHz), synkronointisignaalista, reset-signaalista, ja kahdesta datalinjasta (sdata_out ja sdata_in) [5]. Ainoa signaali, joka prosessorille oli määriteltävä, oli reset-signaali. Muut signaalit olivat jo määriteltyinä prosessorin liitäntöihin.

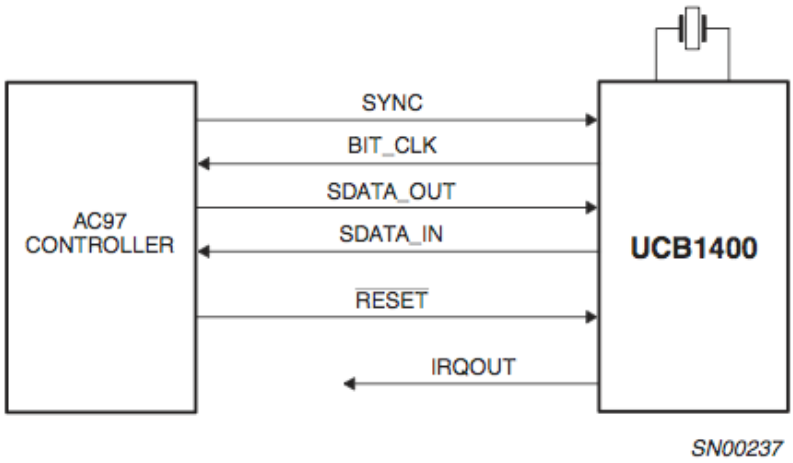
Piiri sisältää myös Line-In/mikrofoniliitynnän, 4 kpl AD-muunninta, 10 kpl GPIO-porttia ja liitännän kosketusnäyttöön [5]. Liitäntää kosketusnäytölle ei asennettu tälle piirille, koska aiemmin mainittu ADS7846-piiri on tarkempi (ADS7846 12bit, UCB1400 10bit), ja ajurit todettiin toimiviksi. Laajennuslevyllä on silti mahdollisuus valita, kumpaa piiriä käytetään kosketuskalvolle. Muutoksen voi toteuttaa kalustamalla 0R vastukset eri paikkoihin piirilevyllä.



Kuva 8: UCB1400 lohkokaavio [5]



Kuva 9: UCB1400 funktionaalinen lohkokaavio [5]

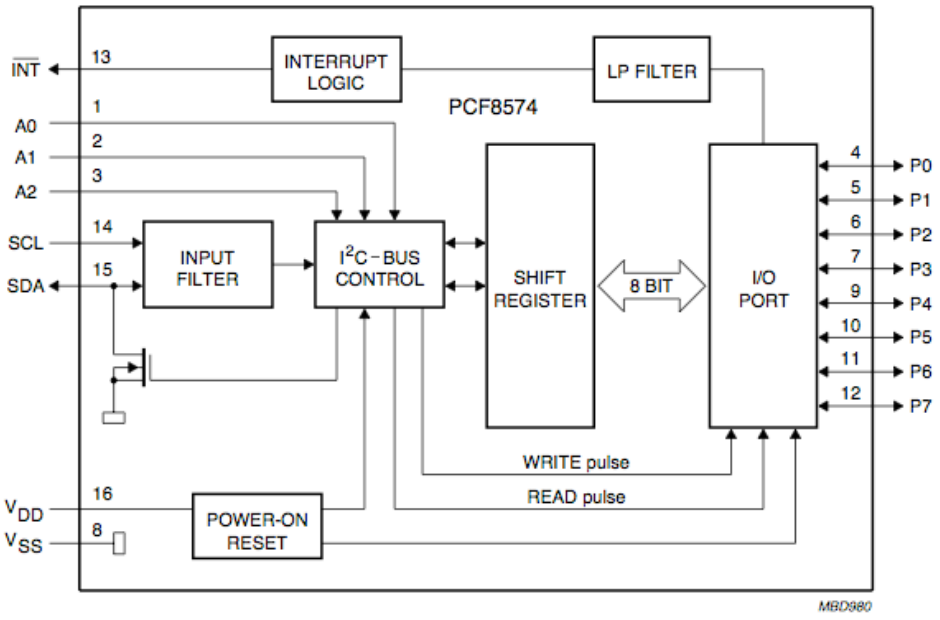


Kuva 10: UCB1400 Liityntärajapinta [5]

3.2.1.5 I/O Expander-piiri

Piirin tarkoituksena on tuoda alustaan lisää ohjelmoitavia digitaalisia nastoja. Tämä piiri lisättiin levyyn, koska projektissa pyrittiin välttämään prosessorin tai UCB1400-piirin vapaiden nastojen käyttämistä ylimääräisiin liitäntöihin. Jos em. piirit menisivät rikki, olisi niiden vaihto kallista ja lisäksi todella hankalaa. Piiriksi valittiin hyvin yleisesti käytetty PCF8574P. Tämä 8-bittinen I/O expander –piiri on edullinen, ja koska se on asennettu liittimien päälle DIL-kotelonsa ansiosta, se voidaan myös vaihtaa todella helposti. Kommunikointi prosessorin ja piirin välillä tapahtuu I^2C-väylän välityksellä.

Työssä on tarkoitus lisätä liityntä virtakytkimeen tämän piirin välityksellä, eli soitin pystytään sammuttamaan ohjelmallisesti. Lisäksi mahdolliset indikointiin tarvittavat LED-valot voidaan ohjata tämän piirin avulla.

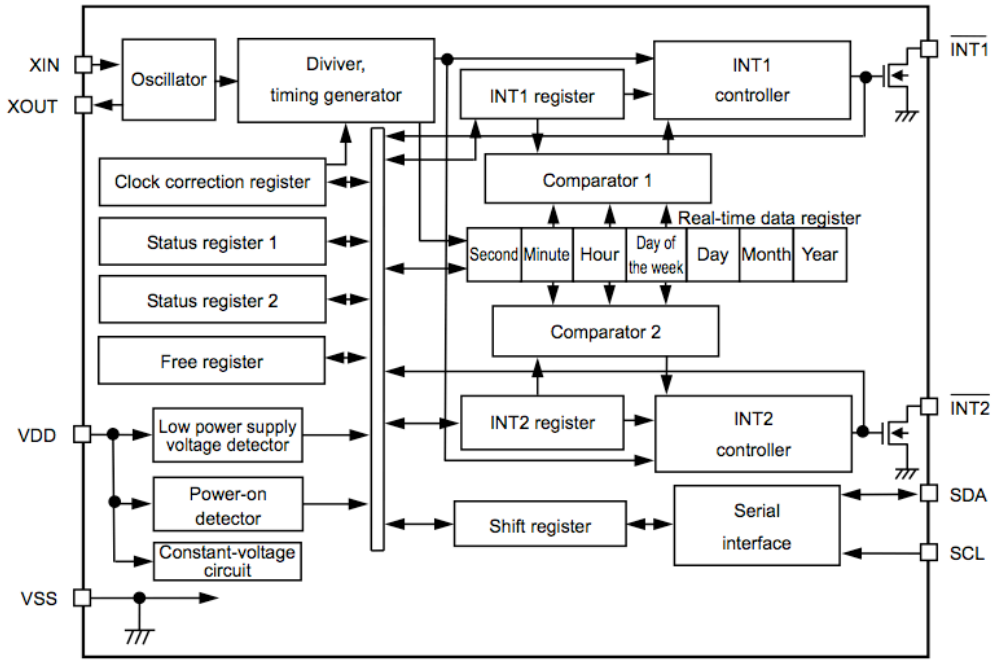


Kuva 11: PCF8574P lohkokaavio [6]

3.2.1.6 Reaaliaikakello

Prosessori sisältää reaaliaikakellon, mutta ongelmana on, että jos kehitysalustasta irrottaa sähköt, kellonaika katoaa. Tästä syystä adapterilevyyn lisättiin erillinen reaaliaikakello. Piirinä toimii Seiko Instruments S-35390A-J8T1G. Kuten I/O expander, tulee tämäkin piiri liittää I^2C-väylään.

Nykyisen kellonajan voi tallentaa ja piiri ”käy” itsenäisesti, vaikka soittimesta kytkisi virran pois. Kun päävirta katkeaa, jatkaa piiriin kytketty 0,33F kondensaattori virransyöttöä piirille, ja tallennettua kellonaikaa ei tästä syystä menetetä.



Kuva 12: S-35390A-J8T1G lohkokaavio [7]

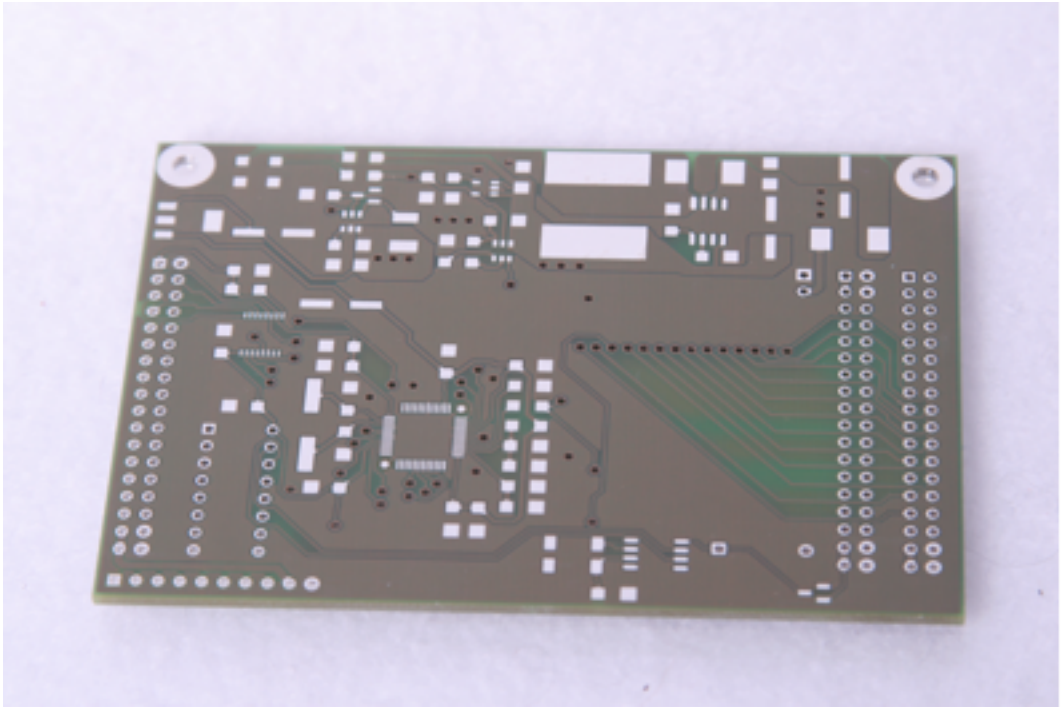
3.2.2 Laajennuslevyn suunnittelu ja valmistus

Piirivalintojen jälkeen kytkentäkaavio piirrettiin PADS 2007 Logic –ohjelmalla. Käytetyt kytkennät ja komponenttiarvot ovat lähes samat, kuin komponenttien datalehdissä ”typical applicationissa” on määritelty. Liityntälevyn piirtämisen voi aloittaa vasta piirikaavion ollessa valmis, koska PADS-ohjelma linkittää Logic puolella määritellyt komponentit ja niiden kytkennät Layout puolelle. Tästä syystä oli hyvin tärkeää piirtää kytkennät sekä komponenttien decalit (ulkomuodot ja kuparipinnat) oikein, koska pienikin virhe tässä vaiheessa pilaa lopulliseen levyyn tulevan kytkennän tai sijoituspaikan. Kytkentäkaavio liitteessä 1.

