POHJOIS-KARJALAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Antti Mauranen

IP-POHJAISEN ASKELMOOTTORIOHJAUKSEN TOTEUTUS ARM- JA EMBEDDED LINUX-YMPÄRISTÖSSÄ

Opinnäytetyö
Marraskuu 2011

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Marraskuu 2011
Tietotekniikan koulutusohjelma

Karjalankatu 3
80200 JOENSUU
p. (013) 260 6800
p. (013) 260 6906

Tekijä
Antti Mauranen

Nimeke
IP-pohjaisen askelmoottoriohjauksen toteutus ARM- ja Embedded Linux-ympäristössä

Toimeksiantaja
Emmecon Oy, Binomi Oy

Tiivistelmä

Opinnäytetyössä perehdyttiin Texas Instrumentsin AM3517-prossessorin evaluointikorttiin sekä otettiin sulautetun Linuxin Arago-ohjelmointiympäristö kehitys- ja tuotantokäyttöön. Lisäksi kehitettiin www-selaimen avulla toimiva sulautettu askelmoottorin ohjausjärjestelmä.

Kehitystyöasemalle ladattiin ja asennettiin Ubuntu Linux. Ubuntu PC:lle ladattiin ja asennettiin Arago-kehitysympäristö, sen vaatimat GNU:n työkalut ja CodeSourcery-työkaluketju. Evaluointikortin latausohjelmat, GNU/Linuxin ydin ja Ångström-juuritiedostojärjestelmä käännettiin Arago-ympäristössä. Aragon alla juuritiedostojärjestelmään käännettiin ja asennettiin www-palvelin.

Evaluointikortista, Trinamic TMC222-askelmoottoriohjaimesta ja askelmoottorista koottiin ohjausjärjestelmän prototyyppi. Evaluointikortille ohjelmoitiin askelmoottoria I2C-väylän kautta ohjaava ohjelma, joka asennettiin evaluointikortilla toimivan www-palvelimen alaisuuteen. Www-palvelin saa moottorin ohjauskäskyt selaimelta ja välittää ne ohjausohjelmalle. Kaikki sovellusohjelmat kehitettiin Arago-ympäristössä.

Kieli
suomi

Sivuja 40

Asiasanat
Linux, sulautetut järjestelmät, askelmoottoriohjaus

NORTH KARELIA
UNIVERSITY OF APPLIED SCIENCES

THESIS
November 2011
Degree Programme in Information Technology
Karjalankatu 3
FIN 80220 JOENSUU
FINLAND
Tel. 358-13-260 6800

Author
Antti Mauranen

Title
IP-based Stepper Motor Control System in ARM and Embedded Linux Environment

Commissioned by
Emmecon Oy, Binomi Oy

Abstract

The goal of this thesis was to familiarize with Texas Instruments AM3517 processor evaluation card and to take Arago programming environment for Linux into development and production use. An embedded stepper motor control system that is controlled by a browser was also developed.

Ubuntu Linux was loaded and installed to a PC. Arago development environment, the necessary GNU tools and CodeSourcery tool chain were loaded and installed to Ubuntu PC. Loading programs for the evaluation card, GNU/Linux kernel and Ångström root file system were compiled in the Arago environment. Using Arago, a www server was compiled and installed into the root file system.

A prototype for the stepper motor controller was assembled by joining the evaluation card, Trinamic TMC222 stepper controller chip and stepper motor. A program to control the motor via I2C line was developed and installed into the evaluation card to operate under the www server. The www server gets the motor control commands from the browser and it forwards the commands to the control program. All application programs were developed in Arago environment.

Language
Finnish

Pages 40

Keywords

Linux, embedded systems, stepper motor control

Sisältö

Lyhenteet

ARM	Advanced RISC Machines, 32-bittinen mikroprosessoriarkkitehtuuri
BSL	Board Support Library, kehityskortin valmistajan ohjelmoima aliohjelmakirjasto kortin ohjelmoinnin avuksi
CAN	Controller Area Network, sarjaliikennetiedonsiirtoväylä
DHCP	Dynamic Host Configuration Protocol, verkkoprotokolla, jonka avulla tietokoneelle annetaan automaattisesti verkko-osoite
EDR	Bluetooth Enhanced Data Rate, Bluetooth-väylän ominaisuus, jolla tiedonsiirtoa voidaan nopeuttaa
GNU	GNU's Not Unix, avoimen lähdekoodin projekti
HDMI	High Definition Multimedia Interface, standardi kuvan ja äänen siirtoon
JTAG	Joint Test Action Group, standardi ohjelmistojen ja laitteistojen testausportille
MPU	Micro Prosessor Unit, mikroprosessoriyksikkö, nykyisin voivat sisältää useita prosessoreja
NFS	Network File System, verkkotiedostojärjestelmä
SD/MMC	Secure Digital/MultiMediaCard, haihtumattoman muistin muistikortti
SDRAM	Synchronous Dynamic Random Access Memory, haihtuvan muistin muistityyppi
TFTP	Trivial File Transfer Protocol, tiedonsiirtoprotokolla
TI	Texas Instruments, perinteikäs elektroniikkaa valmistava yritys Texasista
UART	Universal Asynchronous Receiver Transmitter, sarjaliikennepiiri

1 Johdanto

Opinnäytetyö oli osa Emmecon Oy:n ja Binomi Oy:n projektia, jossa kehitettiin rakenteiden kunnossapitoa avustavaa mittalaitetta. Opinnäytetyössä otettiin käyttöön Texas Instrumentsin AM35xx-prosessoriperhe ja siihen liittyvät ohjelmointiympäristöt ja -menetelmät. Ohjelmistokehitystyössä koko prosessin sujuvuudella on suuri merkitys, joten tässä työssä esitetään kehitysprosessin välineet ja vaiheet.

Kohdejärjestelmänä oli kokonaisuus, jossa verkon läpi voidaan verkkoon liitetyltä PC:ltä tai mobiililaitteelta ohjata etänä www-selaimen avulla askelmoottorin liikkeitä. Www-selain on yleisesti asennettuna PC-koneisiin ja mobiililaitteisiin. Lisäksi selaimen käyttö askelmoottorinohjauksessa ei vaadi mitään ylimääräisiä muiden ohjelmien asennuksia, pelkkä selain riittää.

Kehityskorttialustana oli ARM-prosessoria käyttävä Logic PD:n Zoom AM3517 EVM Development Kit-evaluointipaketti. Ohjelmistonkehitys tehtiin TI:n Arago-projektin kehitysvälinein Ubuntu Linux-työasemalla.

Opinnäytetyö alkoi vaiheessa, jossa oli valittu prosessoriksi TI:n AM3517 ja evaluointikortiksi Logic PD:n AM3517 EVM-evaluointipaketti. Prosessorin ja kortin valintaperusteita tarkastellaan laitteiston yhteydessä.

Tälle työlle asetettiin kolme päätavoitetta:

1. TI:n AM3517-prosessoriin ja Logic PD:n AM3517 EVM-evaluointikorttiin tutustuminen ja sopivuuden selvittäminen oman kortin suunnittelun pohjaksi
2. kehitystyökalujen valitseminen prosessorille ja evaluointikortille ja niiden käyttöön ottaminen
3. selaimen avulla askelmoottoria ohjaavan sulautetun järjestelmän prototyypin kehittäminen käyttäen valittuja kehitystyökaluja

Opinnäytetyössä tutustuttiin TI:n AM3517-prosessorin ominaisuuksiin ja käyttöön sulautettujen järjestelmien osana. Sulautetussa järjestelmässä laitteisto ja ohjelmisto muodostavat yhtenäisen kokonaisuuden, joka suorittaa haluttua älykkyyttä vaativaa toimintaa [1, s. 10].

Opinnäytetyön aihe on saatu suoraan Emmecon Oy:n ja Binomi Oy:n käytännön tarpeesta kehittää asiakkaalle soveltuva mittalaite. Kohdejärjestelmän yhtenä ehdottomana vaatimuksena oli moottorin ohjattavuus etänä verkon läpi ilman erillisiä ohjelmistoasennuksia PC-koneille tai mobiililaitteelle. Tämä vaatimus täyttyi, kun käytettiin www-selainta käyttöliittymänä ohjauksessa. Www-selaimen käyttö vaatii ohjaimen päähän www-palvelimen ja sellainen on tarjolla ilman erillisiä toimenpiteitä sulautetussa Linuxissa. Haluttiin myös monipuolinen ja joustava tiedostojärjestelmä mittaustulosten tallennukseen. Myös tämä ominaisuus on sulautetussa Linuxissa vakio-ominaisuutena.

Sulautetun Linuxin lisäksi harkittiin jotain sopivaa mikrokontrolleria toteutuksen alustaksi, mutta vaatimus www-palvelimesta ja erityisesti tiedostojärjestelmästä käänsi vaa'an sulautetun Linuxin puolelle. Samassa yhteydessä päätettiin toteuttaa kehitystyö avoimen lähdekoodin välinein saatavuuden, verkosta saatavan laajan tuen, aikaisempien hyvien kokemusten ja hinnan takia.

Ehdottomana vaatimuksena oli myös moottorin nopeuden ja paikan ohjauksen tarkkuus ilman erillisiä nopeutta tai paikkaa osoittavia lisälaitteita. Tämä vaatimus saavutetaan askelmoottorin avulla, kun ohjaus tehdään harkiten ja hallitusti. Moottorin paikan ajoittainen kalibrointi on myös välttämätöntä ja se voidaan tehdä helposti ohjaimeen liitettävän ulkoisen mikrokytkimen avulla, kuten ohjelmointiosuudessa näytetään. Askelmoottorin varsinainen ohjain haluttiin mahdollisimman valmiina ja liitännän siihen oli oltava I2C-pohjainen, koska I2C-väylä oli jo käytössä mm. yritysten anturikorttien liitäntäväylänä, joten sen käyttö hallittiin jo hyvin. Trinamic TMC222 täytti vaatimukset ja se valittiin ohjaimeksi.

ARM -arkkitehtuurin merkitys kasvaa jatkuvasti sulautetuissa järjestelmissä, joten siihen liittyvien menetelmien hallitseminen on hyödyllistä ammatillisen osaamisen ja uralla etenemisen kannalta.

