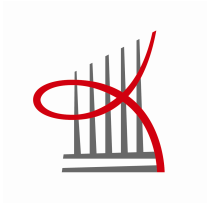


ANTURITIEDON KÄSITTELYOHJELMA SULAUTETULLE LINUXILLE

Harri Hietaranta

Opinnäytetyö
Marraskuu 2011
Tietotekniikka
Ohjelmistotekniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen Ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

HIETARANTA HARRI : Anturitiedon käsittelyohjelma sulautetulle Linuxille

Opinnäytetyö 30s., liitteet 27s.
Marraskuu 2011

Työn tarkoituksena oli tehdä sovellus, joka noutaa palvelimelta Json-tiedoston ja tulkitsee tiedostosta tarvittavat tiedot ja käyttää niitä kulloinkin halutulla tavalla. Kohdealustana sovellukselle oli PC-tietokonetta huomattavasti vähemmän muistia sisältävä, ARM suoritinta käyttävä, sulautettu Linux-laite. Laitteen tehottomuudesta johtuen muistivuotojen ennaltaehkäisy oli tärkeä osa työtä. Lisäksi laite oli kirjoitussuojattu, ja kaikkien sovellusten asentaminen tapahtuu ennalta luotuun levykuvaan. Levykuva luotiin ytimen ja kaikkien sovellusten yhteisellä makefile-tiedostolla. Työhön sisältyi sovellusten rakenteen suunnittelu ja kääntäminen, Python-kielen opettelu, makefile-tiedostojen käytön opettelu ja ARM-kääntäjän käytön opettelu.

Laitteen kehityksestä vastaavien yritysten välillä oli solmittu sopimus, jonka mukaan sovellukset tehdään Python-kielellä. Muistivuotojen välttämiseksi sovelluksen jatkuva käynti suoritetaan C-kielisen sovelluksen sisällä ja varsinainen toiminnallisuus Python-sovelluksen sisällä. C-sovellusten kääntäminen suoritettiin Sourcery G++ Lite, käännösympäristöpaketin, ARM-kääntäjällä. Python-sovellusta ei käännetty, vaan ajettiin Python-tulkilla. Laitteen makefile-tiedosto ja levyosioiden luontiin käytetyt scriptit kuuluivat Finsor Beddit Agent 2 system sovelluspaketin - lähdekoodeihin.

Avainsanat : ARM , Json , Python , Makefile, Scripti

ABSTRACT

Tampereen Ammattikorkeakoulu
Tampere University of Applied sciences
Computer Systems Engineering
Option of Software Engineering

HIETARANTA HARRI : Sensordata Handling Application for an Embedded Linux-system

Bachelor's thesis 30 pages, appedices 27 pages
November 2011

The purpose of this thesis work was to make an application which retrieves a Json-file from server and parses the information from it to use it in various ways. The target platform was a embedded Linux device using an ARM-processor with considerably less memory than a PC. Because of the lack of resources in the device, preventing memory leaks was an important part of the work. In addition the device was on a read-only system and all the installations of applications had to be done in advance to an premade disk image. The disk image was created with a combined makefile for the Linux kernel and all third party applications. Designing the application structure, compiling them, learning Python and the use of makefiles and a cross-compiler, were all part of the work.

There was a deal between all companies involved in the development of the device, that all software will be developed with Python. To prevent memory leaks, the loop structure of the software was implemented with C and the actual functionality with Python. The C-application was compiled with Sourcery G++ Lite cross-compiling environment. The Python-application was not compiled but ran with a Python-interpreter. The makefile for the device and the scripts for creating the disk partitions were part of Beddit Agent 2 system source codes.

Keywords : ARM , Json , Python , Makefile, Scripti

ESIPUHE

Toimeksianto työlle tuli MeshWorks Wirelesiltä. Olin tehnyt yhteistyötä yrityksen kanssa jo edellisenä kesänä, ja sain tänä kesänä uuden toimeksiannon kehittää sulautettua verkkolaitetta hallitsemaan sensoreita ja käsittelemään niistä saatua tietoa. Tietoa olisi tarkoitus käyttää mm. Nokian alulle panemassa SOFIA-projektissa. Työtä oli mielenkiintoista tehdä, sillä en ollut ollut tekemisissä vastaavien laitteiden kanssa aikaisemmin. Lisäksi uuden kielen, eli Pythonin, opettelu oli mielenkiintoista ja toivottavasti myös tulevaisuuden kannalta hyödyllistä. SOFIA-projekti on myös erittäin mielenkiintoinen konsepti, ja olen myöhemmin kehittänyt siihen mm. mobiilisovelluksen.

Marraskuu 2011
Harri Hietaranta

Sisällys

LYHENTEIDEN JA TERMIEN LUETTELO

Json	JavaScript Object Notation. JavaScript-kieleen perustuva tiedon siirtoformaatti.
ARM	Prosessoriarkkitehtuuri, jota mm. mobiililaitteet käyttävät
Python	Tulkattava ohjelmointikieli
Makefile	Tiedosto, joka määrittelee, miten ja minne ohjelma käännetään
Muistivuoto	Sovellus kuluttaa järjestelmän keskusmuistia kasvavalla tahdilla.
Kääntäminen	Lähdekoodin muuntaminen varsinaiseksi sovellukseksi.
Scripti	Yksi tai useampi komento, jotka ovat määritelty tiedostossa, tai komentorivillä. Komentoja luetaan tulkilla, joka riippuu tiedoston tyypistä.
Lipputiedosto	Tiedosto, jonka olemassaolo merkitsee ehdon toteutumista.
SOFIA	Smart Objects for Intelligent Applications. Monikansallinen äly laiteprojekti.
SIB	Service Implementation Bean. Java objekti, jolla toteutetaan web-palvelin.
MicroSD	Micro Secure Digital. Sanoista Mikro Turvallinen Digitaalinen. Digitaalinen korttimallinen tallennusväline.
Unix	Laitteistoriippumaton käyttöjärjestelmä
Ubuntu	Unix-pohjainen käyttöjärjestelmä

1 JOHDANTO

Pyyntö rakentaa kyseinen sovellus tuli langattomia sensoreita valmistavalta Meshworks Wirelessiltä TAMK:ille vastikään perustetulle osuuskunnalle. Tarkoituksena oli kehittää Meshworksin osuutta Nokian alulle panemaan SOFIA-projektiin. SOFIA eli Smart Objects for Intelligent Applications (Viisaita esineitä älykkäisiin sovelluksiin), on monen yrityksen yhteistyöprojekti, jossa tietoa ympäristöstä tuodaan saataville älylaitteille. Sulautettu Linux-laite otettiin käyttöön alustaksi projektin jo alettua. Laite valittiin tehtävään kokeilumielessä, sekä sopivuutensa takia. Opinnäytetyö keskittyykin sovellusten kehittämiseen kyseiselle alustalle enemmän kuin itse SOFIA-projektiin, jota sitäkin sivutaan.

Työssä käsitellään aluksi kohdelaitetta luvussa 2, jotta lukija ymmärtää mitä tämän tyyppiselle laitteelle kehittäminen vaatii. Luvussa käydään läpi laitteen tuomia rajoitteita ja laitteen ominaisuuksia.

Luvussa 3 käydään läpi ohjelmistot, joita sovellusten kehityksessä käytetään. Ohjelmistokehitysympäristöjen lisäksi kaikki apuohjelmat, joita laitteen kanssa tarvitaan, on esitelty.

Neljännessä luvussa selitetään kaikkein tärkein osuus, eli itse ohjelmien toiminta ja rakenne. Sekä C- että Python-sovellus, on kumpikin esitelty erikseen koodinäytteiden kanssa.

Lähdekoodien lisäksi muitakin tiedostoja on muokattava kyseiselle laitteelle kehitettäessä. Luvussa 5 käsitellään makefile-tiedoston sekä scriptien muokkausta. Kyseisiä tiedostoja tarvitaan sovelluksen kääntämiseen, siirtämiseen ja ajamiseen.

Viimeisessä luvussa on yhteenveto opinnäytetyön koko prosessista. Luvussa analysoidaan eri osien toteutusta ja lopputulosta.

2 KOHDELAITE

Kohdelaitteena toimii Convergens Oy:n kehittämä BAM-1 , joka on ARM-prosessori pohjainen sensorihallintaan tarkoitettu verkkolaite. Laitteella on sulautetulle laitteelle ominaisia piirteitä, kuten kirjoitussuojattu järjestelmä, ulkoinen muistikortti kovalevyn sijasta sekä varsinaisten käyttöliittymälaitteiden puuttuminen.

Koska laite on kirjoitussuojattu, kaikkien järjestelmäkansioon asennettavien sovellusten on oltava jo asennettuna järjestelmään ennen sen käyttöönottoa. Kaikkien sovellusten asentaminen hoidetaan Beddit Agent 2 System - sovelluspaketin kautta, jonka toiminta kuvataan tarkemmin luvussa 3. Tietyn tyyppisiä sovelluksia, kuten Python-sovellukset, ei tarvitse kääntää, joten ne voidaan vain siirtää laitteeseen. Helpoin tapa suorittaa siirto on käyttää kopiointikomentoa samassa makefile-tiedostossa, jossa tiedostorakenne levykuvalle luodaan.

Kohdelaite käyttää ulkoista MicroSD - muistikorttia kiintolevyn sijasta. Kaikki järjestelmässä oleva löytyy kuudelta osiolta, jotka kortille luodaan. Osioilla on eri tiedostojärjestelmät, joten kaikki osiot eivät näy muistikorttia luettaessa PC:llä. Osioiden luonti tapahtuu Beddit Agentin sisältämällä scriptillä *create-sd-card.sh*.

Kun käytettävälle muistikortille on luotu oikeat osiot, levykuva on käännetty makefile-tiedoston avulla ja levykuva on siirretty muistikortille, laite on valmiina käynnistettäväksi. Laitteen kanssa ei tarvitse kommunikoida kuin sovellusten kehitysvaiheessa. Lopputuotteessa kaikki sovellukset toimivat automaattisesti scriptien avulla.

3 KÄYTETYT OHJELMISTOT

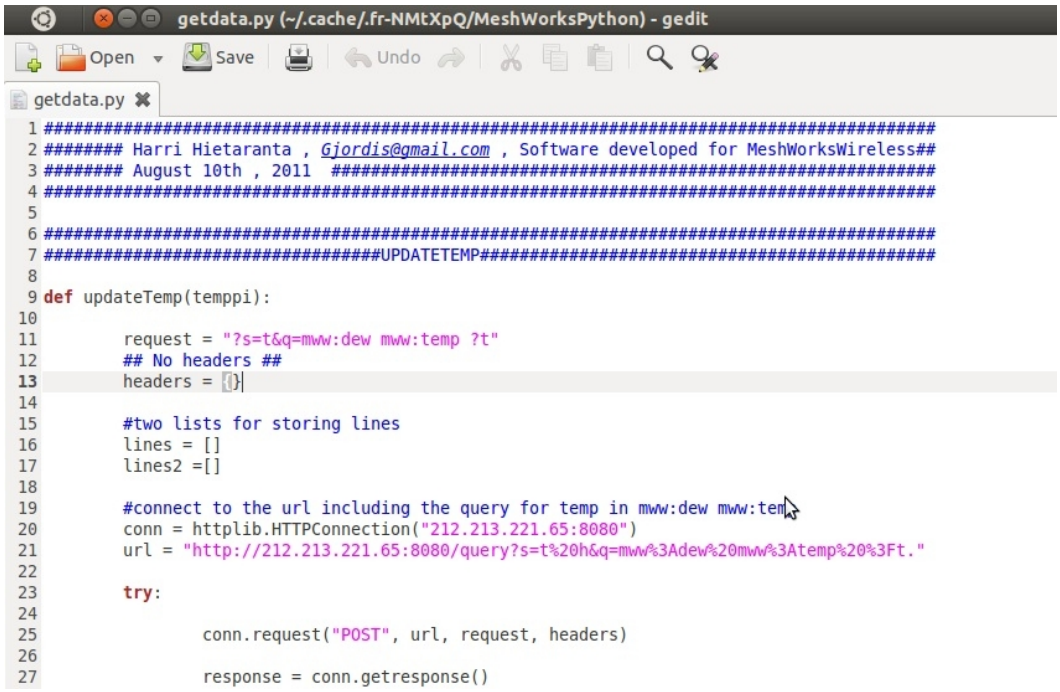
Tässä luvussa käsitellään kehitykseen käytettyjä perussovelluksia sekä kehitystyökaluja. Varsinainen koodi kirjoitettiin Gedit-tekstieditorilla. C-sovellus käännettiin Mentor Graphicsin Sourcery G++ Lite ARM-kääntäjällä. Laitteen muistikortin alustamiseen käytetyt scriptit sekä sovellusten kääntämiseen käytetty makefile-tiedosto kuuluivat Beddit Agent 2 – sovelluspaketin lähdekoodeihin, joka on laitteen kehityksen kannalta kaikkein tärkein kehitystyökalu. Laitteen kanssa kommunikointiin käytettiin Ubuntun sovellusvalikoimasta löytyvää Minicom-sarjaporttisovellusta. Laitteen kanssa kommunikointi on tarpeellista vain sovelluksen toimintaa testatessa.

3.1 Gedit-tekstieditori

Käytetyimpänä kehitystyökaluna toimi Gedit, joka on Unix pohjaisten Gnome-käyttöliittymää käyttävien käyttöjärjestelmien perus tekstieditori. Gedit vastaa hyvin pitkälti Windowsin Notepadia.

Gedit-editorin etuna on eri ohjelmointikielien tunnistus, joka mm. värittää tekstin samalla tavalla kuin kyseisen kielen kehitykseen tarkoitetut työkalut. Tekstin väritystä käytetään mm. erilaisten avainsanojen ja tekstin käyttötarkoituksen tunnistamiseen. Esimerkiksi tekstirivit jotka eivät ole suoritettavia, kuten kommenttirivit, värjäytyvät eri väriseksi kuin toiminnallinen koodi. Lisäksi toiminnallisen koodin eri osat värjäytyvät eri tavoin käyttötarkoituksensa mukaan kuten kuvasta (KUVA 1) on nähtävissä.

Lähes jokaisella ohjelmointikielellä on hieman erilainen syntaksi ja siihen liittyvä värikoodaus. Gedit tunnistaa käytössä olevan kielen joko tiedoston loppupäätteestä tai käyttäjän manuaalisesti tekemästä kielen valinnasta.



KUVA 1: Esimerkki Gedit-tekstieditorin värien käytöstä

Gedit ei sisällä minkäänlaisia käännöstyökaluja vaan sovellukset on käännettävä erillisellä kääntäjällä, jos sovellus vaatii kääntämistä. Windows käyttöjärjestelmällekin on saatavilla vastaavanlaisia ilmaisia sovelluksia kuten Notepad++.