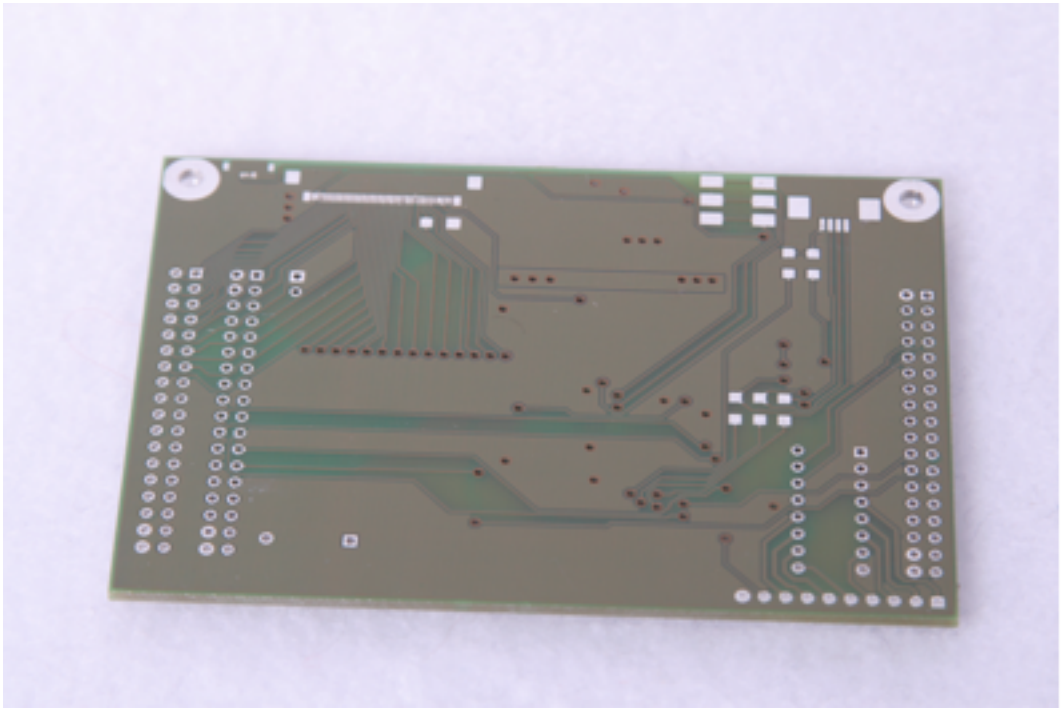
Piirilevyn suunnittelu aloitettiin määrittelemällä levyn ulkomitat, sekä liittimien ja kiinnitysruuvien paikat. Osasijoittelussa pyrittiin ottamaan huomioon, että signaalivedot tulisivat mahdollisimman lyhyiksi, ja että signaali kulkisi luonnollisesti ketjuna. Hakkurivirtalähteen suunnittelussa tulisi erityisesti kiinnittää huomiota osasijoitteluun, jotta hakkurin toiminta pysyisi vakaana, ja että hakkurin kytkemisestä aiheutuneet häiriöt olisivat minimaaliset. Lopputuloksen kannalta pieni virhe tapahtui sijoittamalla lähtökondensaattori hieman sivuun käyttöjännitteen luonnollisesta virtatiestä. Tämä ei kuitenkaan vaikuttanut merkittävästi virtalähteen toimintaan. Myös näytön taustavalon virtalähteessä oleva kela olisi voitu sijoittaa lähemmäs piiriä. Muuten levyn osasijoittelu ja johdotus on toimiva. Liitteissä 2 ja 3 tarkemmat kuvat piirilevyn osasijoittelusta ja kuparikerroksista.

Kun osasijoittelu, johdotus ja kuparialueet olivat valmiit, tilattiin piirilevyt Kamitra Oü:sta (www.kamitra.fi). Piirilevy muutettiin gerbertiedostoiksi, että se voitaisiin valmistaa yrityksessä. Tämä onnistui PADS-ohjelmassa olevalla CAM-työkalulla, jolla saadaan aikaiseksi tarvittavat Gerber-, poraus- ja teräluettelotiedostot. Gerbertiedostot sisälsivät piirilevyn kuparialueiden lisäksi myös piirilevyn ääriviivat, jyrsityt aukot, pinnoitteet ja tilapäiset juotteenestomaskit. Poraustiedostosta ilmenee kaikkien reikien koordinaatit. Poraustiedostoista tulee myös ilmetä ovatko reiät metalloituja vai metalloimattomia. Teräluettelosta ilmenevät kaikki

käytetyt teräkoot. Tämä tieto voi sisältyä myös poraustiedostoon. Piirilevyjä valmistettiin 5 kappaletta.

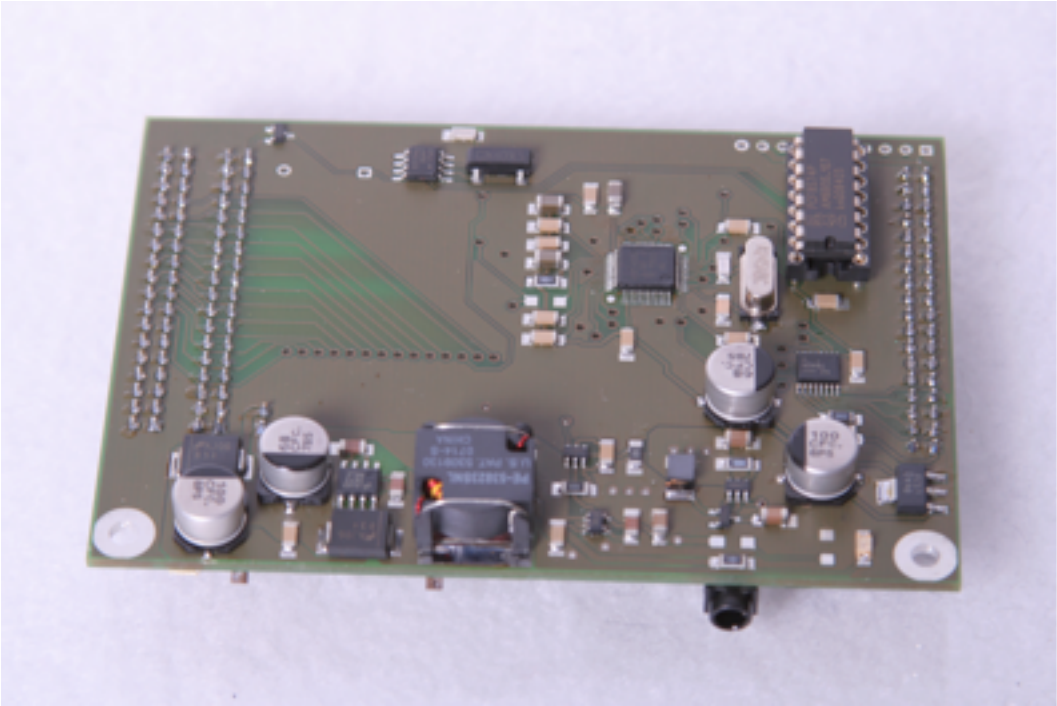


Kuva 13: Valmis piirilevy (yläpuoli)

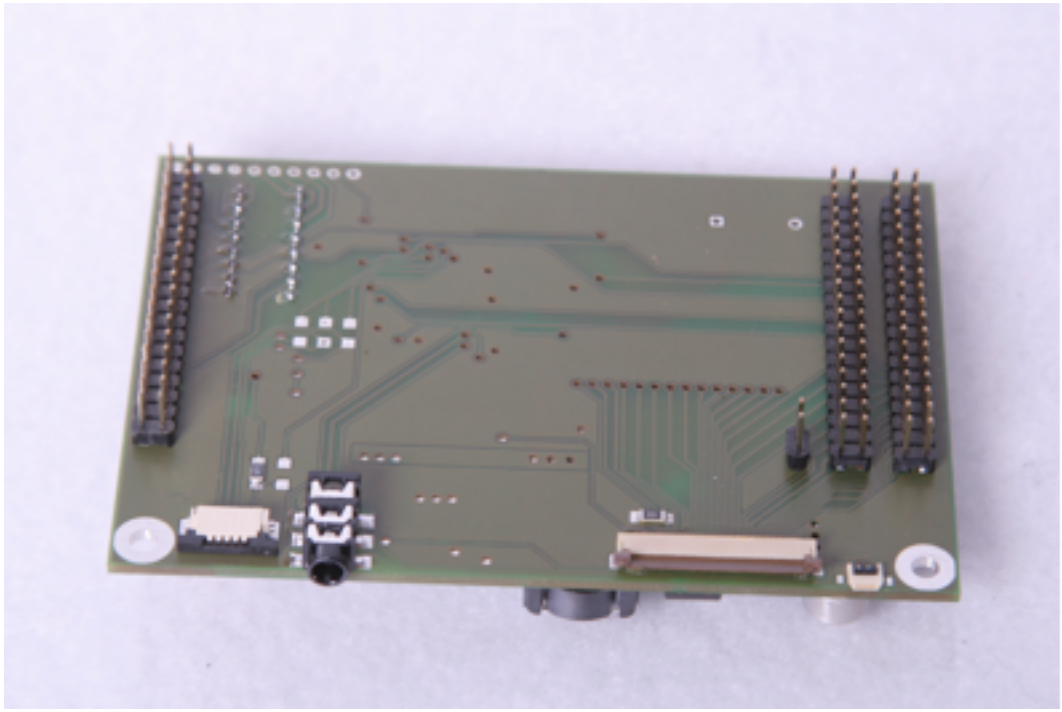


Kuva 14: Valmis piirilevy (alapuoli)

Kokoonpano suoritettiin kahdessa osassa. Virtalähteisiin tarvittavat komponentit ladottiin ensin, jonka jälkeen niiden toiminta testattiin. Toimivuuden toteamisen jälkeen levyn loput komponentit voitiin latoa. Levylle suoritettiin visuaalinen tarkistus, ettei siitä löytyisi oikosulkuja tai muita virheitä. Levyn toiminnallisuuden testaaminen vaati Linux-ympäristön pystyttämisen.