Työskentelyssä on pyritty järjestelmällisyyteen. Pyrittiin etenemään vaihe vaiheelta järjestyksessä:

- kehitysympäristön valinta
- moottoriohjaimen valinta
- kehitysympäristön asentaminen ja käyttöönotto

- koelaitteen suunnittelu ja toteutus
- evaluointikortin ohjelmien kääntäminen ja asennus kortille
- evaluointikortin www-palvelimen valinta ja asennus kortille
- varsinainen ohjelmointityö
    - moottorin ohjausohjelma (I2C-väyläpohjainen ohjaus, tiedon välitys www-palvelimelta ja -palvelimelle cgi-komentorivin kautta)
    - html-sivujen tekeminen

Työmenetelmänä on ollut edetä pienten, yhtä järjestelmän osaa koskevien kokeilujen kautta. Kehitystyössä haluttiin päästä kokeilemaan järjestelmän ominaisuuksia ja ohjelmointia mahdollisimman aikaisessa työn vaiheessa. Eri työvaiheiden aikana työskentely on ollut tyypillistä yrityksen ja erehdyksen kautta etenemistä.

Koska ohjelmankehitysvälineet ovat avoimen lähdekoodin välineitä, on käytetty pääasiallisena tiedon lähteenä verkkohakuja. TI tarjoaa omalta osaltaan hyviä tukifoorumeita [2], joilta löytyy tietoa sekä TI:n prosessoreista että kehitysvälineistä.

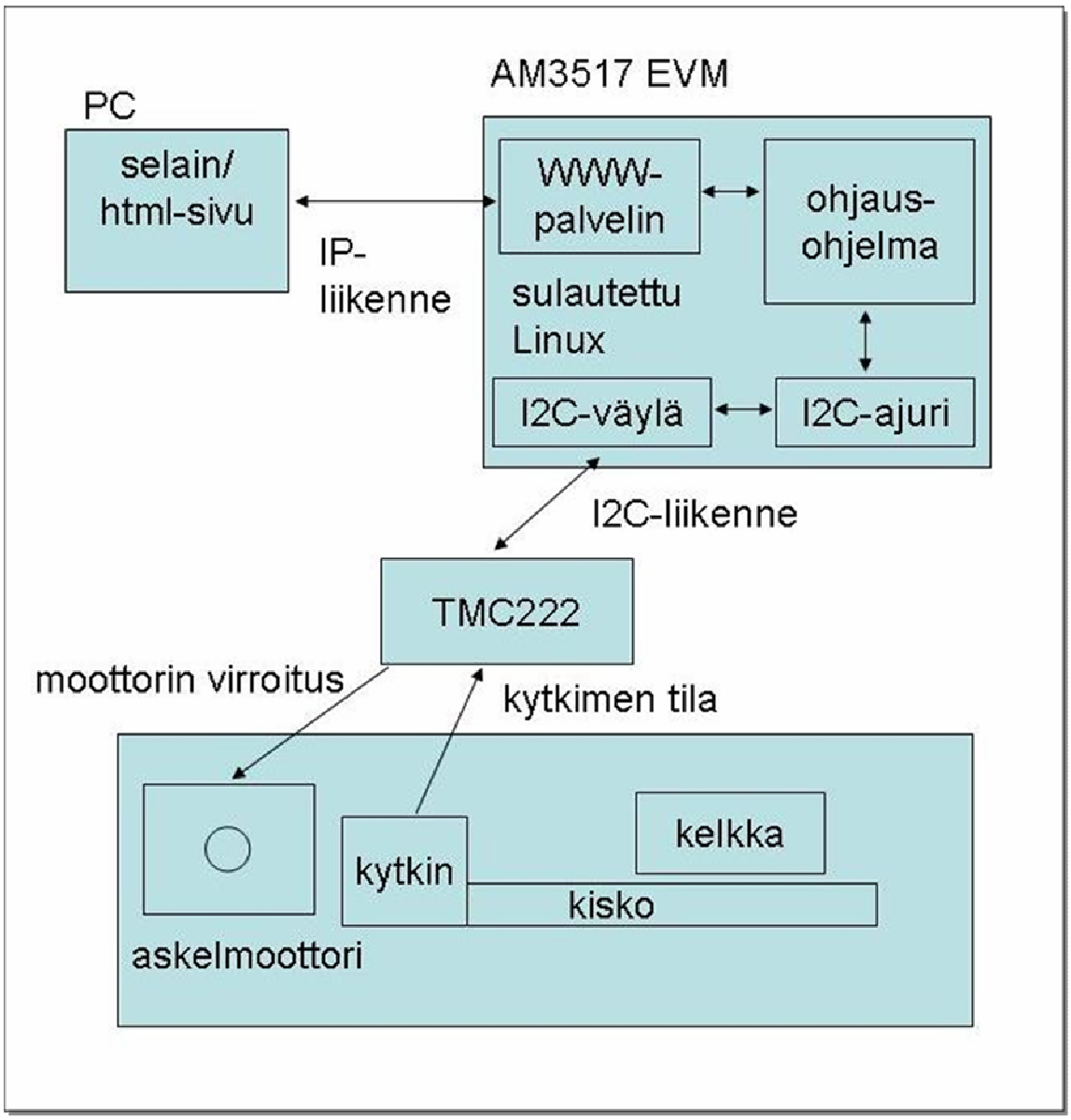
2 Moottorinohjauksen tehtävät ja pääosat

Moottorinohjausjärjestelmä toimii kokonaisuutena seuraavasti:

- käyttäjä ohjaa järjestelmää selaimen avulla
- www-palvelin saa ohjauskäskyt verkon kautta TCP/IP -liikenteen avulla selaimelta ja välittää ne moottoriohjausohjelmalle
- moottoriohjausohjelma lähettää saamansa käskyt I2C-väylän kautta askelmoottoriohjaimelle
- askelmoottoriohjain tulkitsee käskyt ja ohjaa moottoria käskyjen mukaisesti
- moottoriohjausohjelma kyselee askelmoottorin paikka- ja tilatiedot ohjaimelta ja välittää ne takaisin www-palvelimelle
- www-palvelin koostaa paikkatiedoista html-sivun ja lähettää sen selaimelle käyttäjän nähtäväksi.

Moottoriohjausjärjestelmä (kuva 1) muodostaa sulautetun järjestelmän ja siinä on seuraavat osat:

- PC, jossa ohjauksen selainkäyttöliittymä
- kehityskortti, jossa sulautettu Linux
- askelmoottorin ohjain Trinamic TMC222
- 2-vaiheinen bipolaarinen askelmoottori, 400 askelta/kierros, jotta saavutetaan tarvittava askeltarkkuus
- testiasetelma, jossa kehityskortti, ohjain ja askelmoottori ovat testattavissa.



Kuva 1. Askelmoottorin ohjaus

3 Ohjelmistokehitysvälineiden valinta

Opinnäytetyön toteutukseen harkittiin kehitysvälineeksi kolmea eri vaihtoehtoa, OpenEmbedded-ympäristöä, TI:n Arago-ympäristöä ja TI:n Linux SDK-työkalusettiä. OpenEmbedded on avoimeen lähdekoodiin perustuva yleisesti sulautettujen järjestelmien kehittämiseen tarkoitettu ympäristö [3], joka pohjautuu Ångtröm Linux-tiedostojakeluun [4]. Arago on TI:n sisäisesti kehittämä haara OpenEmbedded:stä [5]. Arago keskittyy TI:n prosessoreihin, mutta hyödyntää OpenEmbedded:ssä olevia ei-prosessorikohtaisia ominaisuuksia. Aika ajoin TI liittää kehityshaarat yhteen ja vie Aragon sen hetkiset tulokset yleiseen OpenEmbedded-ympäristöön. TI:n Linux SDK [6] on paketti, johon TI on kerännyt joukon Linuxin työkaluja, GNU/Linuxin ytimen ja tiedostojärjestelmäreferenssin. SDK on kiinteä paketti ja sitä päivitetään harvakseltaan.

GNU-projekti on keskeinen toimija avoimen lähdekoodin kehityksessä ja jakelussa [7]. Sen perusti vuonna 1983 Richard Stallman. GNU-projekti kehittää ja hallinnoi mm. kääntäjäkokoelmaa GNU Compiler Collection GCC, GNU C-kirjastoa ja on osapuolena kehittämässä GNU/Linuxin ydintä.

Pohdintojen jälkeen kehitysympäristöksi valittiin Arago useasta syystä:

- SDK ei ole täydellinen kehitysympäristö, koska kaikki sulautetun Linux-järjestelmän osat eivät ole mukana. Lisäksi ytimen ja tiedostojärjestelmän versiot päivittyvät hitaasti.
- OpenEmbedded-ympäristö päivittyy TI:n prosessorien osalta hitaammin kuin Arago, mutta muilta osin erittäin nopeasti, niin että siinä ilmenee aika ajoin epästabiiliutta, joka on häiritsevää kaupallista kehitystä tehtäessä.
- Arago on yhteensopiva OpenEmbedded-ympäristön kanssa, josta opinnäytetyön tekijällä on kokemusta.
- Arago on suoraan TI:n kehittämä ja tukema ja se on keskittynyt TI:n prosessoreihin, useimmat uudet piirteet tulevat ensin Aragolle.
- Arago (kuten OpenEmbedded) tarjoaa yhdessä paketissa välineet sulautetun Linux-järjestelmän kehitykseen. Kaikki tarvittavat osat ovat mukana eli x-loader, U-Boot, GNU/Linuxin ydin ja Ångström-tiedostojärjestelmä. Erillisiä osasten lataamisia ei tarvita, vaan Arago suorittaa latauket.

- Arago (kuten OpenEmbedded) on erittäin joustava eli kaikki muutokset ketjun osiin on mahdollista tehdä Aragon sisällä, samoin omat sovellukset.

Ohjelmakoodin editointiin päätettiin käyttää gedit-tekstieditoria, koska se oli opinnäytetyön tekijälle tuttu ja siitä oli hyvät kokemukset.

Html-sivujen koostamiseen harkittiin jotain html-sivujen tekoon erikoistunutta editoria, mutta päädyttiin geditiin, koska tarvittavat sivut olivat niin yksikertaisia ja gedit oli käytössä ohjelmakoodin teossa.

Ohjelmoinnissa käytetyt välineet olivat:

- html-sivut
    - gedit-tekstieditori
- Linux-ohjelmointi (kehityskortin ohjelmien lataajat, GNU/Linuxin ydin, Linuxin tiedostojärjestelmä, sovellusohjelmat)
    - Ubuntu PC
    - Arago-ohjelmointiympäristö
    - apuohjelmat (gedit-tekstieditori, kääntäjät, kirjastot)
    - GNU:n apuohjelmat.

Arago-ympäristö toimii Linux-työasemassa, joten työaseman käyttöjärjestelmäksi valittiin Ubuntu, joka on opinnäytetyön tekijälle tuttu aikaisemmista yhteyksistä.