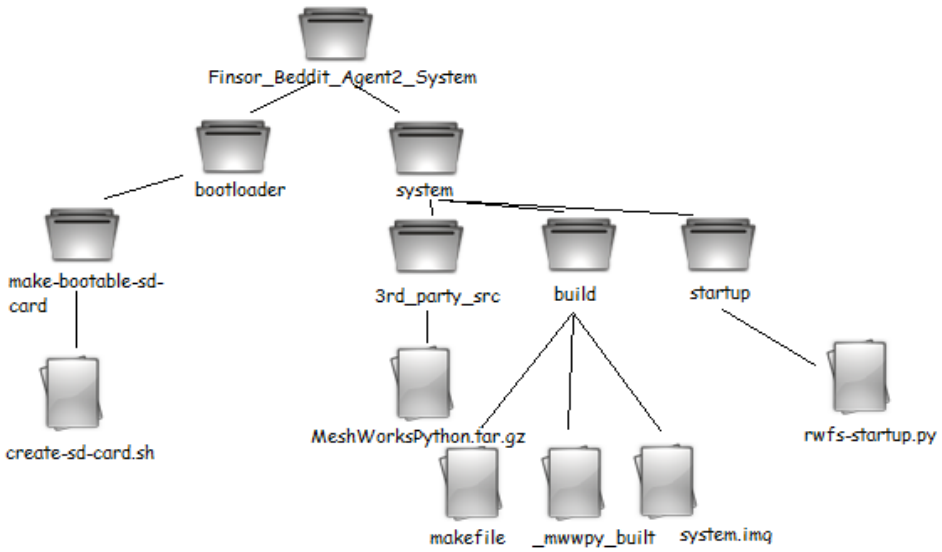
3.2 Sourcery G++ Lite ARM-kääntäjä

Työssä käytetty kääntäjä kuuluu Mentor Graphicsin kääntäjäpakettiin, joka sisältää ARM-arkkitehtuurin eri kääntäjiä. Paketti on ns. cross-compile – ympäristö. Cross-compile tarkoittaa sovelluksen kääntämistä sellaiselle laitteelle tai arkkitehtuurille toimivaksi, joka on eri kuin alusta, jolla käännöstä suoritetaan. Tämän tyyppisiä ympäristöjä käytetään usein, kun kohdelaite on huomattavasti PC:tä tehottomampi tai kun laitteen kanssa kommunikointi on haasteellista. Useammin syynä on se, että sulautettuihin laitteisiin halutaan vain aivan välttämättömät sovellukset, joihin kääntäjä ei kuulu. Opinnäytetyön kohdelaite täytti kaikki edellä mainitut ehdot, sekä on lisäksi kirjoitussuojattu osioilta, joille järjestelmäsovellukset asennetaan.

3.3 Beddit Agent 2 System

Beddit agent 2 sisältää kaikki työkalut käyttöjärjestelmän luomiseksi työssä käytetylle sulautetulle laitteelle. Vaikka laite käyttääkin Linux käyttöjärjestelmää, eikä laitteelle varta vasten suunniteltua kevyempää yksinkertaista käyttöjärjestelmää kuten useat sulautetut laitteet, on ohjelmistojen määrää karsittu huomattavasti verrattuna yleisiin Linux julkaisuihin, kuten Ubuntu. Kaikki sovellukset, jotka laitteelle asentuvat, sekä Linux ydin, on määritelty Beddit Agentin lähdekoodeissa tai Makefile-tiedostossa. Beddit Agent 2 on siis kaikki mitä valmiiksi asennettu laite pitää sisällään.

Sovellukset mitä itse suunniteltiin ja toteutettiin lisättiin siis Beddit Agentin asennustiedostoihin, ja asennettiin kohdelaitteelle ohjelmistopaketin mukana. Paketti haarautuu kahteen osaan, jotka ovat kansiot *system* ja *bootloader,* jotka on esitetty kuvassa. (KUVA 2)



KUVA 2: Kuvaus tiedostorakenteesta

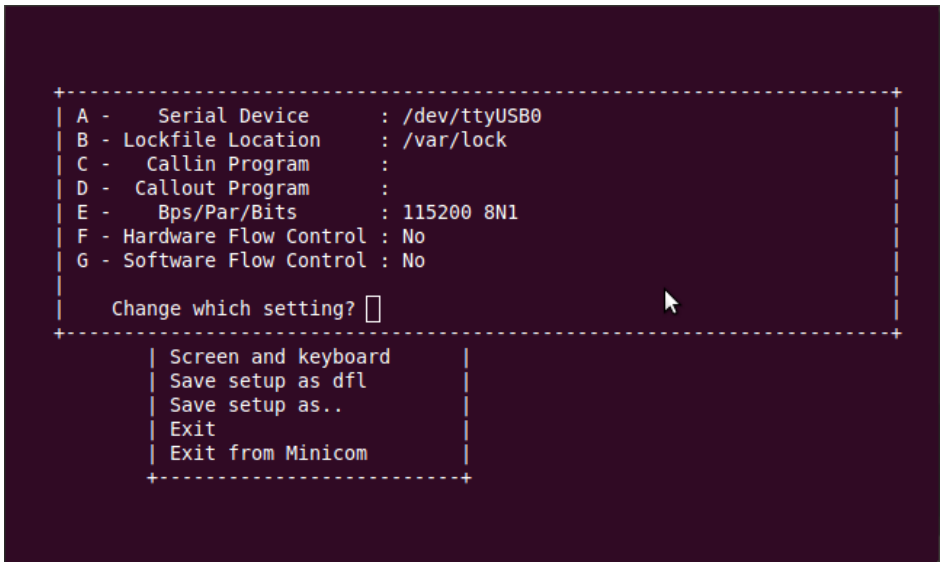
Bootloader hakemisto sisältää kaikki erinäisiä työkaluja levyosioiden luomiseen. Laite tarvitsee tietynlaisen rakenteen käytettävälle muistikortille. Beddit Agent 2 sisältää komentojonot, joilla tämä rakenne luodaan. Muistikortille luodaan 6 osiota, joista 3 on muistikortinlukijalla havaittavissa. Scriptit muodostuvat Linuxin perustyökalujen komennoista.

System hakemistosta löytyy kaikki lähdekoodit, scriptit ja aputiedostot. Tärkein alihakemisto on *build*, jossa sijaitsee Makefile-tiedosto ja kääntämisen jälkeen levykuva. Lisäksi kansioon luodaan lipputiedosto, joka kerta kun joku sovellus saadaan käännetyksi. Tiedostot ovat tyhjiä, mutta ilmaisevat sovelluksen olevan käännetty, ja sitä ei täten käännetä uudestaan. Jos sovellus halutaan kääntää uudestaan, on kyseisen sovelluksen lipputiedosto poistettava.

Build hakemiston sisällä on hakemisto *rootfs*, jonka sisälle muodostetaan levykuvan tiedostorakenne. Kun sovellus käännetään, valmis sovellus siirretään *rootfs* hakemiston sisälle. Lisäksi kaikki scripti-tiedostot on siirrettävä hakemiston sisälle. Makefile-tiedoston viimeinen toiminto on luoda levykuva *rootfs* hakemiston pohjalta.

3.4 Minicom-sarjaporttiterminaali

Ainoa tapa hallinnoida laitetta asentamisen jälkeen on sarjaporttiterminaalilla. Sarjaportti on nykyisin käytössä lähinnä vain tämäntyyppiseen kehitykseen. Nykyisin tietokoneissa itsessään on harvemmin fyysistä sarjaporttia, joten yhteys hoidetaan USB-sovittimen kautta.



KUVA 3: Minicom asetukset

Laitteeseen otetaan yhteyttä millä tahansa sarjaporttiterminaalisovelluksella, tässä tapauksessa Minicomilla. Laitteilla, joiden kanssa voi kommunikoida sarjaportitse, on tietyt asetukset. Asetusten asettamista ei käydä tarkemmin läpi, koska ne eivät kuulu olen-

naisena työn suorittamiseen. Asetukset ovat katsottavissa kuvasta (KUVA 2) tai liitteestä 1.

Sarjaporttiterminaalia käytettiin ohjelmistojen testaamiseen. Vaikka lopputuotteessa kaikki ohjelmistot toimivat automaattisesti, ja kaikki mahdolliset muutokset suoritetaan webkäyttöliittymän kautta, testatessa sovelluksia ajettiin sarjaporttiterminaalin kautta annetuilla komennoilla.

4 C- JA PYTHON-SOVELLUKSET

Sensoridataa hakeva ja sitä eteenpäin lähettävä sovellus toteutettiin Pythonilla ja prosessia toistava osuus C:llä. Kaikkea ei toteutettu Pythonilla, koska käytössä ollut *httplib*-kirjasto kärsii joistakin muistivuodoista. Lähdekoodit käydään tässä kappaleessa yksityiskohtaisesti läpi, jotta lukija ymmärtää sovelluksen toiminnan. Lähdekoodit ovat kokonaisuudessaan liitteinä, C-koodi liitteenä 2, Python-koodi liitteenä 3.

4.1 C-sovellus

Toteutuksen C-kielinen osuus on hyvin yksinkertainen. Sovelluksen ainoana tehtävänä on olla ikuisessa silmukassa käynnistymisestään lähtien, ja kutsua Python-osuutta kerran silmukan joka kierroksella.

```
While(1)
{
	system("python getdata.py");
	sleep(10);
}
```

Silmukan rakenne on perinteinen ikuinen silmukka, jota käytetään C-kielessä. Silmukan ehdoksi on annettu vain 1. Koska kyseessä on numeroarvo, joka ei koskaan muutu mihinkään ja edustaa totuusarvoa *true*, silmukan ehto on aina *true* eli tosi.

Järjestelmäfunktiolla *system()*, voidaan lähettää komentoja samaan tapaan kuin Linuxin komentoriviltä. Komentoriviltä Python-sovellus käynnistettäisiin komennolla : ”*python getdata.py*”, joten käytetään samaa komentoa funktiokutsussa.

Sovelluksen tarkoitus on hakea ja lähettää tietoa tietyin ajoin, joten sovelluksen täytyy odottaa haluttu aika päivitysten välillä. Sovelluksessa käytetään *sleep()*-funktiota hallitsemaan päivitysväliä. Funktio laittaa sovelluksen lepotilaan, jolloin sovellus ei kuluta resursseja, vaan vain odottaa. Lepotilan kesto määräytyy funktiokutsussa annetun luvun mukaan. Funktio vastaanottaa kokonaisluvun, joka on nukuttavien sekuntien määrä. Testausvaiheessa käytetään hyvin pientä aikaväliä, eli 10 sekuntia.

4.2 Python-sovellus

Python-sovellus hoitaa kaiken varsinaisen toiminnallisuuden. Sovellus jakautuu kolmeen suurempaan osaan: Json-tiedoston noutoon, Json-tiedoston parsimiseen ja tiedon uudelleenlähettämiseen tai käyttämiseen. Kaikki toiminnallisuus on tässä vaiheessa toteutettu yhteen lähdekooditiedostoon. Vaikka Python-kielessä sisennys on tärkeää, tässä tekstissä on sisennetty toisiinsa nähden vain samassa kappaleessa olevat koodirivit. Täten koodit joita leipäteksti erottaa, eivät ole toisiinsa nähden sisennetty oikein.

4.2.1 Json-tiedoston noutaminen

Json-tiedoston noutamiseen käytetään Pythonin peruskirjastoihin kuuluvaan *httplib*-kirjastoa. Kyseistä kirjastoa voi käyttää minkä tahansa web-osoitteen sisällön noutamiseen. Esimerkiksi HTML-sivua noudettaessa, saa vastauksena lähdekoodin kyseiseen sivuun. Json-tiedoston tapauksessa tiedosto sisältää Json-formaatissa olevaa tietoa. Json-tiedostoja noutaessa, palvelimelle voi lähettää pyynnön samassa formaatissa. Samaa kirjastoa voi käyttää myös tietojen lähettämiseen. Json-formaattiin voi tutustua tarkemmin osoitteessa http://json.org

Tiedostoa noutaessa parametreiksi annetaan käyttäjänimi, salasana ja toiminto. Pyyntöön käytetään Json-formaattia.

```
{
"request": {
 "identification": {
       "user": "hopeanuoli",
       "pass": "kastepiste",
       "action": "getdata"
}
 "device_filter": {
       "mwwid": [ "3" ]
}
}
}
```

Pyynnössä käytetään toimintoa ”*getdata*” , joka palauttaa kaikki kyseiselle käyttäjänimelle määritellyt laitteet. Parametri *device_filter* rajaa vastauksessa olevia laitteita. Laitteet rajataan *mwwid*:n mukaan laitteeseen numero 3. Pyyntö lähetetään käyttäen *httplib*-kirjastoa.

Ohjelmakoodissa pyyntö kootaan useista eri merkkijonoista helpon muutettavuuden takaamiseksi.

```
identification = '"identification": { "user": "'+user+'",
"pass": "'+passwd+'", "action": "'+action+'" }'
deviceFilter = '"device_filter": { "mwwid": [ "3" ] }'
request = '{ "request": { '+identification+' , '+deviceFil-
ter+' } }'
```

Kun pyyntö on koottu, se pitää lähettää palvelimelle. Palvelimelle tarvitsee aluksi luoda muuttuja ja määritellä osoite.

```
conn = httplib.HTTPConnection("www1.retailhosting.fi")
url =
"http://109.204.225.241/wsn.interface/JsonIfaceV1.jsoniface
"
```

Kun palvelimelle on määritelty kirjaston mukainen muuttuja, sitä voidaan käyttää. *HTTPConnection*-tyyppinen muuttuja sisältää sen toiminnallisuuden, jota yhteyksiin tarvitaan. Käyttäen muuttujan jäsenfunktioita, haetaan Json-tiedosto palvelimelta. Ensiksi lähetetään pyyntö palvelimelle.

```
conn.request("POST", url, request, headers)
```

Kun pyyntö on lähetetty, palvelimelta tuleva vastaus voidaan lukea käyttäen toista jäsenfunktiota.

```
response = conn.getresponse()
data = response.read()
conn.close()
```

Tehdään muuttuja *response* funktiosta *getresponse()*. Luetaan muuttuja ja tallennetaan koko vastaus merkkijonoon data. Seuraavaksi suljetaan yhteys, jotta avonaisia käyttämättömiä yhteyksiä ei jää avonaisiksi.

