Monitasotutkielma sulautetuista järjestelmistä

Teemu Matilainen

Opinnäytetyö
Toukokuu 2015
Tietotekniikka
Sulautetut järjestelmät ja elektroniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

Teemu Matilainen:
Monitasotutkielma sulautetuista järjestelmistä

Opinnäytetyö 34 sivua, joista liitteitä 7 sivua
Toukokuu 2015

Sulautetut järjestelmät koostuvat useista osioista ja tasoista. Näitä mahdollisia tasoja ovat mm. elektroniikka, mikrokontrolleriohjelmointi, tietoliikenne, Linux-ohjelmointi, erilaiset rajapinnat, valmiiden ohjelmistojen konfigurointi, web-ohjelmointi ja skriptikielet. Useat sulautettujen järjestelmien parissa työskentelevät insinöörit erikoistuvat yleensä jossain vaiheessa yhteen näistä tasoista, mutta kattava perusymmärrys eri osioista ja niiden yhteenliittämisestä on tärkeää alalla.

Tässä työssä käsitellään kaikki nämä tasot käytännön toteutuksen kautta. Työ pyrkii osoittamaan koulutuksen tuomaa osaamista ja toisaalta myös työn aikana kertynyttä tietotaitoa. Työ kattaa yksittäisten tasojen lisäksi kokonaisuuden rakentamisen näistä yksittäisistä tasoista ja niiden toteutusten yhteenliittymisen.

Järjestelmä on toteutettu Raspberry Pi –minitietokoneen ja PSoC-mikrokontrollerin ympärille. Lisäksi työssä on toteutettu logiikkatason muuttaja näiden kahden välille. Mikrokontrolleri lukee kahdelta lämpötila-anturilta lämpötilat ja välittää ne SPI-väylää pitkin minitietokoneelle, joka web-kameralta saamansa kuvan ohella päivittää ne verkkosivuille.

Asiasanat: sulautetut järjestelmät, tutkielma, mikrokontrolleri, raspberry pi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information technology
Embedded systems

Teemu Matilainen:
Multilevel Study of Embedded Systems

Bachelor's thesis 34 pages, appendices 7 pages
May 2015

Embedded systems consist of several parts and levels. Possible levels are electronics, microcontroller programming, data transmission, Linux programming, frameworks and APIs, configuration of third party software, web programming and script languages. Several engineers working with embedded systems specialize in one or more of these levels, but a certain level of understanding of all of these levels and their connections is important for everyone working in the field.

This thesis goes through all of these levels via the means of a practical application. The thesis is meant to present the skills brought by the studies, but also the knowledge gained while working on it. In addition to going through all of the levels it also covers building a whole system from them and how they connect to each other.

The system has been has been implemented using a Raspberry Pi minicomputer and a PSoC microcontroller as the basis. Also a logic level converter has been built between them. The microcontroller reads the values of two temperature sensorsa and sends them to the minicomputer via SPI bus. The computer updates the values with a picture from a webcam to a website.

Key words: embedded systems, study, microcontroller, raspberry pi

SISÄLLYS

LYHENTEET JA TERMIT

AD-muunnin	Muuttaa jännitetason digitaaliseksi arvoksi
GPIO	General Purpose Input Output, yleiskäyttöinen sisään- ja ulostuloportti
PSoC	Programmable System-on-chip, Cypress Semiconductorin 8-bittinen mikrokontrolleri
Raspberry Pi	Luottokortin kokoinen minitietokone
SPI-väylä	Serial Peripheral Interface, sarjamuotoinen oheislaiteväylä

1 JOHDANTO

Sulautetut järjestelmät koostuvat useista eri tasoista. Näihin tasoihin kuuluvat mm. elektroniikka, mikrokontrolleriohjelmointi, tietoliikenne, Linux-ohjelmointi, erilaiset rajapinnat, valmiiden ohjelmistojen konfigurointi, web-ohjelmointi ja skriptikielet. Sulautettujen järjestelmien projekti koostuu useimmiten vähintään muutamasta näistä tasosta, joskus jopa kaikista. Vaikka sulautettujen järjestelmien parissa työskentelevä insinööri usein erikoistuukin näistä tasoista yhteen tai useampaan, on perustason ymmärrys jokaisesta niistä tärkeää.

Tässä työssä tavoitteena on käsitellä kaikki nämä tasot käytännön toteutuksen kautta. Samalla työssä on tarkoituksena osoittaa ja syventää koulutuksen tuomaa osaamista jokaisesta näistä tasoista. Yksittäisten tasojen käsittelemisen ja toteuttamisen lisäksi työ kattaa myös tasojen liittymisen toisiinsa sekä kaikista niistä koostuvan kokonaisen järjestelmän rakentamisen.

Toiseksi tavoitteeksi työlle on määritelty tutkimuksen toteutuksellinen osuus eli tasoista koostuva järjestelmä ja sen toteutus. Järjestelmän on tarkoitus olla valvontalaitteisto, joka tuottaa julkaisemalleen verkkosivulle tietoa paikasta, muun muassa lämpötiloja ja web-kameran kuvaa. Lisäksi järjestelmän on suunniteltu tuottavan hälytyksen havaitessaan liikettä. Järjestelmä pyritään suunnittelemaan sellaiseksi, että se on myös muiden helppo toteuttaa.

2 OHJELMISTO-, LAITTEISTO- JA KOMPONENTTIVALINTA

Työn ajatuksena oli oppimistavoitteen lisäksi luoda järjestelmä, jonka perustiedot hallitseva alan ihminen pystyy itse edullisesti ja helposti kokoamaan ja laittamaan toimintakuntoon. Edullisuus ja käytön helppous olivat siis isoja kriteerejä laite- ja komponenttivalintaa tehtäessä. Toisaalta taas työn monitasoinen oppimis- ja tutkimustavoite vaikutti omalta osaltaan valintaprosessiin. Tutkimustavoitteen määrittämiä kriteerejä olivat muun muassa laitteiston tai ohjelmiston merkittävyys ja kyseisen asian vähäinen tuntemus.

Jotta muiden olisi helppo rakentaa oma vastaava järjestelmänsä, on laitteiden, komponenttien ja ohjelmistojen oltava suhteellisen yleisesti käytettyjä ja helposti saatavilla olevia. Myös maksuttomuuden ja avoimen lähdekoodin on katsottu olevan suuria etuja valintoja tehtäessä. Avoin lähdekoodi mahdollistaa myös ohjelmistojen muokkaamisen erilaisiin käyttötarpeisiin.

Nämä taustatekijät asettivat siis yleisiä kriteerejä, jotka koskivat kaikkia osioita ja toimivat pohjana laitteisto- ja komponenttivalinnalle, mutta myös jokainen valittava asia sisälsi luontaisesti omat valintaperusteensa. Vaikka kaikkea ei voi ottaa huomioon, on valintoja tehtäessä myös pyritty pohtimaan tulevaisuuden laajennusmahdollisuuksia ja erilaisia käyttökohteita. Valintatyö suoritettiin pääosin ennen käytännön toteutuksen aloittamista, mutta muutamiin valintoihin tuli työtä tehtäessä muutoksia.

2.1 Minitietokone

Minitietokoneeksi oli tarjolla lukuisia vaihtoehtoja, mukaan lukien Raspberry Pi, Beaglebone Black, Banana Pi, Odroid-U3, Udoo, Raxda Rock ja Hummingboard. Kaikki edellä mainitut minitietokoneet ja useat listan ulkopuolelle jääneistäkin olisivat kyseisen projektin toteutukseen muiden kriteerien osalta soveltuneet, joten valinta täytyi tehdä käyttäen erityisesti hintaa ja helppokäyttöisyyttä valintaperusteina.

Näistä kaikista minitietokoneista Raspberry Pi oli edullisin ja käytetyin. Kyseisen projektin asettamat vaatimukset suoritustehon ja liitäntöjen puolesta täyttyivät, ja jopa ylittyivät Raspberry Pi:ssä, joten korkeammalla hinnalla ei olisi saavutettu mitään merkit-

tävää lisäarvoa. Tarkkoja arvoja ei suoritustehovaatimuksille määritelty, mutta yksinkertaisen skriptin ja web-palvelimen pyörittämiseen komentoliittymäpohjaisessa Linux-jakelussa ei tarvita suurta suoritustehoa. Toisaalta osa vaihtoehdoista oli suoritusteholtaan tarpeettoman suuria. Vähäistä virrankäyttöä ei ole alun perin määritelty valintakriteeriksi, mutta se katsotaan kuitenkin eduksi. Siitä on hyötyä myös, jos järjestelmää aikoo käyttää aurinkokennoilla tai akuilla.

Raspberry Pi:stä on olemassa eri malleja, joista tähän työhön valittiin malli B. Valintaa tehtäessä vaihtoehtona oli myös malli A, mutta kyseisessä mallissa ei ole verkkoporttia ollenkaan, joten sen ei katsottu soveltuvan työhön. Tietysti malli A:ta voi käyttää langattomassa verkossa USB-sovittimen kautta, mutta kyseisessä mallissa on myös puolet vähemmän muistia. Ohjelmisto toimii kyllä myös tässä mallissa samalla tavalla, joten järjestelmän voi periaatteessa toteuttaa myös käyttäen mallia A.

Raspberry Pi on GPIO:n suhteen varsin riittävä, sillä pinnejä on enemmän kuin tarpeeksi, ja tuki SPI:lle on olemassa usean kirjaston avulla. Raspberry Pi ei sisällä AD-muunninta, mutta projektissa ei sille ollut tarvettakaan minitietokoneen päässä, sillä lämpötila-antureiden analogisen jännitteen lukeminen oli alun perinkin tarkoitettu tehtäväksi mikrokontrollerilla. Laitteessa on valmiiksi tuki SPI-väylälle, jonka kautta tietoliikenne on suunniteltu toteutettavaksi työssä.