Kuva 15: Ladottu piirilevy (yläpuoli)



Kuva 16: Ladottu piirilevy (alapuoli)

Kuva 17: Ensimmäinen prototyyppi edestä

Kuva 18: Ensimmäinen prototyyppi takaa

4 OHJELMOINTI

4.1 Sulautettu Linux

Sulautettu Linux tarkoittaa Linux-käyttöjärjestelmän hyödyntämistä sulautetuissa järjestelmissä, kuten matkapuhelimissa, kämmentietokoneissa, mediasoittimissa, digibokseissa, navigointilaitteissa ja muissa vastaavissa laitteistoissa. Sulautettu Linux eroaa pöytäkoneversioista siten, että sen alla oleva laitteisto on yleensä huomattavasti rajoitetumpi, kuten esimerkiksi matkapuhelin. Koska sulautetut laitteet ovat yleensä pienempiä ja halvempia kuin pöytäkoneet, ovat myös niiden ominaisuudet rajoitetumpia (vrt. esim. RAM-muistin ja tallennuskapasiteetin koko). Esimerkiksi pöytätietokoneissa käytetään yleensä kovalevyä, kun taas sulautetussa järjestelmässä käytetään flash-muistia. Edellä mainituista syistä sulautettu Linux on huomattavasti karsitumpi ja enemmän muokattavissa oleva käyttöjärjestelmä. Laitteen kehittäjä voi optimoida järjestelmän juuri sen omiin tarpeisiin sopivaksi. Laitteeseen voidaan valita vain ne ajurit ja ohjelmat, joita tarvitaan. Käyttöjärjestelmän suurimpia etuja on myös sen stabiili kernel, ilmainen tuotetuki, sekä sen royalty- ja lisenssivapaa käyttö. [9]

Työssä käytettyyn Linux-ympäristöön saa ohjeita ja tukea osoitteesta www.avrfreaks.net . Kyseinen sivu sisältää wiki-tyyppisen ohjesivuston, jonne käyttäjät saavat kirjoittaa omia artikkeleitaan.

4.1.1 Kehitysympäristön esittely

Kehitysympäristönä käytettiin Atmelilta saatavaa Linux Ubuntua, joka oli saatavilla VMware image –tiedostona osoitteesta http://www.atmel.no/beta_ware/ . Kyseinen käyttöjärjestelmä ei ollut enää kirjoitushetkellä ladattavissa (26.5.2010). Kehitysympäristö voidaan silti helposti pystyttää Linux-pohjaiseen tietokonee-

seen. Jotta kääntäminen olisi mahdollista, tulee käyttöjärjestelmässä olla asennettuna seuraavat paketit:

- C compiler (GCC)
- C++ compiler for Qtopia (G++)
- GNU make
- sed
- flex
- bison
- patch
- gettext
- libtool
- texinfo
- autoconf (version 2.13 and 2.61)
- automake
- ncurses library (development install)
- zlib library (development install)
- libacl library (development install)
- lzo2 library (development install)

4.1.2 Buildroot – Asennus, asetukset ja kääntäminen

Buildroot on kokoelma Makefile- ja patch-tiedostoja, joiden avulla voidaan ristikääntää helposti toolchain (sarja työkaluja, ensimmäisen työkalun lähtö on seuraavan työkalun tulo jne.) ja root filesystem halutulle Linux-alustalle käyttäen uC-libc C –kirjastoa. Järjestelmä osaa automaattisesti hakea valitut paketit netistä, sekä purkaa ja asentaa ne. Järjestelmä on erittäin hyödyllinen henkilöille, jotka kehittävät pieniä sulautettuja järjestelmiä. [9]

Buildroot on saatavilla sivulta http://buildroot.uclibc.org/ . Työssä käytetty versio oli 2009.08 . Atmelin sivuilta http://www.atmel.no/buildroot/buildroot-src.html oli saatavilla myös buildroot-paketteja. Ne eivät kuitenkaan työhön soveltuneet, sillä versiosta 2.2.1 löytyy äänipiiriin tarvittava AC97-ajuri, mutta uudemmassa 2.3.0 versiossa ei kyseistä ajuria ollut. Kummastakin paketista puuttui lisäksi hyvin oleelliset Qt-ympäristöön tarvittavat paketit. Työ myöhästyi osaksi tästä syystä, koska AC97-ajurit ja Qt-tuki ilmestyivät vasta elokuussa 2009 kernelin versioon 2.6.30 .

4.1.2.1 Alkuasetukset

Buildroot-paketti tulee ladata sivuilta http://buildroot.uclibc.org/ , jonka jälkeen buildroot-2009.08.tar.bz2 –tiedosto tulee purkaa. Puretun kansion juuressa tulee suorittaa komento:

```
make atngw100_defconfig
```

Tämä käsky muodostaa Atmelin NGW100 –alustaan tarvittavat asetukset buidrootin juuressa olevaan .config-tiedostoon. Config-tiedosto sisältää alustaan tarvittavat asetukset ja paketit, esim. prosessorityypin. Komennon jälkeen ohjelma kysyy uusimpaan päivitykseen liittyviä paketinvalintoja. Näissä kysymyksissä alkuasetukset ovat kunnossa muutoin paitsi kernelin version osalta. Työssä käytettävän kernelin version tulee olla 2.6.30, koska tämä versio sisältää työssä tarvittavat AC97-ajurit.

Buildrootin juuressa on erilaisia kansioita, joiden sisällön tietäminen on oleellista. Hakemisto sisältää seuraavat kansiot:

- Config.in
    - Ylimmät määritystiedostot kconfig:ia varten.
- .defconfig
    - Alkuperäisasetukset kääntämistä varten.

- docs/
    - Buildrootin dokumentaatio.
- Makefile
    - Ylin makefile, joka sisältää alkuarvoiset määritykset siitä, miten root filesystem käännetään. Tämä tiedosto sisältää kaikki muut makefile-tiedostot, jotka on hajautettu eri hakemistoihin.
- package/
    - Kaikki makefile-tiedostot, jotka sisältävä tiedon jokaisesta kirjastosta ja ohjelmasta, joka täytyy kääntää.
- project/
    - Projektien kääntämiseen tarvittavia tietoja.
- target/
    - Sisältää tarvittavat tiedot siitä, miten kääntää tietylle alustalle tuleva järjestelmä. Sisältää patch-tiedostoja kernelin konfigurointiin ja muita alustakohtaisia tiedostoja.
- toolchain/
    - Makefile-tiedostoja, jotka määrittelevät, miten järjestelmän tulee kääntää, ja mitä optioita käyttäjä on valinnut toolchainia varten. Ilman toolchainia kääntäminen on mahdotonta.

Onnistuneen kääntämisen jälkeen hakemistoon tulee myös kansiot:

- binaries/
    - Käännetyt image-tiedostot kansiossa binaries/<projektin nimi>/
- build_avr32/
    - Kääntämistä varten tarkoitettu kansio jokaista kirjastoa ja ohjelmaa varten. Oletusarvoisesti sisältää myös staging_dir-kansion, joka sisältää toolchainin ja kirjastot.
- project_build_avr32/
    - Projektikohtainen Linux kernel, busybox ja purettu root filesystem.
- dl/

    - Ladatut lähdetiedostot, jotka tarvitsee hakea vain kerran.
- toolchain_build_avr32/
    - Purettu ja käännetty toolchain

4.1.2.2 Paketit

Buildrootissa käytettävien pakettien valinta tapahtuu buildrootin juurihakemistossa käynnistettävällä komennolla:

```
make menuconfig
```

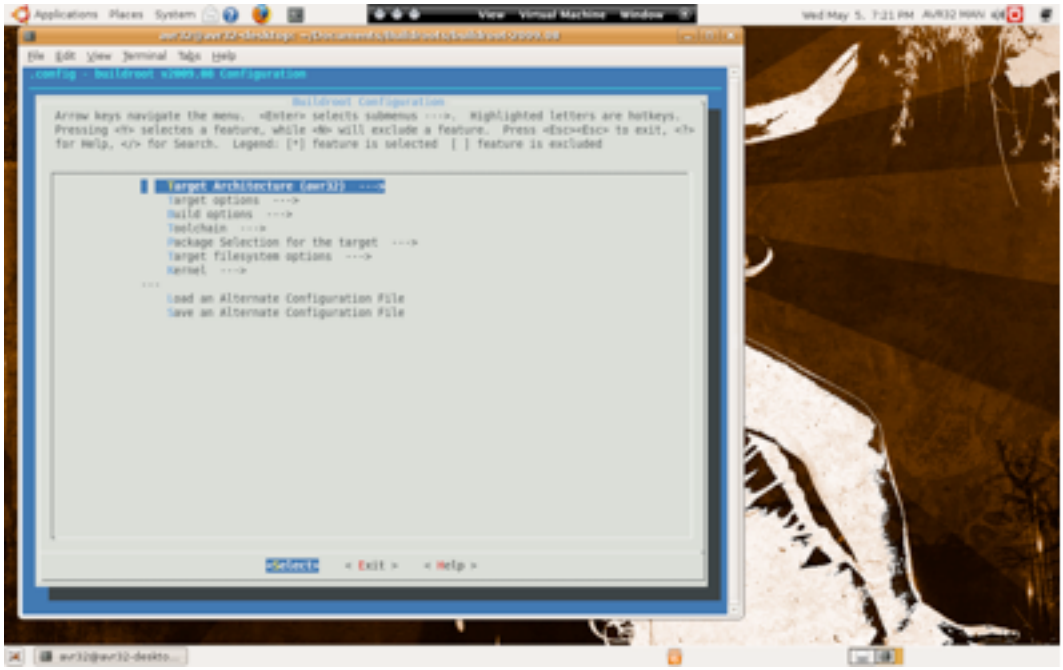
Em. komento käynnistää graafisen ohjelman, joka auttaa hahmottamaan valittavat paketit, ja helpottaa valintaprosessia. Valintojen jälkeen menuconfig-ohjelma muokkaa .config-tiedostoa, josta selviää, mitkä paketit root filesystemiin täytyy ladata ja asentaa. Tämän tiedoston pohjalta tapahtuu itse käännösprosessi.