4 Laitteisto

Laitteisto koostuu Logic PD Zoom AM3517 EVM Development Kit-evaluointipaketista ja Ubuntu Linux -työasemasta. Askelmoottoriohjaus toteutetaan I2C–väyläisellä Trinamic TMC222-askelmoottoriohjaimella. I2C on yksinkertainen, hyvin standardisoitu kaksisuuntainen sarjamuotoinen tiedonsiirtoväylä [8, s. 234]. Siinä on oma linjansa kellolle ja datalle.

4.1 Logic PD Zoom AM3517 EVM Development Kit-evaluointipaketti

ARM-prosessoriperhe oli valittu jo ennen opinnäytetyön alkamista projektiin, koska se on paljon käytetty. ARM:n valinnassa prosessoriksi yhtenä tekijänä oli kehitystyön tu-

loksien jos ei erinomainen niin kumminkin kohtuullinen siirtomahdollisuus eri valmistajien ARM-versioiden välillä erityisesti käytettäessä kortin käyttöjärjestelmänä Linuxia. Linux on Unixin kaltainen GNU/Linuxin ytimeen pohjautuva käyttöjärjestelmäperhe. Linuxia käytetään matkapuhelimista supertietokoneisiin. Linux on avoimen lähdekoodin ohjelmisto [8, s. 5]. Toimiva Linux-kokonaisuus koostuu GNU/Linuxin ytimestä ja siihen liitetystä tiedostojärjestelmästä, joita voivat olla esimerkiksi Debian, Ubuntu (Debian-pohjainen), Fedora ja openSUSE tai meidän tässä työssä käyttämämme Ångström. Linux soveltuu erinomaisesti sulautettujen järjestelmien ohjelmointiin.

AM3517-prossesori ja Logic PD:n TI:n AM3517 EVM-evaluointipaketti oli valikoitunut kehitysalustaksi, koska Emmecon Oy:n yhteistyökumppanit käyttävät sitä omissa projekteissaan. Lisäksi evaluointipaketilla on hyvä saatavuus ja sille on kattava (TI, Logic PD) ohjelmointituki. Evaluointipaketti on myös modulaarinen eli siinä on ydintoiminnot sisältävä myös erikseen myynnissä oleva prosessorikortti ja tukitoiminnot sisältävä emolevy. Tuotantoprojekteissa prosessorikortti on mukana sellaisenaan ja emolevy suunnitellaan tilanteen vaatimusten mukaiseksi itse ja valmistutetaan kolmannella osapuolella.

Evaluointipaketin suositushinta lokakuussa 2011 on Logic PD:n sivuilla USD 999, kortista on myös kevytversio, jonka hinta on USD 299. Tuotetukea kortille saa sekä Logic PD:lta että TI:lta, jolla on laaja TI E2E-tukifoorumi [2].

Evaluointipaketti koostuu kolmesta erillisestä osasta, AM3517 SOM-M2 -prosessorikortista, EVM Baseboard -pohjakortista ja EVM Application Board -sovelluskortista. Pohjakortilla on paikka prosessorikortille ja liittimet sovelluskortin yhdistämiseksi. Paketissa mukana on myös LCD-näyttö, virtalähde, langattoman verkon antenni ja tarvittavat kaapelit.

Evaluointipaketin lisäksi apukorttina oli CreateNewStuff.com:n AM3517-piikkirimakortti, joka kiinnittyy pohjakortin liittimiin ja josta saadaan ulos lähes kaikki prosessorin pinnit.

4.1.1 AM3517 SOM-M2 -prosessorikortti

AM3517 SOM-M2 -prosessorikortilla on TI:n Sitara AM3517 -mikroprosessori ja sen tukipiirit. SOM-moduuli on erittäin järkevä, kompakti ja kätevä kokonaisuus ja valmis

käytettäväksi asiakkaan sovelluksissa joko EVM Baseboard -pohjakortin kanssa tai asiakaan sovellusta varten erityisesti suunnitellun pohjakortin kanssa. Prosessorikortilla on tarvittavat muistit sekä verkko- ja muut liitännät [9]:

- AM3517 600 MHz:n mikroprosessori
- 256 MB DDR2 SDRAM muisti
- 512 MB NAND flash muistia
- 10/100 Base-T Ethernet kontrolleri
- 802.11b/g/n langaton Ethernet kontrolleri
- langattoman verkon antenniliitäntä
- Bluetooth 2.1 + EDR kontrolleri
- USB 2.0 kontrolleri
- CAN-väylä
- kolme I2C-porttia
- kolme SPI-porttia
- neljä UART-sarjaliikenneliitäntää
- näytön grafiikkakiihdytin SGX530
- tuki OpenGL- ja OpenVG-grafiikkakirjastoille
- tuki SD/MMC-korteille

4.1.2 EVM Baseboard -emolevy

EVM Baseboard -pohjakortilla ovat toiminnan kannalta välttämättömät virroitukseen, ulkoisiin kytkentöihin ja näyttöön liittyvät piirit ja toiminnot [10]. Kortista on saatavissa tarpeelliset suunnittelutiedot, jotta sitä voidaan käyttää mallina vastaaville omille pohjakorteille. Kortin tärkeimmät ominaisuudet ja toiminnot ovat:

- liitännät taustavalaistulle kosketusnäytölle
- 4,3 tuuman LCD-kosketusnäyttö
- HDMI video-liitäntä
- RJ45-verkkoliitäntä
- USB 2.0 liitännät (mini AB, A)
- RS-232 portti, maksiminopeus 115,2 kbps
- USB/UART-portti

- MMC/SD-korttipaikka
- JTAG-liitäntä laitteiston testaukseen
- S-Videon liitäntä
- kytkimet kortin käynnistystavan valinnalle

4.1.3 EVM Application Board -sovelluskortti

EVM Application Board -sovelluskortilla on audioon, videoon ja verkkoyhteyksiin liittyviä toimintoja sekä yksinkertainen näppäimistö [10]. Lisäksi kortilla on piikkirimaliittimiä, joista saa ulos osan prosessorin pinnien signaaleista. Myös tästä kortista on saatavissa tarpeelliset suunnittelutiedot, jotta sitä voidaan käyttää mallina vastaaville omille korteille. Kortin tärkeimmät ominaisuudet ja toiminnot ovat:

- audio
    - kaksi kuulokeliitäntää
    - kaksi mikrofoniliitäntää
    - audion linjatasoiset tulo- ja lähtöliitännät
- video
    - S-Videon tuloliitäntä
    - komponenttivideon tuloliitäntä
    - komposiitti RGB tuloliitäntä
    - rinnakkainen kameran tuloliitäntä
- pieni näppäimistö
- 5-suuntainen navigointinappi
- RJ45-verkkoliitäntä
- langattoman verkon lisäkorttiliitäntä
- USB 2.0 liitäntä

4.1.4 Evaluointipaketin sovellustuki

TI ja Logic PD tarjoavat kattavan paketin sovelluskehitykseen. Ohjelmia voidaan tehdä sekä eri käyttöjärjestelmien alle että suoraan prosessorin suoritettavaksi ilman käyttöjärjestelmää.

Tarjolla olevia kehitysvälineitä ovat mm. [10]:

- Android
- Embedded Linux ja GNU/Linux
    - OpenEmbedded/Ångström
    - Arago (TI:n OpenEmbedded-haara)
    - x-loader, kortin toisen tason käynnistysohjelma
    - U-Boot, kortin kolmannen tason käynnistysohjelma/bootloader
- Windows Embedded CE
- Timesys Linux BSP
- QNX Neutrino RTOS
- TI:n kääntäjä ja kortin tukikirjasto BSL suoraan rautakehitykseen

Tässä työssä keskitytään Embedded Linux - ja GNU/Linux -sovelluskehitykseen, erityisesti Arago-haaraan ja kehitystyöhön Ubuntu Linuxin alla.

4.1.5 AM3517-piikkirimakortti

Pieni, mutta tärkeä apuväline evaluointipaketin tukena oli CreateNewStuff.com:n AM3517-piikkirimakortti, jonka avulla lähes kaikki prosessorin pinnit saadaan näkyville piikkirimaliittimeen pohjakortin reunaliittimistä. Tämän tapainen kortti on lähes välttämätön testattaessa evaluointipaketin toimintaa. Kortti on aika kallis verrattuna koko paketin hintaan eli se maksaa USD 99, mutta se helpottaa suuresti prosessorin eri toimintojen ja pinnien testaamista.

4.2 TI AM3517 -prosessori

AM3517 SOM-M2 -prosessorikortilla on TI:n AM3517 ARM Cortex-A8 -mikroprosessori.ARM on 32-bittinen RISC-mikroprosessoriarkkitehtuuri. Se on suosittu mm. sulautetuissa järjestelmissä [1, s. 56]. ARM-prosessoreilla on laaja käyttöjärjestelmätuki.

AM3517-prosessori soveltuu mm. yhden levyn tietokoneisiin, teollisuusautomaatioon ja -kontrolliin sekä teollisuusautomaation käyttöliittymiin [11, s. 138; 12 s. 3]. Prosessorissa on sisäänrakennettuna 3D-grafiikkakiihdytin, tuki DDR2-muisteille, CAN-väylälle USB:lle ja verkolle. Prosessorissa on myös yhdistetty vektorien liukulukulaskentayk-

sikkö, joten se soveltuu nopeaa liukulukulaskentaa vaativiin sovelluskohteisiin. Budjetaarinen hinta 10/2011 on USD 15,85.

Prosessorin tärkeimpiä ominaisuuksia ovat [12, s. 1]:

- MPU
    - 600-Mhz Sitara ARM Cortex-A8 -ydin
    - 1200 ARM MIPS
    - NEON SIMD grafiikka- ja audioapuprosessori
    - liukulukuapuprosessori vektoreille
- muisti
    - 166 MHz:n 16/32 bitin muistiliitäntä DDR/DDR2-muisteille
    - yleiskäyttöinen 83 MHz:n 16 bitin muistiliitäntä
    - 64 KB SRAM
    - 3 MMC/SD/SDIO-korttiliitäntää
- sarjaliikenne
    - CAN-kontrolleri
    - 10/100 Mbit Ethernet MAC
    - 1-Wire liitäntä
    - 4 UART sarjaliikenneliitäntää
    - 3 I2C kontrolleria
    - 4 SPI porttia (McSPI)
    - 186 yleiskäyttöistä IO-pinniä (GPIO)
- ajastimet
    - 12 32-bittistä yleiskäyttöistä ajastinta
    - 1 32-bittinenWatchdog-ajastin
- HD-tasoinen näyttöalijärjestelmä

4.3 Trinamic TMC222-askelmoottorikontrolleri

Trinamic TMC222-piiri on I2C-väyläinen yhdistetty askelmoottorin liikkeen ohjain ja tehonsyöttäjä. Siinä on sisäinen sekä pysyvä että haihtuva konfigurointi- ja parametrimuisti. Kun kontrolleri on alustettu, se suorittaa itsenäisesti moottorin ohjauksen perustuen annettuihin askelmääriin sekä kiihtyvyyksiin ja nopeuksiin.