Seuraavaksi saatu data leikataan riveiksi. Dataa käydään läpi merkki kerrallaan kunnes rivinvaihto tulee vastaan. Merkkijonoa käydään läpi *for*-silmukassa, jossa jokainen merkki kerrallaan sijoitetaan muuttujaan *var*. Rivinvaihtoon asti kaikki merkit tallennetaan *line* merkkijonoon. Mikäli rivi ei ole tyhjä, se tallennetaan taulukkoon *lines*. Tämän jälkeen *line* alustetaan ja sitä aletaan täyttämään taas seuraavasta merkistä.

```
line=""
 for var in data:
```

```
if var != '\n':
      line += var
if var=="\n":
      if line != " " and line != "":
            lines.append(line)
            line = ""
```

4.2.2 Tiedon parsiminen

Rivit on tallennettu taulukkoon jäsentelyn helpottamiseksi. Ennen kuin taulukkoa voidaan alkaa käymään läpi, on luotava muuttujia hallitsemaan parsimista.

```
deviceIndex = -1
attributes = False
inAttr = False
lastWasAttr = False
values = 1
attribute = 0
attrValue = 0
```

Muuttujilla seurataan sitä, missä kohtaa Json-rakennetta ollaan. *DeviceIndex* seuraa sitä, monennettako laitetta käsitellään. Muuttuja *attributes* on tosi, jos käsitellään laitteen attribuutteja, kun taas muuttuja *inAttr* on tosi mikäli käsitellään attribuuttia attribuutin sisällä. Muuttujaa *lastWasAttr* käytetään seuraamaan tyhjiä attribuutteja, joiden sisällä ei ole arvoja, arvo on tosi mikäli edellinen *attr* tunniste on ollut tyhjä. Muuttujalla *values* seurataan laitteen sisältämien arvojen käsittelyä. Muuttuja arvoa kasvatetaan yhdellä aina, kun arvo on käsitelty. Muuttujalla *attribute* pidetään kirjaa siitä, monesko attribuutti on kyseessä. Muuttuja *attrValue* kertoo sen, monennessako arvossa ollaan attribuutin sisällä. Muuttujien merkitys selviää tarkastelemalla vastauksen rakennetta.

```
"dev":
[
{ "data":
[
{ "val": "156"
},
{ "val": "Kerros 4(Group 1)"
},
{ "val": "huone 403 (THB // DAF2)"
},
{ "val": "3"
},
{ "val": "3"
},
{ "val": "0050c2fffeacdaf2"
},
{ "attr":
[
```

```
{ "val": "22,5", "attr":
[
{ "val": "2011-03-18 10:30:23"
},
{ "val": ""
}
]
}
]
},
{ "attr":
[
{ "val": "17,3", "attr":
[
{ "val": "2011-03-18 10:30:23"
},
{ "val": ""
}
]
}
]
},
{ "attr":
[
{ "val": "3,5", "attr":
[
{ "val": "2011-03-18 04:46:28"
},
{ "val": "61,5051083333,23,8117100000"
}
]
}
]
},
{ "attr":
[
]
},
{ "attr":
[
]
},
{ "attr":
[
]
},
{ "attr":
[
]
},
{ "attr":
[
]
},
{ "attr":
[
]
}
]
```

Taulukkoa on helppo käydä läpi alkio kerrallaan. Taulukko on paras käydä läpi samanlaisessa *for*-silmukassa kuin merkkijonotkin, jossa jokainen alkio kerrallaan sijoitetaan alkion tyyppiseen muuttujaan.

```
for line in lines:
```

Jokainen rivi tarkastellaan erikseen, ja jos riviltä löytyy jokin tunniste, jota etsitään, siirrytään toimenpiteisiin. Tunnisteet, joita etsitään ovat *data* , *attr* ja *val*. Tunnisteet edustavat Json-elementtejä, *data* tarkoittaa uuden laitteen tietojen alkua , *attr* viittaa sensorin sisältämään mittausarvoon ja *val* muihin arvoihin kuten laitteen nimeen tai tunnisteeseen.

Tunnisteesta *data* aloitetaan uuden laitteen käsittely. Alustetaan kaikki muut seurantamuuttujat oletusarvoihin paitsi *deviceIndex*, jota kasvatetaan yhdellä. Mikäli laitteiden tietoja tallennetaan, eikä vain lähetetä eteenpäin, uusi laite myös luodaan vektoriin.

```
elif line.find("data") >0:
 #devices.push_back(Device())
 deviceIndex = deviceIndex+1
 values = 1
 attributes = False
 attribute = 0
 attrValue = 1
 inAttr = False
```

Seuraavaksi tutkitaan tunnistetta *attr*. Mikäli tunniste löydetään tutkitaan siihen liittyviä ehtoja. Ensimmäisenä ehtona tutkitaan ollaanko jo yksien attribuuttisulkujen sisällä tutkimalla boolean-muuttujia. Jos ehdot täyttyvät, merkataan, että ollaan kaksinkertaisten attribuuttisulkujen sisällä. Lisäksi kasvatetaan attribuuttien sisältämien arvojen seuranta-arvoa yhdellä.

```
if attributes==True and lastWasAttr==False:
 inAttr = True
 attrValue = attrValue+1
```

Mikäli kyseessä ovat ensimmäisen kerroksen attribuuttisulut, todetaan, että ollaan sulkujen sisällä ja merkitään kyseistä asiaa valvova muuttuja todeksi. Lisäksi kasvatetaan attribuuttien seuranta-arvoa yhdellä ja nollataan attribuutin sisältämien arvojen seuranta-arvo. Merkitään kaksinkertaisia attribuuttisulkuja ilmentävä muuttuja epätodeksi ja todetaan, että attribuuttisuluissa on käyty.

```
elif attributes == False:
 attributes = True
 attribute = attribute+1
 attrValue = 0
 inAttr = False
 lastWasAttr = True
```

Jos boolean-muuttuja *lastWasAttr*, on valmiiksi tosi, tarkoittaa se sitä, että sulut ovat olleet tyhjät, joten kasvatetaan vain attribuutteja laskevaa arvoa.

```
elif lastWasAttr==True:
 attribute = attribute+1
```

Seuraavana ehtona tarkastellaan tunnistetta *val*. Tämä tunniste tarkoittaa siis arvoa laitteessa itsessään, kuten nimeä, tai arvoa attribuutin sisällä, kuten mittaustulosta. Kun tunniste on löydetty, tutkitaan ollaanko yksin- tai kaksinkertaisten attribuuttisulkujen sisällä. Jos boolean-muuttuja *attributes* kertoo, että ei olla yksinkertaisten sulkujen sisällä, ei silloin olla myöskään kaksinkertaisten sulkujen sisällä.

```
elif (line.find("val")!=-1 and attributes==False):
```

Koska tässä vaiheessa tiedetään, että rivillä on arvo, joka halutaan talteen, aletaan turhia merkkejä poistamaan.

```
unwanted_characters = (' ', '"val":', '\n', ' ')
for string in unwanted_characters :
 line = line.strip(string)
```

Silmukka sijoittaa vuorotellen kaikki ei-halutut merkit tai merkkijonot muuttujaan string ja kutsuu Pythonin *string*-luokan jäsenfunktiota *strip*, joka poistaa annetun merkin tai merkkijonon siitä merkkijonosta, jonka jäsenenä sitä kutsutaan. Joihinkin käyttötarkoituksiin halutaan pois enemmän merkkejä kuin toisiin, joten esimerkiksi SOFIA-projektin päivityksiin poistetaan vielä ensimmäinen ja kaksi viimeistä merkkiä.

```
update = line[1:-2]
```

Seuraavaksi tutkitaan muuttujaa *values*, joka ilmoittaa, mikä laitteen arvoista on kyseessä.

```
if values == 1:
 emptyVariable = 0
elif values == 2:
 emptyVariable = 0
elif values == 3:
```

```
 emptyVariable = 0
elif values == 4:
 emptyVariable = 0
elif values == 5:
 emptyVariable = 0
elif values == 6:
```

Kyseisiä arvoja ei tällä kertaa tarvittu, joten ehtojen sisällä ei tehdä mitään järkevää. Pythonin syntaksi vaatii, että ehtolauseiden sisällä tehdään jotain, joten jokaisen sisällä annetaan muuttujalle *emptyVariable* arvo 0. Muuttujaa ei käytetä mihinkään. Lisäksi arvoja seuraavaa *values*-muuttujaa kasvatetaan taas yhdellä.

```
values = values+1
```

Toisessa tapauksessa muuttujat ilmoittavat, että ollaan attribuuttisulkujen sisällä, jolloin käytössä ovat eri ehdot. Aluksi poistetaan taas ei-halutut merkit ja merkkijonot aikaisemmin esitetyllä tavalla. Seuraavaksi on tutkittava, ollaanko yksin- vai kaksinkertaisten attribuuttisulkujen sisällä. Tätä tarkkaillaan *inAttr*-muuttujalla. Seuraavien rivien käsittelytapa on riippuvainen siitä, onko arvo tosi vai epätosi. Jos arvo on epätosi, käsitellään laitteen pääasiallisia mittausarvoja.

```
if attribute == 1:#Temp
 #print "Temperature : \t",line
 updateTemp(update)
 lastWasAttr = False

elif attribute == 2:#Humidity
 #print "Humidity : \t",line
 lastWasAttr = False

elif attribute == 3:#Battery voltage
 #print "Battery : \t",line
 lastWasAttr = False

elif attribute == 4:#Co2
 #print "CO2 : \t\t",line
 lastWasAttr = False

elif attribute == 5:#light
 #print "Light : \t",line
 lastWasAttr = False

elif attribute == 6:#rssirx
 #print "RssiRx : \t",line
 lastWasAttr = False

elif attribute == 7:#rssitx
 #print "RssiTx : \t",line
 lastWasAttr = False
```

Tässä käyttötapauksessa, mittausarvoista vain lämpötilaa on käytetty, ja se on annettu eteenpäin toiselle funktiolle. Kaikissa arvoissa on kommentoituna pois arvon tulostaminen, jota on käytetty testatessa. Lisäksi kaikissa tapauksissa tyhjiä attribuutteja valvova muuttuja merkitään epätodeksi, sillä arvoja on löytynyt.

Jos arvo on tosi, käsitellään attribuuttien arvoilla olevia lisäarvoja, kuten sijainti ja aikaleima. Mikäli kyseessä ei ole *attrValue* 1 tai 2 eli aikaleima tai sijainti, voidaan attribuutti merkitä päättyneeksi. Merkitään sekä ulompia, että sisempiä attribuuttisulkuja ilmaisevat muuttujat epätosiksi.

```
elif(inAttr == True):
 if attrValue == 1 :
       blaa = 1#print "THB Date : \t",line

 if attrValue == 2 :
       blaa = 1#print "Location : \t",line

 else:
       attributes = False
       inAttr = False


 attrValue=attrValue+1;
```

4.2.3 Arvon päivittäminen SIB-palvelimelle

Arvon päivittäminen tehdään erillisessä funktiossa, joka myöhemmässä tuotantovaiheessa tulee olemaan erillisessä tiedostossa. Esimerkkinä päivitetään lämpötila. Funktion määrittelyssä määritellään, että funktio saa parametrina yhden muuttujan *temppi*, jonka nimi tulee englanninkielisestä lämpötilaa tarkoittavasta sanasta *temperature*. Tämä arvo on se lämpötilan arvo, joka päivitysfunktiolle lähetettiin kappaleessa 4.2.2 .

```
def updateTemp(temppi):
```

Päivittämiseen käytetään samaa *httplib*-kirjastoa, kuin tiedon hakemiseenkin. Aluksi luodaan muuttujia, joita käytetään sekä yhteyden käsittelyssä, että tiedon muotoilussa.

```
request = "?s=t&q=mww:dew mww:temp ?t"
headers = {}

lines = []
lines2 =[]

conn = httplib.HTTPConnection("212.213.221.65:8080")
url = "http://212.213.221.65:8080/query?s=t%20h&q=mww%3Adew
%20mww%3Atemp%20%3Ft."
```

Request-muuttuja sisältää HTML-osoitteen osan, joka määrittelee palvelintyypin tarvitsevia muuttujia. Palvelintyyppi ottaa kaikki komennot vastaan suoraan HTML-osoitteessa. *Mww* viittaa nimiavaruuteen ja *dew* muuttujaryhmään. *Temp* viittaa lämpötilaan, joka on muuttuja valitussa muuttujaryhmässä. Muuttujat *lines* ja *lines2* ovat taulukoita, joihin tallennetaan rivejä. *Conn* on samanlainen *httplib*-kirjaston yhteys, jota käytettiin myös Json-tiedoston noutamiseen. Tällä kertaa yhteyteen määritellään myös portti 8080, joka on perus http portti. *Url*-muuttuja sisältää koko osoitteen, myös *request* osan, joka on koodattu.

```
conn.request("POST", url, request, headers)

response = conn.getresponse()
data = response.read()
conn.close()
```

Yhteystyyppinä on taas POST , sillä palvelimelle täytyy lähettää tieto siitä, mitä halutaan tehdä. Palvelimelta saatu vastaus tallennetaan *httpResponse* muuttujaan *response*, jonka jälkeen se luetaan jäsenfunktionsa kautta merkkijonoon data. Vastaus on HTML-lähdekoodia. Haluttu tieto löytyy *<tt>* - tagien välistä.

```
<html>
  <head></head>
<body>
Query results: <font color="green"><i>m3:Success</i></font>
<p /><hr />
<!-- start results -->
<pre><tt>[
  ( "22,0"^^xsd:string )
]
</tt></pre>
<!-- end results -->
<hr />
Go to page
<a href="namespaces">Namespaces</a>,
<a href="update">Update</a>,
<a href="find">Find</a>,
<a href="query">Query</a>,
<a href="sibmgr">SIB manager</a>.
<p />
You can also <a href="download">download</a> or <a
href="upload">upload</a> a snapshot.

</body>
</html>
```

Data käydään läpi samanlaisella silmukkarakenteella kuin saatu Json-tiedosto luvussa 4.2.2 .

```
line=""
for var in data:
```

```
if var != '\n':
      line += var
if var=="\n":

      if line != " " and line != "":
            lines.append(line)
      line = ""
```

Saatua tietoa käydään taas läpi kunnes vastaan tulee rivinvaihto. Kun rivit on tallennettu listaan, käydään lista läpi rivi kerrallaan, ja poimitaan kaikki sulkuja sisältävät rivit toiseen listaan.

```
for line in lines:

 if (line.find("(")!=-1 or (line.find(")")!=-1:
      lines2.append(line)
```

Rivit käydään uudestaan läpi toisesta listasta. Riveistä etsitään lainausmerkki ja kirjain g. Koska vastauksen muoto on aina sama, tiedetään, että sulkuja sisältävillä riveillä, merkistä ” merkkiin g, on haluttu tieto. Merkkijono katkaistaan merkkien kohdalta, ja merkitään arvojen olemassaoloa seuraava muuttuja todeksi.

```
valueExists = 0

for line in lines2:

 found1 = line.find('"')
 found2 = line.find("g")
 erasetemp = line[found1:found2+1]
 valueExists = 1
```

Haluttu arvo tallennettiin muuttujaan *erasetemp*, jota käytetään nimensä mukaisesti poistamaan lämpötila. Palvelimelta pitää poistaa vanhat arvot ennen uusien lisäämistä. Kun muuttuja on tallennettu, luodaan uusi yhteys palvelimeen täysin samalla tavalla, ainoastaan *url*:n pääte on eri.

```
url = "http://212.213.221.65:8080/update?remove=mww%3Adew
%20mww%3Atemp%20"+erasetemp+"%20."
```

Osoitteessa on nyt *remove*-parametri, joka tarvitsee muuttujan ja sen arvon, jotka tallennettiin *erasetemp* -muuttujaan. Käytetään samoja komentoja kuin aikaisemminkin, vastaus on tällä kertaa pelkkä vahvistusviesti, joten sitä ei tarvita.