Työhön tarvittavat lisäpaketit ja asetukset ovat:

- Audio
    - Alsa-lib
    - Alsa-utils
- Video
    - Fbv
    - Qt
        - Approve free licence
        - Enable GIF support
        - JPEG support (system libjpg)
        - PNG support (systen libpng)
        - Mouse driver
            - Tslib

- Phonon module

- Libts



Kuva 19: Buildroot menuconfig

4.1.2.3 Ajurit

Jotta alustassa olevat lisälaitteet toimisivat käyttöjärjestelmässä, täytyy ajurit lisätä kerneliin. Ajurit ovat valittavissa graafisella ohjelmalla, joka käynnistyy komennolla:

```
make linux26-menuconfig
```

Ohjelmassa tulee kiinnittää erityistä huomiota ”*” ja ”M” symboleihin. ”*” tarkoittaa, että kyseinen ajuri tai paketti asennetaan sisäänrakennettuna, ja ”M” merkki tarkoittaa, että ajuri tai paketti asentuu moduulina. Jos haluttua ajuria halutaan käyttää moduulina, täytyy se aina erikseen ladata käyttöjärjestelmään. Moduulin vahvuutena on se, että se voidaan ns. ladata ja purkaa käyttöjärjestelmän ollessa käynnissä. Täten järjestelmä vie vähemmän resursseja. Sisäänrakennetun

ajurin vahvuutena on, että jos kyseistä ajuria tai pakettia käytetään usein, on sen erikseen lataaminen turhaa.

Vaikka ajuri valitaan listasta, se ei silti tarkoita sitä, että kyseinen piiri tai ominaisuus olisi välittömästi käytettävissä. Tiettyjä ajureita täytyy erikseen kutsua tai määritellä käynnistyksen yhteydessä. Tästä lisää seuraavassa luvussa.

Liitteessä 6 on lueteltu työssä käytetyt ajurit ja määritykset. Määritykset on esitetty englanniksi, koska tieto vääristyisi käännösvaiheessa.

4.1.2.4 Alustan konfigurointi

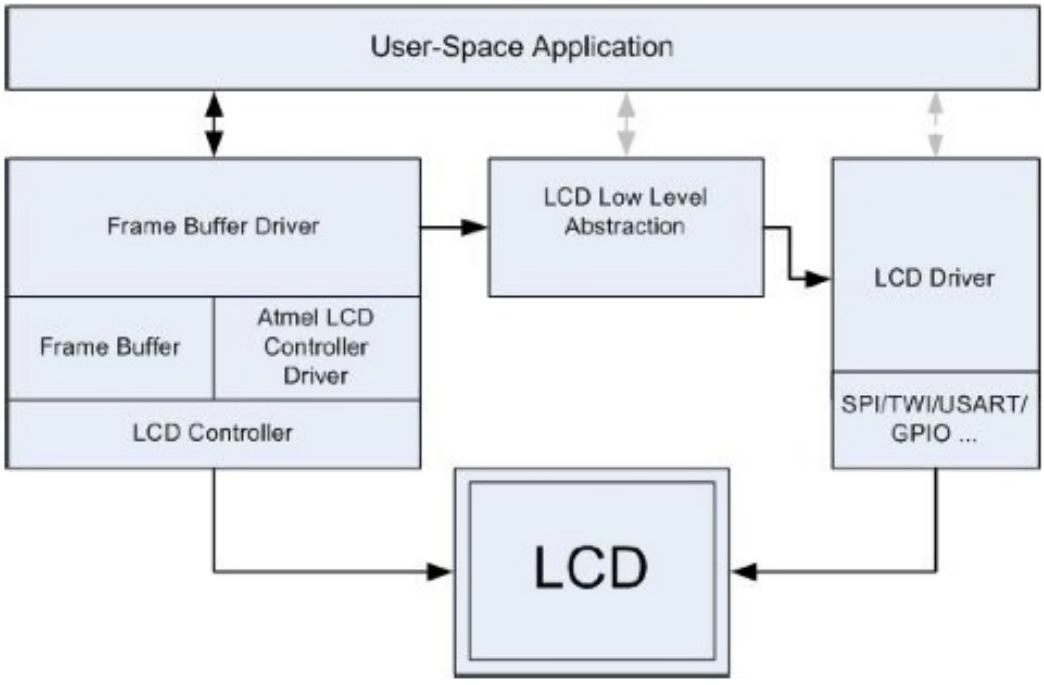
Alustan konfigurointi tapahtuu muokkaamalla ”/project_build_avr32/atngw100/linux-2.6.30.2/arch/avr32/boards/atngw100” kansiosta löytyvää setup.c –tiedostoa. Tämä tiedosto sisältää koodin, joka käynnistyy, kun käyttöjärjestelmä on ladattu muistikortilta. Tässä vaiheessa tehdään viimeiset määrittelyt käynnistettäviin ajureihin. Liitteessä 4 setup.c –tiedosto.

Seuraavista otteista nähdään, miten näytön määritykset asetellaan frame bufferille. Asetuksissa on määritelty mm. näytön resoluutio, päivitysnopeus, värisyvyys ym. Näytön asetuksista ja Linux konfiguroinnista on saatavissa lisää tietoa osoitteesta ”http://www.avrfreaks.net/wiki/index.php/Documentation:Linux/Framebuffer” sekä ”http://www.atmel.com/dyn/resources/prod_documents/doc32105.pdf”.

```
static struct fb_videomode __initdata lq043t3dx02_modes[] = {
        {
                .name           = "480x272 @ 59.94Hz",
                .refresh        = 59.94,
                .xres           = 480,          .yres           = 272,
                .pixclock       = KHZ2PICOS(9000),

                .left_margin    = 2,            .right_margin   = 2,
                .upper_margin   = 3,            .lower_margin   = 9,
                .hsync_len      = 41,           .vsync_len      = 1,

                .sync           = 0,
                .vmode          = FB_VMODE_NONINTERLACED,
        },
};
```

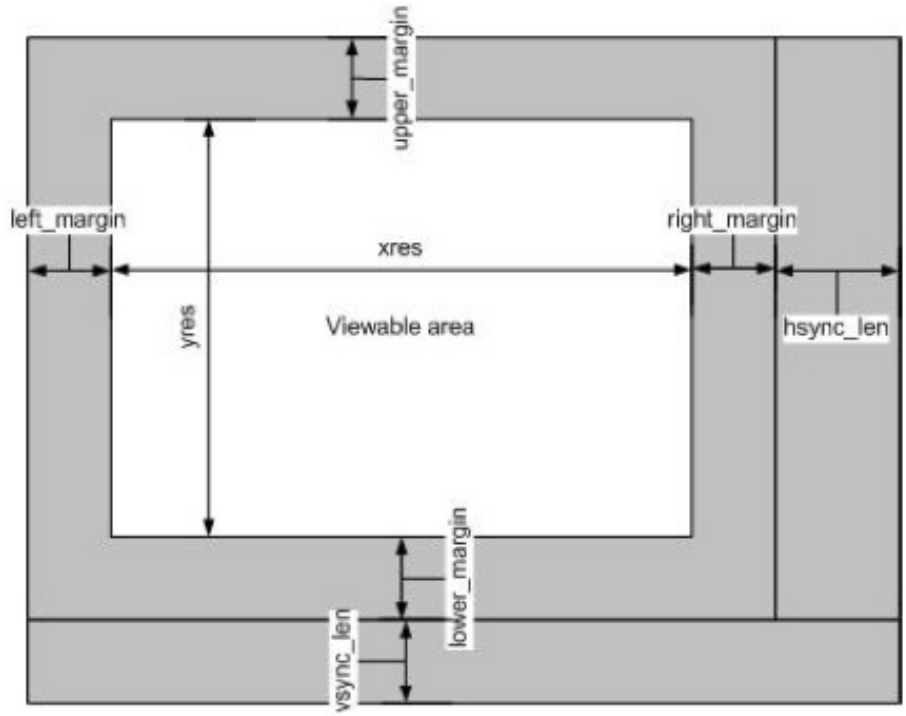


Kuva 20: Frame bufferin toiminta [10]

Parameter	Description
name	Name of the frame buffer video mode
refresh	Refresh rate of the display
xres	Horizontal display size in pixels (active/visible pixels)
yres	Vertical display size in pixels
pixelclock	Desired pixel clock rate in pico seconds. Use the macro KHZ2PICOS() to convert from kHz to pico seconds. The pixel clock can be calculated from: Pixel clock in Hz = (xres + left_margin + right_margin + hsync_len) x (yres + upper_margin + lower_margin + vsync_len) x refresh
left_margin, right_margin	Horizontal blanking in number of pixel clocks (refer to picture…). The minimum is 1 although 0 will be accepted too (both will lead to a register value of 0)
upper_margin, lower_margin	Vertical blanking in scanlines
hsync_len	Horizontal sync pulse width in pixel clock cycles. The minimum is 1 although 0 will be accepted too (both will lead to a register value of 0)
vsync_len	Vertical sync pulse width in pixel clock cycles. The minimum is 1 although 0 will be accepted too (both will lead to a register value of 0)
sync	Sync pulse polarity. Set FB_SYNC_HOR_HIGH_ACT if horizontal sync pulse is active high. Set FB_SYNC_VERT_HIGH_ACT if vertical sync pulse is active high.
vmode	Only non-interlaced is supported. Set this to FB_VMODE_NONINTERLACED

Kuva 21: Fb_videomode selitykset [10]

Alla olevasta kuvasta voidaan huomata ns. blanking-alueet (harmaa väri). Sisäisiin synkronointitarkoituksiin näyttö tarvitsee tätä ”tyhjää” dataa. Tämä data ei sisälly frame bufferiin. Ylimääräisen datan käsittelee vain LCD-kontrolleri. Nämä ajoitukset ovat yleensä esitelty näytön datalehdessä.