Ohjaimen tärkeimmät ominaisuudet ovat [13]:

- tehonsyöttö
    - yhdelle 2-vaiheiselle bipolaarimoottorille
    - 1/2-, 1/4-, 1/8- ja 1/16-askellus
    - maksimi käämivirta 800 mA, PWM-ohjaus
    - syöttöjännite 8 – 29 V
    - maksimi täysaskellustaajuus 1 kHz
    - suojattu ylikuumenemiselta ja oikosululta
- liikkeen ohjaus
    - sisäinen 16-bittinen paikkalaskuri
    - säädettävä nopeus ja kiihtyvyys
    - sisäänrakennettu alkukiihtyvyyden-tasaisen liikkeen-loppuhidastuksen säätö tarkan paikannuksen avuksi
    - ulkoisen kytkimen liitäntä referenssipaikan lukemiseksi
- I2C
    - maksimi nopeus 350 kbps
    - 32 ohjelmoitavaa osoitetta
    - konfigurointi- ja ohjauskäskyt
    - ohjaimen tilakyselyt

Kokeiluja varten tehtiin piirilevy, jossa oli moottorinohjain ja tarpeelliset ulkoiset liitännät.

Moottorin ohjaaminen käydään läpi myöhemmin C-kielisten ohjelmaesimerkkien avulla.

4.4 Askelmoottori

Moottoriksi valittiin Nanotec ST4209S1006 -askelmoottori, josta työn tekijällä oli kokemusta aikaisemmista projekteista. Lisäksi näitä moottoreita oli Binomi Oy:llä varastossa. ST4209S1006-askelmoottori on mahdollista kytkeä unipolaari- tai bipolaarimoodiin [14].

Moottorin tärkeimmät ominaisuudet ovat:

- 400 askelta/kierros
- askellustarkkuus/askel +-5%
- kuulalaakerointi
- paino 0,22 kg

4.5 Koelaite

Testejä varten koottiin pieni koelaite, jossa liukukiskoa pitkin hihnavedolla askelmoottori liikutti kelkkaa. Toisessa päässä kiskoa oli mikrokytkin, joka oli kytketty TMC222:n ulkoisen kytkimen liitäntään. Tämän avulla saatiin kelkan paikka nollattua.

Virroitus TMC222:lle otettiin laboratoriovirtalähteestä.

4.6 Työasema

Kehitystyöasemana oli hp EliteBook 6930p kannettava, jossa on intel CORE Duo -prosessori. Koneen käyttöjärjestelmänä oli sisäiselle kiintolevylle asennettu Ubuntu 10.04 LTS ja Linuxin ytimen versio oli 2.6.32-34-generic.

Kehitystyö tehtiin langallisessa sisäverkossa. Työasemalle annettiin kiinteä IP-osoite ja kehityskortti haki dynaamisesti IP-osoitteensa sisäverkon DHCP-palvelimelta.

5 Kehitysympäristön asennus ja käyttöönotto

Kehitysympäristön asennus ja konfigurointi esitetään siinä järjestyksessä, jossa se todellisuudessa tehtiin.

5.1 Ubuntu

Ubuntu ladattiin ja asennettiin Ubuntun organisaation lataussivulta ja asennettiin sivun ohjeiden mukaisesti [15].

Ubuntu 10.04 LTS kehitystyöaseman käyttöjärjestelmänä on yksi TI:n suosittelemista käyttöjärjestelmistä [16], joten sen käyttäminen ei vaatinut mitään ylimääräisiä toimenpiteitä tässä yhteydessä.

5.2 Gedit

Ohjelmointieditorina käytettiin Ubuntun mukana oletuksena asentuvaa gedit-tekstieditoria. Sitä muokattiin oletusasetuksista Micah Carrick:n blogin [17] ohjeiden mukaisesti paremmin ohjelmointiin sopivaksi. Editorin asetuksiin tehtiin seuraavat muutokset:

- poistettiin tekstin automaattinen rivitys
- näytettiin tiedoston rivinumerot
- korostettiin rivi, jolla kursori on
- korostettiin alkusulku/loppusulku
- näytettiin oikea marginaali sarakkeessa 80
- muutettiin sarkainmerkit tyhjämerkeiksi
- sallittiin automaattinen sisennys
- muutettiin kirjasintyyppi Monospace 10:ksi, jolloin kaikki merkit pysyivät saman levyisinä
- asennettiin lisäosa ”ohjelmasymbolien selaus”
- asennettiin lisäosa ”ulkoiset työkalut”, joka mahdollistaa erilaisten ulkoisten työkalujen liittämisen gedit:iin

Näillä pienentuntuisilla muutoksilla editorin käytettävyys ohjelmoinnissa parani merkittävästi.

5.3 Arago

Arago vaatii useiden ulkopuolisten ohjelmien asennusta ja seuraavaksi käydään läpi Aragon ja apuohjelmien asentaminen [16]. Asennuksia tehdessä ei kannata yleensä suuresti pohtia asennuksen yksityiskohtia vaan tehdä orjallisesti ohjeiden mukaan ja murehtia vasta, jos jokin asia ei toimi.

5.3.1 Apuohjelmat

Vaadittuja apuohjelmia ovat mm.

- git, versionhallintaohjelma verkkolatauksia varten
- gawk, tekstin käsittelyyn erikoistunut ohjelmointikieli
- diffstat, Linux-apuohjelma yksinkertaisten histogrammien tekemiseen
- bison, GNU:n kääntäjägeneraattori
- lukuisia muita

Apuohjelmat saa asennettua kätevästi komennoilla:

```
>sudo apt-get install build-essential subversion ccache sed wget cvs git-core coreutils
>sudo apt-get install  unzip texinfo docbook-utils gawk help2man diffstat file g++
>sudo apt-get install texi2html bison flex htmldoc chrpath libxext-dev xserver-xorg-dev
>sudo apt-get install doxygen corkscrew
```

Komennot hakevat ohjelmat verkosta ja asentavat ne työasemaan. Arago käyttää apuohjelmia taustalla eikä käyttäjän onneksi yleensä tarvitse tietää niistä sen enempää.

5.3.2 CodeSourcery-työkaluketju

Arago ei itse rakenna työkaluketjuaan (kuten OpenEmbedded tekee), vaan käyttää Mentor Graphics:n Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux-työkaluketjua [18], jossa ovat mm. GNU:n C- ja C++-kääntäjä, assembleri, linkkeri ja debuggeri.

Asennus tehtiin työkaluketjun ohjeiden mukaisesti hakemistoon /opt/arm-2009q1. Asennus on suoraviivainen ja se pitää tehdä sudo-komennolla, jotta oikeudet riittävät. Ennen asennusta pitää vaihtaa Linuxin oletus-shell bash:ksi komennolla

```
> sudo dpkg-reconfigure -plow dash
Install as /bin/sh? No
```

Kannattaa huomata, että muut versiot eivät välttämättä toimi Aragon kanssa, esimerkiksi uusin versio 2011 ei kokeillessa toiminut, vaan antoi runsaslukuisen joukon virheilmoituksia käännettäessä Aragon ohjelmia.

5.3.3 Aragon lähdekoodin hakeminen ja käännösympäristön teko

Käyttäjän kotihakemiston alle tehtiin Aragoa varten hakemisto ja siellä annettiin lähdekoodin hakemiseksi komennot

```
>git clone git://arago-project.org/git/arago.git
>git clone git://arago-project.org/git/arago-oe-dev.git
>git clone git://arago-project.org/git/arago-bitbake.git
```

Muodostuu kolme hakemistoa, arago, arago-oe-dev ja arago-bitbake, joissa ovat nyt tarvittavat työkalut ja reseptit bitbake ohjelmaa varten. bitbake on make-ohjelman vastine, joka on käytössä Aragossa. Resepti on make-ohjelman Makefile:a vastaava toimintaohje bitbake-ohjelmalle [16].

Kannattaa tehdä myös komentotiedosto (nimi vaikkapa komentotiedosto.txt), joka ajetaan aina kun aloitetaan työskentely Aragon kanssa. Siinä asetetaan Aragon kotihakemisto, työhakemistoja ja polku CodeSourcery-työkaluketjuun. Alla on esimerkki tiedostosta. Tässäkin pätee neuvo, että ei kannata liikaa miettiä mitä kukin komento tarkoittaa vaan tehdä vain ohjeen mukaan. Polkujen sisältö luonnollisesti riippuu siitä, mihin hakemistoihin eri työkalut on asennettu. Alla olevat ovat yllä olevien ohjeiden mukaisia.

```
# Aragon juurihakemisto
export OEBASE=$HOME/embeddeddevelopment/aragoroot
# työhakemistoja ja polkuja työkaluihin
export BBPATH=$OEBASE/arago:$OEBASE/arago-oe-dev
export PATH=$OEBASE/arago-bitbake/bin/:$OEBASE/arago-utils/:$PATH
# asetetaan Aragon ympäristömuuttujia
export BB_ENV_EXTRAWHITE="OEBASE MACHINE META_SDK_PATH http_proxy
https_proxy ftp_proxy no_proxy GIT_PROXY_COMMAND"
# työkaluketju polkuun
export PATH=/opt/arm-2009q1/bin:$PATH

# siirrytään lopuksi Aragon juureen
cd $OEBASE
```

Kopioidaan Aragon konfigurointitiedoston malli konfigurointitiedostoksi hakemistossa arago/conf eli annetaan komento

```
>cp local.conf.sample local.conf
```

Lopuksi asetetaan voimaan käännösympäristö komennolla.

```
>source komentotiedosto.txt
```

Tämän jälkeen Arago on käytettävissä.

5.3.4 Aragon ottaminen käyttöön

Ensimmäisenä kannattaa tehdä vaikkapa konsoliversio Arago Linuxista. Komento AM3517-evaluointikortille on

```
>MACHINE=am3517-evm bitbake arago-console-image
```

Komento lataa verkosta tarvittavat lähdetiedostot, kääntää kortille sopivat toisen ja kolmannen tason latausohjelmat x-loader ja U-Boot. Samoin komento kääntää kortille GNU/Linuxin ytimen ja Ångströmin tiedostojärjestelmän. Komento paketoi tulostiedostot kortin vaatimaan muotoon. Tiedostojen käyttöön ottaminen käydään läpi myöhemmin esimerkkien avulla.