Arvojen päivitys tehdään myös aivan samalla tavalla, mutta vaihdetaan taas osa *url*:n lopusta.

```
url = "http://212.213.221.65:8080/update?insert=mww%3Adew
%20mww%3Atemp%20%22"+temppi+"%22%20."
```

Osoitteessa on nyt funktion saama *temppi*-muuttuja, joka on sensoreilta saatu ja parsittu lämpötila. Paluuviesti on taas pelkkä vahvistus. Loppuun asti hiotuissa sovelluksissa paluuviestejä kannattaisi käyttää toimenpiteiden vahvistamiseen, mutta tämä sovellus on vielä tutkimusasteella.

5 MAKEFILE- JA SCRIPTI – TIEDOSTOT

Kun sovellus on valmis, se on siirrettävä laitteeseen ja suoritettava. Makefile-tiedoston avulla sovellus käännetään ja siirretään laitteeseen kappaleessa 2 esitetyllä tavalla. Sovellus käynnistetään lisäämällä käynnistyskomento käynnistys-scripti tiedostoon *rwsf.startup.py*. Makefile on kokonaisuudessaan liitteenä 4 ja käynnistys-scripti liitteenä 5.

5.1 Makefile-tiedosto

Makefile-tiedosto hallitsee kaiken sovellusten kääntämisen ja tiedostojen siirtämisen järjestelmän levykuvaan. Sovelluksella on myös oma makefile-tiedosto, jota kutsutaan järjestelmän makefile-tiedoston kautta.

Sovelluksen oma makefile-tiedosto on erittäin yksinkertainen kuten C-sovelluskin, joka sillä käännetään.

```
MeshWorksPython: main.c
 arm-none-linux-gnueabi-gcc main.c -o MeshWorksPython
```

Tiedostossa määritellään aluksi kaikki tiedostot, joita otetaan mukaan. Tässä tapauksessa tiedostoja on vain yksi. Seuraavalla rivillä annetaan itse kääntökomento, joka koostuu useammasta osasta. Ensimmäisenä on kääntäjän komento, joka on armel-kääntäjä. Seuraavana on lähdekooditiedostot, joita on vain yksi. Viimeisenä määritellään tiedostonimi, johon sovellus käännetään.

Järjestelmän levykuvan makefile-tiedosto on huomattavasti monimutkaisempi, mutta perustuu samoihin perustoimintoihin. Tiedoston alussa tehdään muutamia muuttujia, jotta pidempien polkujen ja tiedostonimien käsittely on helpompaa, ja muutosten tekeminen jälkikäteen on helpompaa ja järkevämpää. Tässä dokumentissa käydään läpi vain ne muuttujat, joita tarvitaan tämän sovelluksen kääntämiseen. Koko tiedoston läpikäyminen olisi oman opinnäytetyönsä pituinen dokumentaatio.

```
DIR_3RD_PARTY = $(MYDIR)/../3rd_party_src
MWWPY = MeshWorksPython
MWWPY_ARCHIVE = $(DIR_3RD_PARTY)/$(MWWPY).tar.gz
```

DIR_3RD_PARTY määrittelee kansion, jossa kaikkien asennettavien sovellusten paketit sijaitsevat. *MWWPY* on lyhenne jota käytetään *MeshWorksPython* ohjelmapaketista,

joka sisältää siis tehdyt C- ja Python-sovellukset. *MWWPY_ARCHIVE* määrittelee asennuspaketin ohjelmistopaketille.

Tiedostossa määritellään funktioita, jotka käydään läpi.

```
build_image: \
 _kernel_built \
 _busybox_built \
 _lrzsz_built \
 …
 …
 _mwwpy_built \
```

Funktiot on määritelty myöhemmin tiedostossa, jos saman niminen tiedosto on kansiossa, funktiota ei suoriteta. Seuraavaksi käännetty C-sovellus ja Python-sovelluksen lähdekoodi siirretään tiedostorakenteeseen odottamaan levykuvan luontia.

```
#MeshWorksPython
cp compile/MeshWorksPython/MeshWorksPython/MeshWorksPython
rootfs/bin
cp compile/MeshWorksPython/MeshWorksPython/getdata.py
rootfs/bin
```

Tiedoston lopussa on määritelty funktiot, joita aikaisemmin kutsuttiin.

```
_mwwpy_built: _kernel_built
 mkdir -p compile
 rm -rf compile/MeshWorksPython
 mkdir compile/MeshWorksPython
 cd compile/MeshWorksPython && \
 tar xzf $(MWWPY_ARCHIVE) && \
 cd MeshWorksPython && \
 make
 touch _mwwpy_built
```

Funktion nimen perässä määritellään riippuvuus, *_kernel_built* tiedosto on oltava olemassa, jotta funktio voidaan suorittaa, eli ydin on oltava rakennettu. Funktiossa on perus Linux komentoja. Käydään funktio läpi rivi riviltä. Rivillä yksi luodaan kansio *compile* mikäli sitä ei ole olemassa. Seuraavalla rivillä poistetaan mahdollinen vanha kansio, jossa sovellus on asennettu. Seuraavaksi luodaan sama kansio uudestaan.

Seuraavien rivien komennot on yhdistetty *&&* - merkeillä. Rivit tarvitsee yhdistää, jotta komennot, kuten kansioiden vaihto, pysyvät voimassa seuraavaan komentoon. Aluksi vaihdetaan juuri luotu hakemisto aktiiviseksi, ja puretaan sovelluksen paketti sinne. Paketin sisältä purettiin kansio, jonne siirrytään. Kansiossa suoritetaan *make*-komento. *Make*-komento kutsuu aina siinä hakemistossa olevaa makefile-tiedostoa. Kyseinen tie-

dosto käytiin läpi jo aikaisemmin. Viimeiseksi luodaan tyhjä *_mwwpy_built*-tiedosto, joka merkitsee, että funktio on suoritettu.

5.2 Käynnistys scripti-tiedosto

Koska laitteesta on tarkoitus tulla täysin automatisoitu hallintalaite, sovellusten ohjaaminen on tarkoitus suorittaa scripteillä. Scriptit on kirjoitettu Pythonilla, jolla on helppo kutsua muita Python-scriptejä ja suorittaa järjestelmäkomentoja.

Sovelluksen käynnistys on laitettu käynnistys scripti-tiedostoon *rwsf.startup.py*.

```
import sys
import os
…
…
os.system('MeshWorksPython')
```

Tässä on esitetty vain tärkeimmät kohdat, eli mitkä kirjastot tarvitaan mukaan, ja minkälaisella komennolla sovellus käynnistetään.

6 YHTEENVETO

Sulautetulle laitteelle ohjelmoidessa on otettava huomioon laitteen ominaisuuksien asettamat rajoitteet ja laitteen erikoisominaisuudet. Kyseiselle laitteelle ohjelmoidessa tarvitsi käyttää allekirjoittaneelle uusia tekniikoita ja kieliä. Uusien tekniikoiden käyttö ja opettelu on pitkälti kokeilua ja epäonnistumisia. Makefile-tiedostojen, scriptien ja ARM-kääntäjän käyttäminen olivat kaikki tuntematonta aluetta, mutta varmasti hyödyllisiä tekniikoita. Python-kielen opettelu oli hyvin suoraviivaista, sillä kielen rakenne on yksinkertaisempi kuin koulussa opetetun C++:n.

Kaiken toiminnallisuuden olisi voinut myös toteuttaa ilman Python-osiotakin, mutta laitteelle tehtävien sovellusten yhdenmukaisuuden takia, Python oli valittu yhteisesti käytettäväksi kieleksi. Syy yhteisesti käytettävään kieleen oli sovellusten helppo yhdisteltävyys.

Sovellusten silmukkarakenteen, eli prosessien toistuvuuden, olisi voinut tehdä myös Python-sovelluksen sisällä, mutta käytetty Internet-yhteyksiä hallinnoiva kirjasto kärsii monissa tapauksissa muistivuodoista, joten resurssien säästämiseksi päädyttiin suorittamaan silmukat C-sovelluksella.

Sovellusten toteutus onnistui hyvin. Kaikki sovelluksilta vaadittu toiminnallisuus saatiin toteutettua ja sovellukset asennettua laitteeseen.

Sovellusyhdistelmää tullaan käyttämään ainakin jossain määrin SOFIA-projektissa ja mahdollisesti uuden tuotteen kehityksessä. Sovellusyhdistelmä, tai sen Python-osio saattaa tulla myöhemmin yhdeksi komponentiksi laitteen toimintaa automaatioverkossa. Muun toiminnallisuuden toteuttamisen jälkeen, SOFIA-projektiin on toteutettu mobiilisovellus, joka hakee SIB-palvelimelta tiedot valitusta sensoriryhmästä.

LÄHTEET

Json. Luettu 17.10.2011

http://json.org

URL Encoder/Decoder(URL purkaja/koodaaja), Creative Commons. Luettu 17.10.2011

http://meyerweb.com/eric/tools/dencoder/

Sourcery G++ Lite 2009q1-203 for ARM GNU/Linux, Mentor Graphics 2011, Luettu

17.10.2011 http://www.codesourcery.com/sgpp/lite/arm/portal/release858.

SOFIA-projektin kotisivu, SOFIA 2009, Luettu 26.10.2011
http://www.sofia-project.eu/

Python standard library – Python 2.6.7 Documentation, Python Software foundation

1990-2011, Luettu 26.10.2011

http://docs.python.org/release/2.6.7/library/index.html

Unix -Wikipedia, Wikipedia® . Luettu 18.11.2011

http://fi.wikipedia.org/wiki/Unix

Liite 1. Laitteen käyttöönottodokumentaatio

Beddit Agent SW Ethernetbox Käyttöönotto

(jatkuu)

1 Esiasennettavat ohjelmistot LIITE 1

Näitä ohjelmistoja ei tarvitse laitteen kanssa kommunikointiin, mutta kehitykseen.

- Native C compiler (gcc) and the basic libraries (libc6-dev)
- • patch
- • libglib2.0-dev (alkuperäisessä dokumentissä ibglib, mutta sellaista ei löytynyt)
- • python2.6-dev
- • GNU make
- • Autoconf
- • Automake
- • Libtool
- • Genext2fs
- • U-Boot mkimage (Ubuntu package u-boot-tools)

Ubuntun 11.04 julkaisussa suurin osa ohjelmistoista löytyy valmiiksi perus asennuspaketista tai ovat saatavilla repositorysta.
Repositoryssa olevat paketit pääsee asentamaan, joko synaptic paketinhallinnan kautta tai komentoriviltä komennolla:

```
sudo apt-get install    [ohjelmiston nimi]
                        patch
                        libglib2.0-dev
                        python2.6-dev
                        make
                        autoconf
                        automake
                        libtool
                        u-boot-tools
                        genext2fs
```

Ubuntu kysyy tässä tilanteessa root salasanaa, koska komento on ajettu pääkäyttäjänä.

(jatkuu)

Lisäksi jos haluaa asentaa omia sovelluksiaan tai laittaa alkuperäisen järjestelmän laitteeseen tarvitsee kääntää levykuva eli image. Imagen kääntämistä varten tarvitsee asentaa kääntäjät armel arkkitehtuurille. Käännöspaketti tai toolchain, kuten siihen alkuperäisessä dokumentissa viitataan, on ladattavissa osoitteesta:

http://www.codesourcery.com/sgpp/lite/arm/portal/release858

Ubuntulle kannattaa ladata GNU Linux installer, joka on muotoa *.bin. Kun tiedosto on ladattu sille pitää antaa suoritusoikeudet, joko ubuntun graafisesta käyttöliittymästä tai konsolista. Graafisesti : klikkaa tiedostoa hiiren oikealla painikkeella ja valitse properties (ominaisuudet) , ja sieltä välilehti permissions (käyttöoikeudet/luvat). Sieltä valitse aktiiviseksi checkbox “Allow executing file as program” (Salli suorittaminen sovelluksena tms.) .

Komentoriviltä : Mene kansioon, jossa tiedosto on esim.

cd /home/gjordis/downloads

suorita komento:

chmod +x arm-2009q1-203-arm-none-linux-gnueabi.bin

tiedoston nimi voi muuttua uudemmissa versioissa toolchainista.

Nyt voit ajaa asennusohjelman komentoriviltä. Jos suljit konsolin mene takaisin latauskansioon. Aja komentoriviltä komento:

./arm-2009q1-203-arm-none-linux-gnueabi.bin

Sovelluksella on graafinen asennusohjelma. Asennusohelma käy läpi perusvaiheet:

-hyväksy ehdot

-valitse asennettavat komponentit (oletus asennus on riittävä)

-valitse asennuskansio (oletus kelpaa tässäkin, jos ei syytä muuttaa)

-lisätäänkö PATHiin? (kyllä)

-valitse linkityskansio (oletus kelpaa tässäkin)

Asennusohelman pitäisi onnistua lisäämään tarvittavat komennot Ubuntun PATHiin, eli tehdä niistä suoraan kutsuttavia ilman että tarvitsee olla suoritettavan sovelluksen kan

(jatkuu)

siossa. Jos tämä ei onnistunut, imagen rakentaminen ei onnistu. Käydään siis läpi kansioiden lisääminen manuaalisesti.
Avaa tiedosto bashrc komennolla:

gedit ~/.bashrc

Tiedosto aukeaa graafisen liittymän tekstieditoriin, tässä voi käyttää myös konsolipohjaisia tekstieditoreita geditin sijasta, esim. Nano, vi tai jed, jos nämä ovat asennettuna. Kirjoita tiedoston perään rivi muotoa:

export PATH="$PATH:/home/gjordis/CodeSourcery/Sourcery_G++_Lite/bin"

Lisäämme siis kansion bin asennusohelmassa valitun linkityskansion sisältä ympäristömuuttujaan PATH. Alleviivattu osa voi siis olla eri sen mukaan, mitä asennusohjelmassa valitsi.

