MONITOIMIUUNIN SÄHKÖISTEN TESTIEN AUTOMATISOINTIIN KÄYTETTÄVÄN LAITTEISTON KEHITTÄMINEN

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Valkeakoski, Automaatiotekniikan koulutusohjelma

Syksy, 2016

Sami Nikupeteri

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Automaatiotekniikan koulutusohjelma
VALKEAKOSKI

Tekijä Sami Nikupeteri **Vuosi** 2016

Työn nimi Monitoimiuunin sähköisten testien automatisointiin käytettävän laitteiston kehittäminen

TIIVISTELMÄ

Tämä opinnäytetyö tehtiin R-Menu Oy:lle uunituotantoon, jotta he pystyisivät automatisoimaan monitoimiuunien sähköisiä testejä. Tavoitteena oli suunnitella, koota ja toteuttaa laitteisto, jolla nämä sähköiset testit pystytään automatisoimaan.

Työssä kerrotaan teoriaa opinnäytetyöhöni liittyvistä aiheista, sekä yksityiskohtaisesti laitteiston ja käyttöliittymän toteutuksen suunnittelusta toiminnan testaukseen asti.

Tämän kehitysprojektin lopputuloksena valmistui testauslaitteisto, joka sisältää Raspberry Pi -minitietokoneen oheislaitteineen ja joka kommunikoi uunin kanssa Modbus RTU -sarjaliikenneväylällä. CoDeSys V3.5 -ohjelmalla tehdään ohjelmistoympäristö ja käyttöliittymä testauslaitteeseen, jolloin työn tilaajan on itse mahdollista lisätä monitoimiuunin automatisoituja sähköisiä testejä testauslaitteistoon. Tämä kehitysprojekti on tehty laitteilla, joita ei vielä ole käytetty tällaisessa ympäristössä diagnostiikkatyökaluna.

Avainsanat CoDeSys, Raspberry Pi, Modbus, kehitysprojekti

Sivut 48 sivua, joista liitteitä 2 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Automation Engineering
VALKEAKOSKI

Author Sami Nikupeteri **Year** 2016

Subject Developing hardware for the automatization of electrical testing of multifunctional ovens

ABSTRACT

This thesis was commissioned by R-Menu Ltd. oven production, for them to be able to automate their electrical tests of multifunctional ovens. The aim was to plan, construct and implement a system that could run these automation tests.

In my thesis theories of topics related to this dissertation are introduced. There is also detailed information on the equipment that was used in this project and how it was implemented from the beginning to the end of the project.

The result of this development project was testing equipment, which includes a Raspberry Pi and its peripherals that communicate with the oven through the Modbus RTU serial communication bus. The software and user interface done by using the CoDeSys V3.5 program allowed the commissioner to add the automation tests to this software. This development project was made using equipment that had not previously been used for this purpose.

Keywords CoDeSys, Raspberry Pi, Modbus, development project

Pages 48 pages including appendices 2 pages

SISÄLLYS

Liitteet

Liite 1 CoDeSys-ohjelmarunko

Liite 2 Led valon toiminnan testaus

1 JOHDANTO

Tämän opinnäytetyön tilaajana toimii Kangasalalainen R-Menu Oy niminen yritys, joka valmistaa monitoimiuuneja. Yritys on kehittänyt ainutlaatuisen monitoimiuunin ja nyt yrityksellä on tarvetta kehittää uunituotantoaan. Tämän opinnäytetyön tavoitteena on suunnitella ja rakentaa laitteisto, jolla työn tilaaja pystyy automatisoimaan monitoimiuunien sähköisiä testejä. Kyseinen yritys esitellään tarkemmin opinnäytetyön kolmannessa kappaleessa.

Monitoimiuunin valmistuttua tuotannosta se testataan huolellisesti ennen asiakkaalle luovuttamista. Tällä hetkellä uuniin tehtävät sähköiset testit joudutaan tekemään manuaalisesti ja näistä täytetään paperille tarkastuspöytäkirja. Työn tilaaja hakee uunituotantoonsa ratkaisua, jotta uunien testit voitaisiin automatisoida ja näin tuotantoa kehittää. Työn tilaajan lopullinen tavoite on saada testeistä täysin automatisoituja, jolloin kehitetty laite suorittaa testit automaattisesti ja lähettää testauspöytäkirjan sähköisessä muodossa yrityksen tietokantaan.

Tässä opinnäytetyössä suunnitellaan, kootaan ja ohjelmoidaan laitteisto, jolla testit voidaan jatkossa automatisoida. Tämä tapahtuu Raspberry Pi -minitietokoneeseen tehtävän ohjelmistoympäristön ja käyttöliittymän avulla, johon pystytään jatkossa ohjelmoimaan työn tilaajan haluamia uunin sähköisiä testejä.

Opinnäytetyössä tullaan esittämään teoriaa kehitysprojektiin kuuluvista asioista, kuten Raspberry Pi –minitietokoneesta, CoDeSys-ohjelmasta sekä Modbus-protokollasta. Lisäksi opinnäytetyössä kuvataan perusteellisesti kehitysprojektin kaikki vaiheet suunnittelusta toteutukseen asti. Opinnäytetyössä pohditaan myös työn lopputulosta sekä laitteen jatkokehitysmahdollisuuksia.

2 KEHITYSPROJEKTIN TAVOITE

R-Menu Oy:n uunituotannossa valmiit monitoimiuunit tarkastetaan manuaalisesti siten, että niistä täydennetään testauspöytäkirja saatujen tuloksien perusteella. Nämä testauspöytäkirjat taltioidaan kansioihin ja säilytetään jatkoa varten. Yrityksessä kesällä 2015 alkanut uunituotanto on lähtenyt käyntiin kiitettäväsi ja nyt yritys haluaakin kehittää tuotantoaan. Tällä hetkellä uunien sähköiset testit tehdään tuotannossa manuaalisesti, käyttämällä uunin eri toimilaitteita siten, että niistä saadaan mahdolliset viat selville ennen uunin luovuttamista asiakkaalle. Tällä hetkellä sähköiset testit vievät liian paljon aikaa ja sitovat yhden henkilön koko testien ajaksi tähän työtehtävään. Nyt yrityksen johto toivookin, että manuaalisesti tehdyt sähköiset testit saataisiin kokonaan automatisoitua siten, että tällä hetkellä noin neljä tuntia kestävät testit toteutuisivat nopeammin ja siten ettei se sitoisi yhtä työntekijää tähän työtehtävään. Testien automatisoinnilla saadaan myös poistettua mahdollisten testeissä tapahtuvien virheiden syntyminen.

Tämän kehitysprojektin tavoitteena on kehittää laitteisto ja siihen ohjelmistoympäristö sekä käyttöliittymä, jolla nämä sähköiset testit pystyttään toteuttamaan automaattisesti. Tämän kehitysprojektin tarkoituksena ei siis ole automatisoida testejä vaan suunnitella ja rakentaa laitteisto, johon näitä testejä pystytään tulevaisuudessa automatisoimaan. Laitteiston toiminta varmistetaan kehitysprojektin päätteeksi.

Tässä kehitysprojektissa määritellään testauslaitteiston PLC:nä toimiva Raspberry Pi 3 -minitietokone toimimaan CoDeSys V3.5 kanssa ja kommunikoimaan slave -laitteensa (monitoimiuunin), kanssa Modbus RTU -sarjaliikenneväylän kautta. Tässä opinnäytetyössä kehitettävän laitteiston tarkoituksena on toimia ohjelmistoympäristönä, johon tullaan rakentamaan ns. reseptejä, jossa jokainen resepti vastaa yhden monitoimiuunin toimilaitteen tai toiminnon testaamista. Tässä kehitysprojektissa laitteiston, käyttöliittymän ja ohjelmiston toiminta varmistetaan jollain yksinkertaisella testillä.

Tulevaisuudessa kun laitteistoon on lisätty automaattisia testejä, se tulee toimimaan esimerkiksi siten, että käyttäjän tekemän testiohjelman mukaan testauslaitteisto pakottaa monitoimiuunin toimilaitteita päälle. Samalla kun laite pakottaa toimilaitteita päälle niin se kerää tietoa niiden toiminnasta ja esimerkiksi lämpötiloista. Joitakin uunin I/O-tietoja saadaan luettua ilman toimilaitteiden pakottamista päälle, kuten esimerkiksi onko luukku kiinni vai auki.