Kuva 22: Kuvan määritykset [10]

Seuraavaksi tarvitaan frame bufferin monospecs-asetukset. Tarkemmat tiedot löytyvät include/linux/fb.h –tiedostosta. Alla olevasta taulukosta nähdään karkeasti asetuksien tarkoitukset.

```
static struct fb_monspecs __initdata lq043t3dx02_default_monspecs = {
        .manufacturer       = "SHA",
        .monitor            = "LQ043T3DX02",
        .modedb                 = lq043t3dx02_modes,
        .modedb_len         = ARRAY_SIZE(lq043t3dx02_modes),
        .hfmin              = 14915,
        .hfmax              = 17638,
        .vfmin              = 53,
        .vfmax              = 61,
        .dclkmax            = 9260000,
};
```

Parameter	Description
manufacturer	Display manufacturer name (max. 4 characters)
monitor	Name of the display (max. 14 characters)
modedb	Pointer to the video modes array (fb_videomode structure)
modedb_len	Length of the video modes array. The macro ARRAY_SIZE() can be used to get the length.
hfmin	Lower limit of the horizontal display frequency in Hz
hfmax	Upper limit of the horizontal display frequency in HZ
vfmin	Lower limit of the vertical display frequency in Hz
vfmax	Upper limit of the vertical display frequency in Hz
dclkmax	Upper limit of the pixel clock

Kuva 23: Monospecs-asetukset [10]

Viimeiseksi tarvitaan atmel_lcdfb_info tiedot. Tätä käytetään platform device datana ja täten toimii LCD-kontrollerin asetuksina. Tarkemmat määrittelyt löytyvät include/video/atmel_lcdc.h –tiedostosta.

```
static struct atmel_lcdfb_info __initdata atngw100_lcdc_data = {
        .default_bpp            = 24,
        .default_dmacon                 = ATMEL_LCDC_DMAEN | AT-
MEL_LCDC_DMA2DEN,
        .default_lcdcon2        = (ATMEL_LCDC_DISTYPE_TFT
                        | ATMEL_LCDC_SCANMOD_SINGLE
                        | ATMEL_LCDC_INVLINE_INVERTED
                        | ATMEL_LCDC_INVFRAME_INVERTED
                        | ATMEL_LCDC_INVCLK_NORMAL
                        | ATMEL_LCDC_PIXELSIZE_24
                        | ATMEL_LCDC_CLKMOD_ALWAYSACTIVE
                        | ATMEL_LCDC_MEMOR_BIG),
        .default_monspecs       = &lq043t3dx02_default_monspecs,
        .guard_time             = 2,
};
```

Parameter	Description
default_bpp	Bits per pixel to use. Possible values are 1, 2, 4, 8, 15, 16, 24 and 32. The memory layout for these values is described in the datasheet of the device. For instance 32bit means 3x8bit for the three colors red, green, blue and empty 8bit at the end.
default_dmacon	Setting ATMEL_LCDC_DMAEN is mandatory in order to enable the LCD controller DMA engine. The ATMEL_LCDC_DMA2DEN will activate the double buffering of the frame buffer.
default_lcdcon2	This is the default configuration of the LCD controller register "LCDCON2". The bit-fields INVFRAME, INVLINE and PIXELSIZE are automatically set according to the video mode settings. All other settings of this register may be configured by using the available macros in the include/video/atmel_lcdc.h file.
default_monspecs	Pointer to the monitor specifications structure.
guard_time	Delay in frame periods between applying control signals to the LCD module and setting PWR high, and between setting PWR low and removing control signals from LCD module.
lcd_wiring_mode	Specifies the hardware wiring of the LCD. Possible values are ATMEL_LCDC_WIRING_BGR and ATMEL_LCDC_WIRING_RGB (see chapter "3.6.2.1 RGB frame buffer layout support with hardware swap" for more details)

Kuva 24: atmel_lcdfb_info [10]

Näytön enable-signaali on kytketty prosessoriin (portti A31), minkä vuoksi ennen ”add device” –komentoa näyttö kytketään toimintaan.

```
#define PIN_LCD_DISP       GPIO_PIN_PA(31)
```

```
at32_select_gpio( PIN_LCD_DISP, 0);
        gpio_request(PIN_LCD_DISP, "LCD_ENABLE");
        gpio_direction_output( PIN_LCD_DISP, 0 );
        gpio_set_value( PIN_LCD_DISP, 1 );          /* Enable LCD */
        udelay( 1 );                                /* Wait before init */

        at32_add_device_lcdc(0, &atngw100_lcdc_data,
                fbmem_start, fbmem_size,
                ATMEL_LCDC_ALT_18BIT | ATMEL_LCDC_PE_DVAL);
```

Kosketusnäytön asetukset on esitelty alla. Platform datan lisäksi on tärkeää ilmoittaa portti, johon AD-muuntimelta tuleva keskeytyssignaali tulee.

```
#define PB_EXTINT_BASE  25
#define TS_IRQ          0
#define PIN_TS_EXTINT   GPIO_PIN_PB(PB_EXTINT_BASE+TS_IRQ)
```

```
static int ads7846_pendown_state(void)
{
        return !gpio_get_value( PIN_TS_EXTINT );        /* PENIRQ.*/
}
```

```
static struct ads7846_platform_data ads_info = {
        .model                      = 7846,
        .keep_vref_on               = 0,    /* Use external VREF pin */
        .vref_delay_usecs           = 0,
        .vref_mv                    = 3300, /* VREF = 3.3V */
        .settle_delay_usecs         = 800,
        .penirq_recheck_delay_usecs     = 800,
        .x_plate_ohms               = 750,
        .y_plate_ohms               = 300,
        .pressure_max               = 4096,
        .debounce_max               = 1,
        .debounce_rep               = 0,
        .debounce_tol               = (~0),
        .get_pendown_state          = ads7846_pendown_state,
        .filter                     = NULL,
        .filter_init                = NULL,
};
```

Alla olevat asetukset liittää ADS7846-piirin SPI-väylään.

```
static struct spi_board_info spi0_board_info[] __initdata = {
        {
                .modalias       = "mtd_dataflash",
                .max_speed_hz          = 8000000,
                .chip_select    = 0,
        },
/*************************************************************/
/* Added 11.4.2009 Jani Perkio.                              */
/* ADS7846 Touch Screen at SPI BUS                           */
        {
                .modalias       = "ads7846",
                .chip_select    = 1,
                .max_speed_hz          = 31250*26,
                .bus_num        = 0,
                .platform_data = &ads_info,
                .irq                = AT32_EXTINT(TS_IRQ),
        },
/*************************************************************/
};
```

Viimeiseksi määritellään kosketuksesta aiheutuva keskeytys prosessorille ja piiri rekisteröidään alustalle.

```
at32_select_periph( GPIO_PIOB_BASE, 1 << (PB_EXTINT_BASE+TS_IRQ),
                GPIO_PERIPH_A, AT32_GPIOF_DEGLITCH);
        set_irq_type( AT32_EXTINT(TS_IRQ), IRQ_TYPE_EDGE_FALLING );
        spi_register_board_info(spi0_board_info,ARRAY_SIZE(spi0_board_info)
);
```

Alla on esitelty ne asetukset, joilla äänipiiri on saatettu toimintaan. Oleellisin asetus on lisätä piirille menevä reset-signaalin portti (A30) ac97-ajureihin. Add_device –komento lisää piirin alustaan.

```
#define PIN_AC97_RST_N   GPIO_PIN_PA(30)

…

static struct ac97c_platform_data __initdata ac97c0_data = {
        .reset_pin        = PIN_AC97_RST_N,
};

…

at32_add_device_ac97c(0, &ac97c0_data, AC97C_BOTH);
```

4.1.2.5 Kääntäminen

On suositeltavaa suorittaa ensimmäinen kääntökerta heti, kun pakettien valinta on tehty menuconfig-ohjelmalla. Tämä johtuu lähinnä siitä, että buildroot on hakenut netistä tarvittavat tiedostot, joista oleellisin on kernel. Kun kernel on ladattu ja käännetty buildroottiin, voidaan siihen helposti lisätä tarvittavat ajurit. Kun ajurit ja alustan konfigurointitiedosto on määritelty, voidaan kääntöprosessi suorittaa uudestaan. Kääntäminen tapahtuu komennolla ”make”.

Kun buildroot on suorittanut kääntämisen loppuun, voidaan valmis root filesystem purkaa muistikortille. Binääritiedostot löytyvät oletuksena kansiosta ”/binaries/atngw100”. Kansiosta löytyy ”rootfs.avr32-yyyymmdd.tar” –niminen tiedosto, joka sisältää root filesystemin. Tämä tiedosto voidaan purkaa muistikortille komennolla:

```
tar xf /rootfs.avr32-yyyymmdd.tar –C /muistikortin/sijainti
```

Ennen purkamista täytyy varmistaa, että kyseinen muistikortti on tyhjä ja alustettu “ext2” muotoon.

4.1.3 U-Boot – Käynnistyslataimen esittely ja asetukset

Prosessori voi hyödyntää ainoastaan ohjelmia, jotka on ladattu RAM- tai ROM-muistiin. Kun laitteistoon kytketään virta, ei siinä vielä ole käyttöjärjestelmää ROM- tai RAM-muistissa. Laitteen täytyy käynnistää ROM-muistista ohjelma, jonka avulla sen on mahdollista suorittaa ohjelmia tallennusmediasta (esim. kovalevyltä tai flash-muistista) RAM-muistiin. Tämän ohjelman ainoa funktio on hakea ohjelmia tai dataa RAM-muistiin. Ohjelmaa kutsutaan boot loaderiksi.

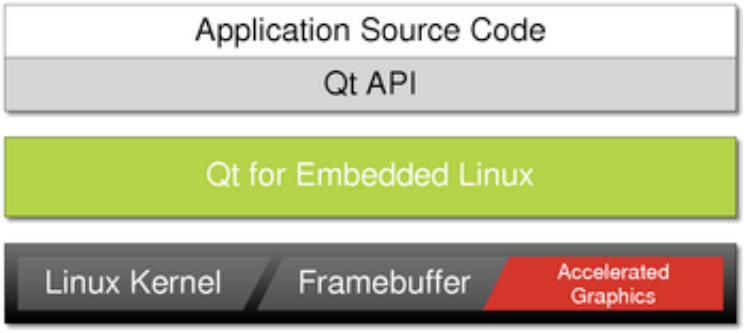
U-Boot (Universal Bootloader) on boot loader, eli käynnistyslatain, erilaisille prosessorityypeille kuten PPC, ARM, AVR32, MIPS, x86 jne. U-Boot:ia pidetään monipuolisimpana, joustavimpana ja kehitetyimpänä, ja siksi se on yksi suosituimmista ja käytetyimmistä boot loadereista. [9]

Työssä käytettävä Linux-käyttöjärjestelmä sijaitsee SD-muistikortilla. Jotta käyttöjärjestelmä voidaan käynnistää, täytyy se ensin ladata alustan RAM-muistiin. U-Boot:in asetuksia voidaan muokata painamalla välilyöntinäppäintä heti käynnistyksen jälkeen. Tämän jälkeen voidaan antaa käynnistyksen yhteydessä tapahtuvia määrittelyjä. Alla annetut komennot:

```
Uboot> askenv bootcmd
Please enter 'bootcmd': mmcinit; ext2load mmc 0:1 0x10400000 /boot/uImage;
bootm
Uboot> set bootargs 'console=ttyS0 root=/dev/mmcblk0p1 fbmem=600k rootwait'
```

Kyseiset komennot mahdollistavat käyttöjärjestelmän lataamisen SD-muistikortilta sekä frame bufferin käytön.