Määriteltyjen kriteerien lisäksi Raspberry Pi:n eduksi lasketaan myös se, että sen ympärillä on iso yhteisö aktiivisia kehittäjiä ja käyttäjiä, mistä johtuen yleisiin ongelmiin saa foorumeilta apua, ja moneen asiaan on olemassa hyvät ohjeistukset. Tämä etu pätee sekä työn toteutusvaiheeseen että mahdollisten rakentelijoiden työhön.

Raspberry Pi ei sisällä sisäistä muistia ollenkaan, vaan käyttöjärjestelmä ja tiedostot sijaitsevat minitietokoneeseen liitettävällä MicroSD-kortilla. Tämä ominaisuus on kehitystyössä eduksi, sillä käyttöjärjestelmän, tiedostot ja niihin tehdyt muutokset saa helposti varmuuskopioitua toiselle muistikortille. Muistikortti tekee helpoksi myös järjestelmän monistamisen levykuvan jakamisen avulla. Järjestelmää rakentaville voi välittää vain levykuvan käyttöjärjestelmästä, johon on jo asennettu kaikki tarvittavat ohjelmistot ja asetukset.

Minitietokoneen ytimenä toimii Broadcom BCM2835 SoC. Piiri sisältää 700 MHz ARM-prosessorin, 512 MB muistia ja VideoCore IV -näytönohjaimen. Liitäntöinä minitietokoneessa on GPIO:n lisäksi kaksi USB-porttia, verkkoportti, ääniulostulo, RCA-video ja HDMI.

Työn toteutuksen aikana on Raspberry Pi:stä tullut uusia malleja, jotka ovat syrjäyttäneet vanhat mallit. Tässä työssä käytettyä mallia ei siis enää valmisteta, mutta sitä on silti myytävänä vielä paljon. Valmistajan mukaan uudet mallit ovat kuitenkin täysin yhteensopivia vanhojen kanssa, eli ohjelman pitäisi toimia myös uudemmissa versioissa. Uudemmissa malleissa on enemmän GPIO-pinnejä ja suorituskykyä, joten ne tarjoavat myös paremmat laajennusmahdollisuudet tulevaisuuden kehitystä varten.

2.2 Mikrokontrolleri

Mikrokontrollerin valintaa varten asetetut vaatimukset eivät olleet suuret, sillä käyttö ei ole kovin raskasta tai monipuolista. Yksinkertainen koodi mahtuu käytännössä minkä tahansa modernin mikrokontrollerin muistiin, ja suorituskyky riittää tähän käyttöön varsin hyvin. Lisäksi tätä projektia varten yksittäisen mikrokontrollerin hinta ei kalleimmillaankaan tule olemaan kovin montaa euroa.

Vaihtoehtoina mikrokontrollerin valinnassa oli lähinnä Cypressin PSoC:it ja Atmelin AVR-perhe. Valinta näiden kahden välillä oli vaikea, sillä molemmat sopivat projektiin yhtä hyvin ja täyttivät kaikki vaatimukset. Valinta kääntyi kuitenkin PSoC:in puoleen, sillä TAMK:issa on käytetty AVR:ää, kun taas PSoC oli suhteellisen vieras. Suorituskyvyltään saman luokan komponenteissa ei ollut suurta eroa hinnassa.

Helppokäyttöisyyden kriteeri tarkoittaa mikrokontrollerin osalta lähinnä sitä, että kotelointi ei voi olla pintaliitostyyppinen, vaan sen on oltava jalallinen eli käytännössä DIP-tyyppiä. Mahdollisia käyttäjän tekemiä ohjelmistollisia muutoksia varten PSoC:in kehitysympäristö on hieman helppokäyttöisempi verrattuna Atmelin vastaavaan. Sekä Atmelin että Cypressin mikrokontrollerit vaativat oman ohjelmointilaitteensa, joka on välttämätön järjestelmän rakentamiseen liittyvä hankinta.

PSoC:eja on laaja valikoima usealla eri kotelotyypeillä, pinnimäärällä, muistin laajuudella ja kellotaajuudella. Tähän työhön valittiin kriteerien mukaisesti DIP-koteloitu mal-

li. Kehitysalustan mukana tuli tähän tarkoitukseen ominaisuuksiltaan varsin riittävä CY8C27443. Kyseisessä mallissa on 16 KB Flash-muisti, 256 tavua RAM:ia sekä kaksitoista analogista ja kahdeksan digitaalista lohkoa käyttäjän moduuleille. (Cypress 2014, 1.)

2.3 Anturit

Lämpötila-anturina suunniteltiin ensin käytettävän yleistä LM35:ttä, mutta se hylättiin, koska kyseisellä anturilla nollan alapuolella olevien lämpötilojen mittaamiseen olisi tarvittu myös negatiivinen jännite. Toinen hylkäämisen peruste oli se, että tuolloin mikrokontrollerin jännitteenä oli tarkoitus vielä käyttää 3,3 volttia, eikä kyseinen anturi olisi soveltunut niin matalalle jännitteelle.

Lämpötila-anturiksi valikoitui lopulta Microchipin MCP9700-E, sillä kyseiselle anturille riittää käyttöjännitteeksi jopa 2,3 V. Lisäksi, toisin kuin valtaosa muista saman hinta- ja ominaisuusluokan antureista, se ei ole pintaliitoskomponentti, vaan käyttää TO-92-koteloa. (Microchip 2009, 3.)

Valitun anturin ulostulo on nollassa 500 mV, ja ulostulon lämpötilariippuvuus on 10 mV/°C (Microchip 2009, 3). Tämä tarkoittaa sitä, että esimerkiksi 20 °C -lämpötilassa anturin ulostulo on 700 mV ja -20 °C -lämpötilassa 300 mV. Tällöin myös pakkasasteet saadaan mitattua ilman tarvetta negatiiviselle jännitteelle. Yksinkertainen lämpötilariippuvuus helpottaa myös mittauksen järjestämistä ja todellisen ulostulon tarkistamista yleismittarilla.

Valintakriteereinä anturille oli siis kyky mitata nollan alapuolella olevia lämpötiloja ilman negatiivista jännitettä, helppokäyttöinen kotelointi ja yksinkertainen ulostulon logiikka sekä alun perin myös tarpeeksi matala käyttöjännite. Valittu sensori täyttää nämä kriteerit hyvin ja on silti edullinen valinta.

2.4 Web-kamera ja lisälaitteet

Web-kameran valinnassa käytettiin apuna yhteisön kokoamaa listaa Raspberry Pin kanssa yhteensopivista USB-käyttöisistä web-kameroista. Ominaisuuksilla ei kameran tapauksessa ole suurta merkitystä, kunhan resoluutio on kohtuullinen, ja laite toimii

Raspberry Pin kanssa. Jälleen kerran siis pääkriteerinä on käytettävä edullisuutta ja helppokäyttöisyyttä.

Lopulta listalta päätettiin hankkia kaksi eri kameraa vertailun vuoksi. Mallit olivat Microsoft LifeCam HD-3000 ja Creative Live! Cam Sync HD. Molemmissa on sama resoluutio, ja niiden pitäisi listan mukaan toimia hyvin Raspberry Pin kanssa. Periaatteessa web-kamera voi olla mikä tahansa yhteensopiva malli. Tämä on hyvä ominaisuus ottaen huomioon tavoitteen siitä, että kuka tahansa alaa tunteva voi rakentaa vastaavan järjestelmän.

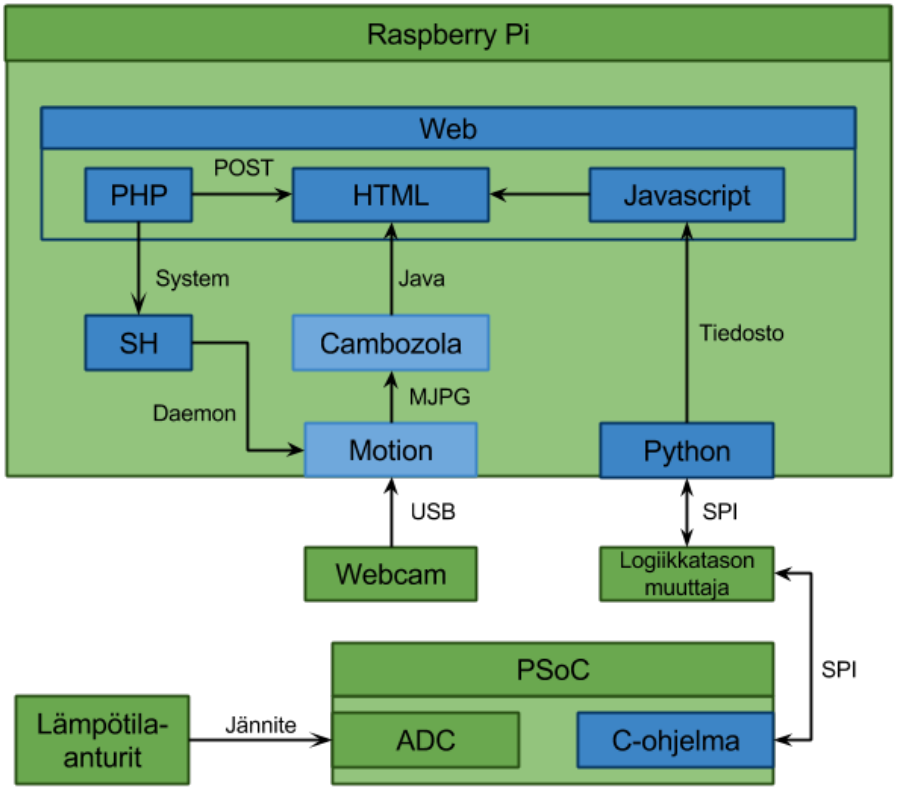
Havaitessaan liikettä ja hälyttäessään siitä oli järjestelmän tarkoitus myös tallentaa videokuvaa ja still-kuvia liikkeestä. Tätä tarkoitusta varten järjestelmään käytettiin myös valmiiksi olemassa olevaa USB-kovalevyä, jossa oli riittävän suuri tallennuskapasiteetti pidemmillekin videonpätkille. USB-kovalevyn suuren virran tarpeen vuoksi järjestelmään hankittiin myös virtalähteellinen USB-hubi. Tarve näille kuitenkin poistui kyseisen ominaisuuden jäädessä pois.