3 TYÖN TILAAJA

3.1 R-Menu Oy

R-Menu Oy on pirkanmaalainen yritys, joka toimii valtakunnallisesti. Perheyritys on aloitettu tuotekehityksellä vuonna 2006 ja vuonna 2009 yritys siirtyi markkinoille. Ennen markkinoille siirtymistä oli tuotekehitystä tehty siten, että tuotteet sekä konsepti oli hiottu valmiiksi. Yrityksen konsepti on ainutlaatuinen Euroopassa ja näin ollen kilpailijoita ei ole joka tarjoisi samaa palvelua. Vuonna 2011 kauppalehti myönsi R-Menu Oy:lle palkinnon jonka saa vuoden paras kasvuyritys. Tällä hetkellä yrityksellä on satoja konseptiasiakkaita Suomessa ja Ruotsissa. Yrityksellä on kaksi toimipistettä Kangasalla. Toisessa toimipisteessä sijaitsee toimistot sekä uunituotannon tilat. Toisessa toimipisteessä sijaitsee ruokatehdas, jossa valmistetaan 40% yrityksen ruokamyynnistä. (Vainionpää 2016.)

Yrityksen ideana oli kehittää ruokapalvelu joka olisi yhtä varmaa, tehokasta ja nopeaa kuin ensiluokkaisen, maistuvan oluen laskeminen hanasta asiakkaalle. Ensimmäiset asiakkaat olivat pubeja, mutta hyvin pian asiakaskunta laajeni hotelleihin, huoltoasemiin ja kahviloihin. (R-Menu Oy n.d.)

3.2 Laitetuotanto

Markkinoille siirryttäessä R-menun konsepti perustui siihen, että asiakas hankki jonkun turbouunin esim. Merrychef e4 (Kuva 1.). Turbouunille ostettiin valmiiksi R-menun annospakattuja ruokia ja näin asiakas sai pakkasesta pöytään ilman hävikkiä ruoka-annoksen. Asiakkaalla ruoka-annoksen tekemiseen ei kulunut kuin muutamia minuutteja. Kuitenkin melko nopeasti yrityksen siirryttyä markkinoille vuonna 2009 alkoi R-Menun johto miettiä, kuinka uuneja pystyisi kehittämään. (Vainionpää 2016.)

Kuva 1. Merrychef e4 -turbouuni (Merrychef n.d.).

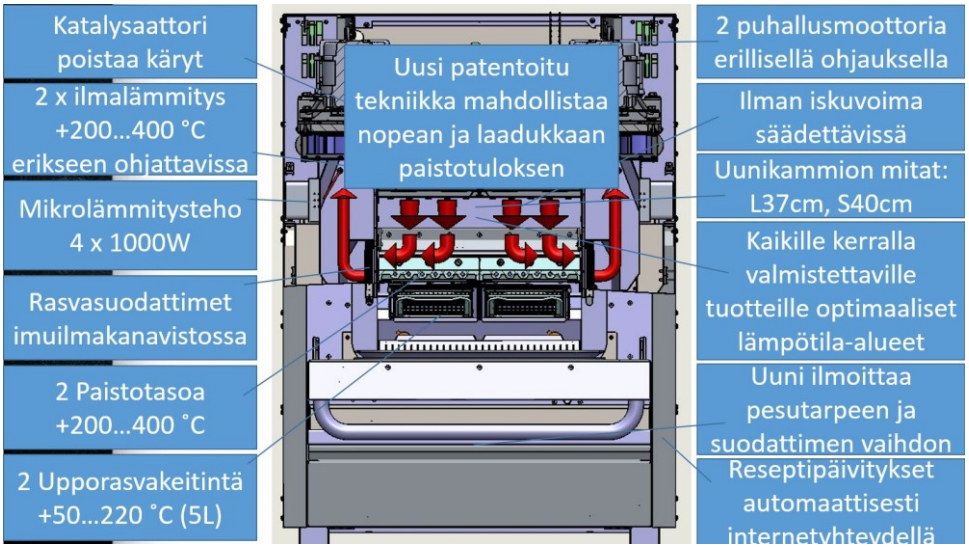
Vuonna 2010 käynnistettiin tuotekehitysprojekti, jonka tarkoituksena oli kehittää entistä parempi ja tehokkaampi monitoimiuuni. Pitkän tuotekehitysprojektin päätteeksi uusi tehokkaampi monitoimiuuni lanseerattiin Fastfood & Cafe messuilla Helsingissä vuonna 2015. (Vainionpää 2016.)

Nykyään uunituotannon hallinnossa työskentelee noin 10 henkilöä ja tuotantoa on Suomessa sekä Baltiassa. Komponentteja monitoimiuuniin saapuu ympäri maailmaa. (Vainionpää 2016.)

3.3 Monitoimiuuni

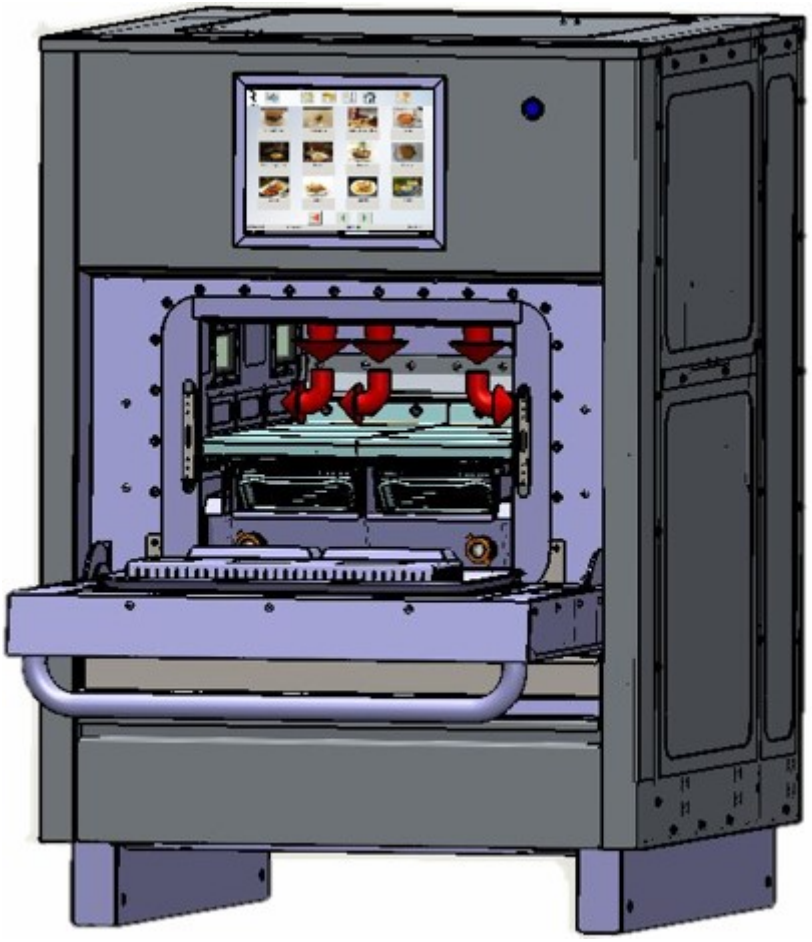
R-Menu Oy:n kehittämä uuniteknologia ja siihen liittyvät logistiikkaratkaisut ovat patentoituna yli 140 maahan. Tämä patentoitu teknologia mahdollistaa nopean ja laadukkaan pakastetun ruuan paistotuloksen. (Vainionpää 2016.)

Yhtenä suurena etuna uunissa on sen katalysaattori, joka poistaa kaikki uunin tuottamat hajut keittiöstä. Tämä tarkoittaa sitä, että keittiöön ei tarvitse hankkia uunin takia paloeristettyä rasvahormia. Kuvassa 2 on esitettynä monitoimiuunin sisältämiä ominaisuuksia ja etuja. (Vainionpää 2016.)



Kuva 2. Monitoimiuunin ominaisuuksia ja etuja (Vainionpää 2016).

Uunin käyttöä on helpottamassa 10” opastava kosketusnäyttö. Tällä kosketusnäytöllä asiakas pystyy tekemään ruokaa ohjeiden mukaan aina samalla laadulla. Uuni tarvitsee toimiakseen 32A pistorasian, sillä sen vaatimat sähkötehot ovat 3x25A. Uuniin on lisäksi ohjelmoitu energiaa säästävä älykäs ohjaus. Asiakkaiden laitteen vikadiagnostiikat tehdään etänä verkkoyhteyden avulla. Verkkoyhteyden kautta asiakkaalle pystytään myös antamaan tarvittavaa etätukea ongelmatilanteissa. Kuvassa 3 on nähtävissä 3D-kuva R-menu Oy:n kehittämästä monitoimiuunista. (Vainionpää 2016.)



Kuva 3. Monitoimiuunin 3D-kuva (Vainionpää 2016).

