Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

JUHO SIMONEN

# Ruokinta-automaatin prototyypin kehitys CCMP15-kehitysalustalla

SÄHKÖ- JA AUTOMAATIOTEKNIIKKA
2024

TIIVISTELMÄ

Simonen, Juho: Ruokinta-automaatin prototyypin kehitys CCMP15-kehitysalustalla
Opinnäytetyö, AMK
Sähkö- ja automaatiotekniikka
Huhtikuu, 2024
Sivumäärä: 39

Arvo-Tec Oy:n toimeksiannosta tehtiin ConnectCore MP15-kehitysalustalla kalojenruokinta-automaatin prototyyppi ja selvitettiin samalla mitä ohjelmistotyökaluja siihen tarvitaan ja miten niitä käytetään. Työn tuloksena syntyi ohjelmistodemo ja ohjeet, kuinka CCMP15-kehitysalustalle voidaan tehdä Linux-järjestelmätiedostot sekä C-kielisiä ohjelmia sen eri prosessoreille siten, että ne pystyvät kommunikoivat keskenään.

Avainsanat: STM32MP15, Yocto Project, Remoteproc, automaatio, ohjelmistokehitys, C, Sulautettu Linux

ABSTRACT

Simonen, Juho: Development of a Feeder Automation System prototype on the CCMP15 Development Kit
Bachelor's thesis
Electrical and Automation Engineering
April, 2024
Number of pages: 38

On behalf of Arvo-Tec Oy, a prototype of an automatic fish feeder was developed using the ConnectCore MP15 development platform, and at the same time, the software tools needed for it and how to use them were explored. As a result of the work, a software demo and instructions were created on how to develop custom image and C-language programs on the CCMP15 development platform for its various processors so that they can communicate with each other.

Keywords: STM32MP15, Yocto Project, Remoteproc, automation, software development, C, Embedded Linux

SISÄLLYS

# 1 JOHDANTO

Arvo-Tec Oy on kehittämässä uuden sukupolven laitteistoa vesiviljelyyn. Uuden sukupolven laitteiston tietojenkäsittelyä ja automatiikkaa on valittu hoitamaan Digi Internationalin ConnectCore MP15 järjestelmämoduuli (engl. system-on-module).

Arvo-Tec Oy on Joroisten Huutokoskella sijaitseva teknologiayhtiö, joka on myynyt kalankasvatusteknologiaa Suomeen ja etenkin ulkomaille usean vuosikymmenen ajan. Arvo-Tec Oy:n tärkein osaamisen keihäänkärki on nykyaikaiset RAS kiertovesilaitokset. Vesiviljely on maailman nopeimmin kasvava elintarviketuotannon sektori ja RAS vuorostaan vesiviljelyn nopeimmin kehittyvä osa-alue.

Opinnäytetyön tarkoitus on kehittää ohjelmistodemo ConnectCore MP15 kehitysalustalla (engl. Development kit) ja samalla määrittää ConnectCore MP15 järjestelmämoduulin (engl. System-on-module) ohjelmistokehityksessä tarvittavat työkalut ja raportoida tiedot tästä prosessista Arvo-Tec Oy:lle.

Ohjelmistodemon on tarkoitus luoda yksinkertainen ruokinta-automaatin käyttöliittymä ja ohjattava GPIO-pinniä käyttäjän antamien tietojen perusteella. Oikeassa kalojenruokintakäytössä GPIO-pinnin tarkoitus olisi ohjata relettä, joka käynnistää moottorin, joka syöttää ruokaa kaloille.

# 2 PROJEKTIN TIETOPERUSTA

## 2.1 Digi ConnectCore MP157 Development Kit

Digi Internationalin valmistama ConnectCore MP15 Development Kit on sulautettujen järjestelmien kehityspaketti, joka tarjoaa kehittäjille valmiin alustan sulautettujen sovellusten prototyyppien rakentamiseen ja testaamiseen. Kehityspaketti on suunniteltu helpottamaan sulautettujen järjestelmien kehittämistä erilaisiin käyttötarkoituksiin, kuten teollisuusautomaatioon, älykkäisiin laitteisiin ja IoT-sovelluksiin. Siinä on useita erilaisia liitäntä mahdollisuuksia kuten RS-232, RS-482, yhden gigabitin Ethernet-liitäntä RJ-45-liittimellä, USB 2.0 ja XBee Cellular. CCMP15 moduulissa on 256 megatavua DDR3 muistia ja 256 megatavua NAND tyyppistä flashmuistia. Sen ydin on STM32MP157 mikroprosessori. (Digi, n.d.-a)

## 2.2 STM32MP157 mikroprosessori

STM32MP157 on mikroprosessori, jonka on kehittänyt ja valmistanut STMicroelectronics. Se kuuluu STM32MP1-perheeseen, joka yhdistää Arm Cortex-A-prosessoriytimen ja Cortex-M prosessoriytimen yhdessä järjestelmässä. Cortex-A7-prosessoriydin, tarjoaa suorituskykyä monipuolisiin tehtäviin, kuten käyttöjärjestelmän suorittamiseen ja monimutkaisempien sovellusten käsittelyyn. Cortex-M4-prosessoriydin on optimoitu reaaliaikaisiin tehtäviin ja sulautettuihin järjestelmiin. STM32MP157 mikroprosessorin suunniteltuja käyttökohteita ovat muun muassa teollisuusautomaatio, älykodit, teollisuus 4.0 ja sähköautojen latausasemat. (St, n.d.-a)

## 2.3 Yocto Project

Yocto-projekti on avoimen lähdekoodin yhteistyöprojekti, joka auttaa kehittäjiä luomaan räätälöityjä Linux-pohjaisia järjestelmiä sulautettuihin tuotteisiin ja muihin kohdennettuihin ympäristöihin (The Linux Foundation, 2024.-a), kuten palvelimiin ja virtuaaliympäristöihin, riippumatta laitteistoarkkitehtuurista (The

Linux Foundation, 2024.-b). Yocto Project tarjoaa monipuoliset työkalut ja ympäristön sulautettujen laitteiden kehittäjille. Yocto-projekti tarjoaa etuja sekä järjestelmien että sovellusten kehityksessä, arkistoinnissa ja hallinnassa. (The Linux Foundation, 2024.-c)

## 2.4 Digi Embedded Yocto

Digi Embedded Yocto (DEY) on avoimen lähdekoodin ja vapaasti saatavilla oleva Yocto Project-pohjainen sulautettu Linux-jakelu. Se on viitejakelu Digi ConnectCore -sarjan sulautetuille järjestelmämoduuleille, ja se perustuu Pokyyn, viite Yocto Project Linux -jakeluun. Siihen sisältyy mukautuksia Digi-laitteistolle sekä valmiiksi asennettuja ohjelmistolaajennuksia, jotka eivät kuulu standardi Yocto Projectiin. (Digi, n.d.-b) Digi Embedded Yocto:lla voidaan rakentaa U-Boot-käynnistyslatain, Linux-ydin ja juuritiedostojärjestelmä(Digi, n.d.-c)

## 2.5 Layer Yocto Projectissa

Layer Yocto-projektissa viittaa lisäosapakettiin tai moduuliin, joka sisältää reseptejä, konfiguraatiotiedostoja ja/tai muita tiedostoja, jotka määrittelevät, miten tietyt ohjelmistokomponentit tai ominaisuudet integroidaan Yocto-projektin työkalujen luomaan sulautettuun Linux-jakeluun. Layerit ovat modulaarisia ja voivat sisältää esimerkiksi koodikirjastoja, sovelluksia, laiteajureita ja käyttöjärjestelmän komponentteja.

## 2.6 Linux-ydin

Linux-ydin, tunnettu myös nimellä Linux-kerneli, on Linux-käyttöjärjestelmän ydin. Se vastaa tietokoneen resurssien hallinnasta ja tarjoaa rajapinnan ohjelmistosovellusten ja laitteiston välille (Red Hat, n.d.-a). Linux-ydin on alun perin kehitetty Linus Torvaldsin toimesta 1990-luvun alussa. Se on avoimen lähdekoodin projekti, mikä tarkoittaa, että sen lähdekoodi on saatavilla ja vapaaehtoiset kehittäjät voivat osallistua sen kehittämiseen ja parantamiseen. Linux-

ytimen keskeisiä tehtäviä ovat muun muassa prosessorin hallinta, muistinhallinta, laiteajurit, tiedostojärjestelmät, prosessien hallinta, verkko- ja turvallisuustoiminnot (Red Hat, n.d.-b). Se toimii käyttöjärjestelmän perustana, ja siihen voidaan lisätä erilaisia komponentteja ja ohjelmistoja, kuten käyttöjärjestelmän käyttöliittymä, sovellukset ja palvelut.

## 2.7 Remoteproc

Remoteproc on Linux-ytimen tarjoama kehys (engl. framework), joka on suunniteltu tukemaan etäprosessorien käyttöä moniprosessorijärjestelmissä. Tämä kehys mahdollistaa etäprosessorin hallinnan ja käytön Linux-ytimen ulkopuolella, mikä on hyödyllistä järjestelmissä, joissa on useita prosessoreita. Remoteproc-kehyksen avulla Linux-ydin voi ladata etäprosessorin käyttöjärjestelmän (esimerkiksi toisen Linux-instanssin tai yksittäisen ohjelman) ja hallita sen käyttöä. Se tarjoaa välineitä etäprosessorin käynnistämiseen, sammuttamiseen ja kommunikointiin etäprosessorin kanssa. (St, n.d.-b)

## 2.8 Rpmsg

RPMsg-kehys on virtio-pohjainen viestintäväylä, joka mahdollistaa paikallisen prosessorin kommunikoinnin järjestelmässä olevien etäprosessoreiden kanssa. Linux RPMsg-kehys on viestintämekanismi, joka on toteutettu virtio-kehyksen päälle kommunikointia varten etäprosessorin kanssa. Se perustuu virtiovringien käyttöön viestien lähettämiseksi/vastaanottamiseksi etäsuorittimelta ja jaettua muistia käyttäen.

Virtiovringit ovat yksisuuntaisia, toinen vring on omistettu viesteille, jotka lähetetään etäprosessorille, ja toinen vring on käytössä viesteille, jotka vastaanotetaan etäprosessorilta. Lisäksi jaetut puskurit luodaan muistitilassa, joka on näkyvissä molemmille prosessoreille.