Raspberry Pi:lle on olemassa suuri valikoima suojakoteloita hyvin kevyistä malleista säänkestävien kautta aina raskaisiin jäähdytyksen sisältäviin koteloihin. Tässä käytössä kone ei kuitenkaan altistu sääolosuhteille, eikä se joudu jäähdytystä vaativiin suorituksiin. Oletuksena on, että laite on pääasiallisesti sisällä sillä poikkeuksella, että web-kamera ja joitain antureita saatetaan sijoittaa ulkopuolelle. Käytännössä siis mikä tahansa kotelo toimii tähän tarkoitukseen. Minitietokonetta tilattaessa RS:stä tilattiin samalla toimittajan oma, edullinen kotelomalli.

3 TOTEUTUS

Lopullinen laite koostuu lukuisista eri osioista, jotka ovat sekä ohjelmistollisia että laitteistollisia. Kaikki nämä osiot ovat näkyvillä kuvassa 1. Lisäksi osioiden liittyminen toisiinsa on ilmaistu nuolilla, ja nuolien läheisyydessä on liittymisen tyyppi.

Ohjelmistolliset osiot on merkitty sinisellä värillä ja laitteistolliset vihreällä. Tummemmalla sinisellä merkityt ohjelmistolliset osiot ovat itse tuotettuja, ja vaaleammalla merkityt valmiita ohjelmistoja, jotka on sisällytetty kokonaisuuteen.



Kuva 1. Lohkokaavio

Konkreettisia osioita laitteessa ovat web-kamera, Raspberry Pi –minitietokone, logiikkatason muuttaja, PSoC-mikrokontrolleri ja lämpötila-anturit. Näiden lisäksi on ohjelmistollisia osiota, joita ovat muiden tuottamat ilmaisen lisenssin Cambozola ja Motion sekä itse tuotetut BASH-skripti, Web-ohjelmisto, joka sisältää PHP:tä, HTML:ää ja Javascriptiä, Pythonskripti ja mikrokontrollerin C-ohjelma.

Toteutukseen kului lopulta huomattavasti aiottua pidempi aika lukuisista vastoinkäymisistä johtuen. Toteutuksellinen osuus oli tarkoitus saada kesän 2014 aikana valmiiksi tai ainakin hyvin pitkälle. Web-kameran kanssa ilmennyt ongelma vei alkukesästä pitkän ajan ja venytti muiden osioiden toteutusta syksyn puolelle. Syksyllä koulun jo alettua alkoi pitkä työskentely ongelman kanssa, jonka oletettiin olevan kommunikaatioongelma mikrokontrollerin ja minitietokoneen välillä, mutta joka paljastuikin aiheutuvan mikrokontrollerin käyttöjännitteestä.

Tämä luku ei ole järjestettynä kronologisesti, sillä osa osioista oli ajallisesti lomittain keskenään. Osiot on järjestetty loogisesti ’matalimmalta’ tasolta ’korkeimpaan’. Tämä tarkoittaa sitä, että ensin on käsitelty konkreettisempia asioita, kuten elektroniikkaa, sulautettua ohjelmointia, tietoliikennettä ja Linux-ohjelmointia. Tämän jälkeen käsitellään vähemmän laitteistoon liittyviä asioita, kuten web-ohjelmointia ja skriptejä sekä ohjelmistojen konfigurointia.

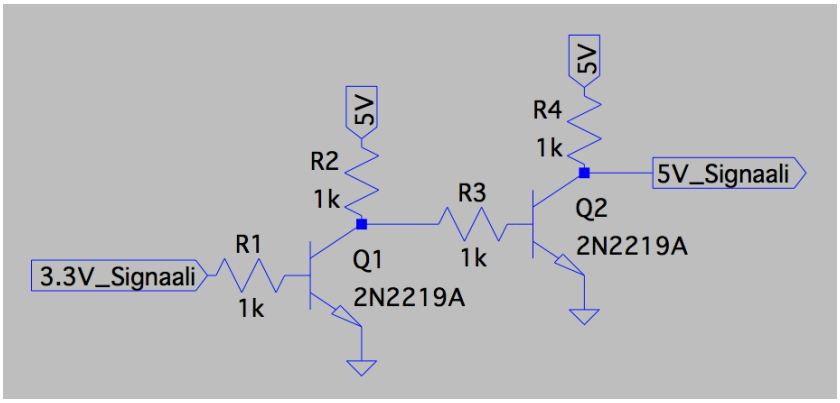
3.1 Elektroniikka

Osiona elektroniikka oli alun perin hyvin suppea, sillä projektissa ei ollut selkeää tarvetta isommalle toteutukselle, eikä sellaista haluttu pakottamalla sisällyttää projektiin. Mikrokontrollerin käyttöjännitteen muutos toi kuitenkin tarpeen logiikkatason muuttajalle, sillä PSoC käyttää lopullisessa versiossa 5 V jännitettä, kun taas Raspberry Pi käyttää 3,3 V jännitettä.

Jotta PSoC:in korkeampi signaalijännite SPI-väylässä ei rikkoisi tai vahingoittaisi Raspberry Pin GPIO:ta, on MISO-signaalin logiikkataso pudotettava 5 V:sta 3.3V:iin. Jotta PSoC pystyy lukemaan Raspberry Pin lähettämää kellopulssia ja MOSI-signaalia, on ne taas vuorostaan nostettava 3,3 V:sta 5 V:iin. Tähän tarkoitukseen on olemassa myös kaupallisia tuotteita, mutta koska elektroniikka osiona oli liian suppea, päätettiin laite suunnitella ja toteuttaa itse.

MISO-signaalin tason pudottaminen tapahtuu yksinkertaisella jännitejaolla. Sekä kellosignaalin, että MOSI-signaalin tason nostaminen tapahtuu kuvassa 2 esitellyllä piirillä. Piiri koostuu kahdesta transistoriasteesta, sillä ensimmäisen asteen jälkeen signaali on invertoitu. Toisen asteen jälkeen signaali vastaa alkuperäistä 3,3 V signaalia, mutta

huippujännite on 5 V. Molemmat signaalikanavat vaativat tietysti oman piirinsä toimiakseen.



Kuva 2. Logiikkatason muuttaja

Koko piiri toteutettiin reikälevylle, jonka koko pyrittiin minimoimaan järjestelmän asennuksen helpottamiseksi. Kaikista yksittäisistä osioista olisi voinut tehdä omat piirilevynsä, mutta se olisi vienyt tilaa ja tehnyt järjestelmästä hankalammin käsiteltävän. Hyppylangat juotettiin suoraan levyyn kiinni, mutta helpomman käsiteltävyyden vuoksi levyyn olisi voinut laittaa myös piikkirimat.

Lämpötila-anturien liittäminen mikrokontrolleriin ei vaadi mitään erityisjärjestelyjä tai –rakennelmia. Anturissa on kolme jalkaa, jotka ovat käyttöjännite, maa ja signaaliulostulo. Signaaliulostulo kytketään mikrokontrollerissa valittuun pinniin. Käyttöjännite otetaan mikrokontrollerin kehitysalustan kautta, tai irtonaisena käytettäessä ohjataan muutoin jänniteläheteltä. Signaalin taso on nollassa +500 mV, ja yhden asteen muutos lämpötilassa tarkoittaa 10 mV:in muutosta jännitteessä. Tämän takia anturia varten ei tarvinnut tuottaa lisäksi negatiivista jännitettä, mikä vähensi työn määrää.

Antureita ostettiin kymmenen kappaletta, mutta vain kahta lopulta käytettiin. Anturei-den keskinäisestä ja absoluuttisesta tarkkuudesta tehtiin vertailu, ja tulosten perusteella valittiin kaksi tarkinta anturia käytettäväksi. Vertailussa kaikkien anturien antamat arvot mitattiin huoneenlämmössä, jääkapissa ja lopulta pakastimessa. Huoneen lämpötila mittaushetkellä oli noin 22 °C. Jo huoneenlämmössä antureissa osoittautui suurta hajontaa, ja kuusi antureista näytti usean asteen virheen sisältävää lukemaa. Neljä tarkinta anturia valittiin tarkempaan vertailuun.

Taulukossa 1 näkyy anturien antama jännite millivoltteina. Arvosta on vähennettävä 500 mV ja jaettava se kymmenellä, jotta saadaan lämpötilaa vastaava arvo. Taulukon tuloksista on nähtävissä, että viimeisten neljän anturin keskinäiset erot ovat alle asteen luokkaa. Tämä on aivan riittävä tarkkuus tähän työhön. Vain huoneenlämmöstä on vertailuarvo lämpömittarilla mitattuna, joten kylmempien lämpötilojen mittaukset kertovat lähinnä antureiden keskinäisestä tarkkuudesta.