Monitoimiuuni valmistaa asiakkaalleen ruoka-annokset aina tasalaatuisena. Lisäksi monitoimiuunin omistavan yrityksen ei tarvitse palkata lisähenkilökuntaa kuten kokkia, koska uunia pystyy käyttämään kuka vain. R-Menu Oy:n puolesta tapahtuva tuotekehitys on jatkuvaa, vaikka tuote on jo markkinoilla. Uunin ominaisuuksia pyritään kehittämään ja parantamaan koko ajan. Näiden lisäksi uusia ruoka-annoksia tuodaan jatkuvasti asiakkaiden saataville. (Vainionpää 2016.)

4 LAITTEISTO

4.1 Raspberry Pi

Raspberry Pi on Raspberry Pi Foundation kehittämä pankkikortin kokoinen yhden piirilevyn tietokone, joka pystyy kaikkeen mihin tavallinenkin tietokone kykenee. Tuttavallisemmin Raspberry Pi -minitietokonetta kutsutaan käyttäjien keskuudessa nimellä Raspi, mitä nimeä käytetään myös tässä opinnäytetyössä. Raspiin pystytään kiinnittämään perinteisen tietokoneen

tavoin näppäimistö, hiiri ja näyttö. Näin Raspilla pystytään esimerkiksi selaamaan internetiä sekä katsomaan korkea laatuisia videoita. (Raspberrypi.org FAQS n.d.)

4.1.1 Historia

Raspin kehittäminen lähti Eben Uptonin ideasta vuonna 2006, kun hän työskenteli Cambridgen yliopistossa. Hän huomasi, että Britannian koulutusjärjestelmässä opetettiin tietokoneella tekstinkäsittelyä, taulukkolaskentaa sekä kotisivujen tekemistä, mutta koulutus ei kuitenkaan sisältänyt oikeaa ohjelmointia. Tästä johtuen hän oli myös huomannut ohjelmointitaitojen heikkenemistä tietotekniikkaa opiskelemaan pyrkivillä opiskelijoilla. Tämän ongelman huomattuaan hän keräsi saman henkisiä ihmisiä sekä opettajia kokoon ja he lähtivät kehittämään Raspia. (Brookes 2012.)

Raspberry Pi Foundation on vuonna 2008 Iso-Britanniaan rekisteröity säätiö. Säätiön tavoitteena on edistää nuorten ja aikuisten osaamista ennen kaikkea tietokoneiden, tietotekniikan ja siihen liittyvällä alalla.
(Mullins 2012.)

Yksi tärkeimpiä asioita tuotteen kehittämisessä oli Raspin hinnan pysyminen matalana, jotta opiskelijat voisivat ostaa sen itselleen ja näin kehittää ohjelmointi taitojaan. Ensimmäisen Raspi mallin hinta oli noin 35 USD ja myös uusimman mallin hinta on pysynyt samana. Laitteessa on tapahtunut kehitystä, mutta silti hinta on pystytty pitämään matalana. (Vilches 2012.)

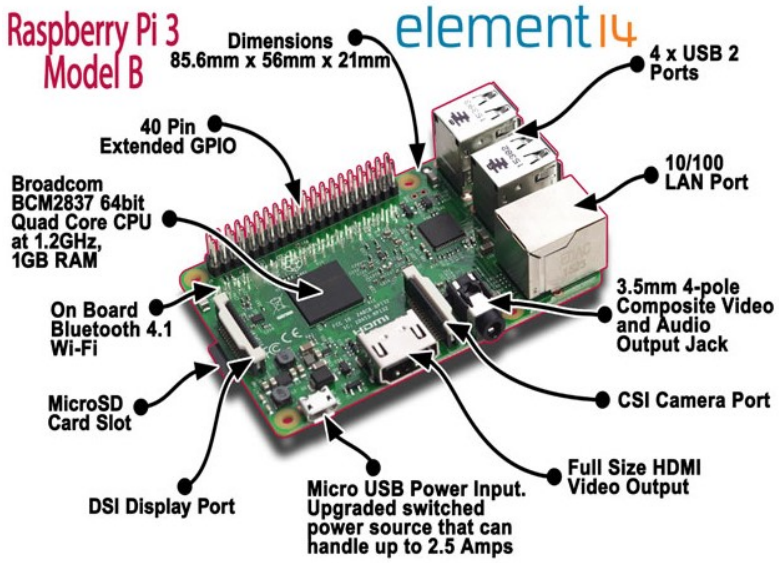
4.1.2 Kehittyminen

Ensimmäisen Raspi mallin tultua markkinoille 29.2.2012, on sitä 29.2.2016 mennessä myyty yli 8 miljoonaa kappaletta. Tämä tekee Raspista eniten myydyimmän Iso-Britannialaisen tietokoneen. Raspberry Pi Foundation on kasvanut muutamasta vapaaehtoistyöntekijästä säätiöksi, joka työllistää tällä hetkellä yli 60 kokopäiväistä työntekijää. (Upton 2016.)

Ensimmäinen julkaistu Raspi malli oli nimeltään Raspberry Pi Model B (Kuva 4.), jossa oli keskusmuistia vain 256 MB. Vuoden 2012 lopulla Raspberry Pi Foundation oli saanut asiakkailtaan paljon toivomuksia koskien keskusmuistin koon kasvattamista. Lokakuussa 2012 Model B:tä alettiinkin valmistaa 516 MB keskusmuistilla. (Upton 2012.)

Kuva 4. Raspberry Pi Model B (EasyIoT n.d.).

Markkinoiden uusin Raspberry Pi 3 Model B (Kuva 5.) julkaistiin 29.2.2016. Tässä mallissa käytetään Broadcomin BCM2837-järjestelmäpiiriä. Uusimman Raspin prosessorina toimii 1.2 GHz 64-bit quad-core ARMv8. Muistia markkinoiden uusimmasta Raspista löytyy 1 GB. Näiden lisäksi laitteessa on myös monia muita ominaisuuksia kuten neljä USB-porttia, 40 GPIO-pinniä, HDMI-portti, ethernet-portti, sisäänrakennettu WLAN, bluetooth 4.1, 3,5mm stereoliitäntä, tuki microSD muistikortille sekä liitännät kameralle ja kosketusnäytölle. Raspberry Pi 3 Model B on myös täysin yhteen sopiva aikaisempien mallien kanssa. (Raspberrypi.org Raspberry Pi 3 Model B n.d.)



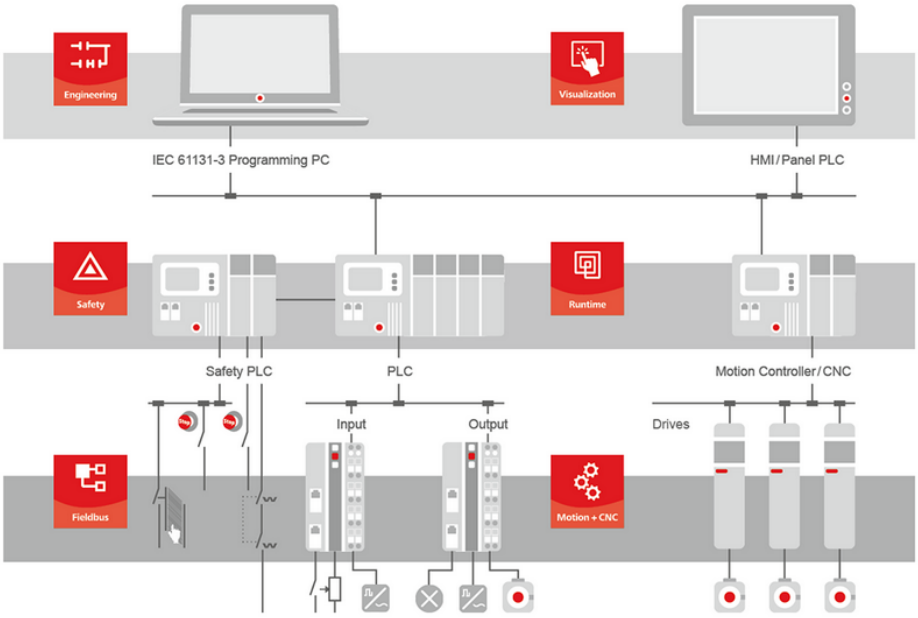
Kuva 5. Raspberry Pi 3 Model B ja sen ominaisuuksia (Harris, 2016).

4.2 CoDeSys

CoDeSys (Controller Development System) on 3S-Smart Software Solutions GmbH:n valmistama ohjelmointiympäristö, joka on laitteistoriippumaton IEC 61131-3 –standardin mukainen ohjelmisto Windows tai Linux pohjaisten laitteiden sekä sulautettujen ja ilman käyttöjärjestelmää toimivien laitteiden ja sovellusten ohjelmointiin. (SKS Group Oy 2016.)