Sen jälkeen postilaatikkokehystä käytetään ilmoittamaan ytimille, kun uusia viestejä odottaa puskurissa. (St, n.d.-c)

Näiden kehysten varassa RPMsg-kehys toteuttaa kommunikaation kanavien perusteella. Kanavat tunnistetaan tekstuaalisella nimellä ja niillä on paikallinen ("lähde") RPMsg-osoite ja etäinen ("kohde") RPMsg-osoite.
Etäprosessorin puolella RPMsg-kehys on myös toteutettava. OpenAMP-kehys (Open Asymmetric Multi-Processing framework) tarjoaa siihen ratkaisun.

# 3 OHJELMISTODEMON KEHITYS JA TOIMINNAN KUVAUS

## 3.1 Tutkimusongelma, kehittämistyön tavoite ja suunnitelma

Tutkimusongelmana on, miten CCMP15-moduulin eri prosessoreille tehdään C-kielellä ohjelmia ja miten prosessoreiden välinen kommunikaatio ja ruokinta-automaatin toiminnallisuus saadaan aikaiseksi.

Tavoitteena on kehittää ohjelmistodemo, joka osaa ruokkia kaloja oikeaan aikaan ja oikean määrän käyttäjän antamien tietojen perusteella ja luoda ohjeistus siitä, miten edellä mainittuun tavoitteeseen päästään. Ohjelmistodemo koostuu kahdesta C-ohjelmointikielellä kirjoitetusta ohjelmasta, jotka kommunikoivat keskenään. Cortex-A7 prosessorin olisi tarkoitus suorittaa ohjelmaa, joka tuottaa yksinkertaisen komentokehotetyyppisen käyttöliittymän, laskee ruokintapulssien määrät ja taukojen pituudet ja lähettää Cortex-M4 prosessorilla suoritettavalle ohjelmalle tarvittavat tiedot, jotta Cortex-M4 prosessorilla suoritettava ohjelma osaa ohjata GPIO-pinniä oikealla tavalla. Cortex-M4 prosessorilla suoritettavan ohjelman on myös ilmoitettava, jokaisen ruokintapulssin jälkeen Cortex-A7 prosessorilla suoritettavalle ohjelmalle, että pulssi on ajettu, jotta Cortex-A7 prosessorilla pyörivä ohjelma tietää paljonko kaloja on ruokittu.

Lopuksi kehitystyöstä laaditaan raportti ja kuvataan, miten tutkimusongelmat on ratkaistu ja kuinka tavoitteeseen on päästy.

Suunnitelma tavoitteeseen pääsemiseksi on seuraavanlainen:

1. Asennetaan kaikki tarvittavat ohjelmistotyökalut kehitysympäristöön
2. Linux-järjestelmätiedostojen (engl. Image) muokkaaminen, kääntäminen ja lataaminen laitteeseen
3. Ohjelmien tekeminen, kääntäminen ja laitteeseen lataaminen
4. Raportin tekeminen

## 3.2 Isäntäjärjestelmän työkalujen asennus ja kohdejärjestelmän käyttöönotto

Yocto Project tukee tällä hetkellä Ubuntu 20.04 LTS, Ubuntu 22.04 LTS Fedora 38, Debian GNU/Linux 11.x (Bullseye) ja AlmaLinux 8 jakeluja. (The Linux Foundation, 2024-d). Digin ohjeistuksen mukaan Digi Embedded Yoctoa voisi käyttää myös Docker konteissa, jotka olisi tehty 64-bittisellä Linux-käyttöjärjestelmällä tai 64-bittisellä Windows 10 käyttöjärjestelmällä.

Dokumentaation mukaan Digi ADE:n asentamisen vaatimuksiin kuuluu 64-bittinen Linux-käyttöjärjestelmä ja suositellaan Ubuntu 20.04 LTS (Digi, n.d.-d)

ST tarjoaa STM32CubeIDE:n asennuspaketteja Linux:ille, Windows:ille ja macOS:lle.

Päädyin asentamaan isäntäjärjestelmään eli tietokoneeseeni Ubuntu 22.04 LTS jakelun pystyäkseni käyttämään Digi Embedded Yoctoa ja Digi ADE:a(Digi Application Development Environment) ja STM32CubeIDE:ä.

Digi Internationalin sivuilta löytyy selkeät ohjeet Digi Embedded Yocto:n ja Digi ADE:n asentamiseen sekä ConnectCore MP15 Development Kit:in käyttöönottoon. ST Microelectronics:in sivuilta löytyy hyvät ohjeet STM32CubeIDE:n asentamiseen.

## 3.3 Linux-järjestelmätiedoston tekeminen Yocto Projectin työkalujen avulla

Ennen tätä vaihetta oletetaan, että ConnectCore MP15 kehitysalusta (engl. Development Kit) on otettu käyttöön laitteen dokumentaation kuvaamalla tavalla eli laitteeseen on asennettu dey-image-webkit niminen Linux-järjestelmätiedosto.

### 3.3.1 Digi Embedded Yocto -projektin perustaminen

Projekteille voi tehdä oman hakemiston. Tässä tapauksessa sen nimi on workspace. Jokaiselle projektille on tämän lisäksi tehtävä vielä oma hakemisto ja projektia perustettaessa hakemiston juuressa on suoritettava mkproject.sh niminen ohjelma ja annettava ohjelmalle parametrina ”ccmp15-dvk”, jotta ohjelma tietää mille laitearkkitehtuurille projekti perustetaan.

```
juho@juho-System-Product-Name:~/workspace$ mkdir -p ${HOME}/workspace/ccmp15-dvk-esimerkki
juho@juho-System-Product-Name:~/workspace$ cd ${HOME}/workspace/ccmp15-dvk-esimerkki
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$ source /usr/local/dey-4.0/mkproject.sh -p ccmp15-dvk
```

*Kuva 1. Tehdään hakemistot projekteille ja yksittäiselle projektille sekä suoritetaan projektinalustusskripti.*

Ohjelma alustaa projektin ja luo conf-hakemiston, jossa on kaksi konfigurointi tiedostoa. Ohjelma tuottaa hakemistoon myös skriptin dey-setup-environment, joka pitää suorittaa aina ennen kuin alkaa käyttämään Digi Embedded Yoctoa ja sen työkaluja, jotta Yocto Projectin työkalut saadaan käyttöön.

```
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$ ls
conf  dey-setup-environment
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$ ls conf
bblayers.conf  local.conf  templateconf.cfg
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$
```

*Kuva 2. Projektihakemistoon syntyy konfigurointi tiedostoja mkproject.sh-skriptin suorittamisen jälkeen.*

### 3.3.2 Konfiguraatio tiedostojen muokkaus

Local.conf tiedostoa muokkaamalla pystyy muun muassa lisätä koottavaa Linux-järjestelmätiedostoon paketteja kuten strace (diagnosointi ja debuggaus

työkalu), ominaisuuksia kuten dey-bluetooth, dey-examples ja dey-network ja vaikuttaa bitbake prosessiin esimerkiksi rajoittamalla bitbake-prosessin aikana käytettävien säikeiden määrää. Tässä projektissa bitbake-prosessi kaatui useaan kertaan, kunnes selvisi, että ongelma oli se, että 32 gigatavua ram-muistia ei riittänyt bitbake-prosessin läpi ajamiseen, koska bitbake-prosessi suoritti tehtäviään käyttäen siihen yli kahdeksaa säiettä. Yocto Projectin dokumentaatiossa suositellaan, että jokaisella säikeellä on käytössään ainakin 4 gigatavua rammuistia, siksi ratkaisu oli rajoittaa bitbake-prosessin käytössä olevat säikeet kahdeksaan, koska tietokoneessani on 32 gigatavua rammuistia. Seuraavan kuvan kaksi alinta riviä kertoo bitbake-prosessille montako säiettä se saa maksimissaan käyttää.

```
#
# By default native qemu will build with a builtin VNC server where graphical output can be
# seen. The line below enables the SDL UI frontend too.
PACKAGECONFIG:append:pn-qemu-system-native = " sdl"
# By default libsdl2-native will be built, if you want to use your host's libSDL instead of
# the minimal libsdl built by libsdl2-native then uncomment the ASSUME_PROVIDED line below.
#ASSUME_PROVIDED += "libsdl2-native"

# You can also enable the Gtk UI frontend, which takes somewhat longer to build, but adds
# a handy set of menus for controlling the emulator.
#PACKAGECONFIG:append:pn-qemu-system-native = " gtk+"

#
# Hash Equivalence
#
# Enable support for automatically running a local hash equivalence server and
# instruct bitbake to use a hash equivalence aware signature generator. Hash
# equivalence improves reuse of sstate by detecting when a given sstate
# artifact can be reused as equivalent, even if the current task hash doesn't
# match the one that generated the artifact.
#
# A shared hash equivalent server can be set with "<HOSTNAME>:<PORT>" format
#
#BB_HASHSERVE = "auto"
#BB_SIGNATURE_HANDLER = "OEEquivHash"

#
# Memory Resident Bitbake
#
# Bitbake's server component can stay in memory after the UI for the current command
# has completed. This means subsequent commands can run faster since there is no need
# for bitbake to reload cache files and so on. Number is in seconds, after which the
# server will shut down.
#
#BB_SERVER_TIMEOUT = "60"

# CONF_VERSION is increased each time build/conf/ changes incompatibly and is used to
# track the version of this file when it was generated. This can safely be ignored if
# this doesn't mean anything to you.
CONF_VERSION = "2"

#
# Enable local PR server
#
PRSERV_HOST = "localhost:0"

#
# Some packages are covered by STM EULA
#
ACCEPT_EULA_ccmp15-dvk = "1"

#threadien rajoitus
BB_NUMBER_THREADS = "8"
PARALLEL_MAKE = "-j 8"
```

*Kuva 3. Kuvan kaksi alinta riviä kertoo bitbake-prosessille, että se saa käyttää maksimissaan kahdeksaa säiettä.*

conf/local.conf

```
IMAGE_INSTALL:append = " strace"
```

*Kuva 4. Lisäämällä kuvassa näkyvä teksti local.conf-tiedostoon bitbake-prosessi lisää strace-ominaisuuden koottavaan Linux-järjestelmätiedostoon.*

conf/local.conf

```
EXTRA_IMAGE_FEATURES = "<feature-name1> <feature-name2>"
```

*Kuva 5. Kuvassa näkyvällä tavalla voi lisätä ominaisuuksia koottavaan Linux-järjestelmätiedostoon.*