1.2 Beddit agent

Imagen rakentamista varten tarvitsee Bedditin ohjelmistot, jotka ainakin tällä hetkellä saa osoitteesta

http://dl.dropbox.com/u/6336829/Finsor_BedditAgent2_System.tar.gz

Osoite saattaa vanhentua tai muuttua.
Pura paketti esimerkiksi kotihakemustoon. Mene komentoriviltä purettuun hakemistoon, joka sisällä system/build hakemistoon.

cd Finsor_BedditAgent2_System/system/build/

Tässä hakemistossa suorittamalla komennon make image rakentuu itsestään hakemistossa olevien scriptien ja makefilen perusteella. Näitä muokkaamalla siis imageen saa esimerkiksi lisää sovelluksia.

Imagen luonti kestää useita minuutteja, noin 15min i3 suorittimella ja normaalilla HDD levyllä.

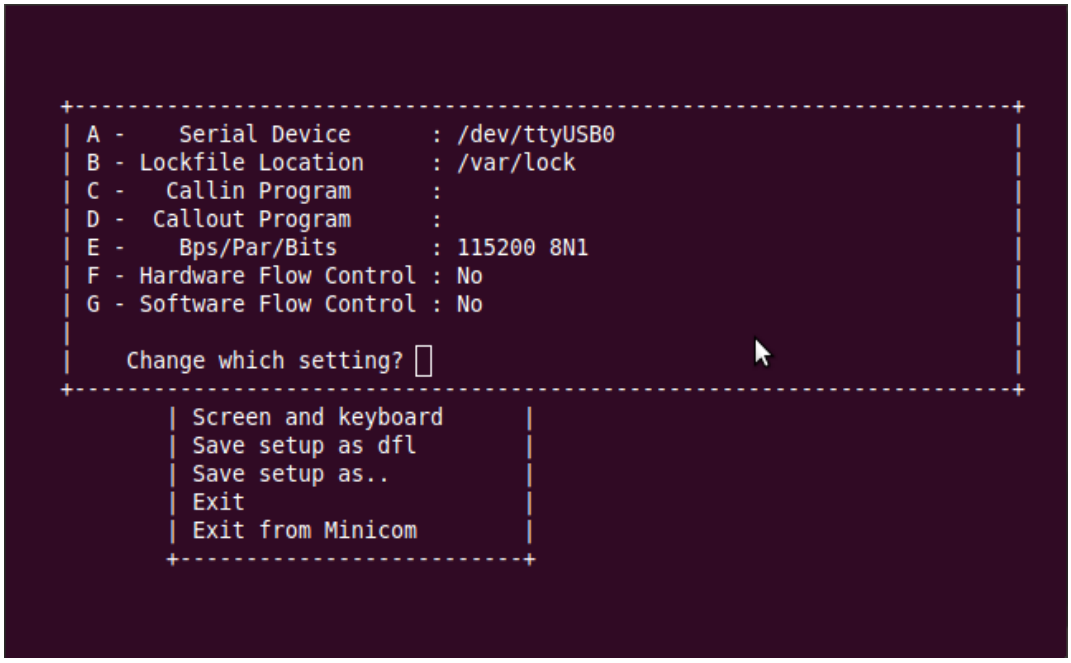
(jatkuu)

Laitteeseen yhdistetään sarjaporttiterminaalilla, hyvä ohjelma on minicom, jonka saa ladattua apt-get komennolla.

Käynnistä minicom ensimmäistä kertaa komennolla

minicom -s

Tämä on parametri asetuksille. Asetukset :



Kuva 4: Minicom asetukset

laite dev/ttyUSB0 viittaa ensimmäiseen usb portissa olevaan sarjaporttiadapteriin. Komennoilla

lsusb

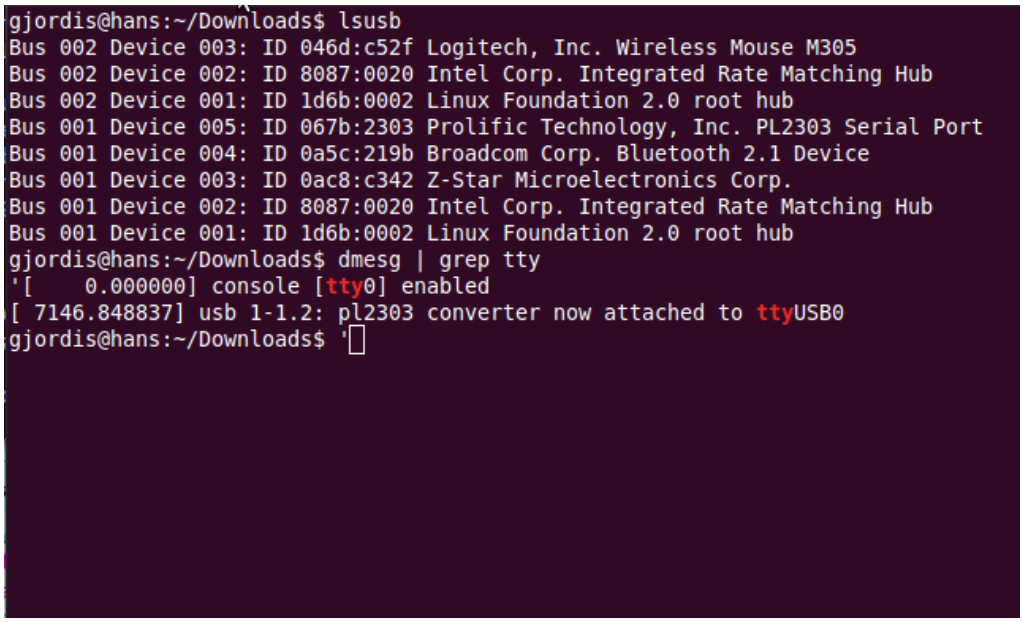
dmesg | grep tty

voit selvittää missä portissa sarjaporttiadapteri on. Lsusb ei ole pakollinen komento, mutta sillä on hyvä tarkastaa liittimen olemassaolo, esimerkiksi jos alempi komento ei

(jatkuu)

anna tuloksia. Komennot voi suorittaa toisessa konsoli-ikunnassa minicomin ollessa auki toisessa.

Laita muuten kuvan mukaiset asetukset. Lisäksi asetukset ikkunasta → Screen and Keyboard → valinta local echo → no. Muuten kirjoittaminen konsolilla on vaikeaa, sillä kaikki kirjaimet kaikuvat takaisin samantien. (sudo make me a sadwich = ssuuddoo mmaakkee mmee aa ssaannddwwiicchh)



Kuva 5: Sarjaportin selvittäminen

Tämän jäkeen minicomin voi käynnistää uudelleen ilman -s parametria. Minicom yhdistää automaattisesti käynnistyessään.

3 Wlan asetukset (ilmeisesti ei toimi kuitenkaan)

WLAN kaiketi ei lähde automaattisesti käyntiin, eikä myöskään hae osoitteita automaattisesti.

Suorittamalla seuraavat komennot, boxin saa kiinni wlaniin.

(jatkuu)

LIITE 1

- # ifconfig wlan0 up (Komento käynnistää sovittiminen, wlan0 on ensimmäinen wlan laite koneessa, järjestysluku voi olla muukin)
- # iwconfig wlan0 essid AndroidAP (annetaan sovittimille wlan verkon nimi, ei suojausta nyt)
- # ifconfig wlan0 192.168.43.233 netmask 255.255.255.0 (määritellään ip osoita manuaalisesti)
- #route add default gw 192.168.43.1
- http://wirelessdefence.org/Contents/LinuxWirelessCommands.htm
- http://www.ghacks.net/2009/04/14/connect-to-a-wireless-network-via-command-line/

Suojauksen pystyisi laittamaan, mutta laite tuntuu tukevan vain hexa avaimia, ja kyseisessä verkossa oli alphanumeerinen salausavain, joten otin vain salauksen pois käytöstä.

4 Levykuvan tekeminen

Uusi system.img eli laitteen levykuva voidaan rakentaa Finsor_BedditAgent2_System:llä. Levykuva tehdään menemällä hakemistoon

/home/gjordis/Finsor_BedditAgent2_System/system/build

Jossa /home/gjordis viittaa hakemistoon minne työkalut on purettu. Hakemistoon pitää navikoida konsolilla. Esimerkiksi Ubuntussa konsoli on oletuksena kotihakemistossa, joten komento:

cd Finsor_BedditAgent2_System/system/build

vie oikeaan hakemistoon. Komento make, luo asetusten mukaisen levykuvan laitetta varten.

Liite 2. C-Lähdekoodi

```
#include <stdlib.h>
#include <stdio.h>

int main()
{
 while(1)
 {
       system("python getdata.py");
       sleep(10);
 }
}
```

Liite 3. Python-Lähdekoodi

```
##########################################################################################
##
######## Harri Hietaranta , Gjordis@gmail.com , Software developed for MeshWorksWire-
less##
######## August 10th , 2011
##########################################################
##########################################################################################
##

##########################################################################################
##
###################################UPDATETEMP#############################################
##

def updateTemp(temppi):

        request = "?s=t&q=mww:dew mww:temp ?t"
        ## No headers ##
        headers = {}

        #two lists for storing lines
        lines = []
        lines2 =[]

        #connect to the url including the query for temp in mww:dew mww:temp
        conn = httplib.HTTPConnection("212.213.221.65:8080")
        url = "http://212.213.221.65:8080/query?s=t%20h&q=mww%3Adew%20mww%3Atemp%20%3Ft."

        try:

                conn.request("POST", url, request, headers)

                response = conn.getresponse()
                data = response.read()
                #print data
                conn.close()

                line=""
                for var in data:

                        if var != '\n':
                                line += var

                        if var=="\n":

                                #for string in unwanted_characters :
                                        #line = line.strip(string)
                                        #line = line.rstrip(string)

                                #hipsu = line.find('"')
                                #newline = line[hipsu+1:]
                                #line = newline
                                #print line
                                #hipsu = line.find('"')
                                #print hipsu
                                #print len(line)
                                #poikki = (len(line)-hipsu)
                                #print poikki
                                #newline = line[:-poikki]
                                if line != " " and line != "":
                                        lines.append(line)
                                line = ""

                for line in lines:

                        if (line.find("("))!=-1 or (line.find(")"))!=-1:
                                lines2.append(line)
                valueExists = 0

                for line in lines2:

                        found1 = line.find('"')
                        found2 = line.find("g")
                        erasetemp = line[found1:found2+1]


                        valueExists = 1
```

(jatkuu)

LIITE 3

```
                #print lines2;
        except: print "MOOOOOOOOOM"


        request = "?s=t&q=mww:dew mww:temp ?t"
        ## No headers ##
        headers = {}

        if valueExists == 1:

                conn = httplib.HTTPConnection("212.213.221.65:8080")
                url = "http://212.213.221.65:8080/update?remove=mww%3Adew%20mww%3Atemp
%20"+erasetemp+"%20."

                try:

                        conn.request("POST", url, request, headers)

                        response = conn.getresponse()
                        data = response.read()
                        #print data
                        conn.close()

                except: print "MOOOOOOOOOM"

        request = "?s=t&q=mww:dew mww:temp ?t"
        ## No headers ##
        headers = {}

        conn = httplib.HTTPConnection("212.213.221.65:8080")
        url = "http://212.213.221.65:8080/update?insert=mww%3Adew%20mww%3Atemp
%20%22"+temppi+"%22%20."

        try:

                conn.request("POST", url, request, headers)

                response = conn.getresponse()
                data = response.read()
                #print data
                conn.close()

        except: print "MOOOOOOOOOM"


################################## END UPDATETEMP
################################################
################################## UPDATEHUM
################################################

def updateHum(humidity):

        request = "?s=t&q=mww:dew mww:temp ?t"
        ## No headers ##
        headers = {}


        #two lists for storing lines
        lines = []
        lines2 =[]

        #connect to the url including the query for temp in mww:dew mww:temp
        conn = httplib.HTTPConnection("212.213.221.65:8080")
        url = "http://212.213.221.65:8080/query?s=t%20h&q=mww%3Adew%20mww%3Ahum%20%3Ft."

        try:

                conn.request("POST", url, request, headers)

                response = conn.getresponse()
                data = response.read()
                #print data
                conn.close()
```

(jatkuu)

LIITE 3

```
                line=""
                for var in data:      # gather raw data into lines

                        if var != '\n':
                                line += var

                        if var=="\n":

                                if line != " " and line != "":
                                        lines.append(line)
                                line = ""

                for line in lines: #just take lines with ()

                        if (line.find("(")!=-1 or (line.find(")")!=-1:
                                lines2.append(line)
                valueExists = 0

                for line in lines2: #parse into better form

                        found1 = line.find('"')
                        found2 = line.find("g")
                        erasehum = line[found1:found2+1]
                        valueExists = 1

                #print lines2;
        except: print "MOOOOOOOOOM"


        request = "?s=t&q=mww:dew mww:temp ?t"
        ## No headers ##
        headers = {}

        if valueExists == 1:
                #connection with erase address to erase old value from server
                conn = httplib.HTTPConnection("212.213.221.65:8080")
                url = "http://212.213.221.65:8080/update?remove=mww%3Adew%20mww%3Ahum
%20"+erasehum+"%20."

                try:

                        conn.request("POST", url, request, headers)

                        response = conn.getresponse()
                        data = response.read()
                        #print data
                        conn.close()

                except: print "MOOOOOOOOOM"

        request = "?s=t&q=mww:dew mww:temp ?t"
        ## No headers ##
        headers = {}
        #connection with insert address to update new value
        conn = httplib.HTTPConnection("212.213.221.65:8080")
        url = "http://212.213.221.65:8080/update?insert=mww%3Adew%20mww%3Ahum%20%22"+hu-
midity+"%22%20."

        try:

                conn.request("POST", url, request, headers)

                response = conn.getresponse()
                data = response.read()
                #print data
                conn.close()

        except: print "MOOOOOOOOOM"

################################## END UPDATEHUM
##################################################
#!/usr/bin/env python
```

(jatkuu)