3S-Smart Software Solutions on Dieter Hessin ja Manfred Wernerin vuonna 1994 perustama yritys, jonka pääkonttori sijaitsee Saksassa. CoDeSys on ilmainen ohjelma, joka tarjoaa käyttäjäystävällisiä ratkaisuja käyttäjän tarpeille erilaisiin automaatioprojekteihin. (3S-Smart Software Solutions GmbH 2016.)

CoDeSys-ohjelmistoympäristöön on integroituna kaikki, mitä automaatioprojektin toteutukseen tarvitsee. Se sisältää kenttäväylän- ja I/O -konfiguroinnin sekä visualisointi- ja liikkeenohjaustyökalun sekä paljon muuta. Näistä osa on nähtävissä kuvassa 6, josta nähdään myös niiden yhteensopivuus. (3S-Smart Software Solutions GmbH 2016.)



Kuva 6. CoDeSys-ohjelmointiympäristö (3S-Smart Software Solutions GmbH 2016).

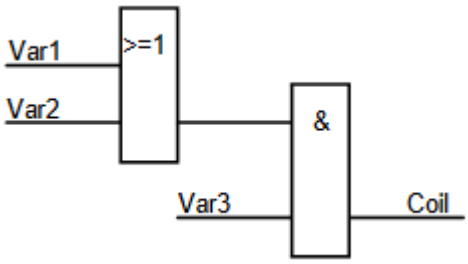
4.3 Ohjelmointikielet

CoDeSys-ohjelmointiympäristössä pystytään ohjelmoimaan viidellä eri ohjelmointikielellä, jotka ovat määriteltynä standardissa IEC 61131-3. Kaksi näistä on tekstimuodossa ja kolme graafisesti toimivia ohjelmointikieliä. Näiden viiden standardissa määritetyn ohjelmointikielen lisäksi CoDeSys-

ohjelmalla ohjelmoitaessa on mahdollista käyttää myös CFC ohjelmointikieltä. (PLC Manual n.d.)

4.3.1 Toimilohkokaavio

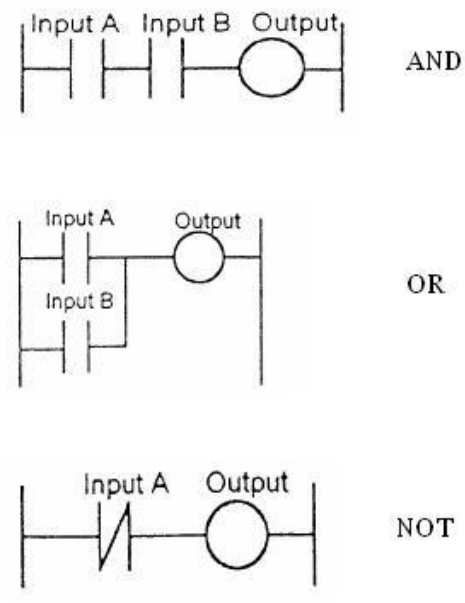
Toimilohkokaavio eli Function Block Diagram (FBD) on graafinen ohjelmointikieli. FBD-kielellä ohjelmoitaessa ohjelma luodaan käyttäen perinteisiä boolean algebran määritelmiä kuten AND, OR, NOT. Kuvan 7 esimerkissä on nähtävässä yksinkertainen ohjelma, joka on tehty FBD-ohjelmoinnilla. Siinä lähtö Coil ei mene päälle, ellei Var1 tai Var2 ja Var3 ole päällä. (Toshiba Corporation 2008.)



Kuva 7. Esimerkki FBD -ohjelmointikielellä tehdystä ohjelmasta (John & Tiegelkamp 2001, 249).

4.3.2 Tikapuukaavio

Tikapuukaavio eli Ladder Diagram (LD) on perinteinen graafinen ohjelmointikieli. Tikapuukaavio-ohjelmointikieltä on alun perin käytetty, koska releohjaukset on ollut helppo muuttaa tälle ohjelmointikielelle. Ohjelmointikielen nimi tikapuukaavio johtuu siitä, että ohjelmoinnin kokonaisuus näyttää tikapuulta, jossa on kaksi pystykiskoa. Tikapuiden jokainen poikkikisko kuvastaa yhtä ohjauspiiriä. Kuvan 8 esimerkissä nähdään kolme eri boolean algebran määritelmän ohjauspiiriä, jotka on toteutettu tikapuukaaviolla. (Kuphaldt n.d.)



Kuva 8. Tikapuukaavio esimerkkejä (Alander n.d.).

4.3.3 Käskylista

Käskylista eli Instruction List (IL) on tekstipohjainen ohjelmointikieli. Käskylista koostuu sarjasta ohjeita. Jokainen ohjelman käsky alkaa uudelta riviltä. Jokainen rivi sisältää käskyn sekä halutun käskyn mukaan joko yhden tai useamman niin sanotun operandin. Nämä operandit tulee erottaa käskylistassa pilkulla. (Beckhoff Instruction List n.d.)

```
LD      17
ST      lint          (* comment *)
GE      5
JMPC    next
LD      idword
EQ      istruct.sdword
STN     test
next:
```

Kuva 9. Käskylistalla tehty esimerkkiohjelma (Beckhoff Instruction List n.d.).

4.3.4 Strukturoitu teksti

Strukturoitu teksti eli Structured Text (ST) on myös tekstipohjainen ohjelmointikieli. Tätä ohjelmointi tapaa pidetään hyvin järkevänä isommissa ohjelmointi kokonaisuuksissa. Graafisissa ohjelmointikielissä, kuten tikapuukaaviossa pidetään ongelmana sen vaikea lukuisuutta verrattuna tekstipohjaisesti tehtyyn ohjelmaan. (PLC Academy 2015.)

Strukturoitu teksti koostuu sarjasta ohjeita, jotka ovat samoja kuin korkeatasoisilla ohjelmointikielillä. Näitä sanoja, joita ohjelmoinnissa käytetään ovat esim. IF, THEN, ELSE. Lisäksi ohjelmoinnissa voidaan käyttää erilaisia silmukoita kuten esim. WHILE. (Beckhoff Structured Text n.d.)

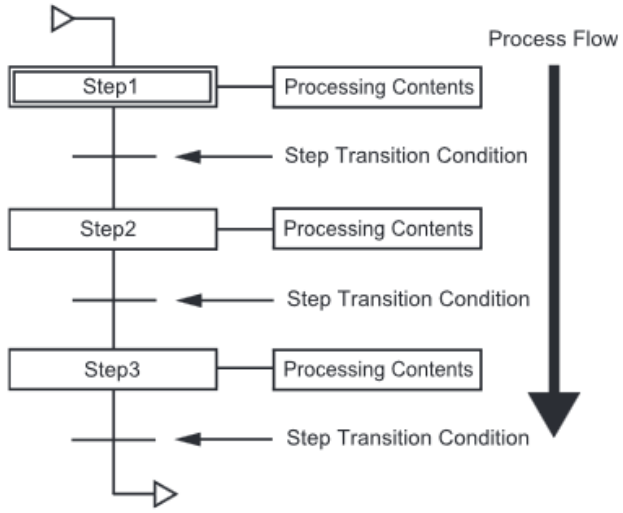
Kuvassa 10 on esitettynä ohjelma, joka on tehty Structured Text ohjelmointikielellä. Kuvan esimerkissä ohjelma summaa luvun 1 ohjelmassa tiedusteltavaan lukuun, jos tämä luku on alle 7. Ohjelma kysyy ja lisää lukua 1 niin kauan kunnes kysyttävä luku on alle 8.

```
IF value < 7 THEN
    WHILE value < 8 DO
        value := value + 1;
    END_WHILE;
END_IF;
```

Kuva 10. Strukturoidulla tekstillä toteutettu ohjelma (Beckhoff Structured Text n.d.).

4.3.5 Sekvenssiohjausohjelmointi

Sekvenssinohjausohjelmointi eli Sequential Function Chart (SFC) on graafinen ohjelmointikieli, jonka käyttö on suosittua sen takia, että siinä ohjelma etenee järjestyksessä askelittain. Tällä ohjelmointikielellä (Kuva 11.) tehdyn ohjelman tilaa on helppo seurata ja ohjelman kokonaisuus on helppo sisäistää. (SFC Introduction Guide n.d.)