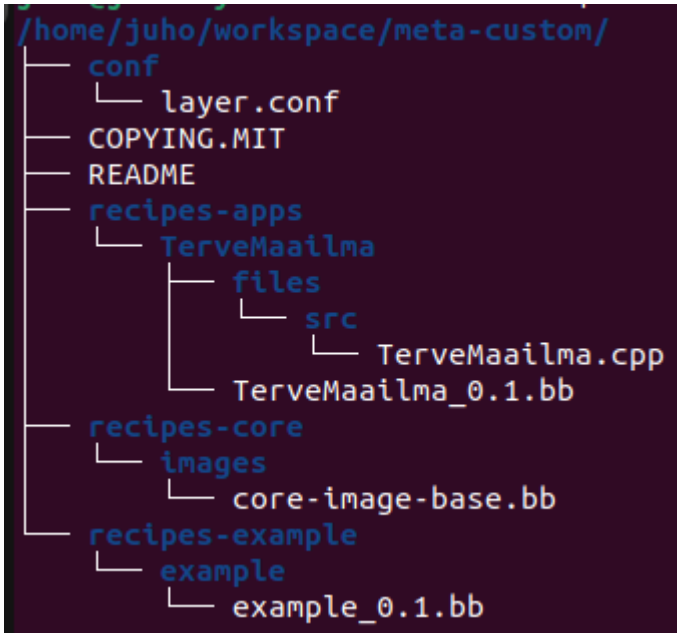
Konfigurointitiedosto bblayers.conf-tiedostossa voi muokata mitä moduuleja koottavaan imageen halutaan mukaan. Moduuleja löytyy valmiina ja niitä voi myös tehdä itse. Esimerkiksi, jos haluaa, että jokin oma sovellus kootaan mukaan Linux-järjestelmätiedostoon, niin silloin on sille tehtävä oma layer ja lisättävä sinne tarvittavat lähdekoodit ja reseptit. Tässä projektissa ei ollut tarvetta tehdä ”custom-layeriä”, eikä TerveMaailma-sovellus ollut osa projektia. Sen tekeminen oli keino esitellä Digi Embedded Yocton ohjelmistotyökaluja.

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
  /usr/local/dey-4.0/sources/poky/meta \
  /usr/local/dey-4.0/sources/poky/meta-poky \
  /usr/local/dey-4.0/sources/poky/meta-yocto-bsp \
  /usr/local/dey-4.0/sources/meta-openembedded/meta-oe \
  /usr/local/dey-4.0/sources/meta-openembedded/meta-python \
  /usr/local/dey-4.0/sources/meta-openembedded/meta-networking \
  /usr/local/dey-4.0/sources/meta-openembedded/meta-webserver \
  /usr/local/dey-4.0/sources/meta-qt5 \
  /usr/local/dey-4.0/sources/meta-swupdate \
  /usr/local/dey-4.0/sources/meta-webkit \
  /usr/local/dey-4.0/sources/meta-timesys \
  /usr/local/dey-4.0/sources/meta-st-stm32mp \
  /usr/local/dey-4.0/sources/meta-st-stm32mpu-ai \
  /usr/local/dey-4.0/sources/meta-digi/meta-digi-arm \
  /usr/local/dey-4.0/sources/meta-digi/meta-digi-dey \
  /home/juho/workspace/meta-custom \
  "
~
~
```

*Kuva 6. Bblayers.conf tiedosto. Alimmalla rivillä näkyy meta-custom -layerin tiedostopolku.*

Lisäämällä ”custom-layerin” tiedostopolku bblayers.conf-tiedostoon Linux-järjestelmätiedoston kokoamisprosessi osaa lisätä meta-custom -nimisen

layer:in koottavaan Linux-järjestelmätiedostoon. Layer on käytännössä hakemisto, jonka rakenne näkyy seuraavassa kuvassa.




*Kuva 7. Meta-custom layern tiedostorakenne.*

```
SUMMARY = "TerveMaailma resepti"
DESCRIPTION = "Custom resepti TerveMaailma.cpp aplikaation kääntämiseen"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

#Mistä löytää source files(voi olla myös esim github)
SRC_URI = "file://src"

#Where to keep downloaded source files(in tmp/work/...)
S = "${WORKDIR}/src"

# Pass argument to linker
TARGET_CC_ARCH += "${LDFLAGS}"

# Cross-compile source code
do_compile() {
   ${CXX} -o TerveMaailma TerveMaailma.cpp
}

# Create /usr/bin rootfs and copy program to it
do_install() {
   install -d ${D}${bindir}
   install -m 0755 TerveMaailma ${D}{bindir}
}
```

*Kuva 8. Resepti TerveMaailma.bb tiedostossa. Resepti kertoo bitbake-prosessille muun muassa sen mihin hakemistoon TerveMaailma-sovellus asennetaan kohdelaitteessa.*

```
//============================================================================
// Name        : TerveMaailma.cpp
// Author      : Simosen Juho
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
        cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
        return 0;
}
~
```

*Kuva 9. Lähdekoodi TerveMaailma.cpp-tiedossa.*

### 3.3.3 Linux-järjestelmätiedoston kokoaminen

Kun Digi Embedded Yocto on aktivoitu dey-setup-environment -skriptillllä, voidaan suorittaa bitbake-prosessi eli Linux-järjestelmätiedoston kokoaminen. Bitbake-prosessi kestää suhteellisen tehokkaallakin tietokoneella useita tunteja. Digi Embedded Yoctoa käytettäessä dey-image-webkit niminen Linux-järjestelmätiedosto tehdään komennolla bitbake dey-image-webkit.

```
juho@juho-System-Product-Name:~$ cd workspace
juho@juho-System-Product-Name:~/workspace$ cd ccmp15-dvk-esimerkki
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$ source dey-setup-environment
Digi Embedded Yocto provides the following image recipes:

  * dey-image-webkit: graphical WebKit image

    By default the image is Wayland-based so it provides a full Weston
    desktop environment.

  * dey-image-qt: graphical QT image

    By default the image is Wayland-based so it provides a full Weston
    desktop environment.

  * core-image-base: a console-only image

    Expansion of native core-image-base by including all the support for the
    target device hardware like firmware files, rootfs customizations, etc.
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$ bitbake dey-image-webkit
NOTE: Started PRServer with DBfile: /home/juho/workspace/ccmp15-dvk-esimerkki/cache/prserv.sqlite3, Address: 127.0.0.1:36235, PID: 3488
Loading cache: 100% |
Loaded 0 entries from dependency cache.
Parsing recipes:  51% |##########################################################################################
```

*Kuva 10. Bitbake-prosessin aloittaminen.*

```
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk-esimerkki$ ls
conf  dey-setup-environment
```

*Kuva 11. Projekti-hakemiston hakemistot ennen bitbake-prosessia.*

```
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk$ ls
bitbake-cookerdaemon.log  cache  conf  dey-setup-environment  downloads  sstate-cache  tmp  workspace
```

*Kuva 12. Projektihakemiston hakemistot bitbake-prosessin jälkeen.*

Bitbake-prosessin jälkeen projektihakemistoon syntyy tmp-hakemisto. Tmp-hakemistosta löytyy lähdekoodit ja kohdelaitteelle käännetyt käynnistyslataimet, linux-ytimet ja juuritiedostojärjestelmät

### 3.3.4 Cortex-M4 suorittimen aktivoiminen Linux-ytimen lähdekoodia muokkaamalla.

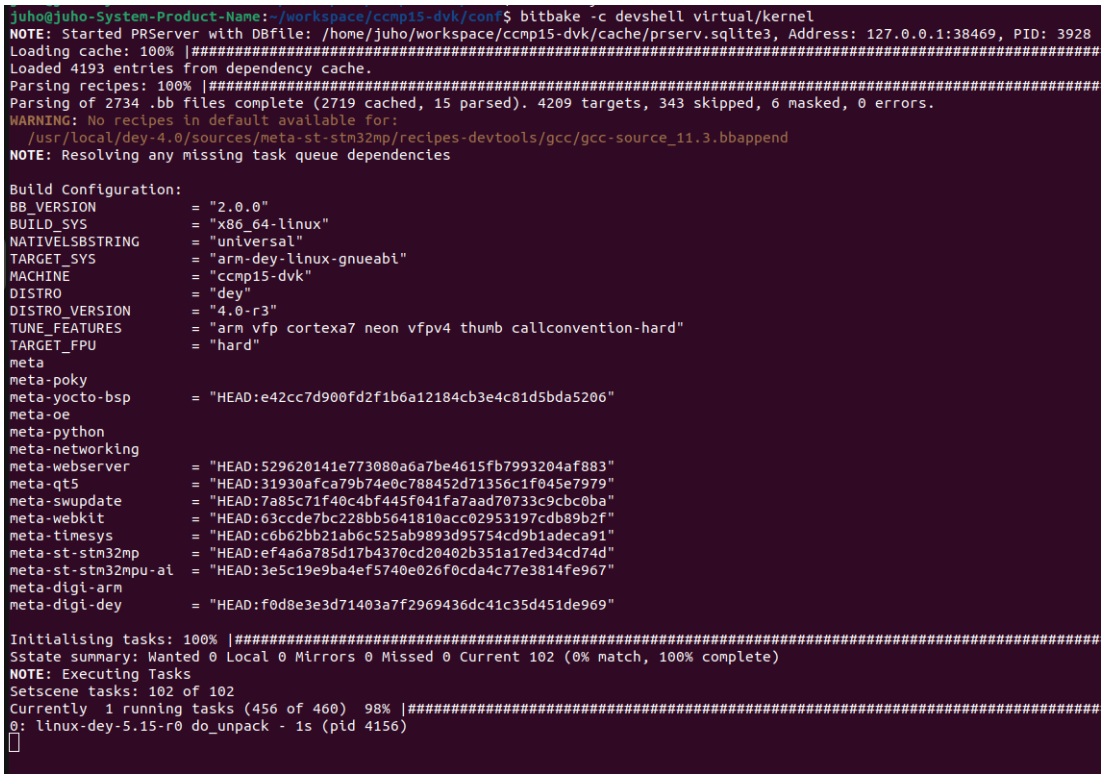
Projektin olennainen osa on Cortex-M4 prosessori ja siinä suoritettava GPIO-pinniä ohjaava ohjelma. Oletuksena dey-image-webkit niminen Linux-järjestelmätiedosto ei tue Cortex-M4 prosessoria. On muokattava Linux-ytimen lähdekoodia, jotta bitbake-prosessilla koottava Linux-järjestelmätiedosto tukisi Cortex-M4 prosessoria. Suorittimen käyttöönotto onnistuu muokkaamalla device-tree:tä. Projekti-hakemistosta löytyy hakemisto dts, jossa on ccmp15.dtsi-tiedosto, johon lisäämällä kuvassa 19 olevat tiedot devshell-ohjelmalla bitbake-prosessilla koottava Linux-järjestelmätiedosto tukee Cortex-M4 prosessoria.