Taulukko 1. Lämpötila-anturien vertailutulokset

	Huoneenlämpö	Jääkaappi	Pakastin
1 Valk.	711	568	297
2 Must.	720	571	299
3 Pun.	720	577	304
4 Sin.	725	578	304

Lopullisten mittaustulosten perusteella päädyttiin käyttämään antureita 3 ja 4, sillä ne vastasivat keskenään parhaiten toisiaan. Toinen vertailumittauksessa hyvin pärjännyt anturipari laitettiin erilleen muista vara-antureiksi. Vaikka datalehdessä luvataan anturin tarkkuudeksi ±4 °C, vaikuttaa testimittauksen perusteella valitut anturit huomattavasti tarkemmilta (Microchip 2009, 1). Huono tarkkuus todennäköisesti viittaakin enemmän eri antureiden väliseen heittelyyn kuin parhaimpien yksilöiden tarkkuuteen. Tässä käytössä valikoinnilla saavutettu tarkkuus on enemmän kuin riittävä.

Vahva osaaminen elektroniikan saralta on tärkeää sulautettujen järjestelmien parissa työskennellessä, sillä toisin kuin muunlaisissa tietoteknisissä projekteissa elektroniikka on läsnä kaikessa suunnittelussa ja toteutuksessa. Kyky suunnitella piirejä ja tuottaa käyttökelpoisia laitteita ei ole kaikissa alan töissä tarpeellista, mutta se on suuri etu ja mahdollistaa työskentelyn projektien elektroniikkasuunnittelun parissa. Myös komponenttien vertailu ja valinta on tärkeä taito elektroniikan parissa työskenteleville.

3.2 Sulautettu ohjelmointi

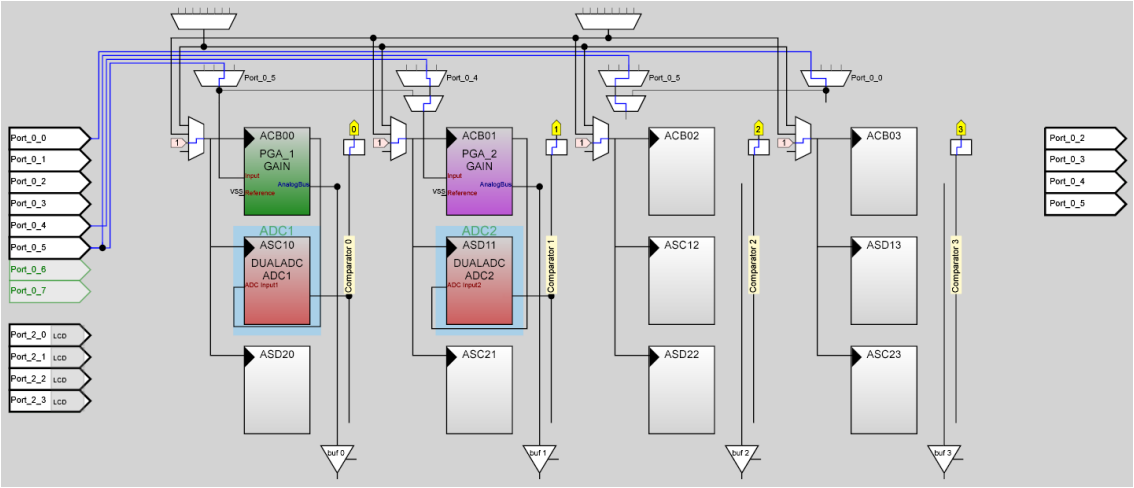
Sulautetun ohjelmoinnin osuus kokonaisuudesta on koodillisesti melko lyhyt, mutta ajallisesti se kesti lopulta pitkään. Aikaa meni yksinkertaisten ongelmien metsästämiseen. Ongelmista pitkäkestoisin paljastui lopulta johtuvan siitä, että piirin ADC ei toimi oikein, jos käyttöjännite on 3,3 V viiden sijaan. Oireiden perusteella ongelma tuntui

selvästi olevan tuotetussa ohjelmassa, eikä missään luetussa dokumentissa mainittu, että kyseinen kaksikanavainen AD-muunnin vaatii viisi volttia käyttöjännitteeksi toimiakseen. Tästä syystä aikaa käytettiin hyvin paljon vian etsintään ohjelmistosta, ja ohjelman kirjoittamiseen useaan kertaan kokonaan uudelleen.

PSoC-ohjelmointi ei ole pelkästään koodillista, vaan koostuu myös pienestä graafisesta osiosta. Liitteessä 2 näkyvässä graafisessa osiossa, joka on nimeltään Chip Level, ei varsinaisesti kuitenkaan tehdä ohjelmointia, vaan siinä otetaan käyttöön erilaisia moduuleja. Moduulien käyttö ja graafinen käyttöliittymä ovat omalla tavallaan työtä helpottavia, erityisesti vähemmän kokemusta omaaville. Kuitenkaan moduulit ja graafinen käyttöliittymä ei tunnu rajoittavan mahdollisuuksia, sillä lähes kaiken pystyy tekemään myös koodissa, ja kaikkeen pystyy vaikuttamaan komennoilla. Pystyyhän PSoC:ia ohjelmoimaan myös assemblyllä.

Moduuleja on kahdenlaisia: digitaalisia ja analogisia. Molemmille eri moduulityypeille on piiristä riippuva määrä vapaita lohkoja, joihin moduulit sijoitetaan. Tässä työssä käytetty CY827443-piiri sisältää kaksitoista vapaata lohkoa analogisille ja kahdeksan digitaalisille moduuleille (Cypress 2014, 1). Jotkin moduulit vievät vain yhden lohkon, kun taas monimutkaisemmat varaavat useamman. Piiristä riippuen on myös hyvin erilainen valikoima moduuleja tarjolla. Isoimmissa ja kalleimmissa piireissä lista on pitkä, ja esimerkiksi AD-muuntimia on monen tyyppisiä.

Kuvassa 3 näkyvät Chip Levelin analogiset lohkot ja niihin sijoitettuina tässä työssä käytetyt moduulit. Kuten kuvasta näkyy, Chip Levelissä tehdään myös sellaisten moduulien kytkennät pinneihin, jotka niitä käyttävät. Ylemmässä rivissä on vahvistinmoduulit, joilla lämpötila-anturien antamia signaaleja vahvistetaan paremman tarkkuuden saavuttamiseksi. Alemmassa rivissä taas sijaitsevat AD-muuntimien moduulit. Kaksisisääntuloinen AD-muunnin vaatii kahden analogisen lohkon lisäksi myös neljä digitaalista lohkoa.



Kuva 3. Analogiset moduulit

Jokaisella moduulilla on omat ohjelmointirajapinnat, jotka ohjelmisto luo automaattisesti. Näiden rajapintojen kautta moduulit käynnistetään ja niiden toimintaa ohjataan itse ohjelmiston puolella. Oletus- ja perusasetukset moduuleille asetetaan Chip Levelissä.

Työssä tuotetun ohjelmiston (liite 1) rakenne itsessään on yksinkertainen. Ohjelmiston alussa varataan tarvittavat muuttujat ja alustetaan moduulit. Pääohjelmasilmukan alussa odotetaan, että SPI:ltä on saapunut käsky, jonka jälkeen tarkistetaan, että AD-muuntimella on data valmiina. Datan ollessa valmiina luetaan molempien kanavien arvot, ja asetetaan AD-muunnin taas lukemaan jännitettä. Tämän jälkeen saadut arvot muutetaan lämpötiloiksi sopivan kertoimen avulla. Myös lämpötila-anturin 500 mV nollapisteestä johtuva offset poistetaan vähentämällä se saadusta arvosta. Tarkistetaan, että SPI:ltä saatu data on luettavissa, ja jos on, niin luetaan se. Saadun arvon perusteella tehdään valinta siitä, kumpi lämpötila-arvoista lähetetään SPI:llä.

Mikrokontrolleriosiota kehitettiin aluksi irrallisena minitietokoneesta, ja SPI-väylän toteutus jätettiin myöhempään vaiheeseen. Tässä alun kehitysvaiheessa piirissä käytettiin 5 V jännitettä, jotta kehitysalustaan liitetty näyttö toimisi. Tietoliikennettä laitteiden välille tehtäessä minitietokone vastaanotti vain hyvin omituisia lukemia, useimmiten vastaanotettu arvo oli 0 tai 255. Koska ohjelma selvästi pystyi lukemaan lämpötila-arvot hyvin, oli ensimmäisenä epäilyn kohteena mikrokontrollerin ohjelmisto. Ohjelmisto tutkittiin tarkasti, ja se jopa kirjoitettiin muutamaan kertaan uudestaan, jotta ongelma ratkeaisi. Lopulta kuitenkin kävi ilmi, ettei ongelma ollutkaan sulautetussa ohjelmistossa, vaan piirin käyttöjännitteessä.

Nykyään sulautettujen järjestelmien ydin on pääasiassa mikrokontrolleri, joten mikrokontrolleriohjelmointi on välttämätön osa sulautettujen järjestelmien projekteja. Elektroniikkaan erikoistuneen osaajan ei välttämättä tarvitse osata tuottaa monimutkaista ja optimoitua sulautettua ohjelmistoa, mutta vähintään perusymmärrys mikrokontrolleriohjelmoinnista on suuri etu.

Toisaalta taas ohjelmistoihin erikoistuneelle työntekijälle on varmasti myös tulevaisuudessa tarvetta. Laitteet monimutkaistuvat toki myös elektroniikan osalta, mutta siinä samassa ohjelmistoihin kohdistuvat tarpeet kasvavat ja monipuolistuvat. Kyky suunnitella ja toteuttaa hyvä mikrokontrolleriohjelmisto on hyvä taito myös, jos projekti on pienempi ja sen parissa yksin työskentelevä ihminen toteuttaa kaikkia osioita.