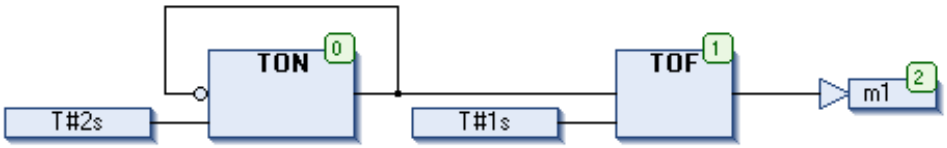


Kuva 11. SFC -ohjelmointikielen kokonaisuus (Omron n.d.).

4.3.6 Sekvenssikaavio-ohjelmointi

Sekvenssikaavio-ohjelmointi, toiselta nimeltään jatkuvatoiminen ohjelmointi eli Continuos Function Chart (CFC) on standardin IEC 61131-3 jatkeena kehitetty graafinen ohjelmointikieli. Tämä ohjelmointikieli perustuu

IEC 61131-3 mukaiseen toimintalohko-kaavio ohjelmointikieleen. Tässä ohjelmointikielessä käytetään vapaasti graafisia elementtejä, jotka mahdollistavat takaisinkytkentätiedon saamisen. Kuvassa 12 on nähtävissä esimerkki siitä, kuinka CFC-ohjelmointikielellä tehty ohjelma rakentuu. (Beckhoff CFC n.d.)



Kuva 12. CFC -ohjelmointikieli (Beckhoff CFC n.d.).

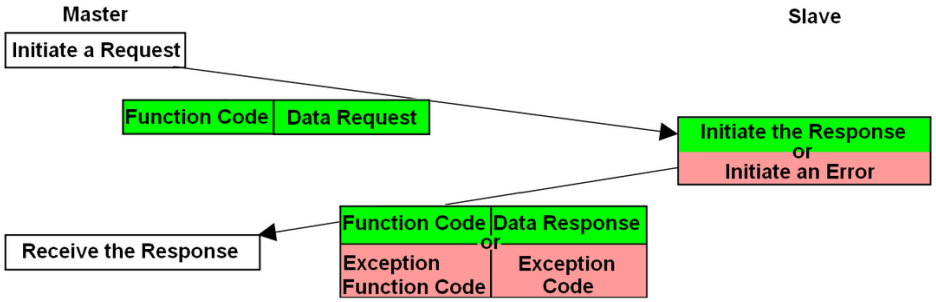
4.4 **Modbus**

Modbus on Modiconin kehittämä sarjaliikenneprotokolla, joka julkaistiin vuonna 1979. Modicon on alun perin kehittänyt Modbusin käytettäväksi heidän omassa ohjelmoitavassa logiikassaan. Modbus on menetelmä, jolla pystytään lähettämään tietoa erilaisten elektronisten laitteiden välillä. (Simply Modbus Frequently Asked Questions 2015.)

Modbus on avoin protokolla joka tarkoittaa, että eri valmistajien on ilmaista käyttää sitä osana laitteitaan. Se on tullut ajan mittaan standardiksi viestintäprotokollaksi teollisuudessa. Modbus on tällä hetkellä eniten käytetty tiedonsiirto tapa eri elektronisten laitteiden välillä. Modbus on tyypillisesti käytettynä sellaisissa laitteissa, jotka keräävät tietoa tuotannosta ja se pitää välittää pääohjaimelle. Esimerkiksi järjestelmä, joka mittaa lämpötilaa sekä kosteutta ja nämä tiedot pitää välittää tietokoneelle. (Simply Modbus Frequently Asked Questions 2015.)

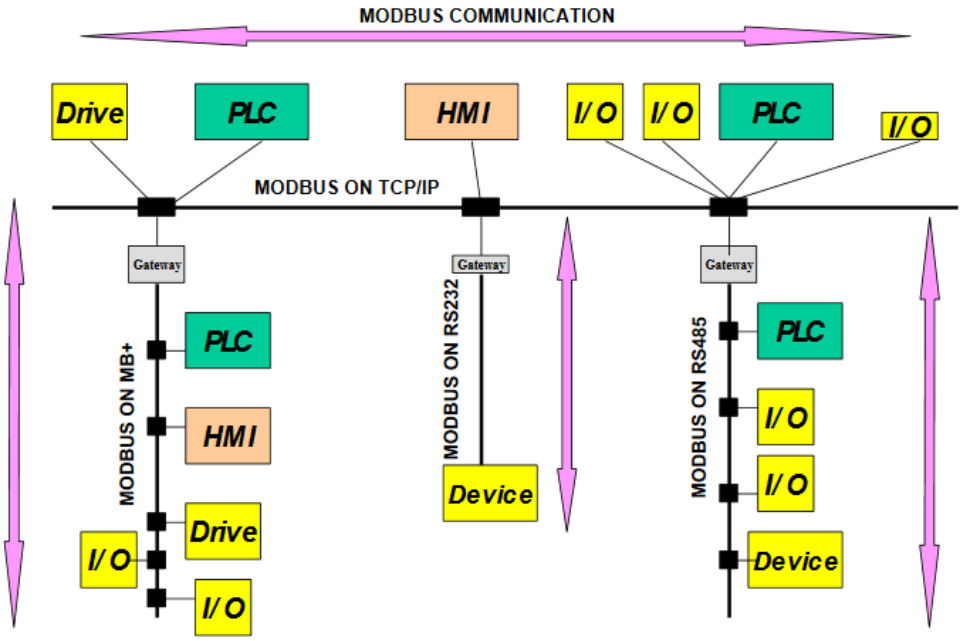
4.4.1 Modbus-protokolla

Modbus-protokolla on määriteltynä master-slave protokollaksi, joka tarkoittaa sitä, että samaan väylään tulee olla kytkettynä yksi master-laite sekä vähintään yksi slave-laite. Master aloittaa aina tiedonsiirron, koska slave-laite ei tähän kykene (Kuva 13). Slave-laitteilla ei myöskään ole mahdollisuutta kommunikoida keskenään. Slave-laitteet eivät myöskään pysty lähettämään mitään master-laitteelle, ellei se sitä erikseen pyydä. (Fläkt woods 2016.)



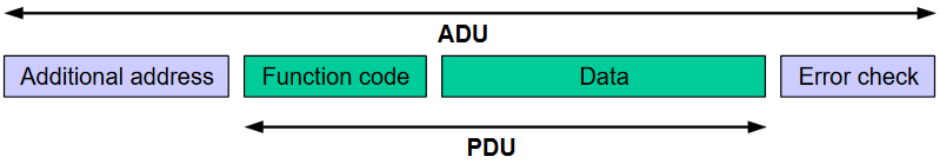
Kuva 13. Modbus-tiedonsiirto (Witthoeft 2013).

Modbus-protokolla mahdollistaa helpon viestinnän erilaisten verkkojen välillä. Kaikenlaiset laitteet kuten PLC:t, ohjauspaneelit, I/O -laitteet jne. pystyvät käyttämään Modbus-protokollaa hyväkseen (Kuva 14). Sama Modbus-viestintä pystytään tekemään sarjaliikenteellä tai ethernetin TCP/IP verkoissa. (Modbus 2012.)



Kuva 14. Esimerkki Modbus-liikenteen arkkitehtuurista (Modbus 2012).

Modbus-protokolla määrittelee yksinkertaisen PDU:n eli Protocol Data Unit:in, joka on itsenäinen yhteyskerroksesta erillään oleva paketti. Se tarkoittaa käytännössä Modbusin viestikehystä. PDU sisältää Function Coden ja datapaketin. ADU eli Application Data Unit sisältää vapaaehtoisia kenttiä, PDU:n sekä virheentarkistuksen (Kuva 15). (Modbus 2012.)

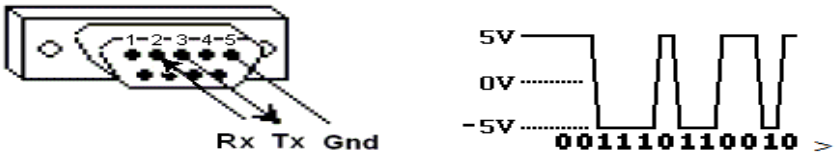


Kuva 15. Yleinen Modbus-viestikehys (Modbus 2012).

4.4.2 Modbus-tiedonsiirto

Modbus-väylällä on kolme mahdollista eri lähetystapaa, jolla tietoa pystytään siirtämään. Nämä lähetystavat ovat Modbus RTU, Modbus ASCII sekä Modbus TCP/IP. (Simply Modbus Frequently Asked Questions 2015.)