```
workspace/ccmp15-dvk/tmp/work-shared/ccmp15-dvk/kernel-source/arch/arm/boot/dts
```

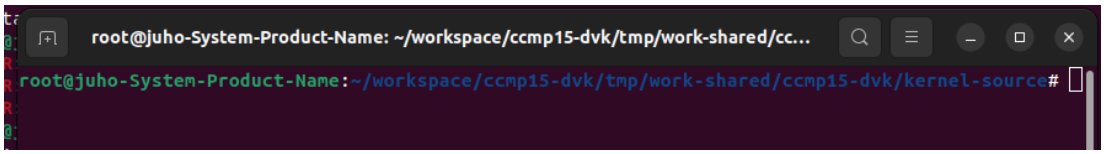
*Kuva 13. Polku, josta ccmp15.dtsi löytyy kehitysympäristössäni.*

```
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk$ source dey-setup-environment
Digi Embedded Yocto provides the following image recipes:

  * dey-image-webkit: graphical WebKit image

    By default the image is Wayland-based so it provides a full Weston
    desktop environment.

  * dey-image-qt: graphical QT image

    By default the image is Wayland-based so it provides a full Weston
    desktop environment.

  * core-image-base: a console-only image

    Expansion of native core-image-base by including all the support for the
    target device hardware like firmware files, rootfs customizations, etc.
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk$ bitbake -c devshell virtual/kernel
```

*Kuva 14. Devshell-ohjelman aloitus.*




*Kuva 15. Näkymä kun devshell ohjelma on alkamassa.*




*Kuva 16. Näkymä, kun ohjelma on käynnistynyt.*


*Kuva 17. Devshell ohjelmassa ccmp15.dtsi-tiedostoa voidaan muokata teks-tieditorilla.*

```
#include "stm32mp15-m4-srm.dtsi"
#include "stm32mp15-m4-srm-pinctrl.dtsi"
.....
&m4_rproc {
   memory-region = <&retram>, <&mcuram>, <&mcuram2>, <&vdev0vring0>, <&vdev0vring1>, <&vdev0buffer>, <&mcu_rsc_table>;
   mboxes = <&ipcc 0>, <&ipcc 1>, <&ipcc 2>;
   mbox-names = "vq0", "vq1", "shutdown";
   interrupt-parent = <&exti>; interrupts = <68 1>;
   wakeup-source;
   status = "okay";
};
```

*Kuva 18. Ccmp15.dtsi-tiedostoon devshell-ohjelmalla lisättävät device-tree-tiedot.*

```
/*
 * Copyright 2022, 2023 Digi International, Inc.
 *
 * The code contained herein is licensed under the GNU General Public
 * License. You may obtain a copy of the GNU General Public License
 * Version 2 or later at the following locations:
 *
 * http://www.opensource.org/licenses/gpl-license.html
 * http://www.gnu.org/copyleft/gpl.html
 */

#include "stm32mp15-m4-srm.dtsi"
#include "stm32mp15-m4-srm-pinctrl.dtsi"
#include <dt-bindings/gpio/gpio.h>
#include <dt-bindings/mfd/st,stpmic1.h>
#include <dt-bindings/rtc/rtc-stm32.h>

/ {
        aliases {
                ethernet0 = &ethernet0;
                i2c1 = &i2c1;
                i2c2 = &i2c2;
                i2c3 = &i2c3;
                i2c4 = &i2c4;
                i2c5 = &i2c5;
                i2c6 = &i2c6;
                mmc0 = &sdmmc3; /* Wireless */
                serial0 = &uart4;
                serial1 = &usart1;
                serial2 = &usart3;
        };

        memory@c0000000 {
                device_type = "memory";
                reg = <0xc0000000 0x20000000>;
        };

        reserved-memory {
                #address-cells = <1>;
                #size-cells = <1>;
                ranges;

                mcuram2: mcuram2@10000000 {
                        compatible = "shared-dma-pool";
                        reg = <0x10000000 0x40000>;
                        no-map;
                };

                vdev0vring0: vdev0vring0@10040000 {
                        compatible = "shared-dma-pool";
                        reg = <0x10040000 0x1000>;
                        no-map;
                };
```

*Kuva 19. Ccmp15.dtsi-tiedosto, johon lisätty tarvittavat tiedot.*

```
234         brcmf: bcrmf@1 {
235                 reg = <1>;
236                 compatible = "brcm,bcm4329-fmac";
237
238                 brcm,broken_sg_support;
239                 brcm,sd_head_align = /bits/  16 <8>;
240                 brcm,sd_sgentry_align = /bits/ 16 <32>;
241         };
242 };
243
244 &m4_rproc {
245         memory-region = <&retram>, <&mcuram>, <&mcuram2>, <&vdev0vring0>,
246                         <&vdev0vring1>, <&vdev0buffer>, <&mcu_rsc_table>;
247         mboxes = <&ipcc 0>, <&ipcc 1>, <&ipcc 2>;
248         mbox-names = "vq0", "vq1", "shutdown";
249         interrupt-parent = <&exti>;
250         interrupts = <68 1>;
251         wakeup-source;
252         status = "okay";
253 };
254
255 &vrefbuf {
256         vdda-supply = <&vdda>;
257 };
258
259 &fmc {
260         pinctrl-names = "default", "sleep";
261         pinctrl-0 = <&ccmp15_fmc_pins_a>;
262         pinctrl-1 = <&ccmp15_fmc_sleep_pins_a>;
263         status = "okay";
264
265         nand-controller@4,0 {
```

*Kuva 20. Ccmp15.dtsi-tiedosto, johon lisätty tarvittavat tiedot.*

```
root@juho-System-Product-Name:~/workspace/ccmp15-dvk/tmp/work-shared/ccmp15-dvk/kernel-source/arch/arm/
boot/dts# exit
```

*Kuva 21. Exit komennolla suljetaan devshell-ohjelma.*

```
$ bitbake -C compile virtual/kernel
```

*Kuva 22. Komento, jolla Linux-ydin kootaan uudestaan.*

Kun ccmp15.dtsi-tiedosto on muokattu onnistuneesti devshell-ohjelmalla, on devshell-ohjelmasta poistuttava exit-komennolla ja kernel on koottava uudestaan.

### 3.3.5 Linux-järjestelmätiedoston asentaminen kohdelaitteeseen

Kun laite on tuotannossa ja käytössä kentällä sen Linux-ytimen ja juuritiedostojärjestelmän muuttaminen tai päivittäminen on mahdollista ohjelmistopäivityspaketin (SWU) avulla. SWU-paketit ovat säiliöitä, jotka paketoivat kaikki tarvittavat tiedostot ja metadatan, joita tarvitaan laitteen ohjelmistojen päivittämiseksi.

Projekti-hakemistossa dey-setup-environment-skriptin ajamisen jälkeen voidaan SWU-paketti luoda dey-image-webkit -Linux-järjestelmätiedostolle seuraavan mukaisella komennolla. Valmis SWU-paketti löytyy tiedostosijainnista:<projekti kansio>/tmp/deploy/images/ccmp15-dvk.

```
$ bitbake dey-image-webkit-swu
```

*Kuva 23. SWU-paketin luominen.*

```
juho@juho-System-Product-Name:~/workspace/ccmp15-dvk/tmp/deploy/images/ccmp15-dvk$ ls -l
total 628284
-rw-r--r-- 2 juho juho       926 loka    4  2023 altboot.scr
drwxr-xr-x 4 juho juho      4096 loka    4  2023 arm-trusted-firmware
-rw-r--r-- 2 juho juho      2295 loka    4  2023 boot.scr
-rw-r--r-- 2 juho juho    104633 loka    6  2023 ccmp157-dvk--5.15-r0.1-ccmp15-dvk-20231006094638.dt
lrwxrwxrwx 2 juho juho        52 loka    6  2023 ccmp157-dvk-ccmp15-dvk.dtb -> ccmp157-dvk--5.15-r0.
lrwxrwxrwx 2 juho juho        52 loka    6  2023 ccmp157-dvk.dtb -> ccmp157-dvk--5.15-r0.1-ccmp15-dv
-rw-r--r-- 2 juho juho    159657 loka    6  2023 config-5.15.118
-rw-r--r-- 2 juho juho   4536735 loka    4  2023 dey-image-recovery-initramfs-ccmp15-dvk-2023100409
-rw-r--r-- 2 juho juho      1323 loka    4  2023 dey-image-recovery-initramfs-ccmp15-dvk-2023100409
-rw-r--r-- 2 juho juho    466052 loka    4  2023 dey-image-recovery-initramfs-ccmp15-dvk-2023100409
lrwxrwxrwx 2 juho juho        79 loka    4  2023 dey-image-recovery-initramfs-ccmp15-dvk.cpio.gz.u-b
lrwxrwxrwx 2 juho juho        70 loka    4  2023 dey-image-recovery-initramfs-ccmp15-dvk.manifest ->
lrwxrwxrwx 2 juho juho        68 loka    4  2023 dey-image-recovery-initramfs-ccmp15-dvk.testdata.js
-rw-r--r-- 2 juho juho 207118336 syys    5  2023 dey-image-webkit-swu-ccmp15-dvk-20230905110307.swu
```

*Kuva 24. Kuvassa alimpana näkyy luotu SWU-paketti*

Kun luotu paketti halutaan asentaa kohdelaitteeseen, on se siirrettävä kehitysympäristöstä muistitikulle tai microsd-kortille ja tallennusväline on sitten liitettävä kohdelaitteeseen ja asennettava laiteelle kuvien 27-28 esittämällä tavalla. Swu-paketti asennetaan kuvan 30 kuvaamalla tavalla.