3.3 Tietoliikenne

Tässä työssä tietoliikenneosio koostuu SPI-väylästä, jonka avulla mikrokontrolleri ja minitietokone kommunikoivat keskenään. Väylän kautta minitietokone lähettää pyyntökäskyjä, joihin mikrokontrolleri vastaa pyydetyllä lämpötila-arvolla. Väylää valitessa tutkittiin myös useita muita kommunikointimuotoja, mm. I^2C-väylä ja UART. SPI todettiin kuitenkin tähän käyttötarkoitukseen soveltuvimmaksi ja helpoiten toteutettavaksi vaihtoehdoksi.

SPI-väylä on täysin kaksisuuntainen synkroninen väylä, jossa sekä isäntälaite että orjalaite voivat yhtä aikaa lähettää ja vastaanottaa dataa. Väylä koostuu kolmesta johtimesta, joissa MOSI-, MISO- ja kellosignaalit kulkevat. Useita laitteita samassa väylässä käytettäessä käytetään SPI:ssä niin sanottua Slave Select tai Chip Select –signaalia jokaiselle laitteelle. Signaalin avulla isäntälaite kertoo orjalaitteille, kenen kanssa väylällä kommunikoidaan milläkin hetkellä. Tässä toteutuksessa ei Chip Select –signaalille ollut tarvetta, sillä vain isäntälaite ja yksi orjalaite käyttävät väylää.

MOSI-signaalin kautta kulkee isäntälaitteelta orjalaitteelle lähetetty data ja MISO-signaalin kautta orjalaitteelta isäntälaitteelle lähetetty data. Koska väylä on synkroninen, liikkuu data molempiin suuntiin kellosignaalin tahtiin. Tässä työssä minitietokone toimii isäntälaitteena ja mikrokontrolleri orjalaitteena. Tähän järjestelyyn päädyttiin, koska minitietokone toimii arvoja pyytävänä osapuolena, ja mikrokontrolleri vastaa isäntälait-

teelta lähetettyyn kyselyyn. Tällöin väylään olisi mahdollista myös liittää muita orjalaitteita.

Vaikka muita orjalaitteita ei tähän toteutukseen lopulta liitettykään, olisi niiden avulla mahdollista laajentaa toteutuksesta esimerkiksi monipuolinen sääasema. Väylässä voisi olla useita mikrokontrollereja, jotka välittävät esimerkiksi ilmankosteusantureiden, tuuliantureiden ja ilmanpaineantureiden tietoja minitietokoneelle. Lisäksi väylän kautta pystyisi laajentamaan myös toiminnallisia osuuksia, esimerkiksi releohjauksen kautta. Työn laajuuden vuoksi johonkin oli kuitenkin vedettävä raja, eivätkä lisälaitteet olisi ainakaan tietoliikenteen osion kannalta enää tuoneet lisää tietoa tai taitoja.

Väylän toiminnan toteutus tapahtui käytännössä valmiiden kirjastojen avulla. PSoC:issa väylän käyttöönotto tapahtuu ohjelmointiympäristön graafisessa chip level –osiossa. Väylä koostuu yhdestä digitaalisesta moduulista, johon kytketään halutut pinnit kullekin signaalille. Väylä alustetaan ja käynnistetään koodissa, ja sitä ohjataan kirjaston komentojen avulla.

Raspberry Pi:ssä väylää käytetään Python-skriptissä. Jotta väylää voi käyttää, pitää väylä ottaa käyttöön järjestelmässä muokkaamalla asetustiedostoa ja asentaa SpiDev Python –moduuli. Moduulin asennustiedosto ladataan kolmannen osapuolen palvelimelta ja ajetaan komentolinjassa. Tämän jälkeen kirjasto on käytettävissä Pythonissa, kunhan vain sisällyttää kirjaston tiedoston alussa ja suorittaa tarvittavat alustuskomennot. Alustuskomennot ovat nähtävissä liitteen 3 alussa.

Ohjelmoinnillisella tasolla tämä osio ei tuonut paljoa uutta, sillä toteutus oli pääosin valmiiden kirjastojen varassa. SPI-väylän toiminta tosin oli jo osittain tuttu ennestään, ja väylän konkreettisesta toiminnasta saatiin syvempi ymmärrys logiikkatason muuttajaa tehtäessä.

Tietoliikenne on erittäin tärkeä osa sulautettujen järjestelmien osaamista, ja hyvä perusosaaminen sen saralta on hyödyllistä. Monissa isommissa projekteissa mikrokontrolleri kommunikoi sisäisesti jonkin muun laitteen komponentin kanssa tai laitteen ulkopuolisten tahojen, kuten tietokoneiden tai toisten laitteiden kanssa. Sekä elektroniikkaan että ohjelmointiin erikoistuneilta sulautettujen järjestelmien osaajilta edellytetään tietolii-

kenteen osaamista, sillä harvoin projektissa on erillistä, pelkästään tietoliikenteen parissa toimivaa työntekijää.

3.4 Python-ohjelmisto

Python kielenä on periaatteessa melko korkean tason kieli, eikä siten ole lähtökohtaisesti tarkoitettu laitteistoläheiseen ohjelmointiin. Kieli soveltuu kuitenkin hyvin sulautettuun Linux-ohjelmointiin, sillä sille on olemassa paljon laitteistokohtaisia kirjastoja, ja esimerkiksi Raspberry Pi:n yleisimmissä käyttöjärjestelmissä tulee Python-tulkki mukana. Python on kielenä lähes täysin riippumaton käyttöjärjestelmästä, joten Linux:illa Python-ohjelmoinnin opeteltua pystyy sitä tekemään myös kaikilla muilla kielen tukemilla käyttöjärjestelmillä.

Python oli tämän projektin tarkoituksiin hyvä, sillä skriptikielenä sitä ei tarvitse kääntää ennen ajoa. Tämä nopeuttaa kehittämistä huomattavasti, sillä kaikki muutokset skriptiin astuvat voimaan heti sen käynnistyessä uudelleen.

Jotta Raspberry Pi:ssa voitaisiin Pythonilla käyttää SPI:tä, vaatii se ensin SPI:n sallimisen järjestelmässä ja SpiDev Python –moduulin asentamisen. SPI:n salliminen tapahtuu kommentoimalla spi-bcm2708 järjestelmän mustalta listalta. Python-moduulin asentaminen tapahtuu lataamalla asennustiedosto ja tarvittava C-kielinen kirjastotiedosto koneelle ja ajamalla asennustiedosto. (Baumann)

Työtä varten tehty Python-ohjelmisto on liitteessä 3. Tiedoston alussa tuodaan tarvittavat kirjastot ja alustetaan SPI. Alun perin, kun PSoC:in oli tarkoitus vielä käyttää 3,3 V jännitettä, oli ajatuksena syöttää käyttöjännite Raspberry Pi:n pinnistä 25. Tätä tarkoitusta varten tehty koodiosuus on vielä nähtävissä kommentoituna: osiossa asetetaan pinni 25 ulostuloksi ja sen ulostulo tilaan 1 skriptin käynnistyessä. Näin saatiin PSoC olemaan päällä vain sitä tarvittaessa.

Itse pääohjelmasilmukassa toistuu kahdesti sama toiminto, mutta vain eri lämpötila-arvoille: SPI:llä lähetetään käsky, joka määrää, kumpaa lämpötilaa pyydetään. Lyhyen odotuksen jälkeen luetaan vastaus SPI:ltä muuttujaan. Debuggausta varten vastaanotettu arvo tulostetaan komentolinjalle. Tiedostonkäsittelijä avaa arvon sisältävän tiedoston.

Tämän jälkeen tiedostossa oleva vanha arvo päällekirjoitetaan SPI:ltä saadulla uudella arvolla. Tämän jälkeen tiedostonkäsittelijä suljetaan.

Ohjelmassa on myös hyvin yksinkertainen poikkeuksenkäsittely, joka ottaa huomioon vain ohjelman lopettamisen crtl-c –näppäinkomennolla. Poikkeuksenkäsittelijä sulkee SPI:n ja asettaa pinnin 25 tilaan 0, sammuttaen samalla virrat PSoC:ilta. Poikkeuksenkäsittelijään olisi voinut rakentaa monenlaisia toimintoja, mutta näin yksinkertaisessa toteutuksessa ei niille ollut käytännössä mitään tarvetta.

Ohjelmisto itsessään ei suuresti käytä Linuxin järjestelmäresursseja tai viestejä, joten ohjelmointi muille ympäristöille olisi lähes samankaltaista. Ainoana erona on se, että ohjelma käyttää GPIO-moduulia ja sen komentoja.

Sulautettu Linux-ohjelmointi ei ole tässä työssä käsitellyistä taidoista välttämättömimpiä, sillä valtaosa työelämän projekteista ei sisällä varsinaista tietokonetta tai korkean tason käyttöjärjestelmää. Kuitenkin minitietokoneiden pienentyessä ja halventuessa, sekä niiden virrankulutuksen pienentyessä, yleistyvät ne huomattavasti alan projekteissa. Linux-ohjelmointi voi siis olla tulevaisuudessa merkittävä taito.

Python on kielenä voimakkaasti yleistyvä ja sen hallitseminen on hyvä taito työmarkkinoilla. Kieltä käytetään kaikkialla hyvin matalan tason laitteistoista monimutkaisiin graafisiin ohjelmiin. Kuitenkin se soveltuu parhaiten yksinkertaisempaan skriptikäyttöön, eli siten kuin sitä on tässä työssä käytetty.