Näistä lähetystavoista perinteisin lähetystapa on RTU (Remote Terminal Unit), jossa tietoa lähetetään sarjaportin (RS-232 tai RS-485) kautta laitteiden välillä. RTU-muodossa lähetetty data kulkee sarjaportin kautta sarjana ykkösiä ja nollia, joita kutsutaan biteiksi. Jokainen näistä lähetettävistä biteistä lähetetään jännitteenä. Ykköset lähetetään negatiivisella jännitteellä ja nollat taas lähetetään positiivisella jännitteellä, kuten kuvassa 16 on esitettynä. Sarjaliikenneportin liittimissä olevat merkinnät Rx tarkoittaa vastaanottavaa pinniä ja Tx dataa lähettävää pinniä. GND on Ground eli maadoitus. (Simply Modbus Frequently Asked Questions 2015.)



Kuva 16. Modbus-tiedonsiirto binäärisenä (Simply Modbus Frequently Asked Questions 2015).

Modbus ASCII -viestit ovat tehottomampia kuin RTU-viestit johtuen siitä, että Modbus RTU käyttää lähettämisessä binäärikoodeja ja sen virheentarkastus menetelmä on CRC. CRC on tehokkaampi kuin Modbus ASCII:n käyttämä LRC-virheentarkistusmenetelmä. Modbus ASCII käyttää nimensä mukaisesti datan siirtämiseen ASCII-merkkejä, joten se on helpommin luettavissa kuin RTU:n käyttämä binäärinen viesti. Modbus ASCII -viestit vaativat kuitenkin kaksi kertaa enemmän tavuja lähettämään saman sisällöin kuin Modbus RTU. (ProSoft Technology 2011.)

Modbus TCP/IP (Transmission Control Protocol/Internet protocol) on internet-liikenteen -protokolla. Kun tietoja halutaan siirtää käyttäen TCP/IP -protokollaa, silloin lähetettävä data siirretään TCP:hen, jossa lähetettävälle datalle annetaan oma IP-osoite. IP sijoittaa lähetettävän datan oikein ja sen jälkeen lähettää sen eteenpäin. TCP:n on muodostettava yhteys ennen kuin se pystyy lähettämään dataa. TCP-protokollassa masteria kutsutaan clientiksi ja se ottaa yhteyden slaveen eli serveriin. (Simply Modbus Modbus TCP/IP 2015.)

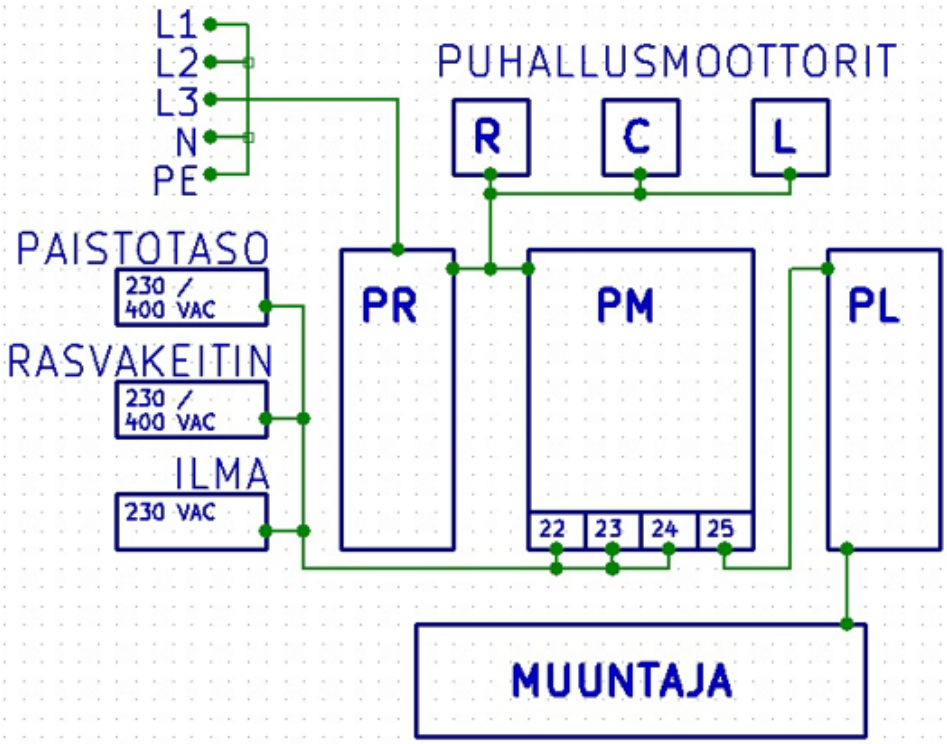
5 KEHITYSPROJEKTI

5.1 Projektin lähtökohdat

Monitoimiuunin sähköisten testien automatisointiin käytettävän laitteiston kehitysprojektin alussa tutustuttiin tarkemmin monitoimiuunin rakenteeseen. Monitoimiuunin rakenteen hahmottaminen kokonaisuudessa oli tärkeää projektin onnistumisen kannalta. Tärkeimpänä tutustumisen aiheena oli uunin sähköinen rakenne, uunin käyttämä ohjelmistosofta sekä uunin kommunikaatiossa käytetty väylätekniikka.

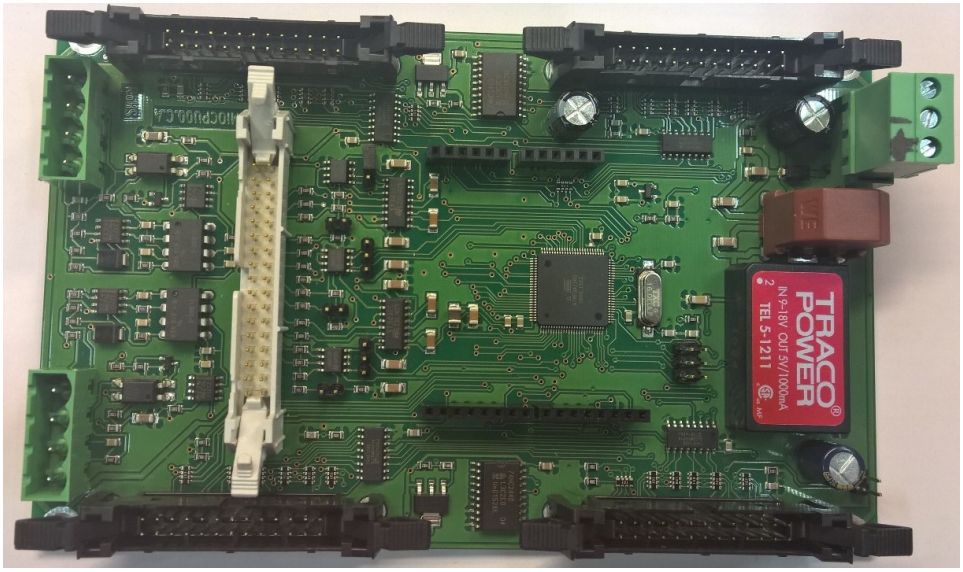
Uunin sähköisten rakenteiden hahmottaminen oli aluksi melko vaikeaa johtuen johtosarjojen ja komponenttien suuresta määrästä. Vaikka uunin sähköistys oli tärkeää tuntea kokonaisuudessaan, keskityttiin tässä kuitenkin uunin sähkömoduuleihin. Niiden toiminta oli tärkeää tuntea, koska ne olivat keskeisessä asemassa kehitysprojektin onnistumisen kannalta. Monitoimiuunissa on yhteensä viisi sähkömoduulia, jotka sisältävät suurimman osan kaikista uunin sähköisistä komponenteista. Sähkökomponentit ovat alun perin sijoitettu moduuleihin, jotta niiden vaihtaminen olisi mahdollisimman helppoa.

Oikean puolen sähkömoduulissa (PR) sijaitsee uunin toimilaitteiden kaikki sulakkeet, riviliittimet sekä liitäntä uunin käyttämän verkkojännitteen syöttökaapelille. Vasemman puolen sähkömoduuli (PL) sisältää sähkönsuodattimia ja tasavirtalähteen, josta se jakaa jännitettä sitä tarvitseville komponenteille. Uunin kolmas sähkömoduuli on nimeltään muuntajamoduuli, joka syöttää käyttöjännitteen uunin neljälle magnetronille. Lisäksi uunissa on PC-moduuli, jossa sijaitsee uunin PC sekä sen kosketusnäyttö, jolla uunin käyttäjä ohjaa uunin eri toimintoja. Takamoduulissa (PM) sijaitsee kaikki uunin ohjaukseen tarvittavat piirikortit ja triac-kortit. Triac-kortit ohjaavat uunin toimilaitteita, ja ne käyttävät 230 tai 400 VAC-jännitettä. Nämä triac-kortit saavat ohjauksen takamoduulissa sijaitsevalta piirikortilta, joka on tärkein osa tämän kehitysprojektin onnistumisen kannalta. Kuvassa 17 on nähtävissä monitoimiuunin sähkömoduulit, toimilaitteet sekä triac-kortit takamoduulissa numeroituna 22-25, jotka toimivat verkkojännitteellä. Kuvasta 17 puuttuu PC-moduuli, mutta se on kytkettynä suoraan uunin takamoduuliin (PM).