```
root@ccmp15-dvk:/mnt# ls /dev
autofs              gpiochip8          mmcblk1          pts              rtc1
block               gpiochip9          mmcblk1p1        ram0             sda
bus                 hwrng              mqueue           ram1             sda1
char                i2c-2              mtd0             ram10            shm
console             i2c-4              mtd0ro           ram11            snd
cpu_dma_latency     i2c-6              mtd1             ram12            spidev0.0
disk                iio:device2        mtd1ro           ram13            stderr
dri                 initctl            mtd2             ram14            stdin
fb                  input              mtd2ro           ram15            stdout
fb0                 kmsg               mtd3             ram2             tee0
fd                  log                mtd3ro           ram3             teepriv0
full                loop-control       mtd4             ram4             tty
galcore             loop0              mtd4ro           ram5             tty0
gpiochip0           loop1              mtd5             ram6             tty1
gpiochip1           loop2              mtd5ro           ram7             tty10
gpiochip2           loop3              mtd6             ram8             tty11
gpiochip3           loop4              mtd6ro           ram9             tty12
gpiochip4           loop5              null             random           tty13
gpiochip5           loop6              ppp              rfkill           tty14
gpiochip6           loop7              ptmx             rtc              tty15
gpiochip7           mem                ptp0             rtc0             tty16
```

*Kuva 25. Kohdelaitteen dev-hakemisto. Laitteeseen kiinnitetty muistitikku on nimeltään sda1.*

Kohdelaitteessa (CCMP15-DVK) on tehtävä hakemisto, johon tikulla olevat tiedot voidaan asettaa mount-komennolla kuvan 28 osoittamalla tavalla.

```
root@ccmp15-dvk:/mnt# mkdir sda
```

*Kuva 26. Sda-nimisen kansion luominen kohdelaitteeseen mnt-hakemiston alle.*

```
root@ccmp15-dvk:/# mount /dev/sda1 /mnt/sda
```

*Kuva 27. dev/sda1 laitteen asettaminen /mnt/sda hakemistoon.*

```
root@ccmp15-dvk:/# ls -l /mnt/sda
-rwxrwx---    1 root     disk        175800 Mar  4 07:15 A7_Ohjelma
-rwxrwx---    1 root     disk       2833624 Mar  3 11:26 CM4_Ohjelma.elf
-rwxrwx---    1 root     disk     207372288 Oct  9 10:27 dey-image-webkit-swu-ccmp15-dvk-20231009102650.swu
```

*Kuva 28. /mnt/sda hakemiston sisältö asettamisen jälkeen.*

```
root@ccmp15-dvk:/# update-firware /mnt/sda/dey-image-webkit-swu-ccmp15-dvk-20231009102650.swu
```

*Kuva 29. SWU-paketin asennus.*

## 3.4 Ohjelmien tekeminen Cortex-A7 ja Cortex-M4 suorittimille

### 3.4.1 Ohjelmistodemon vaatimukset

Cortex-A7 suorittimella suoritettava ohjelman on vastattava käyttöliittymästä, koska se on niin sanottu pääprosessori ja se ohjaa Cortex-M4 prosessoria. Cortex-M4 prosessorilla suoritettavan ohjelman on vastattava GPIO-pinnin ohjauksesta, koska se kykenee ajallisesti paljon tarkempaan ohjaukseen kuin Cortex-A7 prosessori. Ohjelmien on myös kommunikoitava keskenään.

Annostelijoita ohjataan ajastetulla on/off tekniikalla eli annostelija syöttää ruokaa kaloille tai ei.

Käyttäjä asettaa tiedot:

- Minimi tauko (off)
    - Käyttäjä voi asettaa minimin (1- 200 s)
    - Tauon pituudella muutetaan annostelun määrää
- Pulssi (on)
    - Käyttäjä määrää pulssin pituuden (1- 250 s) Oletusarvo 5s
    - Pulssi ajetaan aina kokonaan. Ei lyhennetä eikä venytetä
- Päivä-annos
    - Käyttäjä asettaa päiväannoksen kilogrammoina
    - Se voidaan asettaa kolmen desimaalin tarkkuudella
- Annostelunopeus
    - Käyttäjä asettaa annostelunopeuden (g/s)
    - Se voidaan asettaa kolmen desimaalin tarkkuudella
- Jaksojen määrä
    - Käyttäjä voi asettaa 1-8 ruokintajaksoa
    - Joka jaksolla on aloitusaika ja lopetusaika
    - Jakson pituus vähintään 60 sekuntia
    - Joka jaksolle voidaan asettaa, kuinka monta prosenttia päivän annoksesta siinä ruokitaan.

Annostellessa:

- Joka sekunti lisätään päivän ruokittuihin
- Ennen pulssin alkamista tarkistetaan, onko päivän annos tullut täyteen
    - Jos on niin ei annostella

### 3.4.2 Ohjelmien yhteistoiminta

Yksi haaste ohjelmistodemon tekemisessä on se, kuinka eri suorittimilla pyörivät ohjelmat saadaan vaihtamaan tietoa keskenään ja kuinka apusuorittimelle saa ohjelman ladattua, käynnistettyä ja sammutettua. Linux maailmassa remoteproc ja rpmsg kehyksiin kuuluu koodikirjasto, jolla voidaan lähettää merkkijonomuotoista tietoa suorittimien jakamalla muistialueella sijaitsevalle vring-tyyppiselle muistipuskurillle. Mikrokontrolleri maailmassa OpenAmp-projekti tarjoaa lähes vastaavan toiminnallisuuden tarjoavan koodikirjaston. Internetin syövereistä löytyi sopivat funktiot niin Cortex-A7 kuin Cortex-M4 prosessoreilla suortitettaviin ohjelmiin ja tietojen lukeminen muistipuskureilta onnistui niiden avulla. Kuvissa näkyy ohjelmistodemossa tarvitut koodikirjastot ja funktiot.

main.c

```c
#include <math.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/rpmsg.h>
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <string.h>
#include <time.h>
#include <sys/time.h>
#include <locale.h>
#include <ctype.h>
#include "cJSON.h"



const char sepHourMin[2] = ":";

int askInt(char whatToAsk[40]) {
    int num;
    printf("%s: ",whatToAsk);
    if (scanf("%d", &num) != 1) {
        printf("Error: Invalid input\n");
        exit(1);
    }
    return num;
}
```

*Kuva 30. (Cortex-A7 ohjelma) Sisällyttämällä linux/kernel.h-, linux/module.h ja linux/rpmsg.h tiedostot main.c-lähdekooditiedostoon. Saadaan etäprosessorille tietojen lähettämiseen tarvittavat funktiot käyttöön.*

```c
int openTtyRpmsg(int ttyNb)
{
    int FdRpmsg;
    char devName[50];
    sprintf(devName, "/dev/ttyRPMSG%d", ttyNb);
    FdRpmsg = open(devName, O_RDWR |  O_NOCTTY | O_NONBLOCK);
    if (FdRpmsg < 0) {
        printf(" Error opening ttyRPMSG%d, err=-%d\n", ttyNb, errno);
        return (errno * -1);
    }
    return FdRpmsg;
}
```

*Kuva 31. (Cortex-A7 ohjelma) openTtyRpmsg-funktiolla "avataan" Rpmsg-kanava.*

```
370     int channel0 = openTtyRpmsg(0);
```

*Kuva 32. (Cortex-A7 ohjelma) rpmsg-0 -kanavan avaaminen.*

```
74  int closeTtyRpmsg(int FdRpmsg)
75  {
76      close(FdRpmsg);
77      return 0;
78  }
```

*Kuva 33. (Cortex-A7 ohjelma) closeTtyRpmsg-funktiolla suljetaan kanava.*

```
79  int writeTtyRpmsg(int FdRpmsg, int len, char* pData)
80  {
81      int result = 0;
82      if (FdRpmsg < 0) {
83          printf("CA7 : Error writing ttyRPMSG, fileDescriptor is not set\n");
84          return FdRpmsg;
85      }
86
87      result = write(FdRpmsg, pData, len);
88      return result;
89  }
```

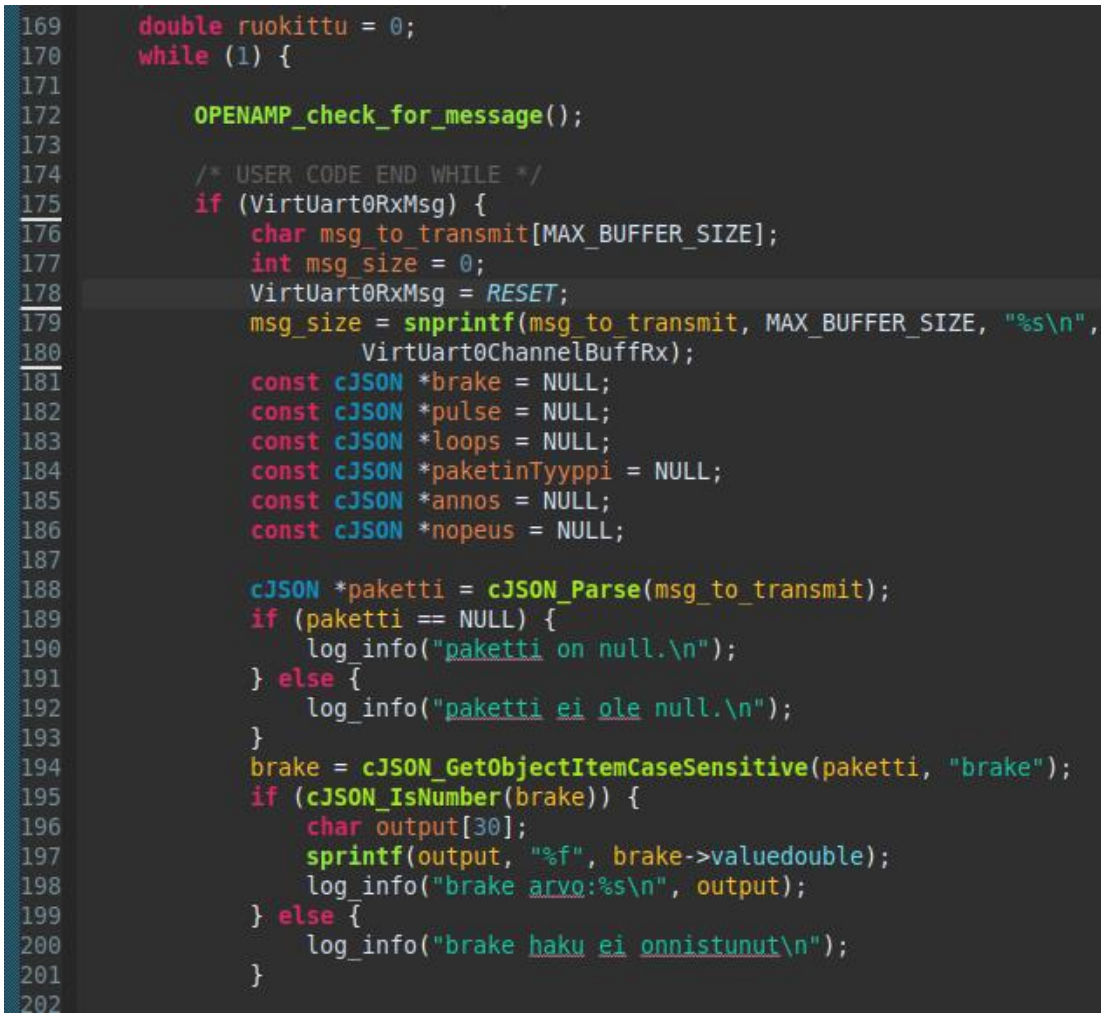
*Kuva 34. (Cortex-A7 ohjelma) WriteTtyRpmsg-funktiolla kirjoitetaan muistipuskurille merkkijonomuotoista tietoa.*