LIITE 3

```
import httplib, urllib, sys ,time , SIBpy


## Login data ##

user = "tamk2"
passwd = "zigbee"
action = "getdata"

## Request fields ##
identification = '"identification": { "user": "'+user+'", "pass": "'+passwd+'",
"action": "'+action+'" }'
deviceFilter = '"device_filter": { "mwwid": [ "3" ] }'
#attributeFilter = '"attribute_filter": { "data": [ "devname", "meas" ], "meas":
[ "temp" ] }'
deviceFilter = '"device_filter": { }'

#request = '{ "request": { '+identification+' , '+deviceFilter+' , '+attributeFilter+' }
}'
request = '{ "request": { '+identification+' } }'
## No headers ##
headers = {}
nro = 0;
################ VARIABLES FOR GETTING Json
#############################################

conn = httplib.HTTPConnection("www1.retailhosting.fi")
#url = "http://109.204.225.241/wsn.interface/JsonIfaceV1.jsoniface"
url = "http://seemoto.mww.fi/interface/JsonIfacev1.jsoniface"
unwanted_characters = (' ', '"val":', '\n',",")
lines = []
print("GG")


#################### CONNECTION PART , GET Json FILE
#####################################
try:
        conn.request("POST", url, request, headers)
        file = open("data.txt", "w")
        response = conn.getresponse()
        data = response.read()
        file.write(data)
        file.close()
        conn.close()
        nro = nro+1
        print("GG")

        ## Check if verbose mode ##
        try:
            if(sys.argv[1] == "-v" or sys.argv[1] =="-V"):
                print data

            if(sys.argv[1] == "-l" or sys.argv[1] =="-L"):
                print "#",nro
        except:
            pass

        line=""
        for var in data:

                if var != '\n':
                        line += var

                if var=="\n":

                        #for string in unwanted_characters :
                                #line = line.strip(string)
                                #line = line.rstrip(string)
                        if line != " " and line != "":
                                lines.append(line)
                        line = ""

################################END CONNECTION PART ###################################
################################STARTING PARSING PART###################################
```

(jatkuu)

LIITE 3

```
        deviceIndex = -1 # Initialize parsing variables
        attributes = False
        inAttr = False



        values = 1



        attribute = 0
        attrValue = 0

        for line in lines:

                if (line.find("{"))!=-1 or (line.find("}"))!=-1:
                        line=""

                #found = line.find("data")>0

                elif line.find("data") >0:

                                #devices.push_back(Device())      #push new device to vector
                                deviceIndex = deviceIndex+1       #increase the indexing int
for the vector by 1;
                                values = 1                      #initialize the tracking int of
which value in question;
                                attributes = False #set false, meaning not inside attr brac-
kets
                                attribute = 0   #set attribute indexing int to 0
                                attrValue = 1   #set value inside attribute brackets indexing
int to 1;
                                inAttr = False  #set false, meaning not inside attr inside
attr

                #found = line.find("attr")

                elif (line.find("attr") != (-1)):

                        if attributes==True and lastWasAttr==False:  #if one set of attr
has been found and it has contained values

                                inAttr = True                           #true, that we are in-
side brackets inside brackets
                                attrValue = attrValue+1                 #increase the inde-
xed value



                        elif attributes == False:           #if there has been no attr

                                attributes = True          #now there has been
                                attribute = attribute+1    #increase index monitorin of
which attribute in question
                                attrValue = 0              #new attribute, reset value
                                inAttr = False             #not inside brackets inside
brackets
                                lastWasAttr = True         #this one was a attr, its the
last one to the next one


                        elif lastWasAttr==True:             #and if there was a attr, inc-
rease monitoring int

                                attribute = attribute+1

                #found = line.find("val")

                elif (line.find("val")!=-1 and attributes==False):   # if we get "val"
and are not inside attr

                        unwanted_characters = (' ', '"val":', '\n', ' ')
```

(jatkuu)

```
                        for string in unwanted_characters :
                                line =line.strip(string)
```

LIITE 3

```
                        #print line
                        update = line[1:-2]
                        #print update



                        if values == 1:
                                blaa = 1#print "Device : ",deviceIndex," Asset Id :
\t",line


                        elif values == 2:
                                blaa = 1#print "Device : ",deviceIndex," Asset Name :
\t",line
                        elif values == 3:
                                blaa = 1#print "Device : ",deviceIndex," Device Name :
\t",line
                        elif values == 4:
                                blaa = 1#print "Device : ",deviceIndex," ZigBee ID :
\t",line
                        elif values == 5:
                                blaa = 1#print "Device : ",deviceIndex, "MWW ID : \t",line
                        elif values == 6:
                                blaa = 1#print "Device : ",deviceIndex, "Address : \t",line


                        values = values+1 # increase monitoring value
                #found = line.find("val")
                elif ((line.find("val")!=-1) and attributes==True):    # if there is
"val" and we are inside attr


                        unwanted_characters = (' ', '"val":', '\n', ' ')

                        for string in unwanted_characters:
                                line =line.strip(string)

                        if(inAttr == False): #if inside attr but not inside atrr-atrr


                         # set right attribute to class representing current device


                                update = line[1:-3]

                                if attribute == 1:#Temp
                                        #print "Temperature : \t",line
                                        updateTemp(update)
                                        lastWasAttr = False


                                elif attribute == 2:#Humidity
                                        #print "Humidity : \t",line
                                        lastWasAttr = False

                                elif attribute == 3:#Battery voltage
                                        #print "Battery : \t",line
                                        lastWasAttr = False

                                elif attribute == 4:#Co2
                                        #print "CO2 : \t\t",line
                                        lastWasAttr = False

                                elif attribute == 5:#light
                                        #print "Light : \t",line
                                        lastWasAttr = False

                                elif attribute == 6:#rssirx
                                        #print "RssiRx : \t",line
                                        lastWasAttr = False

                                elif attribute == 7:#rssitx
```

(jatkuu)

LIITE 3

```
                                #print "RssiTx : \t",line
                                lastWasAttr = False

                        elif(inAttr == True): #if inside inner attr

                                if attrValue == 1 :
                                        blaa = 1#print "THB Date : \t",line




                                if attrValue == 2 :
                                        blaa = 1#print "Location : \t",line

                                else:
                                    attributes = False        #nothing to insert into any


device, meaning empty attribute
                                    inAttr = False    #just mark booleans to know that
attribute has ended

                                attrValue=attrValue+1;  #//increase the attrvalue represen-
ting either date or location

############################### END OF PARSING PART
#################################################################
except:
  print "Program execution failed for some reason, most likely for missing Internet con-
nection"


############################### START OF SIB UPLOAD PART
#################################################################
```

Liite 4. Laitteen Makefile-tiedosto

```
#
# Makefile for the Beddit Agent root file system
#

CROSS_HOST = arm-none-linux-gnueabi
CROSS_COMPILE = $(CROSS_HOST)-
ARCH=arm

STRIP = $(CROSS_COMPILE)strip

GENEXT2FS = genext2fs
MKIMAGE = mkimage

BUILD_ARCH=$(shell uname -m)

ROOTFS_EXT2_INODES=1024
ROOTFS_EXT2_BLOCKS=32768

MYDIR = $(shell pwd)

KERNEL_SRC = $(MYDIR)/../linux-2.6.35.3

#KERNEL_CONFIG = imx28evk_defconfig
KERNEL_CONFIG = beddit_agent_defconfig

DIR_3RD_PARTY = $(MYDIR)/../3rd_party_src

BUSYBOX=busybox-1.17.4
BUSYBOX_ARCHIVE = $(DIR_3RD_PARTY)/$(BUSYBOX).tar.bz2
LRZSZ=lrzsz-0.12.20
LRZSZ_ARCHIVE = $(DIR_3RD_PARTY)/$(LRZSZ).tar.gz
DROPBEAR=dropbear-0.52
DROPBEAR_ARCHIVE = $(DIR_3RD_PARTY)/$(DROPBEAR).tar.gz
LIBNL=libnl-1.1
LIBNL_ARCHIVE=$(DIR_3RD_PARTY)/$(LIBNL).tar.gz
IW=iw-0.9.21
IW_ARCHIVE=$(DIR_3RD_PARTY)/$(IW).tar.bz2
WIRELESS_TOOLS=wireless_tools.30
WIRELESS_TOOLS_ARCHIVE=$(DIR_3RD_PARTY)/wireless-tools_30~pre9.orig.tar.gz
WPA_SUPPLICANT=wpa_supplicant-0.6.10
WPA_SUPPLICANT_ARCHIVE=$(DIR_3RD_PARTY)/$(WPA_SUPPLICANT).tar.gz
ETHTOOL=ethtool-3
ETHTOOL_ARCHIVE=$(DIR_3RD_PARTY)/$(ETHTOOL).tar.gz
PPP=ppp-2.4.4
PPP_ARCHIVE = $(DIR_3RD_PARTY)/$(PPP).tar.gz
PYTHONVER=2.6.6
PYTHONVER1=2.6
PYTHON=Python-$(PYTHONVER)
PYTHON_ARCHIVE=$(DIR_3RD_PARTY)/$(PYTHON).tar.bz2
PYTHON_PATCH = $(DIR_3RD_PARTY)/Python-2.6.6-xcompile.patch
PYTHON_SETUP_PATCH = $(DIR_3RD_PARTY)/python-2.6.6-setup.py.diff
PYTHONSERIAL=pyserial-2.5
PYTHONSERIAL_ARCHIVE=$(DIR_3RD_PARTY)/$(PYTHONSERIAL).tar.gz
OPENSSL=openssl-0.9.8q
OPENSSL_ARCHIVE = $(DIR_3RD_PARTY)/$(OPENSSL).tar.gz
PYNETLINUX=pynetlinux-1.0
PYNETLINUX_ARCHIVE=$(DIR_3RD_PARTY)/$(PYNETLINUX).tar.gz
ZLIB=zlib-1.2.5
ZLIB_ARCHIVE=$(DIR_3RD_PARTY)/$(ZLIB).tar.bz2
NTP=ntp-4.2.6p2
NTP_ARCHIVE=$(DIR_3RD_PARTY)/$(NTP).tar.gz
E2FSPROGS=e2fsprogs-1.41.13
E2FSPROGS_ARCHIVE=$(DIR_3RD_PARTY)/$(E2FSPROGS).tar.gz
NUMPY=numpy-1.4.1
NUMPY_ARCHIVE=$(DIR_3RD_PARTY)/$(NUMPY).tar.gz
NUMPY_PATCH=$(DIR_3RD_PARTY)/numpy-1.4.1-cross1.patch
NUMPY_BUILD_SCRIPT=$(DIR_3RD_PARTY)/numpy-build.sh
MWWPY = MeshWorksPython
MWWPY_ARCHIVE = $(DIR_3RD_PARTY)/$(MWWPY).tar.gz
DEVICEREAD=deviceread
DEVICEREAD_ARCHIVE=$(DIR_3RD_PARTY)/$(DEVICEREAD).tar.gz

FINSOR_PYTHONLIB_DIR=$(MYDIR)/../../../Finsor_PythonLib
```

(jatkuu)

LIITE 4

```
PYTHON_MODULES =  abc.py atexit.py base64.py codecs.py \
               collections.py ConfigParser.py copy.py copy_reg.py \
               decimal.py fnmatch.py functools.py genericpath.py \
               getopt.py glob.py hashlib.py httplib.py keyword.py \
               linecache.py md5.py mimetools.py numbers.py os.py \
               pickle.py posixfile.py posixpath.py pty.py quopri.py \
               random.py rfc822.py re.py shutil.py site.py socket.py \
               stat.py sre_compile.py sre_constants.py \
               sre_parse.py ssl.py string.py StringIO.py struct.py \
               subprocess.py tempfile.py textwrap.py threading.py \
               traceback.py types.py unittest.py urllib.py \
               urlparse.py UserDict.py warnings.py zipfile.py \
               _abcoll.py __future__.py

PYTHON_SO_MODULES = array.so cPickle.so cStringIO.so \
               datetime.so fcntl.so itertools.so _json.so math.so \
               operator.so resource.so select.so syslog.so \
               termios.so time.so \
               _collections.so _ctypes.so _functools.so \
               _md5.so _random.so _sha.so _sha256.so \
               _sha512.so _socket.so _struct.so

build_image: \
       _kernel_built \
       _busybox_built \
       _lrzsz_built \
       _libnl_built \
       _iw_built \
       _wireless_tools_built \
       _wpa_supplicant_built \
       _openssl_built \
       _python_built \
       _ntp_built \
       _e2fsprogs_built \
       _extadc_driver_built\
       _intadc_driver_built \
       _hsadc_mic_driver_built \
       _numpy_built \
       _dropbear_built \
       _mwwpy_built \
       _deviceread_built \
       devtable.txt \
       conf-files/fstab conf-files/group conf-files/inittab \
       conf-files/passwd conf-files/rcS conf-files/services \
       conf-files/fstab conf-files/hosts conf-files/host.conf \
       conf-files/UTC

       rm -rf rootfs
       mkdir rootfs

       mkdir rootfs/dev rootfs/bin rootfs/lib rootfs/lib/modules \
              rootfs/share rootfs/share/udhcpc \
              rootfs/usr rootfs/etc rootfs/etc/init.d \
              rootfs/usr/local rootfs/debug rootfs/lib/firmware

       ln -s /lib rootfs/usr/lib
       ln -s bin rootfs/sbin
       ln -s /bin rootfs/usr/bin
       ln -s /bin rootfs/usr/sbin
       ln -s /share rootfs/usr/share
       ln -s UTC rootfs/etc/localtime
       ln -s /proc/mounts rootfs/etc/mtab
       ln -s bin/busybox rootfs/init
       ln -s bin/busybox rootfs/linuxrc
       ln -s lsz rootfs/bin/sz
       ln -s lrz rootfs/bin/rz
       ln -s /tmp/resolv.conf rootfs/etc/resolv.conf

       cp conf-files/UTC conf-files/fstab \
              conf-files/group conf-files/host.conf \
              conf-files/hosts conf-files/inittab \
              conf-files/passwd conf-files/services \
              conf-files/nsswitch.conf \
                     rootfs/etc
```

(jatkuu)