3.5 Web-ohjelmointi ja skriptit

Web-koodi tässä työssä koostuu kolmesta eri kielestä, jotka ovat HTML, PHP ja JavaScript. Kaikki nämä ovat kirjoitettuna yhteen tiedostoon, mutta käytännössä ne toimivat hyvin eri tavalla ja jopa eri paikassa. Kun tietokoneen selaimeen syötetään sivun osoite, pyytää selain palvelimelta sivun tiedostoja ja saa vastauksena HTML- ja JavaScript- osiot sivusta. Nämä osiot eivät siis sijaitse palvelimella, vaan ne näytetään tai ne pyörivät selaimessa paikallisesti. HTML on muotoilukieli, joka kertoo selaimelle sen, mitä ja missä kohdassa sivua näytetään. JavaScript on asiakaspään skriptikieli ja voi esimerkiksi muuttaa sisältöä sivulla paikallisesti. PHP sen sijaan on niin kutsuttu palvelinpään skriptikieli, eli se ei missään vaiheessa saavu selaimeen ja asiakaskoneelle.

Tässä työssä PHP:n (liitteen 4 alussa) tehtävänä on odottaa asiakaskoneelta käskyä, jonka saatuaan se ajaa palvelimella sijaitsevan skriptitiedoston kayn-sam.sh system-komennolla. Käsky välitetään asiakas-palvelin-mallin starndardiviestinnän POST-metodilla, kun selaimessa on painettu asianmukaista nappia. PHP-skripti on melko suppea, mutta laajempaa toiminnallisuutta ei siihen ollut tarkoituksenmukaista sisällyttää.

Sivuston ulkomuoto, asioiden sijoittelu ja keskinäiset suhteet muodostetaan HTML:n avulla. HTML sisältää lämpötilataulukon alustuksen ilman arvoja, painikkeen sijoittamisen ja videokuvan kehyksen luomisen ja sen sisällön tuottavan Java-appletin kutsun. Kun asiakaslaite pyytää sivua palvelimelta, se saa ensimmäisenä HTML-tiedoston, jonka avulla se luo sivunäkymän selaimeen. Vasta tämän jälkeen muut toiminnot, kuten skriptit, ladataan ja ne alkavat vaikuttaa sivun sisältöön. Tästä johtuen hyvin hitaalla yhteydellä eivät välttämättä lämpötila-arvot ja web-kameran kuva näy välittömästi selaimessa, vaan niiden tilalla on tyhjää tai placeholder-arvot.

JavaScript sijaitsee HTML-osion sisällä head-osiossa olevassa script-osiossa. Skripti koostuu vain yhdestä funktiosta, joka on määritetty ajettavaksi ensimmäistä kertaa heti sivua ladattaessa. Funktio pyytää palvelimelta lämpötilatiedostot ja asettaa niiden sisällön lämpötilataulukon oikeisiin kohtiin. Funktion lopussa sen seuraava ajoajankohta määritellään sekunnin päähän. JavaScript on asiakaspään laitteessa toimiva skripti, eli se ei sijaitse palvelimella. Sen sijaan se kommunikoi palvelimen kanssa ja pyytää siltä tarvittavia tietoja. Skripti vaikuttaa myös asiakasohjelmistossa näkyvään sivuun muun muassa päivittämällä sen sisältöä.

Palvelinpään skripti, jonka PHP ajaa pyydettäessä, on BASH-skripti, eli käytännössä sen sisältö on komentolinjakomentoja. Ensin skripti tarkistaa, onko videopalvelinohjelman palvelu käynnissä. Jos on, niin se sammutetaan, ja jos ei, niin se käynnistetään. Palveluita voi käynnistää ja sammuttaa vain jos on pääkäyttäjäoikeudet. Tästä syystä käynnistys- ja sammutuskomennot ajetaan sudo-komennon avulla.

Suurimmat ongelmat web-osiossa liittyivät lämpötilatietojen välittämiseen palvelimelta asiakasohjelmalle. Asiaa tutkittiin muun työn ohessa pitkään, ja siihen kyseltiin neuvoa myös foorumeilta. Kaikki ehdotetut vaihtoehdot ja normaalit toimintamallit olivat tähän

tarkoitukseen aivan liian raskaita: ne vaativat erillisiä frameworkejä ja satoja rivejä koodia molempiin päihin, jotta voitaisiin välittää kaksi yksinkertaista arvoa. Tätä ei pidetty kohtuullisena ratkaisuna, vaan uutta ja järkevämpää ratkaisua lähdettiin etsimään ja kehittelemään itse.

Lopulta pitkän pohdinnan ja asiaan perehtymisen jälkeen yksinkertainen ratkaisu tuli mieleen, ja se jopa saatiin toimimaan. Monimutkaisten rajapintojen sijaan arvot välitetään palvelimelta asiakaslaitteelle yksinkertaisesti pyytämällä JavaScriptin avulla sieltä tiedostot, jotka sisältävät nämä arvot. Python skripti kirjoittaa arvot tiedostoon, ja tiedostosta arvot päivitetään JavaScriptillä asiakaslaitteella näkyvään sivustoon. Tämä keino on yksinkertainen, mutta toimii hyvin tällaisessa yksinkertaisessa kahden arvon välityksessä. Kyseisessä sovelluksessa ei myöskään ole tarvetta erityisen tehokkaalle tietoturvalle tai suurille tietorakenteille, sillä välitettävät arvot eivät sisällä mitään suojattavaa tai monimutkaista tietoa.

Web-ohjelmoinnista tuttua ennalta oli ainoastaan HTML, jota ei tässä työssä lopulta käytetty kuin sivun ulkoasullisen sisällön luomiseen. PHP, JavaScript ja BASH-skriptaus olivat kaikki lähes täysin uutta. PHP ja asiakas-palvelin-malli olivat melko kevyitä opeteltavia, mutta niiden toteutus toi kuitenkin tärkeää osaamista palvelimen ja asiakaslaitteen välisestä kommunikoinnista. JavaScript oli toteutuksellisesti haastavampi ja vaati paljon kielen opettelua ja harjoitusta jotta toiminto saatiin toimimaan. BASH-skriptaus oli periaatteessa perustason Linux-kokemuksella melko yksinkertaista, sillä monet komennot olivat jo entuudestaan tuttuja.

Tässä osiossa käytetyt taidot eivät ole sulautettujen järjestelmien osaajalle välttämättömiä, ja monissa järjestelmissä ne voivat olla täysin tarpeettomiakin. Näin on ollut varsinkin aiemmin, sillä sulautetut järjestelmät eivät ole perinteisesti olleet liitettyinä verkkoon vaan ovat kommunikoineet muulla tavoin. Tämä on kuitenkin muuttumassa nopeasti, ja verkko-ohjelmoinnin perustaitojen osaaminen voi tuoda etulyöntiaseman nykypäivän työelämässä.

Erityisesti yksityiskäyttöön suunnitelluissa järjestelmissä kuluttajat arvostavat enenevissä määrin etähallinnan ja -valvonnan mahdollisuutta sekä selainpohjaista käyttöliittymää. Usein kuitenkin web-ohjelmoijat eivät ole perehtyneitä laitteistoläheiseen ohjel-

mointiin, eivätkä siten pysty välttämättä toimimaan sulautettujen järjestelmien projektissa ilman isompaa perehtymistä ohjelmiston liittymisestä konkreettisempaan tasoon.

3.6 Web-kamera ja Motion

Alun perin työn tarkoitus oli huomattavasti enemmän painottunut web-kameran ja sen kautta valvonnan ympärille. Tämä olikin siksi luonteva osio toteuttaa ensimmäisenä kokonaisuudesta, sillä se ei myöskään välttämättä vaatinut muita osioita toimiakseen. Tämä työvaihe lomittui pitkälti myös toteutusta edeltävän selvitystyön kanssa.

Web-kameran lukemiseen ja kuvavirran ohjaamiseen valittiin ohjelmaksi ilmainen avoimen lähdekoodin ohjelmisto nimeltä Motion. Siinä on laajat säätömahdollisuudet, ja se pystyy nimensä mukaisesti tunnistamaan liikettä kuvasta. Motionin tunnistaessa liikettä kuvasta oli tarkoitus tallentaa kuvaa niin kauan, kun liikettä havaitaan ja lähettää asetettuihin sähköpostiosoitteisiin ilmoitus.

Käytettyjen kameroiden ja Motionin sekä Linux-ajurin keskinäinen toiminta kuitenkin osoittivat liiketunnistuksen hyvin hankalaksi. Kamera itse, ajuri ja ohjelma pyrkivät kaikki jossain määrin säätämään kuvan kirkkautta ja valkotasapainoa. Näiden säätöjen keskinäinen vaikutus aiheuttaa sen, että kaikilla kokeilluilla kameroilla säännöllisin väliajoin kuvan kirkkaus säätyy ensin puhki palaneen kirkkaaksi, siitä täysin mustaksi ja vasta sitten palautuu normaaliksi.

Tämänkaltainen kirkkauden muutos tietysti tulkitaan liikkeenä, ja virheilmoituksia syntyy jatkuvasti. Huolimatta usean viikon työstä ei asetuksia saatu siten, että kuva säätyisi vain tarvittaessa ja silloinkin hallitusti. Tämän epäonnistumisen myötä työn painotusta siirrettiin enemmän monitasotutkielmaan kuin videovalvontajärjestelmään.

Toiseksi ongelmaksi muodostui se, että Motion julkaisee web-kameran kuvan MJPEG-virtana. Monet selaimet, mukaan lukien Google Chrome, ovat luopuneet MJPEG-virran tuesta, ja se ei siksi toimi yleisessä käytössä. Tämän ongelman kiertämiseen löytyi ratkaisuksi Cambozola-niminen Java-pohjainen ohjelma, joka muuttaa virran Javaksi. Tämän avulla videokuva saadaan helpommin näkymään sivuilla.

Työn toteutuksen aikana myös tuki Javalle selaimissa on vähentynyt, ja kuvavirran toimiminen asiakaslaitteella vaatii Javan asentamisen sekä sallimisen selaimessa. Tämä hankaloittaa hieman käyttöä, mutta se on silti hyvin kohtuullisesti käyttöönotettava ja käyttöönoton jälkeen helppokäyttöinen ratkaisu. Cambozolan tilalle etsittiin myös ratkaisuja, jotka eivät olisi riippuvaisia Javasta, mutta sellaista ei löydetty.