```
90  int readTtyRpmsg(int FdRpmsg, int len, char* pData)
91  {
92      int byte_rd, byte_avail;
93      int result = 0;
94      if (FdRpmsg < 0) {
95          printf("CA7 : Error reading ttyRPMSG, fileDescriptor is not set\n");
96          return FdRpmsg;
97      }
98      ioctl(FdRpmsg, FIONREAD, &byte_avail);
99      if (byte_avail > 0) {
100         //printf("byte_avail = %d\n",byte_avail);
101         if (byte_avail >= len) {
102             byte_rd = read(FdRpmsg, pData, len);
103             //printf("byte_rd1 = %d\n",byte_rd);
104         } else {
105             byte_rd = read(FdRpmsg, pData, byte_avail);
106             //printf("byte_rd2 = %d\n",byte_rd);
107         }
108         result = byte_rd;
109     } else {
110         result = 0;
111     }
112     return result;
113 }
```

*Kuva 35. (Cortex-A7 ohjelma) readTtyRpmsg-funktiolla luetaan merkkijonomuotoista tietoa muistipuskurilta.*

```
370     int channel0 = openTtyRpmsg(0);
371     int num = 0;
372     time_t aikaNyt;
373     cJSON *msgFromM4;
374     while (num < periodCntI) {
375         aikaNyt = time(NULL);
376         aikaNyt= aikaNyt+7200;//koska UTC aika on 2h jäljessä
377         if (aikaNyt >= messages[num].send) {
378             printf("Lähetetään M4:lle viesti: %s \n", messages[num].msg);
379             printf("\n");
380             printf("\n");
381             writeTtyRpmsg(channel0, strlen(messages[num].msg),
382                     messages[num].msg);
383             num = num + 1;
384         }
385         char rpmsgBuff[200];//tähän readTtyRpmsg sijoittaa m4:n lähettämän viestin
386         int readRet;
387         readRet = readTtyRpmsg(channel0, 200, rpmsgBuff);
388         if (readRet > 0) {
389             msgFromM4 = cJSON_Parse(rpmsgBuff);
390             if (msgFromM4 == NULL) {
391                 printf("m4 viestin purku ei onnistunut\n");
392             } else {
393                 printf("Viesti m4:ltä vastaanotettu.\n");
394             }
395             struct tm *aikaNytTm;
396             t = time(NULL);
397             t = t +7200;//koska UTC 2h jäljessä
398             aikaNytTm = localtime(&t);
399             char aikaStr[50];
400             strftime(aikaStr, 50, "%X", aikaNytTm);
401             printf("%s \n",aikaStr);
```

*Kuva 36. (Cortex-A7 ohjelma) Kuvassa näkyy muun muassa, kuinka writeTtyRpmsg- ja readTtyRpmsg-funktioita voidaan käyttää.*



```
169     double ruokittu = 0;
170     while (1) {
171
172         OPENAMP_check_for_message();
173
174         /* USER CODE END WHILE */
175         if (VirtUart0RxMsg) {
176             char msg_to_transmit[MAX_BUFFER_SIZE];
177             int msg_size = 0;
178             VirtUart0RxMsg = RESET;
179             msg_size = snprintf(msg_to_transmit, MAX_BUFFER_SIZE, "%s\n",
180                     VirtUart0ChannelBuffRx);
181             const cJSON *brake = NULL;
182             const cJSON *pulse = NULL;
183             const cJSON *loops = NULL;
184             const cJSON *paketinTyyppi = NULL;
185             const cJSON *annos = NULL;
186             const cJSON *nopeus = NULL;
187
188             cJSON *paketti = cJSON_Parse(msg_to_transmit);
189             if (paketti == NULL) {
190                 log_info("paketti on null.\n");
191             } else {
192                 log_info("paketti ei ole null.\n");
193             }
194             brake = cJSON_GetObjectItemCaseSensitive(paketti, "brake");
195             if (cJSON_IsNumber(brake)) {
196                 char output[30];
197                 sprintf(output, "%f", brake->valuedouble);
198                 log_info("brake arvo:%s\n", output);
199             } else {
200                 log_info("brake haku ei onnistunut\n");
201             }
202
```

*Kuva 37. (Cortex-M4 ohjelma.) Kuvassa näkyy kuinka viesti luetaan muistipuskurilta ja formatoidaan JSON-muotoon ja sitten puretaan tietoa JSON-paketista.*

Seuraava ongelma ohjelmistodemoa tehtäessä oli, kuinka muistipuskureilta luettavat merkkijonot voidaan muuntaa rakenteiseksi tiedoksi. Ratkaisu tähän oli muuntaa lähettävä tieto JSON-tyyppiseen muotoon, jolloin merkkijonoon pystyi sisällyttämään kokonais- ja desimaalilukuja, string-tyyppistä tietoa ja listoja, joissa oli mahdollista olla lukuja ja merkkijonoja. Ei ole järkevää keksiä pyörää uudestaan, joten sitten piti etsiä JSON-kirjasto, joka oli mahdollista kääntää CCMP15-laitteelle. JSMN-kirjastossa oli liian suppea toiminnallisuus ja JSON-C-kirjastoa ei saatu loppuen lopuksi toimimaan, koska sitä ei pystynyt kääntämään CCMP15-laitteelle. Onneksi löytyi riittävän monipuolisen toiminnallisuuden omaava JSON-kirjasto nimeltä cJSON, eikä tarvinnut alkaa kirjoittamaan uutta kirjastoa. cJSON-kirjaston funktioilla pystyi vaivattomasti muuntamaan

C-kielen char-array-tyyppistä tietoa JSON-muotoon ja JSON-muodosta takaisin char-array tyyppiseksi.

```
327     struct PacketToM4Info messages[8];
328     for (int i = 0; i < periodCntI; i++) {
329         char *annosteluOhjelmaS = NULL;
330         cJSON *annosteluOhjelma = cJSON_CreateObject();
331         cJSON *pulseJson = NULL;
332         cJSON *brakeJson = NULL;
333         cJSON *dayPulsesJson = NULL;
334         cJSON *paketinTyyppi = NULL;
335         cJSON *nopeus = NULL;
336         cJSON *annos = NULL;
337         pulseJson = cJSON_CreateNumber(PeriodPrograms[i].pulseLen);
338         brakeJson = cJSON_CreateNumber(PeriodPrograms[i].taukoLen);
339         dayPulsesJson = cJSON_CreateNumber(PeriodPrograms[i].loops);
340         paketinTyyppi = cJSON_CreateNumber(1); //jos yksi niin paketi
341         nopeus = cJSON_CreateNumber(feedSpeed);
342         annos = cJSON_CreateNumber(dayDose);
343         cJSON_AddItemToObject(annosteluOhjelma, "pulse", pulseJson);
344         cJSON_AddItemToObject(annosteluOhjelma, "brake", brakeJson);
345         cJSON_AddItemToObject(annosteluOhjelma, "pulses", dayPulsesJs
346         cJSON_AddItemToObject(annosteluOhjelma, "paketinTyyppi", paket
347         cJSON_AddItemToObject(annosteluOhjelma, "dayDose", annos);
348         cJSON_AddItemToObject(annosteluOhjelma, "feedSpeed", nopeus);
349         annosteluOhjelmaS = cJSON_PrintUnformatted(annosteluOhjelma);
```

*Kuva 38. (Cortex-M4 ohjelma.)  Kuvassa näkyy kuinka cJSON-objekti ja sinne lisättäviä tietoja luodaan sekä kuinka cJSON-objekti muutetaan merkkijonomuotoon.*

.