U-Boot:in käynnistymistietoihin annettu ”fbmem=600k”, määrittelee frame bufferin muistin koon. Jos komento olisi ”fbmem=600k@10200000” niin frame bufferin käyttämä muistialue alkaisi muistin osoitteesta 10200000. Muistialue sisältää yhden kuvan (frame) datan. Frame bufferin toiminta perustuu siihen, että ohjelmat voivat siirtää kuvatiedon suoraan kernelille. Esimerkiksi projektissa käytetty Qt:lla toteutettu graafinen käyttöliittymä hyödyntää tätä tekniikkaa. Qt-ohjelma siirtää näytettävän kuvan suoraan frame bufferin muistialueeseen, josta kernel tulostaa sen edelleen näytölle.



Kuva 25: Qt:n ja sulautetun Linuxin rajapinta [8]

Root filesystemin juuressa oleva kansio ”boot” sisältää tiedoston ”uImage-yyyymmdd”. Tämä tiedosto täytyy nimetä ”uImage” –nimiseksi, koska U-Boot :ssa määritelty kernelin lataus on em. nimiselle tiedostolle. Ilman tätä määritystä bootloader ei pysty lataamaan kerneliä muistiin. Jos tiedostoa ei löydy, tai se on vahingoittunut, U-Boot antaa virheilmoituksen ”bad magic number”.

4.1.4 init.d – Kansion sisältöön tehtävät muutokset

init.d –kansiosta voidaan löytää kaikki käynnistyksen ja sammutuksen yhteydessä tapahtuvat komennot. Nämä komennot ovat tiedostoissa, jotka on nimetty joko S- tai K-alkuisiksi, ja kirjaimen jälkeen tulee juokseva numerointi. S-alkuiset tiedostot tarkoittavat käynnistyksessä tapahtuvia komentoja, ja K-alkuiset sammutuksen yhteydessä tapahtuvia komentoja.

Buildrootin kääntämiä käynnistystiedostoja täytyy muuttaa laitteen saattamiseksi toimintakuntoon. Kansiosta tulee poistaa S15localfs- ja K90localfs-tiedostot, koska näiden sisältävät komennot kiinnittää kehitysalustan muistissa olevia tiedostoja ja kansioita SD-muistikortilla olevien kansioiden päälle. Lisäksi kansioon luotiin S98user-niminen tiedosto, joka sisältää seuraavat komennot:

```
#!/bin/sh
if [ "$1" = start ]
then
        # Display an image
        fbv -c -i /Startup.jpg &

        # Touch screen setup
        export TSLIB_FBDEVICE=/dev/fb0
        export TSLIB_CONSOLEDEVICE=none
        export TSLIB_CONFFILE=/etc/ts.conf
        export TSLIB_CALIBFILE=/etc/pointercal
        export TSLIB_TSDEVICE=/dev/input/event0
        export QWS_MOUSE_PROTO=tslib:/dev/input/event0
        export SDL_MOUSEDRV=TSLIB
        export SDL_MOUSEDEV=/dev/input/tslib0
        export SDL_NOMOUSE=1
fi
```

Komento ”fbv -c -i /Startup.jpg &” asettaa juurihakemistossa olevan kuvan ”Startup.jpg” laitteen näytölle. ” # Touch screen setup” alla olevat export-komennot aktivoivat kosketusnäyttöön liittyvät asetukset.

4.1.5 Libts – Kosketusnäytön asetukset

Libts on asennuspaketti, jonka valinta esiteltiin kappaleessa ”Paketit”. Tämä paketti mahdollistaa kosketusnäytön toiminnan. Paketin mukana tulee kirjastot, joiden avulla ajureilta vastaanotettu data voidaan käsitellä, ja edelleen välittää ohjelmistolle. Kosketusnäyttö ei toimi suoraan kääntämisen jälkeen, vaikka ajurit ja paketit olisivatkin oikein asennettu. Root filesystemin /etc/ -kansiossa olevaa ts.conf –tiedostoa täytyy muokata, ja tiedoston sisältöön täytyy tehdä määritykset, miten ajureilta vastaanotettu data viedään eteenpäin. Työssä käytetyt määritykset ovat:

```
# Uncomment if you wish to use the linux input layer event interface
module_raw input

module pthres pmin=1
module variance delta=30
module dejitter delta=100
module linear
```

Määritysten jälkeen kirjastolle täytyy kertoa näyttölaitteen ja A/D-muuntimen ajurit, sekä ts.conf- ja kalibrointitiedoston sijainti. Nämä asetukset tapahtuvat edellisessä kappaleessa olevan S99User-tiedoston latauksen yhteydessä (ks. Kohta # Touch screen setup). Em. export-komennot määrittelevät näytöllä ohjaamiseen tarvittavat tiedot tslib-kirjastolle ja Qt:n kirjastolle. Kosketusnäyttö täytyy myös kalibroida ennen käyttöönottoa. Kalibrointi tapahtuu komennolla ”ts_calibrate”, jonka jälkeen kosketetaan esim. styluskynällä mahdollisimman tarkasti näytössä näkyviin risteihin. Kalibroinnin suoritettua ohjelma luo /etc/ -hakemistoon ”pointercal” nimisen tiedoston, joka sisältää kalibrointidatan. Näiden asetusten jälkeen kosketusnäyttö on käyttövalmis.

Kosketusnäytön kalibrointi ei onnistunut projektissa, koska kalibrointiohjelman tuetut värisyvyydet olivat 16bpp ja 32bpp. Koska työssä käytettävä näyttö on 24bpp, kalibrointiohjelma käynnistyy, mutta kalibroinnissa olennaisesti tarvittavaa kuvaa ei näy. Tästä syystä projektissa olevaa näyttöä ei saatu kalibroitua oikein, ja tämän seurauksena näytön toiminta jäi rajalliseksi.

Tilannetta yritettiin korjata kopioimalla toisilta käyttäjiltä pointercal-tiedostoa, mutta tällä päästiin vain tyydyttäviin tuloksiin. Näytön tarkkuus ei ollut toivotulla tasolla ja kynällä täpäytys (tai klikkaus) ei toiminut.

4.2 Qt

4.2.1 Yleistä Qt:sta

Qt on Nokian omistama, monelle käyttöjärjestelmälle käännettävissä oleva ohjelma- ja käyttöliittymärajapinta. Kehitystyökalut ja dokumentaatio on ladattavissa osoitteesta http://qt.nokia.com/ . Ohjelmointikieli on C++ -pohjainen. Qt:lla kirjoitettu koodi voidaan kääntää eri alustoille ja käyttöjärjestelmille. Tuetut käyttöjärjestelmät ovat:

- Sulautettu Linux
- Mac OS X
- Windows
- Linux / X11
- Windows CE / Mobile
- Symbian
- Maemo

Qt:n suurena vahvuutena on sen lisenssivaihtoehdot. Ohjelmaa voidaan käyttää kahdella eri lisenssillä. Vaihtoehtoina ovat joko LGPL tai Commercial. Lyhyesti selitettynä ainoa ero lisensseillä on se, että LGPL-lisenssin alla tehdyn ohjelman lähdekoodi täytyy olla julkinen tai antaa pyydettäessä. Kolmas osapuoli voi kopioida tai muokata tämän lisenssin alla tehtyä koodia. Commercial-lisenssin alaisena tehty koodi voidaan pitää yksityisenä, mutta tästä lisenssistä täytyy maksaa Nokialle. LGPL-lisenssi sen sijaan on ilmainen. [8]

Kehitystyökaluna on saatavilla Qt Creator –ohjelma, jonka avulla ohjelmien lähdekoodi ja graafinen käyttöliittymä on helppo toteuttaa. Ohjelma on saatavilla Windows, Linux ja OS X -käyttöjärjestelmille. Ohjelman asennusvaiheessa ei tarvitse tietää, että mille alustalle ohjelman täytyy kääntää, mutta koodin muodostamisessa täytyy olla tietoinen alustan ominaisuuksista. Näytön koko ja alustan tu-

kemat paketit täytyy olla tiedossa ennen kuin ohjelmointi aloitetaan. Ohjelmakoodin kääntäminen AVR32-alustalle on esitelty kappaleessa 4.2.5 .

4.2.1.1 Tiedostorakenne

Qt:n tiedostorakenne on yksinkertainen. Projekti aloitetaan luomalla .pro – tiedosto, josta selviää kaikki projektiin liittyvät tiedostot. Qt Creator lisää tähän tiedostoon automaattisesti projektiin lisätyt tiedostot. Työssä tarvittu phonon kirjasto täytyi lisätä erikseen projektitiedostoon, koska ohjelma ei tätä automaattisesti osannut tehdä.

Projektitiedoston lisäksi projektista löytyy forms-, header-, source- ja resources-tiedostoja. Header- ja source-tiedostot sisältävät ohjelman koodin. Forms-tiedostot sisältävät Qt Creatorilla tehdyt graafisen käyttöliittymän komponentit ja resources-osuudesta löydetään projektin kaikki kuvamateriaali.

4.2.2 Graafisen käyttöliittymän ominaisuudet

Ohjelman perusominaisuuksiin tulee kuulua play, pause, next track ym. Ominaisuuksiin kuuluu myös albumin vaihto. Jotta kosketusnäytön ominaisuuksia voitaisiin hyödyntää, esitetään levyn kansikuva näytöllä, ja levyn vaihto tapahtuu interaktiivisesti koskettamalla seuraavan levynkannen kuvaa. Lisäominaisuudeksi valittiin myös ns. SeekSlider, jonka avulla voidaan hypätä kappaleen eri kohtiin. Ohjelman komponentit tulevat olla siis seuraavat:

- Kappalelistan haku ja -päivitys ennalta määritellystä sijainnista
- Levynkansien lataus
- Toiston ohjaus (play, pause jne.)
- SeekSlider
- Soitettavan kappaleen nimi ja toistoaika

4.2.3 Ohjelman tiedostorakenne

Ohjelma on rakennettu kahdeksaan eri tiedostoon:

- MediaPlayer.pro
- Mainwindow.ui
- Mainwindow.h
- Pictureflow.h
- Resources.qrc
- Main.ccp
- Mainwindow.ccp
- Pictureflow.ccp

Liitteessä 5 ohjelman lähdekoodi.