Komento kannattaa antaa yötä vasten sillä verkosta ladattavia ja käännettäviä tiedostoja on tuhansia ja niiden yhteinen koko on gigatavuja. Kannattaa myös välillä tarkistaa (erityisesti juuri ennen nukkumaan menoa), ettei käännöstyö ole katkennut johonkin virheeseen (esimerkiksi verkon jumiutumiseen), jos on, auttaa yleensä komennon uudelleen antaminen. Arago jatkaa automaattisesti kohdasta, jossa virhe tapahtui.

5.4 Verkkopalvelimet

Kehitystyötä tukemaan kannattaa kehityskoneeseen asentaa kaksi verkkopalvelinta TFTP-palvelin ja NSF-palvelin. Niitä käytettäessä ei uusia versioita ytimestä eikä tiedostojärjestelmästä ja yksittäisistä ohjelmista tarvitse tallentaa kortin NAND flash-muistille, vaan ne voidaan ladata kortille kehityskoneen levyltä verkon kautta. Tämä joustavoittaa kehitystyötä merkittävästi.

5.4.1 TFTP-palvelin

Tämän palvelimen avulla kortin on mahdollista hakea GNU/Linuxin ydinverkkolevyltä. Palvelin asennetaan kehityskoneeseen komennolla [19]:

```
>sudo apt-get install xinetd tftpd tftp
```

Luodaan tiedosto /etc/xinetd.d/tftp ja talletetaan sinne sisältö

```
service tftp
{
protocol      = udp
port          = 69
socket_type = dgram
wait          = yes
user          = nobody
server        = /usr/sbin/in.tftpd
server_args  = /tftpboot
disable       = no
}
```

Luodaan hakemisto /tftpboot ja annetaan käyttöoikeudet kaikille käyttäjille. Lopuksi palvelin käynnistetään komennolla

```
>sudo /etc/init.d/xinetd start
```

Tämän jälkeen palvelin käynnistyy automaattisesti aina kun kehityskone käynnistetään.

5.4.2 NFS-palvelin

Tämän palvelimen avulla kehityskortin on mahdollista käyttää GNU/Linuxin tiedosto-järjestelmää kehityskoneen levyltä. Palvelin asennetaan kehityskoneeseen komennolla [20]:

```
>sudo apt-get install nfs-kernel-server
```

Lisätään tiedostoon /etc/exports rivi

```
/nfsboot *(rw,sync,no_root_squash,no_subtree_check)
```

Luodaan hakemisto /nfsboot ja annetaan oikeudet kaikille käyttäjille. Käynnistetään palvelin komennolla

```
>sudo /etc/init.d/nfs-kernel-server start
```

Tämän jälkeen palvelin käynnistyy automaattisesti.

6 Ajoympäristö

6.1 Evaluointikortin käynnistyminen

Kortti käynnistetään viidessä vaiheessa [11, s. 2680]. Ensin tapahtuu prosessorin alustus ja kellojen käynnistys. Sitten käynnistyy latausohjelma prosessorin sisäisestä ROM:sta. Tämä ohjelma käynnistää toisen tason latausohjelman x-loader käynnistystavan valinnan mukaisesti joko SD/MMC-kortilta tai NAND flash:stä. x-loader käynnistää sitten valinnan mukaisesti joko SD/MMC-kortilta tai NAND flash:stä kolmannen tason latausohjelman U-Boot.

Das U-Boot on yleisesti käytetty latausohjelma [21] ja sen toimintaa ohjataan ympäristömuuttujien avulla. U-Boot:lla on komentorivikäyttöliittymä, jonka avulla sille voidaan antaa komentoja.

U-Boot lataa GNU/Linuxin ytimen valinnaisesti MMC/SD-kortilta, NAND flash:stä tai verkkolevyltä TFTP-palvelimesta. Testeissämme käytimme lataamista verkkolevyltä tavan joustavuuden takia. Latauksen jälkeen ydin käynnistetään ja tiedostojärjestelmä liitetään mukaan taas valinnan mukaan joko MMC/SD-kortilta, NAND flash:stä tai verkkolevyltä NFS-palvelimesta. Testeissä tiedostojärjestelmä oli NFS-palvelimelle.

Ytimen ja tiedostojärjestelmän lataaminen verkkolevyltä suoritetaan seuraavien U-Boot:n ympäristömuuttujien ja käskyjen avulla [22]:

```
#verkkopalvelimen (kehityskoneen) IP-osoite
serverip 192.168.2.110
#U-Boot ajaa automaattisesti bootcmd-ympäristömuuttujaan tallennetun komennon,
#nyt ajetaan nfsboot
bootcmd run nfsboot
#ydin ladataan TFTP-palvelimelta tiedostosta uImage
bootfile ${serverip}:uImage
#asetetaan ytimen käynnistysparametrit, muistin koko, konsoli, oma IP-osoite,
#NFS-palvelimen IP-osoite, tiedostojärjestelmän polku palvelimella
bootargs mem=256M console=${console} noinitrd ip=dhcp rw root=/dev/nfs
nfsroot=${serverip}:/nfsboot/rootfs,nolock
#käytetään DHCP-palvelinta, ladataan ja käynnistetään ydin
nfsboot "dhcp; bootm”
```

6.2 GNU/Linuxin ydin

Käytämme testeissämme Aragon GNU/Linuxin ydintä AM3517-kehityskortin oletuskonfiguraatiossa.

I2C-ajurit ja I2C-väylät (i2c-1, -2 ja -3) ovat mukana oletuksena, joten niiden konfiguroimiseksi ei tarvitse tehdä erillisiä toimenpiteitä. I2C-laitteet näkyvät /dev-hakemiston laiteajureina, joten ne ovat käytettävissä mm. käyttäjätilan komentoriviohjelmissa.

6.3 Tiedostojärjestelmä

Testeissä tiedostojärjestelmänpohjana on reseptin arago-console-image mukainen tiedostojärjestelmä. Siinä ovat mukana tarvittavat osat konsolikäyttöön. Tarvittavat palvelimet ja ohjelmat asennetaan tähän tiedostojärjestelmään.

6.4 Www-palvelin thttpd

Kehityskortin www-palvelimeksi valittiin thttpd-palvelin [23], koska se on kevyt, luotettava ja helppo konfiguroida. Lisäksi se on mukana Aragossa, vaikkakaan sitä ei asenneta oletuksena testeissä käytettävään arago-console-image:en, joten käännämme ja asennamme sen erikseen. Kääntäminen tapahtuu käskyllä:

```
>MACHINE=am3517-evm bitbake thttpd
```

Arago tallentaa palvelimen asennuspaketin tiedostopuussaan olevan deploy-hakemiston alihakemistoon tiedostonimellä thttpd_2.25b-r8-arago1.6_armv7a.ipk. Kopioimme sen verkkolevylle NFS-palvelimelle evaluointikortin käyttäjän root kotihakemistoon /home/root. Asennus kehityskortille tapahtuu kehityskortin komentoriviltä root-käyttäjän hakemistossa komennolla:

```
>opkg install thttpd_2.25b-r8-arago1.6_armv7a.ipk
```

Palvelinta hallitaan komennolla

```
/etc/init.d/thttpd {start | stop | restart}
```

Aragossa oleva thttpd:n versio käynnistetään thttpd-scriptissä komennolla

```
>start-stop-daemon --start --quiet --exec $thttpd -- -d /srv/www –u root -c "/cgi-bin/*"
```

Juurihakemisto palvelimelle on /srv/www (html-sivut tänne) ja cgi-komentoriviohjelmat haetaan juurihakemiston alihakemistosta /cgi-bin (cgi-komentoriviltä ajettavat scriptit ja ohjelmat tänne). Nämä hakemistot asennusohjelma myös automaattisesti tekee levylle. Palvelin käynnistyy automaattisesti aina kehityskortin käynnistyessä.

Koska palvelin käynnistetään root-käyttäjän oikeuksin, tarvitsemamme kortin resurssit, kuten I2C väylät, ovat automaattisesti palvelimen käytettävissä. Tietoturvan kannalta palvelimen käynnistäminen root-käyttäjän oikeuksin on ongelmallista, koska palvelin saa täydet oikeudet käyttöjärjestelmän ja kortin resursseihin. Samoin moottorinohjauksen käyttäminen ilman käyttäjätunnus/salasana -järjestelmää on ongelmallista, koska kuka tahansa käyttäjä voi ohjata moottoria, jos tietää ohjaimen IP-osoitteen. Näiden ongelmien takia järjestelmää ei voi asettaa esille julkiseen verkkoon. Opinnäytetyön yhteydessä päätettiin ajan ja resurssien säästämiseksi tyytyä näihin ratkaisuihin ja tietoturvaongelmiin palataan aikanaan jatkokehityksessä.

7 Askelmoottorin ohjaus

Askelmoottoria ohjataan Trinamic TMC222 -piirin, I2C-väylän ja selainyhteyden avulla. Ohjelmankehitysprosessin kulku esitetään esimerkkien avulla. Ohjelmointi aloitetaan fyysisestä askelmoottorin ohjauksesta ja edetään lopuksi selaimen suuntaan. Lopuksi kaikki kerätään yhdeksi Aragon alaiseksi kokonaisuudeksi, joka on käännettävissä Aragossa yhdellä käskyllä ja asennettavissa evaluointikortille normaalien Aragon tuottamien asennuskäytäntöjen mukaisesti.

7.1 Askelmoottorin ohjausohjelma

Ohjelmista alimmalla tasolla ohjauksessa toimii komentoriviohjelma, joka toisaalta ottaa käskyjä thttpd-palvelimelta ja toisaalta antaa käskyjä I2C-väylän kautta TMC222-piirille. Yksittäisiä kirjastofunktio- ja -aliohjelmakutsuja ei jatkossa selitetä kovin yksityiskohtaisesti, vaan lukijaa kehotetaan halutessaan tekemään funktion tai aliohjelman nimellä verkkohaku. Esimerkiksi getenv-funktion tietojen haku merkkijonolla ”Linux C getenv” antaa listan osumia, joiden perusteella kutsun tarvitsemat header-tiedostot, tarkka syntaksi, tarkoitus ja käyttö selviävät. Tärkeimmät toiminnot ohjelmissa selitetään lyhyesti sanallisesti ja ohjelmaesimerkkien kautta.