```
double dayDoseG = annos->valuedouble * 1000.0;
double nopeusD = nopeus->valuedouble;
double pulseD = pulse->valuedouble;
double pulseG = nopeusD*pulseD;
log_info("pulseG = %f \n",pulseG);
//double ruokittu = 0; siirretään ulos luupista
if (paketinTyyppi->valueint == 1) {
    log_info("Paketin tyyppi on 1\n");
    for (int i = 0; i < loops->valueint; i++) {
        log_info("loopissa ollaan %d kierroksella. \n",i);
        BSP_LED_Off(LED2);
        HAL_Delay(brake->valuedouble * 1000.0);
        if (ruokittu >= dayDoseG) {
            char ruokintaValmisA7[50];
            sprintf(ruokintaValmisA7, "Ruokinta valmis! Ruokittu %.2fg/%.2fg \n", ruokittu,
                    dayDoseG);
            int ruokintaValmisLen = strlen(ruokintaValmisA7);
            VIRT_UART_Transmit(&huart0, ruokintaValmisA7,
                    ruokintaValmisLen);
            log_info(
                    "Päivän ruokinta annos ruokittu(%.2f grammaa). \n",
                    ruokittu);
            break;
        }
        BSP_LED_On(LED2);
        HAL_Delay(pulse->valueint * 1000);
        BSP_LED_Off(LED2);
        ruokittu = ruokittu + pulseG;
        cJSON *msgToA7 = cJSON_CreateObject();
        if (msgToA7 == NULL) {
            log_info("msgToA7 = null");
        }
        cJSON *payload = NULL;
        char *msgToA7S = NULL;
        char viestiA7[50];
        sprintf(viestiA7, "Tänään ruokittu %.2fg/%.2fg \n",
```

*Kuva 39. (Cortex-M4 ohjelma.) Kuvassa näkyy, kuinka tietoa lähetetään muistipuskurille ja kuinka GPIO-pinni voidaan laittaa päälle ja pois.*

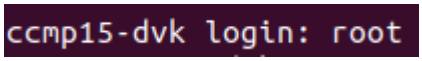
### 3.4.3 Projektien perustaminen Eclipse IDE:llä ja STM32CubeIDE:llä

Ccmp15-laitteeseen voidaan tehdä Eclipse IDE:llä ohjelmia monella tavalla. Tässä projektissa Cortex-A7 prosessorille tehtiin C-kielinen ohjelma, tekemällä Eclipse IDE:ssä managed makefile-projekti. Tällä tavoin Eclipse IDE luo kaikki ohjelman kohdelaitteelle kääntämiseen tarvittavat makefilet automaattisesti. Ohjelma voidaan kääntää vaivattomasti kohdelaitteelle, koska Eclipse IDE:en asennettu Digi Application Development Environment -lisäosapaketti sisältää kaikki CCMP15-laitteelle ohjelman kääntämiseen tarvittavat tiedot ja työkalut.

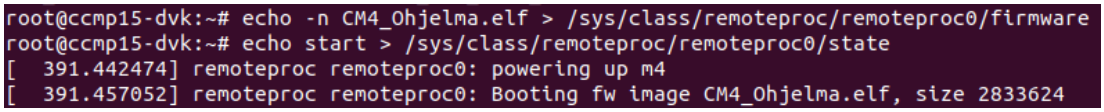
STM32CubeIDE:llä projektia perustaessa täytyy ensiksi valita mille piiriarkkitehtuurille ohjelmaa ollaan tekemässä. CCMP15-laitteessa on STM32MP157CAC3 tai STM32MP157CAC3T piiri. Tässä projektissa aloitin

Cortex-M4 prosessorille tehtävän ohjelman tekemisen OpenAMP_TTY_echo nimisen esimerkkiohjelman pohjalta, koska siinä oli valmiina kaikki viestin lähettämiseen ja vastaanottamiseen sekä GPIO-pinnien ohjaukseen tarvittava konfiguraatio.

## 3.5 Sovelluksen toiminnan kuvaus




*Kuva 40. Kuvassa näkyy, kuinka laitteelle kirjaudutaan. Käyttäjänimi on tässä tapauksessa ”root”, eikä salasanaa tarvitse antaa.*



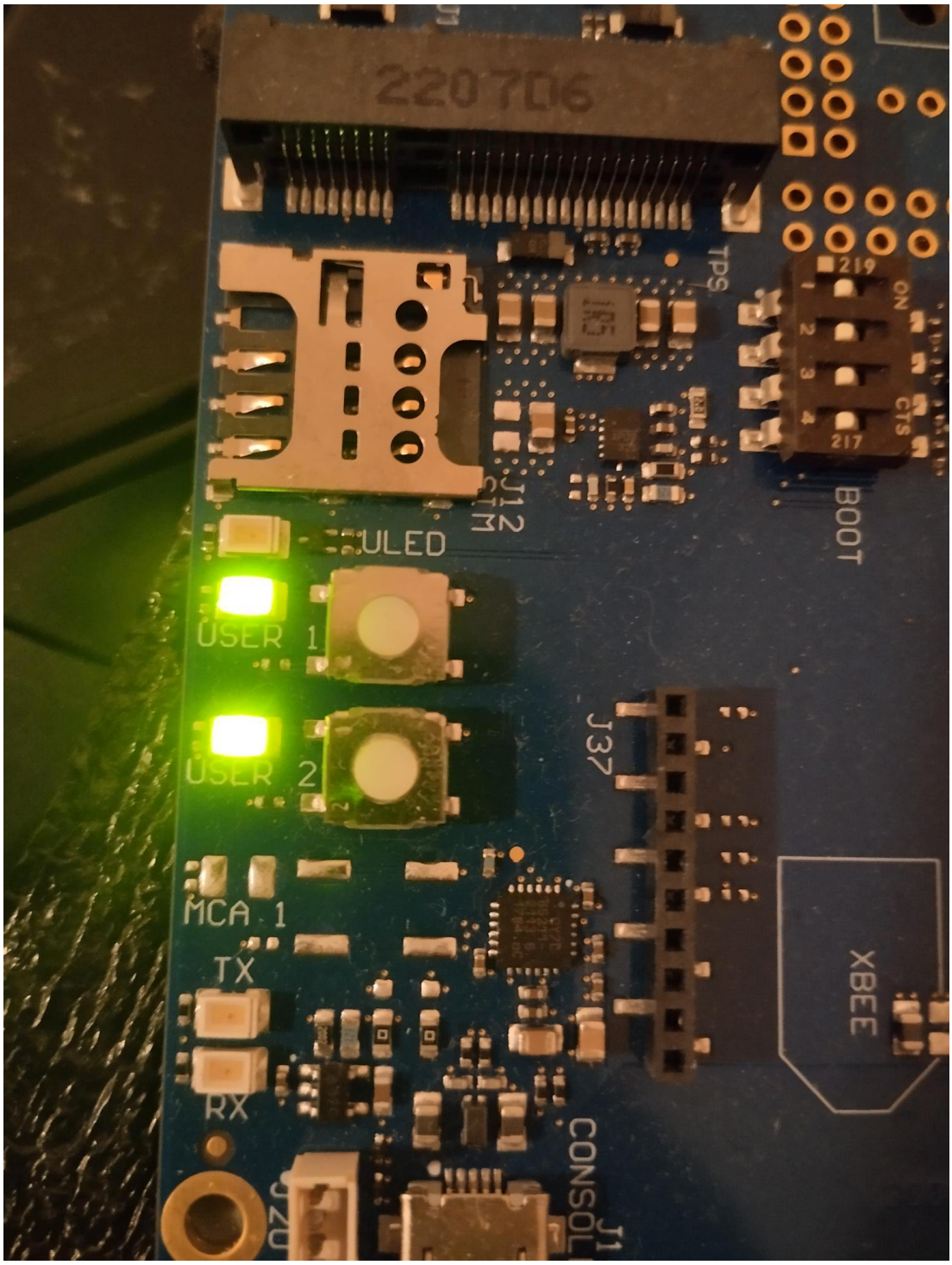

*Kuva 41. Kuvassa näkyy, kuinka Cortex-M4 -ohjelma käynnistetään.*

```
root@ccmp15-dvk:/mnt/sda# ./A7_Ohjelma


Mon Mar  4 12:56:37 2024
Syötä ruokintanopeus(g/s): 6.25
Syötä päivän ruokinta-annos(kg): 4
Syötä minimi tauon pituus(1-200s): 2
Syötä pulssin pituus(1-250s): 20
Jaksotetaanko annostelua(yes or no):yes
Kuinka moneen jaksoon ruokinta jaetaan?(0-8): 4

Syötä jakson 1 painotus: 25
Syötä jakson 1 aloitusaika: 13:00

Syötä jakson 2 painotus: 25
Syötä jakson 2 aloitusaika: 13:04

Syötä jakson 3 painotus: 25
Syötä jakson 3 aloitusaika: 13:08

Syötä jakson 4 painotus: 25
Syötä jakson 4 aloitusaika: 13:12
Syötä ruokinnan lopetusaika(HH:MM): 13:16


Paketti 1. Lähetysaika: 13:00:00.
Paketin sisältö: {"pulse":20,"brake":10,"pulses":8,"paketinTyyppi":1,"dayDose":4,"feedSpeed":6.25}
Paketti 2. Lähetysaika: 13:04:00.
Paketin sisältö: {"pulse":20,"brake":10,"pulses":8,"paketinTyyppi":1,"dayDose":4,"feedSpeed":6.25}
Paketti 3. Lähetysaika: 13:08:00.
Paketin sisältö: {"pulse":20,"brake":10,"pulses":8,"paketinTyyppi":1,"dayDose":4,"feedSpeed":6.25}
Paketti 4. Lähetysaika: 13:12:00.
Paketin sisältö: {"pulse":20,"brake":10,"pulses":8,"paketinTyyppi":1,"dayDose":4,"feedSpeed":6.25}


Lähetetään M4:lle viesti: {"pulse":20,"brake":10,"pulses":8,"paketinTyyppi":1,"dayDose":4,"feedSpeed":6.25}


Viesti m4:ltä vastaanotettu.
13:00:30
Tänään ruokittu 125.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:01:00
Tänään ruokittu 250.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:01:30
Tänään ruokittu 375.00g/4000.00g

Viesti m4:ltä vastaanotettu.
```

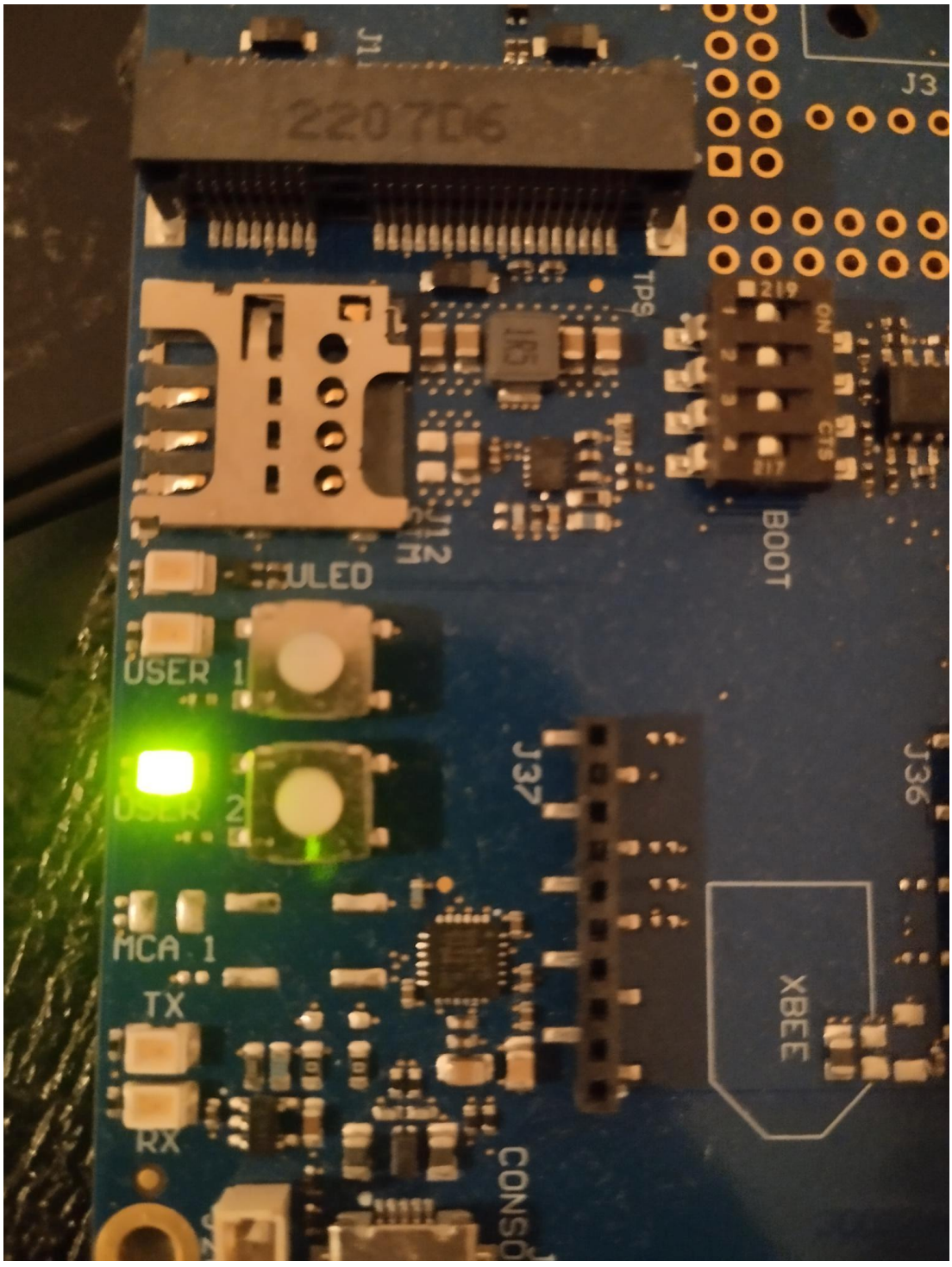
*Kuva 42. Kuvassa näkyy kuinka Cortex-A7 ohjelma käynnistetään ja miltä ohjelman käyttöliittymä näyttää.*