Kolmannen osapuolen laitteiden ja ohjelmistojen käyttö projekteissa on ollut aina melko yleistä, mutta se on yhä yleisempää varsinkin laajemmissa projekteissa. Myös kertaluontoisemmat räätälöidyt ratkaisut vaativat usein vähempien resurssien vuoksi kolmannen osapuolen laitteiden ja ohjelmistojen käyttöä. Taito konfiguroida ja liittää tällaisia laitteita ja ohjelmistoja järjestelmään ja sen toimintaan on tärkeä isoissa projekteissa, joissa kaikkea ei tehdä itse.

4 YHTEENVETO

Työssä oli päätavoitteena tutkia tärkeimpiä sulautettuihin järjestelmiin erikoistuneen tietotekniikkainsinöörin työn osa-alueita. Ne ovat elektroniikkaa, mikrokontrolleriohjelmointia, tietoliikennettä, Linux-ohjelmointia, erilaisia rajapintoja, valmiiden ohjelmistojen konfigurointia, web-ohjelmointia ja skriptikieliä.

Näistä osa-alueista elektroniikka jäi työn alkuvaiheissa hyvin vähälle, sillä sille ei ollut helposti perusteltavissa olevaa toteutuskohdetta. Työn edetessä kävi kuitenkin ilmeiseksi, että tarvitaan logiikkatason muuttaja mikrokontrollerin ja minitietokoneen välille. Tämän logiikkatason muuttajan suunnittelu ja toteutus muodostui laajuudeltaan ja haastavuudeltaan sopivaksi tuomaan elektroniikan muiden osa-alueiden tasolle.

Mikrokontrolleriohjelmointi jäi lopputulokseltaan oletettua suppeammaksi. Ohjelman pituus koodirivillisesti on hyvin vähäinen ja toiminta siten myös yksinkertaista, mutta tarvetta tätä laajemmalle ohjelmalle ei ollut. Kuitenkin ohjelman tekemiseen kului yllättävän kauan johtuen ongelmista, joiden lähde lopulta löytyi käyttöjännitteestä eikä koodista. Koodin suunnittelun ja toteutuksen laajuus ei siis ollut kovin suuri, mutta ongelmanratkaisu ja sen yhteydessä tehty ohjelmiston toimintojen tarkka analysointi ja useasti uudelleen kirjoittaminen tutustuttivat syvemmin mikrokontrolleriohjelmointiin.

Logiikkatason muuttajan suunnittelu vaati myös tarkkaa perehtymistä SPI-väylään ja sen asettamiin tarpeisiin ja rajoitteisiin. Tämä laajensi tietoliikenteen osiota työssä ja lisäksi kasvatti ymmärrystä SPI-väylän toiminnasta. Ohjelmallisesti osiossa tehty oma toteutus jäi melko pieneksi johtuen valmiiden kirjastojen käytöstä. Väylän toimintaan kyllä perehdyttiin kirjastojen ja muiden tietolähteiden avulla.

Linux-ohjelmoinnin yhteydessä tutustuttiin syvemmin Python-kieleen ja sen käyttöön Linux-ympäristössä. Ohjelman koodauksessa tärkeimmät opitut asiat olivat tiedostojen käsittely, SPI:n käyttö kirjastojen avulla ja poikkeusten käsittely. Laajuudeltaan osio oli tasapainossa muiden kanssa.

Rajapintojen kanssa toimiminen rajoittui web-kameran kanssa olleiden ongelmien takia ohjelmissa avoimien kirjastojen käyttöön. Tarkoituksena oli käyttää Motion-ohjelman

rajapintaa, jonka avulla liikkeen tunnistuksesta olisi voinut lähettää sähköpostia. Tämä osio jäi siis aiottuun suhteessa hyvin vähäiseksi.

Valmiin ohjelmiston konfigurointi toteutui Motionin asetuksia hakiessa. Asetusten kanssa oli lukuisia ongelmia, eivätkä kaikki ratkenneet työn aikana, mistä johtuen osa suunnitelluista ominaisuuksista jouduttiin jättämään pois. Tämä johtui myös osittain siitä, että samaan asiaan vaikutti kolme eri tahoa eikä kaikkia niitä ei pystytty säätämään.

Web-ohjelmointi muodostui melko kattavaksi osioksi, mutta sen osuus on silti kohtuullinen suhteessa muihin. Isoimman ongelman muodosti dynaaminen sisältö, eli lämpötilojen päivittäminen sivuille. Tätä osaa tehtäessä turvauduttiin myös keskustelupalstoihin, mistä ei kuitenkaan löytynyt apua: kaikki tarjotut ratkaisut olivat turhan monimutkaisia muutaman arvon siirtämiseen ohjelmalta toiselle. Lopulta asian hetken hauduttua löytyi yksinkertaiseksi ja toimivaksi ratkaisuksi arvojen välittäminen tiedoston avulla.

Skriptikielien osio oli osittain päällekkäinen web-ohjelmoinnin kanssa, sillä koodiriveissä mitattuna tässä projektissa laajemmin käytetty skriptikieli JavaScript on nimenomaan web-kieli. Toisaalta myös komentoskripri on päällekkäinen Linux-ohjelmoinnin kanssa. Kaikesta huolimatta tämä osio toteutui hyvin ja oli riittävän kattava.

Vastoinkäymiset työn aikana saivat opettelemaan monia tietoja ja taitoja, jotka ovat varmasti tulevaisuudessa hyödyksi moniosaisten projektien parissa. Tiedon saattaminen analogisesta jännitteestä verkkosivujen kautta asiakastietokoneelle on monimutkainen prosessi, ja sen kaikkien vaiheiden periaatteiden ymmärtäminen tuo etuja sulautettujen järjestelmien parissa työskenneltäessä.

Työssä onnistuttiin näyttämään jo koulussa saatua osaamista, ja lisäksi opittiin paljon uusia tarpeellisia taitoja työelämää varten, sekä saavutettiin ymmärrystä erillisten osioiden ja niihin liittyvän osaamisen yhdistämisestä monitasoiseksi kokonaisuudeksi.

Työn toisesta tavoitteesta eli valvontajärjestelmän luomisesta jouduttiin karsimaan web-kameraongelman vuoksi. Lukuun ottamatta hälytysominaisuutta tavoitteessa kuitenkin onnistuttiin hyvin, sillä valmiilla laitteella voidaan tarkkailla lähes reaaliajassa web-kameran kuvaa kohteesta sekä kahta lämpötilaa, esimerkiksi sisä- ja ulkolämpötiloja.

LÄHTEET

Microchip Technology Inc. 2009. MCP9700 Datasheet. Julkaistu 11.2005. Päivitetty 04.2009. Luettu 28.09.2014.
http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf

Cypress Semiconductor Corporation. 2014. CY8C27X43 Datasheet. Julkaistu 01.07.2003. Päivitetty 19.09.2014. Luettu 12.10.2014.
http://www.cypress.com/?docID=50148

Elinux Wiki. 2015. RPi USB Webcams. Luettu 20.05.2014.
http://elinux.org/RPi_USB_Webcams

Thomas Baumann. SpiDev Documentation. Luettu 17.08.2014.
http://tightdev.net/SpiDev_Doc.pdf