Kuva 17. Uunin moduulit ja toimilaitteet (Sundström 2015).

Edellisessä kappaleessa mainittu uunin piirikortti, joka ohjaa uunin toimilaitteita ja sijaitsee takamoduulissa, on nähtävissä kuvassa 18. Kyseinen piirikortti sisältää uunin käyttämän ohjelmistosoftan. Uunin PC-moduulista ollaan yhteydessä tähän piirikorttiin Modbus-väylän kautta. Tämä tarkoittaa sitä, että uunin käyttäjän antamat käskyt välittyvät kosketusnäytöltä PC:n kautta uunin piirikortille, joka käsittelee käyttäjän ohjaukset. Sieltä se lähettää käskyt triac-korteille, jotka taas ohjaavat uunin toimilaitteita. Tätä piirikorttia käytettiin kehitysprojektin toiminnan varmistamisessa, koska yhteyden toimiessa tähän piirikorttiin se toimisi myös uuniin.



Kuva 18. Uunin ohjaamiseen käytettävä piirikortti

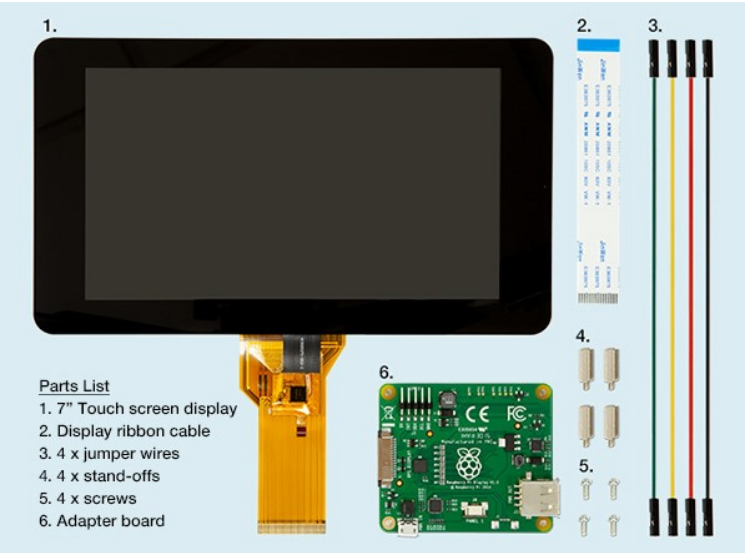
Uunin väylätekniikan tunteminen oli kehitysprojektin onnistumisen kannalta tärkeää, koska testilaitteiston tuli kommunikoida uunin kanssa. Testilaitteiston ja uunin piirikortin kommunikaation toimiessa testauslaitteisto pystyy ohjaamaan uunin toimilaitteita sekä lukemaan uunista saatavia I/O-tietoja. Monitoimiuunissa väylätekniikkana toimii Modbus RTU RS232 ja tätä samaa väylätekniikkaa käytettiin myös yhdistettäessä testilaitteistoa uuniin.

5.2 Kehitysprojektissa käytetyt laitteet

Kehitysprojektin alussa laitteiston valinta vaati paljon tutustumista eri vaihtoehtoihin, koska sen piti olla mahdollisimman helposti saatavissa yhteensopivaksi uunin kanssa. Kehitysprojektin laitteistoa valittaessa tärkeänä asiana oli myös sen mahdollisuus kommunikoida monitoimiuunin kanssa Modbus RTU -väylätekniikan avulla. Testauslaitteiston valinnassa vaikuttivat myös sen helppokäyttöisyys, kehittämismahdollisuudet sekä hinta. Eri vaihtoehtoihin tutustumisen jälkeen tässä kehitysprojektissa käytettäväksi laitteistoksi valikoitui seuraavaksi esitellyt laitteet.

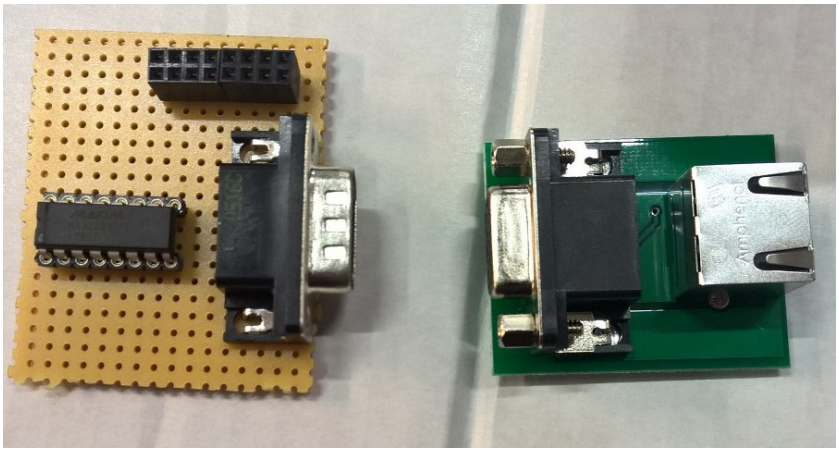
Testauslaitteiston pääkomponenttina toimi jo tässäkin opinnäytetyössä luvussa 4.1 tarkemmin esitelty Raspberry Pi 3 Model B. Raspiin hankittiin erikseen virtalähde, jonka lähtövirta on 2,5A. Raspi, kosketusnäyttö ja kommunikointiin vaadittavat piirilevyt suojattiin niille suunnitellulla suojakotelolla.

Kuvassa 19 on tässä kehitysprojektissa käytetty Raspin oma 7” kosketusnäyttö ja sen mukana tulleet osat. Näytön mukana tuli nauhakaapelia, neljä liitäntä johtoa, korotus- ja kiinnitysruuvit. Näiden lisäksi mukana tuli näytön piirilevy, jolla se tullaan kytkemään Raspiin. Kosketusnäytön resoluutio on 800 x 480. (element14 2016)



Kuva 19. Raspin 7” kosketusnäyttö ja varusteet (element14 2016).

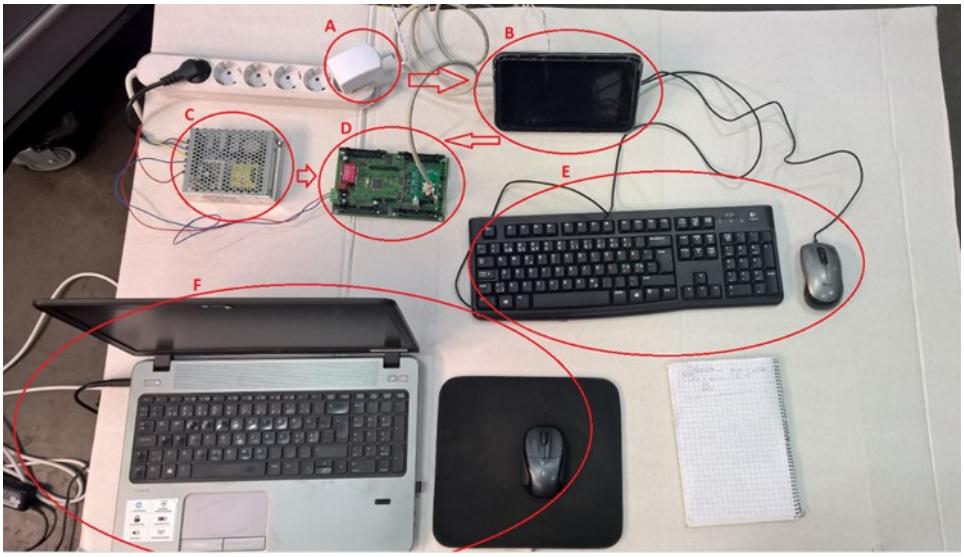
Jotta Raspi pystyi kommunikoimaan uunin takamoduulissa sijaitsevan piirikortin kanssa Modbus-sarjaliikenteen kautta, tarvitsi se kaksi erilaista piirilevyä. Toinen näistä piirilevyistä oli itse rakennettu ja se muutti Raspin GPIO-pinneistä tiedon RS232-muotoon eli D9-liittimelle. Toinen piirilevy muutti taas datan D9-liittimeltä RJ45-liittimelle, josta pystyttiin muodostamaan yhteys uunin piirikortille CAT6-kaapelilla. Myöhemmin tässä dokumentissa tullaan tarkemmin kertomaan, miten Raspi 3 ja uunin piirikortti saatiin kommunikoimaan keskenään. Nämä kaksi tiedonsiirtoon tarvittavaa piirilevyä on nähtävissä kuvassa 20.