```
Tänään ruokittu 2625.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:11:00
Tänään ruokittu 2750.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:11:30
Tänään ruokittu 2875.00g/4000.00g

Lähetetään M4:lle viesti: {"pulse":20,"brake":10,"pulses":8,"paketinTyyppi":1,"dayDose":4,"feedSpeed":6.25}


Viesti m4:ltä vastaanotettu.
13:12:00
Tänään ruokittu 3000.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:12:30
Tänään ruokittu 3125.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:13:00
Tänään ruokittu 3250.00g/4000.00g

Viesti m4:ltä vastaanotettu.
13:13:30
```

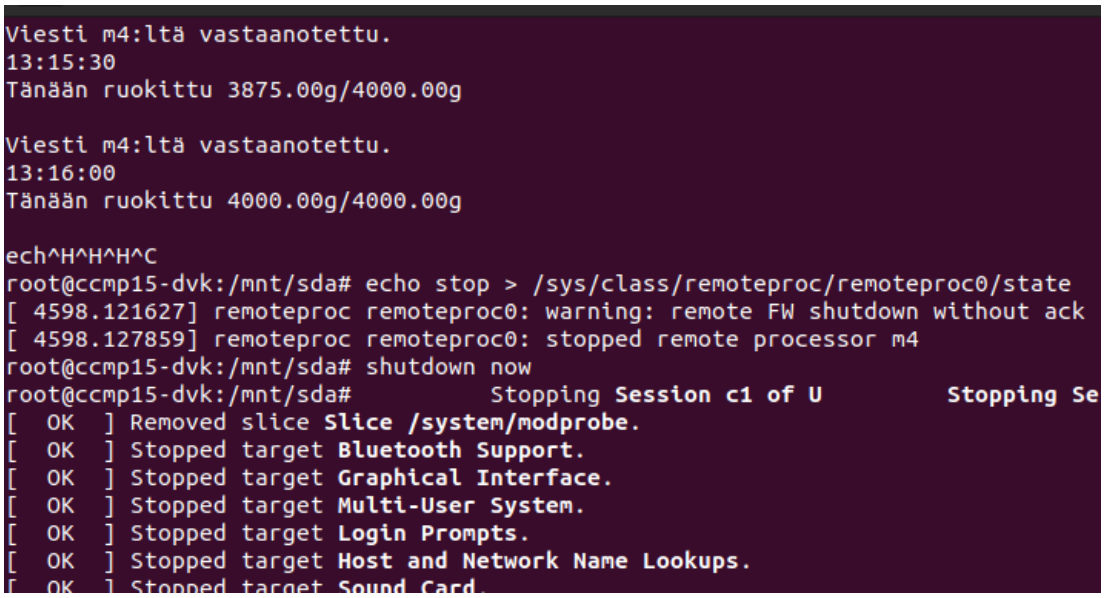
*Kuva 43. Kuvassa näkyy ohjelmien välistä tiedonvaihtoa.*




*Kuva 44. Kuvassa näkyy kuinka ledi USER 1 on päällä, kun GPIO-pinni on päällä.*




*Kuva 45. Kuvassa näkyy kuinka ledi USER 1 on pois päältä, kun GPIO-pinni on ohjattu pois päältä.*




*Kuva 46. Kuvassa näkyy, kuinka ohjelma* lopetetaan *ja kuinka laite sammutetaan.*

# 4 LOPPUPÄÄTELMA OHJELMISTODEMON KEHITYKSESTÄ JA SEN TOIMINNASTA

Opinnäytetyön aihe on mielestäni hyvä, koska prosessoreiden välistä kommunikointia remoteproc-kehyksen avulla on käsitelty vähän. En löytänyt Theseus.fi opinnäytetyöarkistosta yhtään asiaa käsittelevää opinnäytetyötä. En myöskään löytänyt internetistä yhtään sulautettuun Linux-järjestelmään tehtyä ohjelmaa, jossa käytetään remoteproc-kehystä. CCMP15 laite on myös melko uusi, eikä sitä käsittelevää materiaalia ole vielä paljoa.

Opinnäytetyö antaa mielestäni tyydyttävän vastauksen tutkimusongelmaan eli siihen, miten CCMP15-moduulilla voidaan tehdä sovelluksia sen Cortex-A7 ja Cortex-M4 prosessoreille siten, että ne pystyvät kommunikoimaan keskenään. Ohjelmat yhdessä muodostavat sovelluksen, joka tekee alkeellisen ruokinta-automaatin tehtävän. Ohjelmat ovat kuitenkin raakileita. Käyttöliittymä ei salli vääränlaisia syötteitä ja joissakin tapauksessa ohjelma laskee väärin, mutta koska opinnäytetyön tavoite oli tehdä ohjelmistodemo, eikä myyntiin menevä tuote, olen tyytyväinen lopputulokseen.

Opinnäytetyön tekeminen oli todella opettavainen prosessi. Ennen projektin alkua en ollut kuullut Yocto Project:ista tai Remoteproc kehyksestä (engl. framework). En myöskään ollut käyttänyt Linux-käyttöjärjestelmää tai tehnyt ohjelmia C-kielellä. Projektin aikana tuli vastaan ylitsepääsemättömän oloisia ongelmia, mutta projekti opetti, että niistäkin voi päästä yli kovalla työllä tai apua pyytämällä.

# LÄHTEET

Digi, (n.d.-a). Tuotteen hardware reference manual. Haettu 19.4.2024 osoitteesta https://www.digi.com/resources/documentation/digidocs/90002511/#reference/devboard/r_features-functionality_dvk.htm?TocPath=About%2520the%2520ConnectCore%2520MP15%2520DVK%257C_____1

Digi. (n.d.-b). CCMP15-laitteen Digi Embedded Yocto -manuaalin verkkosivut. Haettu 19.4.2024 osoitteesta https://www.digi.com/resources/documentation/digidocs/embedded/dey/4.0/ccmp15/index.html

Digi. (n,d.-c). CCMP15-laitteen Digi Embedded Yocto -manuaalin verkkosivut. Haettu 19.4.2024 osoitteesta https://www.digi.com/resources/documentation/digidocs/embedded/dey/4.0/ccmp15/yocto-system-development_r.html

Digi. (n.d.-d). CCMP15-laitteen Digi Embedded Yocto -manuaalin verkkosivut.. Haettu 19.4.2024 osoitteesta https://www.digi.com/resources/documentation/digidocs/embedded/dey/4.0/ccmp15/yocto-dade-set-up-environment_t.html

Red Hat. (n,d.-a) What is the Linux kernel?. Haettu 19.4.2024 osoitteesta https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel

Red Hat. (n,d.-b) What is the Linux kernel? Haettu 19.4.2024 osoitteesta https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel

St. (n.d.-a) STM32MP15 microprocessor. Haettu 19.4.2024 osoitteesta https://wiki.st.com/stm32mpu/wiki/STM32MP15_microprocessor

St. (n.d.-b). Linux remoteproc framework overview. Haettu 19.4.2024 osoitteesta https://wiki.st.com/stm32mpu/wiki/Linux_remoteproc_framework_overview

St. (n.d.-c). Linux RPMsg framework overview. Haettu 19.4.2024 osoitteesta https://wiki.st.com/stm32mpu/wiki/Linux_RPMsg_framework_overview

The Linux Foundation. (n,d.-a). Getting Started: The Yocto Project Overview . Haettu 19.4.2024 osoitteesta https://www.yoctoproject.org/development/technical-overview/

The Linux Foundation. (n,d-b). Project Overview. Haettu 19.4.2024 osoitteesta https://www.yoctoproject.org/about/project-overview/

The Linux Foundation. (n,d-c). Introducing the Yocto Project. Haettu 19.4.2024 osoitteesta https://docs.yoctoproject.org/overview-manual/yp-intro.html#components-and-tools

The Linux Foundation. (n,d-d) System Requirements. Haettu 19.4.2024 osoitteesta https://docs.yoctoproject.org/kirkstone/ref-manual/system-requirements.html

# LIITE:1 OPINNÄYTETYÖN PALAUTE 24.4.2024

Juho Simonen, Arvo-Tec Oy Ruokinta-automaatin prototyypin kehitys CCMP15-kehitysalustalla.

Opinnäytetyö kattaa annetun tehtävän kiitettävästi. Työ kertoo seikkaperäisesti, kuinka Digi Embedded Yocto käyttöjärjestelmä muokataan ja otetaan käyttöön CCMP15 laitteistolle sekä millaisilla työkaluilla asia hoidetaan. Lisäksi työ kertoo tehtävän aikana ilmenneistä muutamista sudenkuopista sekä kuinka ne kierretään. Tämä opinnäytetyö auttaa yritystä uuden järjestelmän suunnittelussa, komponenttien ja työkalujen valinnassa.

Terho Jalkanen
Automaatioinsinööri
Arvo-Tec Oy