7.1.1 Cgi-käskyjen tulkinta

Thttpd-palvelin välittää selaimelta ja html-lomakkeelta saamansa tiedot ympäristömuuttujien kautta [24]. Ympäristömuuttujassa QUERY_STRING ovat lomakkeelta poimitut

kenttien nimet ja syötetyt kenttien arvot. Myöhemmin esitettävässä html-lomakkeessa on kaksi kenttää, first_position ja second_position. Tässä ei tarkisteta arvojen oikeellisuutta, vaan luotetaan käyttäjän hyväntahtoisuuteen. Kenttien arvot poimitaan alla olevan mukaisesti:

```
char * data;
long lFirstPosition, lSecondPosition;

//getenv palauttaa osoittimen ympäristömuuttujan sisältävään merkkijonoon
data = getenv("QUERY_STRING");

//etsitään kenttien arvot datasta
if(data == NULL)
{
		//jos ei dataa, virheilmoitus ja oletuspaikka
		printf("<P>Error! Error in passing data from form to script.\n");
		lFirstPosition = 25300;
		lSecondPosition = 5000;
}
//meillä on dataa, katsotaan, löytyykö oikeanlaista
//sscanf etsii annetun ohjeen mukaisesti arvoja merkkijonosta
else if(sscanf(data, "first_position=%ld&second_position=%ld\n", &lFirstPosition, &lSecondPosition)
!= 2)
{
		//jos ei, virheilmoitus ja oletuspaikka
		printf("<P>Error! Invalid data. Data must be numeric.\n");
		lFirstPosition = 15000;
		lSecondPosition = 25300;
}
else
{
		//dataa löytyi, kiitokset käyttäjälle
		printf("<P>Position 1: %ld\n", lFirstPosition);
		printf("<P>Position 2: %ld\n", lSecondPosition);
}
```

Näin meillä on selvillä kaksi paikkaa, joihin askelmoottori ajaa koelaitteemme kelkan. Printf-funktion kirjoittamat merkkijonot menevät palvelimelle osaksi koostettavaa html-sivua, joka lähetetään selaimelle. Sivun rakenne esitetään myöhemmin.

7.1.2 I2C-väylän alustus

I2C-väylä on ensin avattava. Käytämme avaukseen yleistä open-käskyä ja avaamme väylän luku-/kirjoitustilaan. Väylänumero fd on määritelty globaaliksi muuttujaksi.

```
//väylän nimi, jos virhe, virheilmoitus
fd = open("/dev/i2c-1", O_RDWR);
if (fd < 0)
{
		printf("<P>Error! Error on opening the device file.\n");
		return 0;
}
```

Kun linja on saatu auki, voidaan ioctl-kutsulla asettaa linjan ominaisuuksia.

```
ioctl(fd, I2C_TIMEOUT, 1); //asetetaan timeout
ioctl(fd, I2C_RETRIES, 1);   //asetetaan uudelleenyritysten määrä
```

7.1.3 I2C-väylän lukeminen ja kirjoittaminen

Väylän lukemiseen ja kirjoittamiseen on tehty pienet aliohjelmat. Ennen lukua tai kirjoitusta asetetaan moottoriohjaimen I2C-osoite, joka on asetettu moottoriohjaimelle ohjainkortilla. Osoite on esimerkiksi

```
#define STEPPER_Y_ADDR 0xC2
```

Osoitetta käytetään, kun tehdään luku- tai kirjoituskutsu. Lukemiseen ja kirjoittamiseen käytetään kirjastofunktioita read ja write.

```
void i2cMasterSend(unsigned char ucMotor, unsigned char ucLen, unsigned char * pucBuf)
{
        ioctl(fd, I2C_SLAVE, ucMotor);          /* osoite */
        //kirjoitus
        if(write(fd, pucBuf, ucLen) != ucLen)
                printf("<P>failed to write\n");
}
```

Luku tehdään vastaavasti:

```
void i2cMasterReceive(unsigned char ucMotor, unsigned char ucLen, unsigned char * pucBuf)
{
        ioctl(fd, I2C_SLAVE, ucMotor);          /* osoite */
        if(read(fd, pucBuf, ucLen) != ucLen)
                printf("<P>failed to read\n");
}
```

7.1.4 Moottoriohjaimen ohjauskäskyt

TMC222 tuntee joukon käskyjä, joiden avulla sitä voidaan ohjata ja kysellä sen tilaa [25, s. 29]. Käskyt on annettu alla lyhyin kommentein ja C-kielen define-muodossa. Esimerkeissä käytetään tässä määriteltyjä nimiä.

```
//moottoriohjaimen komennot
#define GetFullStatus1        0x81      //kyselee ohjaimen tilan
#define GetFullStatus2        0xFC      //kyselee todellisen paikan, kohdepaikan ja
                                        //turvallisen paikan
#define GetOTPParam           0x82      //kyselee OTP-parametrien arvon
#define GotoSecurePosition    0x84      //siirtää moottorin turvalliseen paikkaan
#define HardStop              0x85      //pysäyttää moottorin ilman hidastusvaihetta
#define ResetPosition         0x86      //asettaa todellisen paikan nollapaikaksi
#define ResetToDefault        0x87      //korvaa RAM-parametrien arvon OTP-
                                        //parametrien arvoilla
```

```
#define RunInit          0x88      //alustaa  ohjaimen
#define SetMotorParam    0x89      //asettaa ohjaimen parametrit
#define SetOTPParam      0x90      //asettaa pysyvän OTP-muistin arvon
#define SetPosition      0x8B      //asettaa ohjaimen kohde- ja turvallisen paikan
                                   //arvot
#define SoftStop         0x8F      //pysäyttää moottorin hidastusvaiheen kanssa
```

7.1.5 Moottoriohjaimen alustaminen

Ennen kuin moottoriohjain on käyttövalmis, pitää se nollata, suorittaa tilakysely 1 ja asettaa meille sopivat oletusarvot ohjaimen parametreille.

Nollaus tehdään käskyllä

```
ResetMotor(STEPPER_Y_ADDR);
```

ja vastaava funktio on

```
void ResetMotor(unsigned char ucMotor)
{
        unsigned char ucLen = 0;
        //piirin valmistajan oletusasetukset
        pucI2CBuf[ucLen++] = ResetToDefault;
        i2cMasterSend(ucMotor, ucLen, pucI2CBuf);
}
```

Käytetään globaalia taulukkoa I2C-käskyn välittämiseen:

```
unsigned char pucI2CBuf[32];          //i2c buffer
```

TMC222-piiri vaatii toimiakseen alkuun tilakysely 1 käskyn suorittamisen, itse tilakyselyn tulosta ei käytetä mihinkään. Tilakysely tehdään käskyllä

```
GetFullStat1(STEPPER_Y_ADDR);
```

ja funktio on

```
void GetFullStat1(unsigned char ucMotor)
{
        unsigned char ucLen = 0;

        pucI2CBuf[ucLen++] = GetFullStatus1;

        //annetaan tilakyselykäsky
        i2cMasterSend(ucMotor, ucLen, pucI2CBuf);
        //luetaan tila
        i2cMasterReceive(ucMotor, 9, pucI2CBuf);
}
```

Lopuksi asetetaan ohjain meille sopiviin parametrien arvoihin, käsky on

```
SetMotorPar4Init(STEPPER_Y_ADDR);
```

ja funktio on

```
void SetMotorPar4Init(unsigned char ucMotor)
{
        unsigned char ucLen = 0;

        pucI2CBuf[ucLen++] = SetMotorParam;
        pucI2CBuf[ucLen++] = 0xFF;              //dummy
        pucI2CBuf[ucLen++] = 0xFF;              //dummy
        pucI2CBuf[ucLen++] = 0xF6;              //Irun & Ihold
                                                //ajovirta & pitovirta
        pucI2CBuf[ucLen++] = 0xD1;              //Vmax & Vmin
                                                //maksiminopeus&
                                                //miniminopeus
        pucI2CBuf[ucLen++] = 0x05;              //7,6,5 = SecPos
                                                //turvallinen paikka,bitit 10 - 8
                                                //4 = shaft
                                                //peruspyörimissuunta
                                                //3,2,1,0 = acceleration
                                                //kiihtyvyys/hidastuvuus
        pucI2CBuf[ucLen++] = 0x00;              //SecPos
                                                //turvallinen paikka,bitit 7 - 0
        pucI2CBuf[ucLen++] = 0x0C;              //4 = AccShape,
                                                //alkukiihdytys,kyllä/ei
                                                //3, 2 = StepMode
                                                //askellustapa 11 =1/16-askellus
        i2cMasterSend(ucMotor, ucLen, pucI2CBuf);
}
```

Tämän jälkeen moottori on valmis ottamaan vastaa liikekäskyjä. Alustusten lopuksi moottori ajetaan kotiasemaan, jotta kelkan fyysinen paikka saadaan tarkasti tietoon. Kelkan kotiasema määritellään ulkoisen mikrokytkimen suhteen. Ensin ajetaan kelkka kiinni mikrokytkimeen ja peruutetaan sitten siitä irti niin, että kykin juuri ja juuri aukeaa. Sen jälkeen ajetaan mikrokytkimeltä pois annettu määrä askelia. Jos kotiasema on kytkimen vieressä, 0 askelta, jos muualla, annettu määrä askelia. Kotiasema saavutetaan käskyllä

```
#define Y_STEPS_TO_HOME             0           //kotiasema on nyt samalla
                                                //puolella liukukiskoa kuin
                                                //mikrokytkin

StepHome(STEPPER_Y_ADDR, Y_STEPS_TO_HOME);
```

Vastaava funktio on

```
void StepHome(unsigned char ucMotor, short sLimit)
{
        unsigned char ucLow;

        //ajetaan ensin mikrokytkimelle
        StepToExternalSwitch(ucMotor, -30000);

        //arvot oletuksiin
        SetMotorPar4Init(ucMotor);
```

```
		//ajetaan mikrokytkimestä irti annettu askelmäärä
		StepToX(ucMotor, sLimit);

		//odotellaan, että ollaan käsketyssä asemassa eli tässä tutkitaan,
		//moottorin liiketilaa, jos liikkuu, ei olla perillä
		ucLow = GetMotorMotionStatus(ucMotor);
		while(ucLow > 0) {
				ucLow = GetMotorMotionStatus(ucMotor);
		}

		//annetaan maailman rauhoittua eli nukutaan hetki
		usleep(50000);
		//paikka nollataan
		DoResetPosition(ucMotor);
		GetFullStat1(ucMotor);
}
```

Moottorin liiketila haetaan funktiolla

```
uint8 GetMotorMotionStatus(uint8 uint8Motor) {
		//liiketila haetaan tilakysely 1:llä
		GetFullStat1(uint8Motor);

		//moottorin liiketilaa, jos ei perillä, liiketila <> 0, ks. [25, s. 20]
		return (puint8I2C_buf[5] >> 5);
}
```