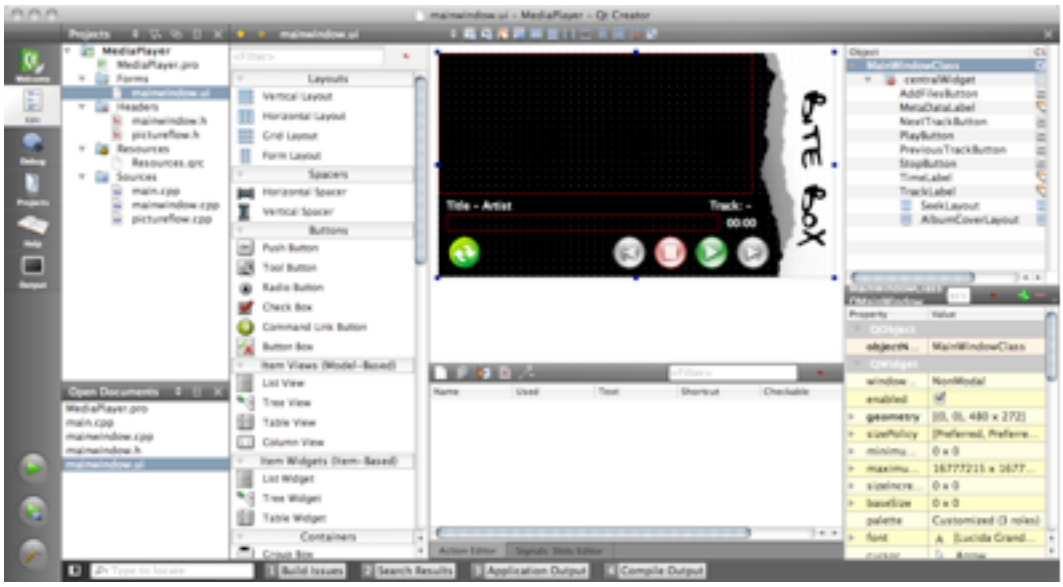
4.2.3.1 MediaPlayer.pro

Tiedosto sisältää kaikki projektiin tarvittavat tiedot. Alla esitettynä tiedoston sisältö.

```
# ------------------------------------------------- #
Project created by QtCreator 2009-04-13T11:59:44
# ------------------------------------------------- #
QT += phonon
TARGET = MediaPlayer
TEMPLATE = app
SOURCES += main.cpp \     mainwindow.cpp \     pictureflow.cpp
HEADERS += mainwindow.h \     pictureflow.h
FORMS += mainwindow.ui
RESOURCES += Resources.qrc
```

4.2.3.2 Mainwindow.ui

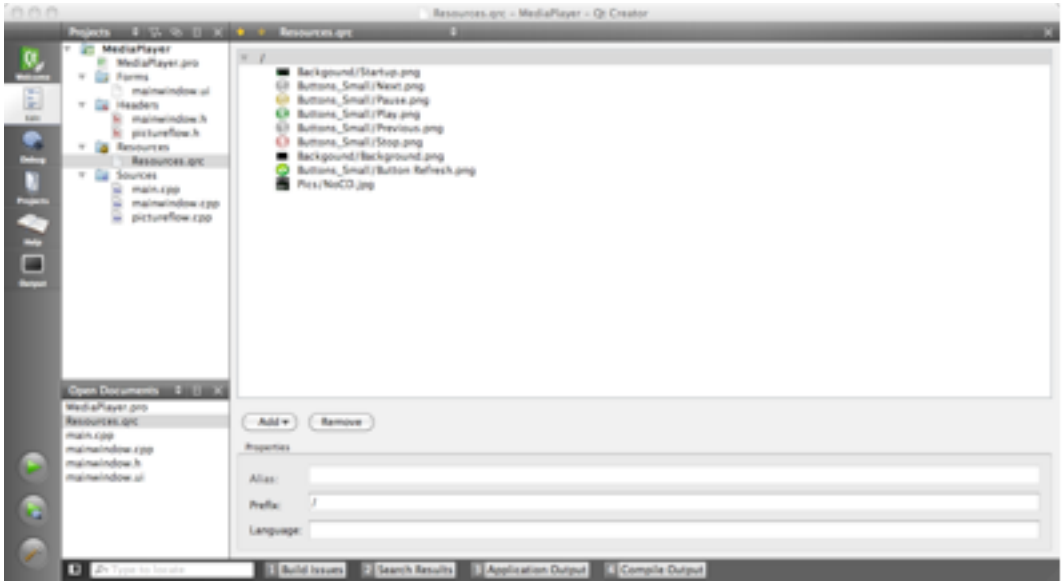
Tiedostossa on esitetty graafisen käyttöliittymän komponentit. Kehitystyökalulla luotiin projektiin tausta, napit ja ns. widget-alueet. Em. alueisiin tuodaan koodissa määritellyt albumien kuvat, soivan kappaleen numero ja esittäjä, sekä kappaleen nimi.



Kuva 26: Graafisen käyttöliittymän kehitystyökalu

4.2.3.3 Resources.qrc

Projektiin liittyvät kuvat on määritelty tähän tiedostoon. Kuvat ovat projektikansiossa ja qrc-tiedosto sisältää vain polun kyseisiin tiedostoihin. Koodia ja graafista käyttöliittymää tehtäessä kaikki kuvalliset resurssit on hyvä olla määriteltyinä, koska tämä helpottaa kehitystyökalun käyttämistä. Alla olevasta kuvasta nähdään kaikki projektiin käytetyt kuvaresurssit. Listasta nähdään myös resurssin nimeä vastaava kuva.



Kuva 27: Resurssienshallinta

4.2.3.4 Main.ccp

Tiedosto sisältää koodin, joka käynnistää ohjelman ja siihen tarvittavan ikkunan. Alla esitetty tiedoston sisältämä koodi.

```
int main(int argc, char *argv[])
{
        QApplication a(argc, argv);
        MainWindow MediaPlayer;
        MediaPlayer.show();
        return a.exec();
}   /* End of main */
```

4.2.3.5 Pictureflow.ccp

Soittimen levyjen esittäminen ja vaihto on toteutettu kuvin elävöitetyllä selaustekniikalla, joka on saman tyyppinen kuin Applen ”cover flow” tekniikka, jota käytetään mm. iTunes-, Finder-, iPhone- ja iPod-tuotteissa. Koodi on hankittu osoitteesta http://code.google.com/p/pictureflow/ . Ohjelma on avoimen lähdekoodin alainen, joten sitä voidaan käyttää projektissa. Ohjelma on widget-

tyyppinen, eli se voidaan lisätä objektina ohjelmaan. Mainwindow.ui tiedostossa oli varattu alue tälle widgetille ja Mainwindow.ccp -tiedostossa määriteltiin alue tämän toiminnon käyttämiseen.

4.2.3.6 Mainwindow.ccp

Soittimen pääasialliset toiminnot on toteutettu Mainwindow.ccp –tiedostoon. Koodi on pyritty toteuttamaan mahdollisimman helppolukuiseksi.

Koodin alussa alustetaan phonon-kirjasto käyttövalmiiksi, jonka jälkeen luodaan graafisen käyttöliittymän komponentit ja yhdistetään ne funktioihin. Lisäksi määritellään erikoistapahtumat, kuten esimerkiksi, mitä tapahtuu kun kappale loppuu, tai kun soitettava kappale vaihtuu.

Ensimmäinen ja monimutkaisin funktio onkin soittolistan haku. Ensimmäiset komennot pysäyttävät musiikin toiston ja tyhjäävät olemassa olevan soittolistan ja albumien kansien tiedot. Haku aloitetaan ennalta määritellystä hakemistosta, joka on määritetty define-komennolla aivan koodin alussa joko, OSX_MUSIC_DIR tai LINUX_MUSIC_DIR kohdassa (riippuen siitä, onko EMBEDDED_LINUX -lippu aktiivisena). Haussa on käytetty hyväksi Qt:n mukana tulevaa dirIterator-funktiota. Haussa listataan kaikki tiedostot ja kansiot määritellystä hakemistosta. Tämän jälkeen tiedostoista suodatetaan kaikki .mp3 –loppuiset tiedostot (musiikkitiedostot). Musiikkitiedostojen hakemistot selvitetään ja tarkastetaan, löytyykö kyseisestä kansiosta tiedostoa nimeltä cover.jpg, joka sisältää kyseisen levyn kansikuvan. Jos kansikuvaa ei löydy, asetetaan ennalta määritetty ”no cover available” –kuva albumille. Käytännössä ohjelmassa luodaan kaksi taulukkoa; toisessa taulukossa on kappaleiden polut, ja toisessa vastaavan kappaleen kansikuvan tieto. Funktion lopussa toistettavat tiedostot lisätään phonon:in soittolistaan.

Play/Pause-nappia painaessa funktiossa tarkastetaan ensin, soiko kappale, jonka jälkeen kappale joko keskeytetään tai aloitetaan toisto. Lisäksi napin kuva vaihtuu, joko play- tai pause-kuvaksi. Stop-funktiossa musiikin toisto lopetetaan ja play/pause-nappiin asetetaan play kuva.

Vaihtaessa kappaletta ohjelma hakee soittolistan seuraavan kappaleen ja tarkastaa, onko osoitin mennyt soittolistan rajojen yli. Tämän jälkeen selvitetään, onko soitin toistotilassa vai ei. Soitin pysäytetään ja asetetaan seuraava tai edellinen kappale soittolistaan. Lopuksi asetetaan soitin toistamaan kappaletta, mikäli soitin oli aiemmin käynnissä.

AboutToFinish-funktio aktivoituu viisi sekuntia ennen kappaleen loppua. Tämä antaa ohjelmalle aikaa asettaa seuraava kappale soittolistaan. Em. funktion ongelmana on, jos kappaletta vaihdetaan esimerkiksi 3 sekuntia ennen loppua, on seuraava soitettava kappale väärä, koska indeksiä on vaihdettu kaksi kertaa. Tick-funktio pyörittää soivan kappaleen aikaa näytöllä.