LIITE 4

```
	cp conf-files/rcS rootfs/etc/init.d
	cp conf-files/default.script rootfs/share/udhcpc

	# drivers
	cp ../ext-adc-driver/extadc.ko rootfs/lib/modules
	cp ../int-adc-driver/intadc.ko rootfs/lib/modules
	cp ../hsadc-mic-driver/hsadc_mic.ko rootfs/lib/modules

	cp ../startup/rwfs-startup.py rootfs

	sed "s/<<BUILDDATE>>/`date -u +%Y-%m-%dT%H:%M:%SZ`/" conf-files/rcS
>rootfs/etc/init.d/rcS

	# Finsor config
	mkdir -p rootfs/etc/finsor/defconfig
	cp ../default-config/* rootfs/etc/finsor/defconfig

	cp compile/busybox/$(BUSYBOX)/busybox rootfs/bin

	cp compile/lrzsz/$(LRZSZ)/src/lrz \
		compile/lrzsz/$(LRZSZ)/src/lsz rootfs/bin
	$(STRIP) rootfs/bin/lrz rootfs/bin/lsz

	#cp compile/ppp/$(PPP)/pppd/pppd \
	#	compile/ppp/$(PPP)/chat/chat rootfs/bin
	#$(STRIP) rootfs/bin/pppd rootfs/bin/chat

	# ethtool
	#cp compile/ethtool/$(ETHTOOL)/ethtool rootfs/bin
	#$(STRIP) rootfs/bin/ethtool

	# Wireless tools
	cp compile/wireless-tools/$(WIRELESS_TOOLS)/iwconfig rootfs/bin
	cp compile/wireless-tools/$(WIRELESS_TOOLS)/iwlist rootfs/bin
	cp compile/wireless-tools/$(WIRELESS_TOOLS)/iwpriv rootfs/bin

	# libnl, iw
	cp compile/libnl/$(LIBNL)/lib/libnl.so.1.1 rootfs/lib
	$(STRIP) --strip-debug rootfs/lib/libnl.so.1.1
	ln -s libnl.so.1.1 rootfs/lib/libnl.so.1
	ln -s libnl.so.1.1 rootfs/lib/libnl.so

	cp compile/iw/$(IW)/iw rootfs/bin

	# WPA supplicant
	(cd compile/wpa-supplicant/$(WPA_SUPPLICANT)/wpa_supplicant && cp wpa_supplicant
wpa_cli $(MYDIR)/rootfs/bin)
	$(STRIP) rootfs/bin/wpa_supplicant rootfs/bin/wpa_cli

	# Dropbear
	cp compile/dropbear/dropbear-0.52/dropbearmulti rootfs/bin
	ln -s dropbearmulti rootfs/bin/dropbear
	ln -s dropbearmulti rootfs/bin/dropbearkey
	ln -s dropbearmulti rootfs/bin/scp
	ln -s dropbearmulti rootfs/bin/dbclient

	#deviceread
	cp compile/deviceread/deviceread/deviceread rootfs/bin

	#MeshWorksPython
	cp compile/MeshWorksPython/MeshWorksPython/MeshWorksPython rootfs/bin
	cp compile/MeshWorksPython/MeshWorksPython/getdata.py rootfs/bin

	# Dropbear host key
	mkdir rootfs/etc/dropbear
	cp conf-files/dropbear_rsa_host_key rootfs/etc/dropbear

	# ntpd
	cp compile/ntp/$(NTP)/ntpd/ntpd rootfs/bin
	$(STRIP) rootfs/bin/ntpd

	# ntp.conf
	cp conf-files/ntp.conf rootfs/etc
```

(jatkuu)

LIITE 4

```
	# e2fsprogs (mkfs, fsck)
	(cd compile/e2fsprogs/$(E2FSPROGS) && cp e2fsck/e2fsck misc/mke2fs \
		$(MYDIR)/rootfs/bin)

	# zlib
	cp -a compile/tmp-install/lib/libz.so* rootfs/lib

	sed 's|^.*/\(.*\)|ln -s busybox rootfs/bin/\1|' compile/busybox/$(BUSYBOX)/busy-
box.links |sh

	mkdir -p rootfs/python/lib/python$(PYTHONVER1)/site-packages \
		rootfs/python/lib/python$(PYTHONVER1)/logging \
		rootfs/python/lib/lib-dynload \
		rootfs/python/lib/python$(PYTHONVER1)/site-packages/serial

	ln -s ../lib-dynload rootfs/python/lib/python$(PYTHONVER1)/lib-dynload

	cp compile/python/$(PYTHON)/python rootfs/bin

	(cd $(MYDIR)/compile/python/$(PYTHON)/Lib && \
	cp $(PYTHON_MODULES) $(MYDIR)/rootfs/python/lib/python$(PYTHONVER1))

	cp compile/python/$(PYTHON)/Lib/logging/* \
		rootfs/python/lib/python$(PYTHONVER1)/logging
	cp -r compile/python/$(PYTHON)/Lib/ctypes \
		rootfs/python/lib/python$(PYTHONVER1)
	cp -r compile/python/$(PYTHON)/Lib/json \
		rootfs/python/lib/python$(PYTHONVER1)
	(cd rootfs//python/lib/python$(PYTHONVER1)/ctypes/test && \
		rm -rf test macholib)
	(cd rootfs//python/lib/python$(PYTHONVER1)/json && \
		rm -rf tests)

	mkdir -p rootfs/python/lib/python$(PYTHONVER1)/encodings
	(cd compile/python/$(PYTHON)/Lib/encodings && \
	    cp hex_codec.py ascii.py aliases.py __init__.py utf_8.py \
		$(MYDIR)/rootfs/python/lib/python$(PYTHONVER1)/encodings)

	(cd $(MYDIR)/compile/python/$(PYTHON)/build/lib.linux-$(BUILD_ARCH)-$(PYTHONVER1)
&& \
	cp $(PYTHON_SO_MODULES) \
		$(MYDIR)/rootfs/python/lib/lib-dynload)

	for x in $(PYTHON_SO_MODULES); \
		do $(STRIP) --strip-debug rootfs/python/lib/lib-dynload/$$x; \
	done

	# python-serial
	rm -rf compile/python-serial
	mkdir compile/python-serial
	(cd compile/python-serial && \
	tar xzf $(PYTHONSERIAL_ARCHIVE) && \
	cd $(PYTHONSERIAL)/serial && \
	cp __init__.py serialposix.py serialutil.py \
	  $(MYDIR)/rootfs/python/lib/python$(PYTHONVER1)/site-packages/serial)

	# simplejson (currently compiled separately)
	#(cd rootfs/python//lib/python$(PYTHONVER1)/site-packages && \
	#	tar xzf $(MYDIR)/../simplejson/simplejson-arm.tar.gz)

	# numpy
	cp -r $(MYDIR)/compile/numpy/numpy-1.4.1/build/lib.linux-arm-2.6/numpy \
		rootfs/python/lib/python$(PYTHONVER1)/site-packages

	(cd $(MYDIR)/compile/numpy/numpy-1.4.1/build/lib.linux-$(BUILD_ARCH)-2.6 && \
	    tar cf - `find . -type f -name '*.py' \! \( -path '*/doc*' -or \
		-path '*/numarray/*' -or -path '*/oldnumeric/*' -or \
		-name 'setup*.py' -or -name 'scons*.py' \)` ) | \
	(cd rootfs/python/lib/python2.6/site-packages && tar xf -)

	# python-wpactrl
	# (compiled every time, but fast)
	(cd ../python-wpactrl && sh compile.sh && \
		cp build/lib.linux-arm-2.6/wpactrl.so \
```

(jatkuu)

LIITE 4

```
$(MYDIR)/rootfs/python/lib/python$(PYTHONVER1)/site-packages)

        # pynetlinux
        rm -rf compile/pynetlinux
        mkdir compile/pynetlinux
        cd compile/pynetlinux && \
        tar xzf $(PYNETLINUX_ARCHIVE) && \
        cd $(PYNETLINUX) && \
        cp -r pynetlinux $(MYDIR)/rootfs/python/lib/python$(PYTHONVER1)/site-packages

        # patch to make it possible to find "non-physical" devices
        # with findif (necessary because eth0 is not properly set
        # as a "physical" device

 in the current kernel)
        sed -i 's/    for br in iterifs(True):/    for br in iterifs(False):/'
rootfs/python/lib/python$(PYTHONVER1)/site-packages/pynetlinux/ifconfig.py

        cp $(MYDIR)/conf-files/hotplug rootfs/bin
        chmod 755 rootfs/bin/hotplug

        cp $(MYDIR)/../firmware/* rootfs/lib/firmware

        #mkdir -p rootfs/lib/finsor/finsorlib rootfs/lib/finsor/daqagent

        #python finsor-app-files.py

        #cp ../finsor/run-app.py rootfs
        #chmod 755 rootfs/run-app.py

        # Finsor python extension
        # (compiled every time, but it is fast, so it does not matter)
        #rm -rf compile/finsor-python-ext
        #mkdir -p compile/finsor-python-ext
        #cp -r $(FINSOR_PYTHONLIB_DIR)/extensions/src/* compile/finsor-python-ext
        #(cd compile/finsor-python-ext && \
        # $(CROSS_COMPILE)gcc -c -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2
-Wall -Wstrict-prototypes -fPIC -Wall -O2 -I$(MYDIR)/compile/tmp-install/include/python$
(PYTHONVER1) protocol_impl.c -o protocol_impl.o && \
        # $(CROSS_COMPILE)gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions
protocol_impl.o -o protocol_impl.so && \
        #    cp protocol_impl.so \
        #    $(MYDIR)/rootfs/lib/finsor/finsorlib)

        # Finsor config
        #mkdir -p rootfs/etc/finsor/defconfig
        #cp ../finsor/config/* rootfs/etc/finsor/defconfig

        # python module compile
        # NOTE: this assumes the host python to be the same version
        # as the one on the target!
        #python$(PYTHONVER1) -m compileall \
        #       rootfs/python/lib/python$(PYTHONVER1)
        #(find rootfs/python/lib/python$(PYTHONVER1) -name '*.py' | \
        #       xargs rm -f)

        #python$(PYTHONVER1) -m compileall rootfs/lib/finsor
        #(find rootfs/lib/finsor -name '*.py' | xargs rm -f)

        python2.6 libs.py $(CROSS_COMPILE)

        chmod 755 rootfs/etc/init.d/rcS
        #chmod 755 rootfs/lib/ld-uClibc-0.9.31.so
        chmod 755 rootfs/lib/ld-2.8.so

        cp `find $(KERNEL_SRC) -name '*.ko'` rootfs/lib/modules

        grep UTS_RELEASE $(KERNEL_SRC)/include/generated/utsrelease.h|sed 's/^.*"\
(.*\)".*$$/\1/' >_kernelver
        ln -s . rootfs/lib/modules/`cat _kernelver`


        cd $(KERNEL_SRC) && \
        make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm uImage && \
        cp $(KERNEL_SRC)/arch/arm/boot/uImage $(MYDIR)
```

(jatkuu)

LIITE 4

```
	$(GENEXT2FS) -U -d rootfs -D devtable.txt -i$(ROOTFS_EXT2_INODES) -b$
(ROOTFS_EXT2_BLOCKS) rootfs.ext2

	python generate_system_image.py


_kernel_built:
	cd $(KERNEL_SRC) && \
	make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm $(KERNEL_CONFIG) && \
	make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm modules
	touch _kernel_built

_busybox_built: $(BUSYBOX_ARCHIVE) busybox.config
	mkdir -p compile
	rm -rf compile/busybox
	mkdir compile/busybox
	cd compile/busybox && \
	tar xjf $(BUSYBOX_ARCHIVE) && \
	cd $(BUSYBOX) && \
	cp $(MYDIR)/busybox.config .config && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) oldconfig && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) busybox.links
	touch _busybox_built

_lrzsz_built: $(LRZSZ_ARCHIVE)
	mkdir -p compile
	rm -rf compile/lrzsz
	mkdir compile/lrzsz
	cd compile/lrzsz && \
	tar xzf $(LRZSZ_ARCHIVE) && \
	cd lrzsz-0.12.20 && \
	CC=$(CROSS_COMPILE)gcc AS=$(CROSS_COMPILE)as LD=$(CROSS_COMPILE)ld \
		AR=$(CROSS_COMPILE)ar RANLIB=$(CROSS_COMPILE)ranlib \
		STRIP=$(CROSS_COMPILE)strip CFLAGS=-Wstrict-prototypes \
		./configure --host=$(HOST) && \
	 $(MAKE)
	touch _lrzsz_built

_libnl_built: $(LIBNL_ARCHIVE)
	mkdir -p compile
	rm -rf compile/libnl
	mkdir compile/libnl
	cd compile/libnl && \
	tar xzf $(LIBNL_ARCHIVE) && \
	cd $(LIBNL) && \
	sed -i '/inttypes\.h/s/$$/\n#include <limits.h>/' include/netlink-local.h && \
	./configure --host=$(CROSS_HOST) --prefix=$(MYDIR)/compile/tmp-install && \
	make && \
	make install
	touch _libnl_built

_iw_built: _libnl_built $(IW_ARCHIVE)
	rm -rf compile/iw
	mkdir compile/iw
	cd compile/iw && \
	tar xjf $(IW_ARCHIVE) && \
	cd $(IW) && \
	 PKG_CONFIG_PATH=$(MYDIR)/compile/tmp-install/lib/pkgconfig make CC=$(CROSS_COM-
PILE)gcc CFLAGS+="-I$(MYDIR)/compile/tmp-install/include -D_GNU_SOURCE" && \
	$(STRIP) iw
	touch _iw_built

_wireless_tools_built: $(WIRELESS_TOOLS_ARCHIVE)
	mkdir -p compile
	rm -rf compile/wireless-tools
	mkdir compile/wireless-tools
	cd compile/wireless-tools && \
	tar xzf $(WIRELESS_TOOLS_ARCHIVE) && \
	cd $(WIRELESS_TOOLS) && \
	$(MAKE) CC=$(CROSS_COMPILE)gcc BUILD_STATIC=y && \
	$(STRIP) iwconfig iwlist iwpriv
	touch _wireless_tools_built

_wpa_supplicant_built: $(WPA_SUPPLICANT_ARCHIVE)
```

(jatkuu)