Mikrokytkimeen ajetaan aliohjelmassa StepHome käskyllä

```
StepToExternalSwitch(ucMotor, -30000);
```

ja vastaava funktio on

```
void StepToExternalSwitch(unsigned char ucMotor, short sLimit)
{
		unsigned char ucAtHome;

		SetMotorPar4Init(ucMotor);

		DoResetPosition(ucMotor);
		GetFullStat1(ucMotor);

		//ajetaan pitkälle
		StepToX(ucMotor, sLimit);

		//kytkimen tila = auki
		ucAtHome = 0;
		//ajetaan, kunnes kytkin menee kiinni
		while(!ucAtHome) {
				//kysytään kytkimen tila, ks. [25, s. 20]
				GetFullStat1(ucMotor);
				ucAtHome = (pucI2CBuf[5] & 0b00010000) >> 4;
		}

		//ollaan kiinni kytkimessä, pysäytetään moottori ja nollataan tila
		DoHardStop(ucMotor);
		DoResetPosition(ucMotor);

		GetFullStat1(ucMotor);
```

```
		//peruutetaan pois kytkimeltä
		StepToX(ucMotor, 1000);

		//kytkimen tila = kiinni
		ucAtHome = 1;
		//ajetaan, kunnes kytkin aukeaa
		while(ucAtHome) {
				GetFullStat1(ucMotor);
				ucAtHome = (pucI2CBuf[5] & 0b00010000) >> 4;
		}

		//kytkin aukesi, nollataan
		DoHardStop(ucMotor);
		DoResetPosition(ucMotor);

		GetFullStat1(ucMotor);

		//fiilistellään vielä vähän
		StepToX(ucMotor, 1);
		usleep(50000);

		DoResetPosition(ucMotor);
		GetFullStat1(ucMotor);
}
```

7.1.6 Moottorin siirtokäskyt

Kelkka siirretään kotiasemasta kohtaan 1 seuraavilla käskyillä:

```
//ajetaan kohtaan 1
StepToX(STEPPER_Y_ADDR, (short)lFirstPosition);

//tutkitaan paikka, odotetaan, kunnes perillä
while(GetMotorPosition(STEPPER_Y_ADDR) != (short)lFirstPosition)
{
		//perillä?
}
```

Sitten siirrytään kohtaan 2 käskyillä

```
//kuten kohta 1
StepToX(STEPPER_Y_ADDR, (short)lSecondPosition);
while(GetMotorPosition(STEPPER_Y_ADDR) != (short)lSecondPosition)
{
		//perillä?
}
```

Lopuksi palataan kotiasemaan:

```
//Home Sweet Home
StepHome(STEPPER_Y_ADDR, Y_STEPS_TO_HOME);
```

Moottorin paikka kysytään funktiolla

```
uint16 GetMotorPosition(uint8 uint8Motor) {
        int16 int16High;

        //tutkitaan paikka, tilakysely 2, ks. [25, s. 31]
        GetFullStat2(uint8Motor);

        //paikka vie 2 tavua
        int16High = ((uint16) puint8I2C_buf[1]) << 8;

        return int16High | ((int16) puint8I2C_buf[2]);
}
```

Haluttuun kohtaan ajetaan funktion StepToX avulla, jonka sisältö on

```
void StepToX(unsigned char ucMotor, short sX)
{
        unsigned char ucLen = 0;

        //alustetaan käsky ja haluttu paikka
        pucI2CBuf[ucLen++] = SetPosition;
        pucI2CBuf[ucLen++] = 0xFF;
        pucI2CBuf[ucLen++] = 0xFF;
        pucI2CBuf[ucLen++] = (unsigned char) (sX >> 8);
        pucI2CBuf[ucLen++] = (unsigned char) (sX);

        i2cMasterSend(ucMotor, ucLen, pucI2CBuf);
}
```

7.1.7 Vastaussivun kirjoittaminen selaimelle

Thttpd-palvelin koostaa vastaussivun selaimelle stdout:iin kirjoitetuista riveistä. Sivun rakenne on normaali html-sivun rakenne eli siinä on head- ja body-osa. Alla on sivun kirjoitus koottuna yhteen. Esimerkkinä on normaali tilanne, jossa käyttäjä on antanut sekä ensimmäisen että toisen paikan.

```
printf("Content-type: text/html\n");
printf("\n");
printf("<html>\n");
printf("    <head>\n");
printf("                <title>Welcome to thttpd i2ctest</title>\n");
printf("    </head>\n");
printf("    <body>\n");
printf("                <P>Position 1: %ld\n", lFirstPosition);
printf("                <P>Position 2: %ld\n", lSecondPosition);
printf("                <p>EURO is saved: stepper.cgi is running!\n");
printf("    </body>\n");
printf("</html>\n");
```

Talletetaan lopuksi ohjelma tiedostoon stepper.c määrittelyineen, pää- ja aliohjelmineen.

7.2 Selainsivu paikkaparametrien antamiseen

Sivulla on määritelty kaksi kenttää kelkan paikan määrittelyyn. Sivulla olevan formin tiedot lähetetäänGET-metodilla, jolloin niiden sisältö saadaan ympäristömuuttujasta QUERY_STRING, kuten stepper.c-ohjelmassa tehtiin.

```
<html>
	<head>
		<title>Set stepper steps, save the EURO</title>
	</head>
	<body>
		<form method = get action="/cgi-bin/stepper.cgi">
		  <div><label>First position: <input name="first_position" size="5"></label></div>
		  <div><label>Second position: <input name="second_position" size="5"></label></div>
		  <div><input type="submit" value="Set positions"></div>
		</form>
	</body>
</html>
```

Talletetaan sivu nimellä steppervalues.html.

7.3 Ohjelman kääntäminen ja asennuspaketin tekeminen

Käyttäjän tekemät ohjelmat ja reseptit talletetaan Aragon juurihakemiston alla olevan arago-custom -hakemiston alle. Me käytämme rakennetta

```
arago-custom
		packages
				stepper
				- reseptit
						files
						- ohjelmien lähdekoodit
						- html-sivujen koodi
```

Kääntäminen ja asennuspaketin tekeminen tapahtuu bitbake reseptillä stepper_1.0.0.bb:

```
DESCRIPTION = "stepper test program"

PR = "r1"

DEPENDS = ""

#lähdekoodit
SRC_URI += " \
  file://stepper.c \
  file://steppervalues.html \
"
```

```
S = "${WORKDIR}"

#mitä käännetään
do_compile () {
   ${CC} ${CFLAGS} ${LDFLAGS} -o stepper.cgi stepper.c
}

#mitä asennetaan
do_install () {
   #tarkistetaan /srv/www/cgi-bin, luodaan jos ei ole olemassa
   install -d ${D}${servicedir}/www/cgi-bin/
   #asennetaan stepper.cgi, exe oikeudet
   install -m 0755 ${S}/stepper.cgi ${D}${servicedir}/www/cgi-bin/
   #asennetaan html-sivu
   install -m 0664 ${S}/steppervalues.html ${D}${servicedir}/www/
}

#asennuspakettiin talletettavat tiedostot
FILES_${PN} = " \
  ${servicedir}/www/cgi-bin/stepper.cgi \
  ${servicedir}/www/steppervalues.html \
"
```

Kääntäminen tapahtuu käskyllä

```
>MACHINE=am3517-evm bitbake stepper
```

Arago tallentaa stepper:n asennuspaketin tiedostopuussaan olevan deploy-hakemiston alihakemistoon tiedostonimellä stepper_1.0.0-r0.6_armv7a.ipk. Kopioimme sen verkko-levylle NFS-palvelimelle käyttäjän root kotihakemistoon /home/root. Asennus kehitys-kortille tapahtuu kehityskortin komentoriviltä root-käyttäjän hakemistossa komennolla

```
>opkg install stepper_1.0.0-r0.6_armv7a.ipk
```

Lopuksi voidaan testata kokonaisuuden toiminta selaimen avulla antamalla selaimen komentoriville käsky (oletetaan, että kortin IP-osoite on 192.168.2.5)

```
http://192.168.2.5/steppervalues.html
```

Selaimeen avautuu kahden kentän lomake, johon syötetään kaksi liukukelkan pysäkin paikkaa ja lähetetään lomake palvelimelle. Kelkka ajaa pyydettyihin paikkoihin ja palaa kotiasemaan.

8 Pohdinta

Tässä luvussa pohditaan opinnäytteen tavoitteiden toteutumista ja saavutettuja tuloksia. Tarkastellaan toteutusta ja menetelmiä, arvioidaan mitä opittiin ja miten sekä saavutettujen tulosten tulevaa käyttöä ja jatkokehitystä.

8.1 Tavoitteiden toteutuminen

Työlle oli asetettu projektin alussa kolme päätavoitetta:

1. tutustua TI:n AM3517-prosessoriin ja Logic PD:n AM3517 EVM-evaluointikorttiin ja selvittää sen sopivuutta oman kortin suunnittelun pohjaksi
2. valita prosessorille ja evaluointikortille kehitystyökalut ja ottaa ne käyttöön
3. kehittää evaluointikortin ja valittujen kehitystyökalujen avulla prototyyppi selaimen avulla ohjattavasta askelmoottorin ohjausjärjestelmästä

Tavoitteet toteutuivat hyvin. Tarkastellaan tavoitteiden toteutumista kohdittain.

1. TI AM3517 -prosessori sekä Logic PD AM3517-kehityskortti tulivat tutuiksi. Työtä tehtäessä prosessorin dokumentointi, rakenne ja toiminta käytiin laajasti läpi. Evaluointikortin rakenne ja toiminta todettiin sopivaksi oman kortin suunnitteluun ja ensimmäiset versiot onkin jo toteutettu omasta kortista.
2. Kehitysympäristöksi valittiin Ubuntu ja TI:n Arago-projektin työkalut. Ubuntu kehitysympäristönä todettiin toimivaksi ja käteväksi. Sen käyttöönotto sujui ilman suuria kommelluksia. TI:n Arago-projektin tarjoama sulautetun Linuxin kokonaisvaltainen kehittämistapa todettiin joustavaksi ja toimivaksi. Sen avulla on mahdollista hallita sulautetun Linux-järjestelmän kehitys aivan ensimmäisestä toisen tason lataajasta koko Linux-järjestelmän asentamiseen sovelluskortille.
3. Työn aikana koottiin koelaitteisto askelmoottorin ohjaukseen ja toteutettiin selainpohjainen askelmoottorin ohjausohjelmisto onnistuneesti. Työssä valittu kehitysympäristö ja järjestelmän kokoonpano todettiin onnistuneeksi ja toimivaksi. Ohjelmiston toteutuksessa suurena apuna olivat sulautetun Linuxin tarjoamat mahdollisuudet kuten thttpd-www-palvelin, monipuolinen verkkoliikenteen hallinta sekä valmiit I2C-ajurit.