SourceChanged-funktio esiintyy aina, kun soitettava kappale vaihtuu. Funktion tarkoituksena on pitää soivan kappaleen tiedot näytöllä. Tietoina on esimerkiksi kansikuva, kappaleen nimi, esittäjä ja kappaleen numero. Mikäli kappaleesta ei löydy metadataa, esitetään sen tiedoissa tiedoston nimi.

CoverChanged-funktio toteutuu, kun levyä vaihdetaan. Tämän jälkeen selvitetään, onko soitin toistotilassa vai ei. Soitin pysäytetään ja asetetaan levyn ensimmäinen kappale soittolistaan. Lopuksi asetetaan soitin toistamaan kappaletta, mikäli soitin oli aiemmin käynnissä.

4.2.4 Graafisen käyttöliittymän ulkoasu

Soittimen graafinen käyttöliittymä esitetään alla olevassa kuvassa. Kuvan vasemmassa alakulmassa on soittolistan päivitysnappi, ja sen vieressä kaikki tutut kappaleen vaihtoon ja keskeytykseen tarvittavat painikkeet. Kuvassa ylimpänä ja hallitsevimpana osana onkin Pictureflow-menetelmällä toteutettu levykansien esittäminen. Levyjen alle toteutettiin kappaleen tietokenttä ja SeekSlider. Aivan ylimpänä näkyvää MainWindow-tekstiä ja sen alla olevaa palkkia ei lopullisessa näytössä ole, koska alla oleva kuva on kaapattu OS X –käyttöjärjestelmässä.

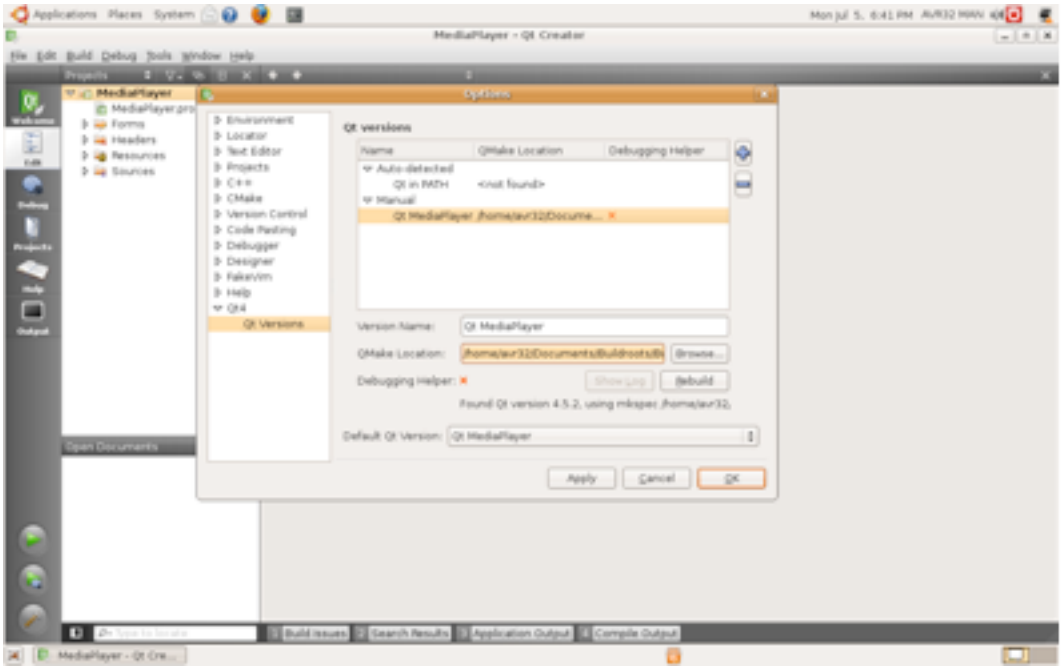
Kuva 28: Soittimen graafinen käyttöliittymä

4.2.5 Kääntäminen AVR32-alustalle

Ohjelman kääntäminen AVR32-alustalle tulee toteuttaa samassa Linux-ympäristössä, jossa aiemmin tarvittu Buildroot on konfiguroitu ja käännetty. On olemassa useampia vaihtoehtoja siihen, miten kääntäminen alustalle voidaan toteuttaa, mutta projektin ohjelma käännettiin QtCreator-ohjelman avulla.

QtCreator:in asetuksiin täytyy lisätä buildrootissa sijaitseva qMake-tiedoston sijainti, joka löytyy yleensä hakemistosta /build_avr32/staging_dir/usr/bin/qmake. Sijainti tulee lisätä cMake- ja QtVersions-kohtiin. Jos näitä asetuksia ei suorita, ohjelma käännetään vain Linux yhteensopivaksi, ei AVR32-alustalle sopivaksi. Alla olevassa kuvassa valikko, minne asetukset tulee määritellä.

Valmiit binäärit kopioitiin alustan SD-muistikortille. Tämän jälkeen ohjelma voidaan suorittaa normaalina Qt-ohjelmana sulautetussa Linux-ympäristössä.



Kuva 29: QtCreator-kääntäjän määrittely

5 YHTEENVETO

Opinnäytetyössä toteutin alkeellisen prototyypin mediasoitinlaitteesta, jonka nimeksi tuli BiteBox. Työ koostui sen suunnittelusta, toteutuksesta ja testauksesta. Työtä aloittaessa minulla ei ollut kokemusta työn osa-alueissa tarvittavista komponenteista ja ohjelmista. Projektin todellinen tarkoitus oli siis oppia paljon uutta ja sen ohessa toteuttaa omaan käyttöön soitin, joka olisi helppokäyttöinen.

En ole esittänyt erikseen kaikkia toteutukseen liittyviä ongelmia ja niiden ratkaisuja, koska niitä ilmeni todella paljon. Kirjoitusosuus kertoo vain työn toteutuneesta osuudesta.

Työn alkuvaiheessa kaikki aika meni tiedonhakuun ja toteutuksen suunnitteluun. Suunnittelua hankaloitti se, että henkilökohtainen Linux-osaamiseni oli heikkoa ja tästä syystä työhön tarvittava tieto oli vaikeasti luettavissa. Elektroniikkatietämykseni oli sen sijaan paremmalla tasolla, ja se näkyi myös projektin aikataulussa. Elektroniikan suunnitteluun ja toteuttamiseen tarvittu aika oli noin kymmenesosa koko projektiin tarvittavasta ajasta.

Työn toteutuksessa suurimmaksi esteeksi muodostui Buildroot ja sen pakettien puuttuminen tai toimintahäiriöt. Ensimmäinen työhön soveltuva versio, joka sisälsi kriittisen Qt:n phonon-kirjaston ja Kernel 2.6.30 –version, ilmestyi syyskuussa 2009. Työn tekemistä vaikeutti myös se, että ohjelmallinen tuki on todella puutteellista. Ainoa tuki oli AVRFreaks-foorumi. Kirjoitushetkellä oleva Buildroot 5/10 –versio on edelleen virheellinen, koska käännösvaiheessa ilmenee Qt:n kohdalla ongelmia. Tämä ongelma on raportoitu ja korjaus tulee seuraavaan versioon. Valitettavaa on etenkin se, että yhden ongelman korjaannuttua, tulee tilalle yleensä toinen.

Projekti valmistui siihen pisteeseen, että kaikki muu paitsi kosketusnäyttö oli toimintakunnossa, Näytössä näkyvää kursoria voidaan liikuttaa, mutta klikkaaminen ei toimi. Elektronisesti kaikki on toiminnassa, koska näytöllä oleva kursori on liikuteltavissa. Ratkaisu tähän olisi todennäköisesti näytön kalibrointi, mutta tätä

ei voida tehdä, koska kalibrointiohjelma on suunniteltu ainoastaan 16bpp ja 32bpp oleville näytöille (työssä käytettävä näyttö oli 24bpp). Yritin muokata kalibrointiohjelmaa toimimaan 24bpp-näytölle foorumeilta saamani avun voimin, mutta tuloksetta.

Ottaen huomioon projektin laajuuden, pääsin toteutuksessa mielestäni pitkälle ja suurimmat ongelmat eivät johtuneet omista virheistäni, vaan ohjelmallisista puutteista. Työ oli erittäin haastava ja monimutkainen, mutta samalla todella opettavainen, ja sen toteuttaminen oli erittäin mielenkiintoista. Tietoisuus siitä, että olen omalla alallani vahvistui tämän projektin myötä.

LÄHDELUETTELO

[1] Atmel Corporation, *NGW/NGW100 Hardware reference*, [viitattu 13.10.2010] <http://www.avrfreaks.net/wiki/index.php/Documentation:NGW/NGW100_Hardware_reference>

[2] Atmel Corporation, *AVR®32 32-bit Microcontroller AT32AP7000*, [viitattu 13.10.2010] <http://www.atmel.com/dyn/resources/prod_documents/32003s.pdf>

[3] Hantouch, *How it works: 4-wire analog-resistive touch screens*, [viitattu 13.9.2010] <http://www.sparkfun.com/datasheets/LCD/HOW%20DOES%20IT%20WORK.pdf>

[4] Texas Instruments, *Touch screen controller*, [viitattu 9.11.2010] <http://focus.ti.com/lit/ds/symlink/ads7846.pdf>

[5] Philips Semiconductors, *Audio codec with touch screen controller*, [viitattu 9.11.2010] <http://www.datasheetcatalog.org/datasheet/philips/UCB1400BE.pdf>

[6] Philips Semiconductors, *Remote 8-bit I/O expander for I2C-bus*, [viitattu 9.11.2010] <http://www.datasheetcatalog.org/datasheet/philips/PCF8574P.pdf>

[7] Seiko Instruments Inc. , *2-wire real-time clock*, [viitattu 9.11.2010] <http://datasheet.sii-ic.com/en/real_time_clock/S35390A_E.pdf>

[8] Nokia Corporation , *Qt for Embedded Linux*, [viitattu 26.1.2011] <http://doc.qt.nokia.com/4.7-snapshot/qt-embedded-linux.html>

[9] Yaghmour, Masters, Ben-Yossef & Gerum. *Building Embedded Linux Systems (2008)*. 2nd Ed. O'Reilly Media inc.

[10] Atmel Corporation, *AVR32 AP7 Linux LCD Panel Customization*, [viitattu 3.2.2011] <http://www.atmel.com/dyn/resources/prod_documents/doc32105.pdf>

LIITELUETTELO