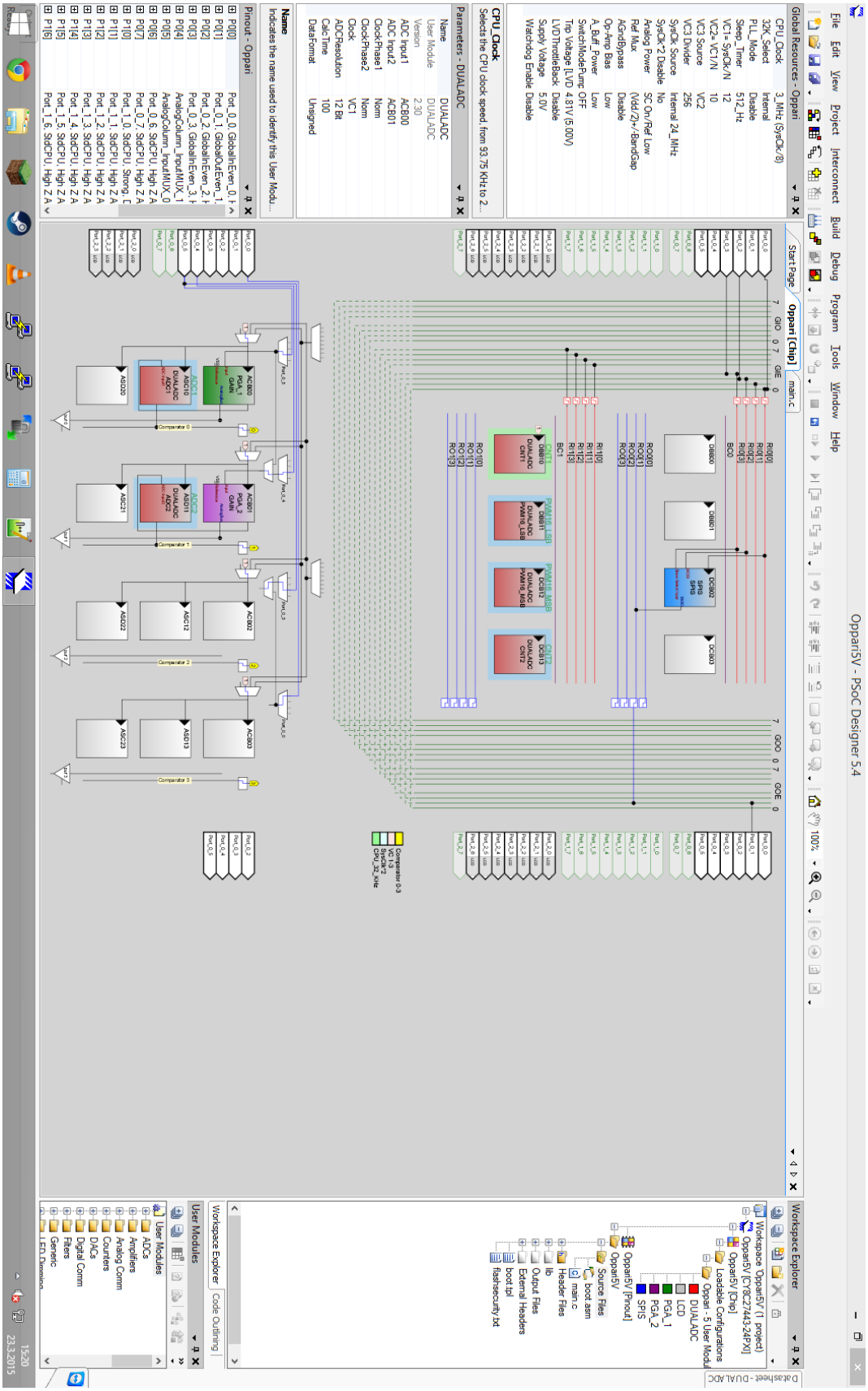
LIITTEET

Liite 1. Sulautettu ohjelmisto

```
#include  <m8c.h>
#include  "PSoCAPI.h"
#include <stdlib.h>

void main(void){

     char bData = 0;
     char Temp[3];
     int t1, t2 = 0;
     signed int temp1, temp2 = 5;

     // Sallitaan keskeytykset
     M8C_EnableGInt;
     // Alustetaan SPI
     SPIS_Start(SPIS_SPIS_MODE_0 | SPIS_SPIS_MSB_FIRST);
     // Alustetaan vahvistimet
     PGA_1_Start(PGA_1_MEDPOWER);
     PGA_2_Start(PGA_2_MEDPOWER);
     // Alustetaan ADC
     DUALADC_Start(DUALADC_HIGHPOWER);
     DUALADC_SetResolution(12);
     DUALADC_GetSamples(0);
     LCD_Start();


     //Pääohjelmasilmukka
     while(1){

          // Odotetaan että dataa saapuu SPI:ltä
          while( !( SPIS_bReadStatus() & SPIS_SPIS_SPI_COMPLETE ) );

                    // Tarkistetaan onko data valmista
          if(DUALADC_fIsDataAvailable != 0){
               // Luetaan molempien kanavien datat
               t1 = DUALADC_iGetData1();
               t2 = DUALADC_iGetData2();
               DUALADC_ClearFlag();

                    // Muunnetaan arvot lämpötilaksi
               temp1 = (t1-(3827*0.5))/38.27;
               temp2 = (t2-(3827*0.5))/38.27;

               // Tulostellaan lämpötilat näytölle
               LCD_Position(0,5);
               itoa(Temp, temp1, 10);
               LCD_PrString(Temp);
               itoa(Temp, temp2, 10);
               LCD_PrString(Temp);
          }

          // Tarkistetaan onko  SPI:stä saatu data luettavissa
          if(SPIS_bReadStatus()&SPIS_SPIS_RX_BUFFER_FULL){

               // Luetaan data */
               bData = SPIS_bReadRxData();

               // tulostetaan saatu arvo näytölle
               LCD_Position(1, 5);
```

```
                itoa(Temp, bData, 10);
                LCD_PrString(Temp);

                // Lähetetään pyydetty arvo SPI:llä
                switch (bData){
                     case 0x01:

                          // Lähetetään lämpötila 1
                          SPIS_SetupTxData(temp1);
                          PRT1DR &= ~0x01;
                          break;
                     case 0x02:
                          // Lähetetään lämpötila 2
                          SPIS_SetupTxData(temp2);
                          break;
                }
           }
            }
}
```

Liite 2. PSoC Designerin Chip Level



Liite 3. Python-ohjelmisto

```
# coding=utf-8
# moduulien tuonti: SPI-kirjasto, aikakirjasto ja GPIO-kirjasto
import spidev
import time
import RPi.GPIO as GPIO

#SPI:n alustus
spi = spidev.SpiDev()
spi.open(0,0)
spi.bits_per_word=8
spi.max_speed_hz=10000

#VANHAA! PSoC:in virransyötyön käynnistys: asetetaan GPIO-moodi, ase-
tetaan pinni 25 ulostuloksi, laitetaan ulostulo päälle ja odototetaan
PSoC:in käynnistymistä.

#GPIO.setmode(GPIO.BCM)
#GPIO.setup(25, GPIO.OUT)
#GPIO.output(25, True)
#time.sleep(1)

try:
     #Pääsilmukka, loppuu vasta sammutettaessa.
     while True:
          #lähetetään lämpötilan 1 pyyntö PSoC:ille
          spi.writebytes([0x01])
          time.sleep(0.02)
          #luetaan vastaus vastaus-muuttujaan
          vastaus = str(spi.readbytes(1)[0])
          #tulostetaan vastaus
          print("1: saatiin " + vastaus)
          #avataan tiedosto
          f = open('/var/www/lampotila','w')
          #kirjoitetaan saatu lämpötila-arvo puskuritiedostoon
          f.write(vastaus)
          #suljetaan tiedosto
          f.close()
          #lähetetään lämpötilan 2 pyyntö PSoC:ille
          spi.writebytes([0x02])
          time.sleep(0.02)
          #luetaan vastaus vastaus-muuttujaan
          vastaus = str(spi.readbytes(1)[0])
          #tulostetaan vastaus
          print("2: saatiin " + vastaus)
          #avataan tiedosto
          f = open('/var/www/lampotila2','w')
          #kirjoitetaan saatu lämpötila-arvo puskuritiedostoon
          f.write(vastaus)
          #suljetaan tiedosto
          f.close()
          #odotetaan sekunti
          time.sleep(1)

#ohjelman keskeytymisen yhteydessä suoritettavat lopputoimenpiteet
except KeyboardInterrupt:
     #suljetaan SPI
     spi.close()
     #Sammutetaan virrat PSoC:ista
     #GPIO.output(25, False)
     #palautetaan GPIO:n oletusasetukset
     #GPIO.cleanup()
```

Liite 4. Web-koodi

```
<?php
     //tarkistetaan onko käynnistys/sammutuskäsky annettu
     if (isset($_POST['kayn-sam'])){
       //käynnistetään puskuri nappaamaan vastaus
       ob_start();
       //lähetetään palvelimelle käsky ajaa skripti
       $dump = system('/var/www/kayn-sam.sh', $paluuarvo);
       //tyhjennetään puskuri
       ob_end_clean();
       //poistetaan annettu käsky
       unset($_POST['kayn-sam']);
     }
?>
<html>
     <head>
       <script type="text/javascript">
         function paivitaLampo(){
           // luodaan tiedostonkäsittelijä
           var tiedKas = new XMLHttpRequest();
           //haetaan esimmäinen tiedosto palvelimelta
           tiedKas.open("GET", "lampotila", false);
           tiedKas.send(null);
           //päivitetään arvo sivulle
           document.getElementById("lampotila1").innerHTML = tied-
Kas.responseText + " °C, ";
           //haetaan toinen tiedosto palvelimelta
           tiedKas.open("GET", "lampotila2", false);
           tiedKas.send(null);
           //päivitetään arvo sivulle
           document.getElementById("lampotila2").innerHTML = tied-
Kas.responseText + " °C";
           //asetetaan seuraava suoritusajankohta 1s
           setTimeout('paivitaLampo()',1000);
         }

       </script>
       <!-- metatiedot -->
       <meta content="text/html; charset=UTF-8" http-equiv="content-
type">
       <title>Teemu Matilainen - Valvontapalvelin</title>
       <meta content="Teemu Matilainen" name="author">
       <link title="style" rel="stylesheet" href="style.css" ty-
pe="text/css">

     </head>
     <!-- sivun runko, ajetaan lämpötilanpäivitysfunktio ensimmäisen
kerran heti avatessa -->
     <body onLoad="paivitaLampo()">

       <table align="center">
         <tr>
           <td>
             <!-- Lämpötilataulukko -->
             <table id="lampotilat">
               <tr>
                    <td><b>Lämpötilat: </b></td>
                    <td>Ulko: </td>
                    <td id="lampotila1">NULL</td>
                    <td>Sisä: </td>
                    <td id="lampotila2">NULL</td>
               </tr>
             </table>
           </td>
```

```
          </tr>
          <tr>
            <td>
              <!-- nappi videopalvelimen käynnistykseen/sammuttamiseen
-->
              <form method="post">
                <button name="kayn-sam">Sammuta/käynnistä videopalve-
lin</button>
              </form>
            </td>
          </tr>
        </table>
        <!-- videopalvelimen syöte -->
        <div id="kuva">
          <applet code=com.charliemouse.cambozola.Viewer
            archive=cambozola.jar width="1280" height="720"  sty-
le="border-width:1; border-color:gray; border-style:solid;"> <param
name=url value="http://192.168.1.39:8081">
          </applet>
        </div>

     </body>
</html>
```

Liite 5. BASH-skripti

```
#!/bin/sh
#Palvelun nimi
SERVICE='motion'
#tarkistetaan onko palvelu käynnissä
if ps ax | grep -v grep | grep $SERVICE > /dev/null
#jos on, niin pysäytetään se
then
    sudo /etc/init.d/motion stop
#jos ei, niin käynnistetään
else
    sudo /etc/init.d/motion start
fi
```