8.2 Tiedonhaku- ja työskentelytavat

Heti työn alussa kävi selväksi, että tämä maailma on verkossa. Kaikki dokumentointi, asiaan liittyvä keskustelu ja ihmiset ovat siellä. Niinpä tiedon hankinnan lähteenä on käytetty pelkästään verkkoa. Hankaluutena (kylläkin tavanomaisena työelämässä) oli aiheen tavaton laajuus. Kokonaisuutena ketju

Ubuntu <-> Arago <-> sulautettu Linux <-> thttpd-www-palvelin <-> moottorin ohjaus

on mieluummin galaksi kuin aurinkokunta. Lisäksi laitteistoon ja kehitysvälineisiin liittyvän dokumentoinnin laajuus (esimerkiksi AM3517-prosessorin ns. datalehdessä on tällä hetkellä 2742 sivua, sisällysluettelo/taulukkolista loppuu sivulle127) ja dokumentoinnin hajanaisuus sekä tekijän joskus raivon partaalle vienyt ristiriitaisuus hankaloittivat työn etenemistä. Toisaalta lohtua toi nopeasti tehty huomio siitä, että jos törmäät johonkin ongelmaan, niin et varmasti ole ensimmäinen, joka on siihen törmännyt. Ongelmana on vain löytää se toinen onneton, jolla on ollut sama ongelma. Yleensä ratkaisu on löytynyt ja jos ei, sitten on valittu jokin toinen lähestymistapa.

Työskentelytapana oli edetä pienin askelin, kokonaisuus ja vaihe kerrallaan. Aivan työprosessin alusta asti on tehty pieniä prototyyppiohjelmia, joilla eri asioita on testattu. Kun on ilmennyt jokin uusi tarve, ensimmäisenä on tehty asiaa koskeva verkkohaku ja yleensä on löytynyt yksi tai useita esimerkkiohjelmia, joita soveltamalla on saatu aikaan prototyyppi. Tärkeässä osassa on ollut myös Arago-ympäristön hakemistorakenteessa olevat järjestelmät ja niiden reseptit, mallia omiin resepteihin ja ohjelmakoodeihin on etsitty niistä.

8.3 Kehitysympäristön käyttövarmuus

Ohjelmistokehitysketju on toiminut yllättävän hyvin. Ubuntu-käyttöjärjestelmäympäristöstä johtuneita ongelmia ei ole ollut yhtään. Arago-projektin ohjelmista näkyy positiivisesti, että TI ja sen kaupallinen intressi ovat Aragon takana eli kokonaisuus on toiminut erittäinkin luotettavasti. Ajoittaisia ongelmia on ollut verkon syövereistä ladattavien, sulautettuun Linuxiin kuuluvien eri osien lähdekoodien latauksen kanssa. Palvelimia on ollut alhaalla tai muita verkko-ongelmia on esiintynyt. Asiaa on auttanut, että yleensä lähdekoodeilla ei ole yhtä ainoaa latauspaikkaa vaan useita vaihtoehtoisia. Joskus on mieleen kumminkin hiipinyt epäilys, kun neljäskään latauspalvelinvaihtoehto ei ole vastannut latauspyyntöön.

8.4 Oppiminen

Opinnäytetyön tekijä on saanut syvällistä kokemusta digitaalitekniikan mahdollisuuksista ja menetelmistä nykyaikaisessa sulautettujen järjestelmien kehitystyössä. Valmiudet verkkoyhteisön tarjoamien mahdollisuuksien käyttöön ovat parantuneet merkittävästi. Uusi prosessoriperhe on tullut tutuksi (tutuille vitsailenkin, että työni on perhetyötä), samoin sen ohjelmointi ja ohjelmointivälineet.

8.5 Jatkokehitys

Opinnäytetyön tuloksia hyödynnetään kaupallisessa toiminnassa sekä Emmecon Oy:ssä että Binomi Oy:ssä. Opinnäytetyön tulokset luovat hyvän pohjan käyttää sulautettua Linuxia yhtiöiden toiminnassa ja tuotteissa.

Jatkokehityksessä erityistä painoa asetetaan tietoturvalle. Thttpd-palvelimelle luodaan oma käyttäjä ja tälle käyttäjälle annetaan oikeudet vain tarvittaviin käyttöjärjestelmän ja kortin resursseihin. Samaten luodaan käyttäjätunnus/salasana -järjestelmä moottoriohjauksen käyttäjän tunnistamiseksi.

Jatkokehityksessä on itse asiassa jo edetty pitkälle tämän opinnäytetyön aiheen ja tulosten ohi. Käytössä ovat jo myös 1-Wire- ja SPI-väylät. Samoin koko Aragon arsenaali U-Boot:n, GNU/Linuxin ytimen ja tiedostojärjestelmän kehittämisen ja muokkauksen osalta on käytössä. Olemme myös ohjelmoineet ajureita GNU/Linuxin ytimeen SPI-väyläpohjaisille antureille. Näin opinnäytetyö on ollut erinomaisen hyödyllinen osa yritysten tuotekehitystyötä.

Lähteet

1. Hallinan, C. Embedded Linux Primer. Boston. USA. 2007. 537 s. ISBN 978-0-13-167984-9.
2. Texas Instruments. TI E2E Community. Verkkosivu. 1995-2011. [Viitattu 3.11.2011]. Saatavissa: http://e2e.ti.com/
3. OpenEmbedded. Welcome to OpenEmbedded. Verkkosivu.Päivitetty 11.5.2011. [Viitattu 31.10.2011].Saatavissa: http://www.openembedded.org/index.php/Main_Page
4. Angstrom Project. FAQ for Angstrom project. Verkkosivu. Päivitetty 5.10.2011. [Viitattu 1.11.2011]. Saatavissa: http://www.linuxtogo.org/gowiki/AngstromFaq
5. Arago Project. Arago Project Main Page. Verkkosivu. Päivitetty 19.10.2011. [Viitattu 1.11.2011]. Saatavissa: http://arago-project.org/wiki/index.php/Main_Page
6. Texas Instruments. AM3517. Tuotesivu. [Viitattu 31.10.2011]. Saatavissa: http://www.ti.com/product/am3517
7. GNU Operating System. Verkkosivu. Päivitetty 1.11.2011. [Viitattu 20.11.2011]. Saatavissa: http://www.gnu.org/
8. Venkateswaran, S. Essential Linux Device Drivers. Boston. USA. 2008. 537 s. ISBN 978-0-13-239655-4.
9. Logic PD. AM35x SOM-M2. Moduulin esittelysivu. [Viitattu 31.10.2011]. Saatavissa: http://www.logicpd.com/products/system-on-modules/am35x-som-m2/
10. Logic PD. Zoom AM3517 EVM. Evaluointipaketin esittelysivu. [Viitattu 31.10.2011]. Saatavissa: http://www.logicpd.com/products/system-on-modules/zoom-am3517-evm/
11. Texas Instruments. AM35x ARM Microprocessor Technical Reference Manual Version B (Rev. B). PDF- dokumentti. 2009. Päivitetty 7.7.2010. Saatavissa: http://www.ti.com/lit/ug/sprugr0b/sprugr0b.pdf
12. Texas Instruments. AM3517/05 Sitara ARM Microprocessors. PDF-dokumentti. 2009. Päivitetty 03.2011. Saatavissa: http://www.ti.com/lit/ds/sprs550c/sprs550c.pdf
13. TRINAMIC. TMC222. Tuotesivu. 2006. [Viitattu 31.10.2011]. Saatavissa: http://www.trinamic.com/tmc/render.php?sess_pid=211
14. Nanotec. ST4209S1006. Datalehti. 2006. PDF-dokumentti. Päivitetty 26.2.2007. Saatavissa: http://en.nanotec.com/downloads/pdf/1138/ST4209S1006.pdf
15. Ubuntu. Download Ubuntu. Verkkosivu. [Viitattu 1.11.2011]. Saatavissa: http://www.ubuntu.com/download/ubuntu/download
16. Arago Project. Building with Arago. Verkkosivu. Päivitetty 16.8.2011. [Viitattu 1.11.2011]. Saatavissa: http://arago-project.org/wiki/index.php/Building_with_Arago
17. Micah Carrick. Another blog. Customizing gedit as a Web Developer's IDE. 2007. [Viitattu 1.11.2011]. Saatavissa: http://www.micahcarrick.com/gedit-html-editor.html
18. Mentor Graphics. Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux. Lataussivu. 2004-2011. [Viitattu 1.1.2011]. Saatavissa: https://sourcery.mentor.com/sgpp/lite/arm/portal/release858
19. David Sudjiman. Installing and setting TFTPD in Ubuntu. Verkkosivu. 2006. [Viitattu 1.11.2011]. Saatavissa: http://www.davidsudjiman.info/2006/03/27/installing-and-setting-tftpd-in-ubuntu/
20. Ubuntu Documentation. SettingUpNFSHowTo. Verkkosivu. [Viitattu 1.1.2011]. Saatavissa: https://help.ubuntu.com/community/SettingUpNFSHowTo

21. DENX. Das U-Boot -- the Universal Boot Loader. Verkkosivu. Päivitetty 1.10.2010. [Viitattu 2.11.2011]. Saatavissa: http://www.denx.de/wiki/U-Boot
22. Texas Instruments. AM35x-OMAP35x-PSP 04.02.00.07 UserGuide. Verkkosivu. Päivitetty 21.7.2011. [Viitattu 2.11.2011]. Saatavissa: http://processors.wiki.ti.com/index.php/AM35x-OMAP35x-PSP_04.02.00.07_UserGuide
23. ACME Labs. thttpd man page. Verkkosivu. 2000. [Viitattu 2.11.2011]. Saatavissa: http://acme.com/software/thttpd/thttpd_man.html
24. The Computer Technology Documentation Project. CGI Variables. Verkkosivu. [Viitattu 2.11.2011]. Saatavissa: http://www.comptechdoc.org/independent/web/cgi/cgimanual/cgivars.html
25. TRINAMIC. TMC222 – DATASHEET V. 1.12. PDF-dokumentti. Päivitetty 7.3.2011. [Viitattu 20.11.2011]. Saatavissa: http://www.trinamic.com/tmc/media/Downloads/integrated_circuits/Tmc222/TMC222_datasheet.pdf