LIITE 4

```
	mkdir -p compile
	rm -rf compile/wpa-supplicant
	mkdir compile/wpa-supplicant
	cd compile/wpa-supplicant && \
	tar xzf $(WPA_SUPPLICANT_ARCHIVE) && \
	cd $(WPA_SUPPLICANT)/wpa_supplicant && \
	echo CONFIG_DRIVER_WEXT=y >.config && \
	echo CONFIG_CTRL_IFACE=y >>.config && \
	make CC=$(CROSS_COMPILE)gcc
	touch _wpa_supplicant_built

_ethtool_built: $(ETHTOOL_ARCHIVE)
	mkdir -p compile
	rm -rf compile/ethtool
	mkdir compile/ethtool
	cd compile/ethtool && \
	tar xzf $(ETHTOOL_ARCHIVE) && \
	cd ethtool-3 && \
	CC=$(CROSS_COMPILE)gcc AS=$(CROSS_COMPILE)as LD=$(CROSS_COMPILE)ld \
		AR=$(CROSS_COMPILE)ar RANLIB=$(CROSS_COMPILE)ranlib \
		STRIP=$(CROSS_COMPILE)strip CFLAGS="-O2 -static" \
		./configure --host=arm-linux --enable-static && \
	make
	touch _ethtool_built

_ppp_built: $(PPP_ARCHIVE)
	mkdir -p compile
	rm -rf compile/ppp
	mkdir -p compile/ppp
	cd compile/ppp && \
	tar xzf $(PPP_ARCHIVE) && \
	cd $(PPP) && \
	sed -i 's/^FILTER=y/#FILTER=y/' pppd/Makefile.linux && \
	./configure && \
	$(MAKE) HAVE_MULTILINK= NEEDDES= PLUGINS= \
	CC=$(CROSS_COMPILE)gcc \
	"CFLAGS+=-I. -I.. -I../.. -I../include -I../include/net"
	touch _ppp_built

_python_built: _openssl_built _zlib_built $(PYTHON_ARCHIVE)
	mkdir -p compile
	rm -rf compile/python
	mkdir compile/python
	cd compile/python

	cd compile/python && tar xjf $(PYTHON_ARCHIVE)
	sed -i 's/^    \.pad #16/     UNWIND .pad #16/' compile/python/$
(PYTHON)/Modules/_ctypes/libffi/src/arm/sysv.S

	cd compile/python/$(PYTHON) && ./configure

	@# Compile the pgen and parser of the specific version of Python. These are
	@# used for generating some of the Python source code. If the host system
	@# has the exact same Python version as the target, this build step could
	@# be omitted, and the host system's version of pgen could be used.
	cd compile/python/$(PYTHON) && make python Parser/pgen && \
	mv python hostpython && \
	mv Parser/pgen Parser/hostpgen

	cd compile/python/$(PYTHON) && make distclean

	cd compile/python/$(PYTHON) && patch -p1 <$(PYTHON_PATCH)
	cd compile/python/$(PYTHON) && patch -p1 <$(PYTHON_SETUP_PATCH)
	sed -i 's|<<TMPINSTALL>>|$(MYDIR)/compile/tmp-install|' \
		compile/python/$(PYTHON)/setup.py

	cd compile/python/$(PYTHON) && \
	./configure --host=$(CROSS_HOST) --build=$(BUILD) --prefix=/python

	sleep 3

	cd compile/python/$(PYTHON) && \
	sed -i 's|^#SSL=/usr/local/ssl|SSL=$(MYDIR)/compile/tmp-install|' Modules/Setup
&& \
	sed -i 's|^#_ssl|_ssl|' Modules/Setup && \
```

(jatkuu)

LIITE 4

```
sed -i 's|^#   -DUSE_SSL -I\$$(SSL)/include| -DUSE_SSL -I\$$(SSL)/include/openssl|' Mod-
les/Setup && \
        sed -i 's|^#   -L\$$(SSL)/lib|       -L$$(SSL)/lib|' Modules/Setup && \
        sed -i 's|^#zlib zlibmodule|zlib zlibmodule|' Modules/Setup && \
        sed -i 's|^#binascii|binascii|' Modules/Setup

        cd compile/python/$(PYTHON) && \
        make HOSTPYTHON=./hostpython HOSTPGEN=./Parser/hostpgen \
                BLDSHARED='$(CROSS_COMPILE)gcc -shared' \
                CROSS_COMPILE=$(CROSS_COMPILE) CROSS_COMPILE_TARGET=yes \
                CFLAGS+=-I$(MYDIR)/compile/tmp-install/include

        $(STRIP) compile/python/$(PYTHON)/python

        ln -f -s ../pyconfig.h compile/python/$(PYTHON)/Include/pyconfig.h

        ln -f -s $(MYDIR)/compile/python/$(PYTHON)/Include \
                    $(MYDIR)/compile/tmp-install/include/python$(PYTHONVER1)

        touch _python_built

_openssl_built: $(OPENSSL_ARCHIVE)
        mkdir -p compile
        rm -rf compile/openssl
        mkdir compile/openssl
        cd compile/openssl && \
        tar xzf $(OPENSSL_ARCHIVE) && \
        cd $(OPENSSL) && \
        ./Configure --prefix=/usr linux-generic32 -DL_ENDIAN && \
        $(MAKE) AR="$(CROSS_COMPILE)ar r" RANLIB=$(CROSS_COMPILE)ranlib \
                CC=$(CROSS_COMPILE)gcc && \
        ln -s . lib

        ln -sf $(MYDIR)/compile/openssl/$(OPENSSL)/include/openssl $(MYDIR)/compile/tmp-
install/include/openssl
        ln -sf $(MYDIR)/compile/openssl/$(OPENSSL)/libssl.a $(MYDIR)/compile/tmp-
install/lib/libssl.a
        ln -sf $(MYDIR)/compile/openssl/$(OPENSSL)/libcrypto.a $(MYDIR)/compile/tmp-ins-
tall/lib/libcrypto.a
        ln -sf . $(MYDIR)/compile/openssl/$(OPENSSL)/include/openssl/openssl
        touch _openssl_built

_zlib_built: $(ZLIB_ARCHIVE)
        mkdir -p compile
        rm -rf compile/zlib
        mkdir compile/zlib
        cd compile/zlib && \
        tar xjf $(ZLIB_ARCHIVE) && \
        cd $(ZLIB) && \
        CC=$(CROSS_COMPILE)gcc AR=$(CROSS_COMPILE)ar RANLIB=$(CROSS_COMPILE)ranlib ./con-
figure --prefix=$(MYDIR)/compile/tmp-install && \
        $(MAKE) && $(MAKE) install
        touch _zlib_built

_ntp_built:
        mkdir -p compile
        rm -rf compile/ntp
        mkdir compile/ntp
        cd compile/ntp && \
        tar xzf $(NTP_ARCHIVE) && \
        cd $(NTP) && \
        ./configure --host=$(CROSS_HOST) && \
        sed -i 's/__adjtimex/adjtimex/' util/tickadj.c && \
        sed -i 's/HAVE___ADJTIMEX/linux/' util/tickadj.c && \
        make && \
        $(STRIP) ntpdate/ntpdate
        touch _ntp_built

_e2fsprogs_built:
        mkdir -p compile
        rm -rf compile/e2fsprogs
        mkdir compile/e2fsprogs
        cd compile/e2fsprogs && \
        tar xzf $(E2FSPROGS_ARCHIVE) && \
        cd $(E2FSPROGS) && \
```

(jatkuu)

LIITE 4

```
	./configure --host=$(CROSS_HOST) && \
	make && \
	$(STRIP) e2fsck/e2fsck misc/mke2fs
	touch _e2fsprogs_built

_extadc_driver_built: _kernel_built
	cd ../ext-adc-driver && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) ARCH=$(ARCH) clean && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) ARCH=$(ARCH) && \
	$(STRIP) --strip-debug extadc.ko
	touch _extadc_driver_built

_intadc_driver_built: _kernel_built
	cd ../int-adc-driver && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) ARCH=$(ARCH) clean && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) ARCH=$(ARCH) && \
	$(STRIP) --strip-debug intadc.ko
	touch _intadc_driver_built

_hsadc_mic_driver_built: _kernel_built
	cd ../hsadc-mic-driver && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) ARCH=$(ARCH)

 clean && \
	$(MAKE) CROSS_COMPILE=$(CROSS_COMPILE) ARCH=$(ARCH) && \
	$(STRIP) --strip-debug hsadc_mic.ko
	touch _hsadc_mic_driver_built

_dropbear_built: _zlib_built $(DROPBEAR_ARCHIVE)
	mkdir -p compile
	rm -rf compile/dropbear
	mkdir compile/dropbear
	cd compile/dropbear && \
	tar xzf $(DROPBEAR_ARCHIVE) && \
	cd $(DROPBEAR) && \
	ln -s $(MYDIR)/compile/tmp-install/include/zlib.h zlib.h && \
	ln -s $(MYDIR)/compile/tmp-install/include/zconf.h zconf.h && \
	CC=$(CROSS_COMPILE)gcc AR=$(CROSS_COMPILE)ar RANLIB=$(CROSS_COMPILE)ranlib
STRIP=$(CROSS_COMPILE)strip ./configure --host=arm-linux --with-zlib=$
(MYDIR)/compile/zlib/zlib-1.2.5 && \
	$(MAKE) PROGRAMS="dropbear dropbearkey dbclient scp" MULTI=1 && \
	$(STRIP) dropbearmulti
	touch _dropbear_built
_numpy_built: $(NUMPY_ARCHIVE) $(NUMPY_PATCH) $(NUMPY_BUILD_SCRIPT) _python_built



mkdir -p compile
	rm -rf compile/numpy
	mkdir compile/numpy
	cd compile/numpy && tar xzf $(NUMPY_ARCHIVE)

	cd compile/numpy/$(NUMPY) && patch -p2 <$(NUMPY_PATCH)

	cd compile/numpy/$(NUMPY) && python2.6 setup.py build
	sed -i 's/^\#define SIZEOF_PY_INTPTR_T.*$$/\#define SIZEOF_PY_INTPTR_T 4/' \
		compile/numpy/$(NUMPY)/build/src.linux-$(BUILD_ARCH)-2.6/numpy/core/inc-
lude/numpy/config.h
	sed -i 's/^\#define HAVE_LDOUBLE_INTEL_EXTENDED.*$$/\#define
HAVE_LDOUBLE_IEEE_DOUBLE_LE 1/' compile/numpy/$(NUMPY)/build/src.linux-$(BUILD_ARCH)-
2.6/numpy/core/include/numpy/config.h

	sed -i 's/^\#define NPY_SIZEOF_PY_INTPTR_T.*$$/\#define NPY_SIZEOF_PY_INTPTR_T
4/' \
		compile/numpy/$(NUMPY)/build/src.linux-$(BUILD_ARCH)-2.6/numpy/core/inc-
lude/numpy/_numpyconfig.h
	sed -i 's/^\#define NPY_SIZEOF_LONGDOUBLE.*$$/\#define NPY_SIZEOF_LONGDOUBLE
8/' \
		compile/numpy/$(NUMPY)/build/src.linux-$(BUILD_ARCH)-2.6/numpy/core/inc-
lude/numpy/_numpyconfig.h
	sed -i 's/^\#define NPY_SIZEOF_COMPLEX_LONGDOUBLE.*$$/\#define NPY_SIZEOF_COMP-
LEX_LONGDOUBLE 16/' \
		compile/numpy/$(NUMPY)/build/src.linux-$(BUILD_ARCH)-2.6/numpy/core/inc-
lude/numpy/_numpyconfig.h
```

(jatkuu)

LIITE 4

```
	ln -s src.linux-$(BUILD_ARCH)-2.6 compile/numpy/$(NUMPY)/build/src.linux-arm-2.6



	cd compile/numpy/$(NUMPY) && sh $(NUMPY_BUILD_SCRIPT)
	touch _numpy_built

_deviceread_built: _kernel_built
	mkdir -p compile
	rm -rf compile/deviceread
	mkdir compile/deviceread
	cd compile/deviceread && \
	tar xzf $(DEVICEREAD_ARCHIVE) && \
	cd deviceread && \
	make
	touch _deviceread_built

_mwwpy_built: _kernel_built
	mkdir -p compile
	rm -rf compile/MeshWorksPython
	mkdir compile/MeshWorksPython
	cd compile/MeshWorksPython && \
	tar xzf $(MWWPY_ARCHIVE) && \
	cd MeshWorksPython && \
	make
	touch _mwwpy_built

clean:
	(cd ../ext-adc-driver && make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm clean)
	(cd ../int-adc-driver && make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm clean)
	(cd ../hsadc-mic-driver && make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm clean)
	(cd $(KERNEL_SRC) && make CROSS_COMPILE=$(CROSS_COMPILE) ARCH=arm distclean)
	rm -rf _*_built compile rootfs rootfs.ext2 rootfs.ext2.temp uImage \
		../python-wpactrl/build _kernelver boot.src boot.src.img \
		system.img
```

Liite 5. Laitteen käynnistys-scripti

```
#!/bin/python
#

import sys
import os
import subprocess
import select
import time



def filesystem_check(devname):
    """
    Checks the given filesystem using e2fsck.

    @param devname: Name of the block device to check.
    @return: True if check OK, False if check fails
    """
    sel_timeout = 1.0

    print "Checking %s" % devname

    fsck_res = []
    pobj = subprocess.Popen(('/bin/e2fsck', '-y', '-f',  devname),
                            stdout=subprocess.PIPE,
                            stderr=subprocess.STDOUT)

    t0 = time.time()

    while True:
        if time.time() > t0 + 60.0:
            print 'fsck takes too long time, assuming filesystem corrupted'
            pobj.kill()
            return False

        rdl, wrl, exl = select.select([pobj.stdout], [], [], sel_timeout)
        if rdl == []:
            continue

        data = pobj.stdout.readline()
        if data == '':
            break
        fsck_res.append(data)
        if len(fsck_res) > 20:
            print 'Too much output from fsck, assuming filesystem corrupted'
            pobj.kill()
            return False

    pobj.wait()
    ret = pobj.returncode
    print "fsck returncode: %s" % repr(ret)
    if ret > 3:
        print "fsck returned error (returncode %d)" % ret
        return False

    print "fsck OK"
    return True

def filesystem_check_and_mount(devname, mountpoint):
    reformat = True
    if not filesystem_check(devname):
        print "%s: Filesystem check failed" % devname
    else:
        res = os.system('mount %s' % mountpoint)
        if res != 0:
            print "%s: Mount failed" % devname
        else:
            if (mountpoint == '/var/cache' and \
                    (not os.path.isdir('/var/cache/daqagent'))) or \
                    (mountpoint == '/etc/finsor/config' and \
                         (not os.path.isfile('/etc/finsor/config/daqagent.cfg'))):
                print "%s: %s: file/directory check failed" % (devname, mountpoint)
                os.system('umount %s' % mountpoint)
            else:
```

(jatkuu)

LIITE 5

```
        print "%s: %s: mounted OK" % (devname, mountpoint)
                return


                print "%s: reformatting" % devname
    os.system('mke2fs -t ext4 %s' % devname)
    res = os.system('mount %s' % mountpoint)
    if res != 0:
        print "%s: %s: mount failed after format - rebooting" % (devname, mountpoint)
        os.system('reboot')

    if mountpoint == '/var/cache':
        os.system('mkdir -p /var/cache/daqagent /var/cache/log')
    elif mountpoint == '/etc/finsor/config':
        print "Copying default config"
        os.system('cp /etc/finsor/defconfig/* /etc/finsor/config')

    # set the reformatted flag
    os.system('touch %s' % os.path.join(mountpoint, 'reformatted'))

    print "%s: %s: reformatted and mounted OK" % (devname, mountpoint)
    return

#
if __name__ == '__main__':
    filesystem_check_and_mount('/dev/mmcblk0p5', '/etc/finsor/config')
    filesystem_check_and_mount('/dev/mmcblk0p6', '/var/cache')

#os.system('deviceread')
```