Kuva 20. Raspiin liitettävät piirilevyt

Näiden laitteiden lisäksi kehitysprojektissa tarvittiin kannettavaa tietokonetta, jolla Raspi 3:sta ohjelmoitiin. Tietokoneen tuli olla tarpeeksi tehokas pyörittääkseen CoDeSys V3.5 -ohjelmistoa. Ohjelmointi vaiheessa Raspissa käytettiin USB-johdolla toimivaa näppäimistöä ja hiirtä, vaikkakin se olisi toiminut kosketusnäytönkin kautta. Raspin käyttöjärjestelmän asentamiseen tarvittiin muistikorttia, jolla käyttöjärjestelmä siirrettiin kannettavalta tietokoneelta Raspiin.

Kuvassa 21 on kirjaimilla merkittynä laitteet, jotka kuuluivat testauslaitteiston kehittämiseen. A-kirjaimella kuvaan on merkitty Raspin oma virtalähde. Se syötti virtaa Raspille ja sen kosketusnäytölle, joka on merkattuna kuvassa kirjaimella B. Raspia pystyttiin käyttää myös kosketusnäytöltä, mutta projektin kehitysvaiheessa oli käytännöllisempää käyttää USB-hiirtä ja -näppäimistöä, jotka ovat merkattuna kuvaan kirjaimella E. Raspi oli kytkettynä CAT6-kaapelilla uunin piirikortille. Uunin piirikortti on merkattuna kuvaan kirjaimella D ja se otti käyttöjännitteensä erillisestä 12V virtalähteestä C. Lisäksi kuvaan on merkattu kirjaimella F kannettava tietokone, jolla tässä kehitysprojektissa toteutettiin testauslaitteiston ohjelmointi.



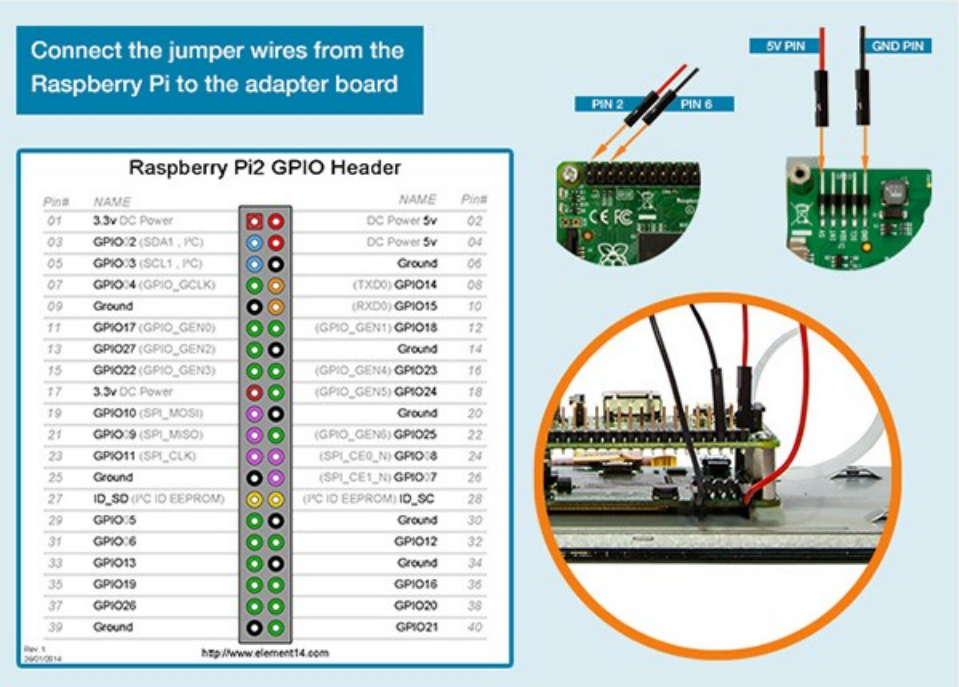
Kuva 21. Opinnäytetyössä käytettävä laitteisto

Laitteistoon päädyttiin, koska CoDeSys on ilmainen ohjelma ja Raspi on halpa ja sitä voidaan käyttää tässä tarkoituksessa diagnostiikkatyökaluna. Tässä kehitysprojektissa kannettavalla tietokoneella tehtiin ohjelma ja käyttöliittymä käyttäen CoDeSys V3.5 -ohjelmaa. CoDeSys yhdistettiin Raspiin käyttäen sen lisäosaa nimeltä CoDeSys Control for Raspberry Pi. Raspiin eli testauslaitteistoon ladattiin CoDeSysillä tehty ohjelma ja käyttöliittymä internetin välityksellä WebVisuna. Työn tilaaja tulee tekemään jatkossa sähköisten testien automatisoinnin samalla periaatteella monitoimiuuniin. Tässä projektissa Raspi määritettiin toimimaan laitteiston masterina ja uunin piirikortti slavena.

5.3 **Laitteiston käyttöönotto**

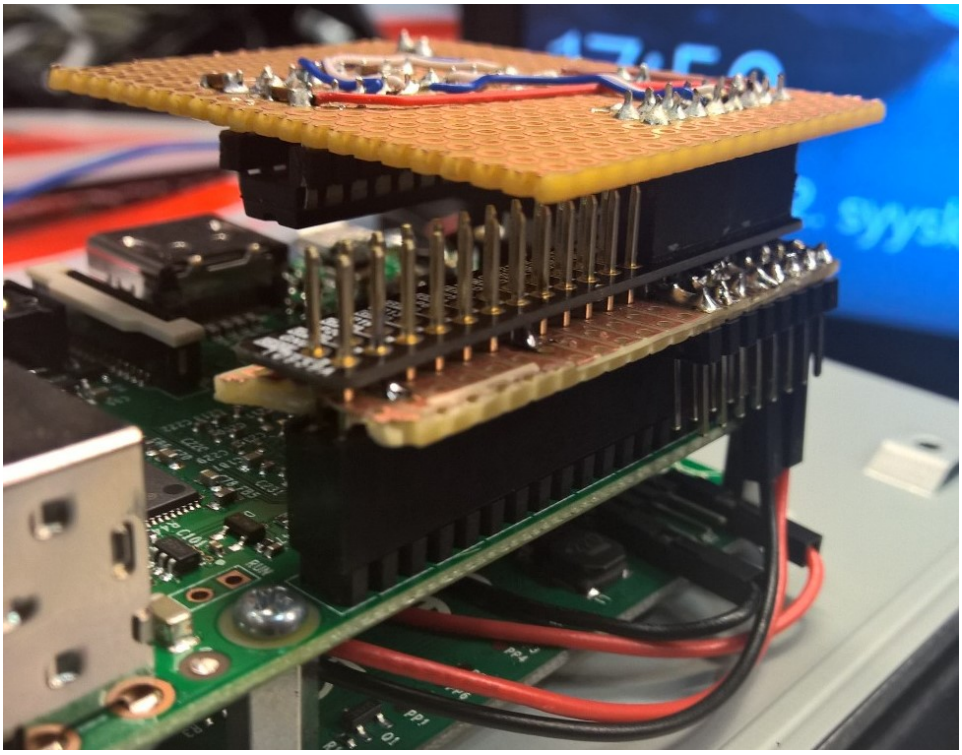
5.3.1 Fyysiset kytkennät

Ennen kun kehitysprojektissa testauslaitteistona toiminutta Raspia pystyttiin ottamaan käyttöön, kytkettiin se oheislaitteisiinsa kiinni. Ensimmäisenä Raspiin kiinnitettiin sen oma 7” kosketusnäyttö. Siinä Raspi kiinnitettiin korotusruuvien avulla kosketusnäytön piirikorttiin ja tämän jälkeen kahden hyppyjohdon avulla Raspin GPIO-pinneistä otettiin virransyöttö kosketusnäytön piirikortille. Jotta näyttöön saatiin Raspista kuva, niin se kytkettiin nauhakaapelia käyttäen, siihen tarkoitettujen liittimien kautta. Nämä kytkennät ovat nähtävissä kuvassa 22.



Kuva 22. Raspin ja sen kosketusnäytön kytkennät (element14 2016).

Tässä kehitysprojektissa Modbus RTU:ta käytettiin kommunikaatiossa kehitetyn testilaitteiston ja uunin piirikortin välillä. Raspiin tuli tämän takia lisätä kaksi piirilevyä, jotta tieto saataisiin kulkemaan oikealla tavalla. Nämä piirilevyt ovat esitettynä tarkemmin kappaleessa 5.2. Ensimmäinen, itsetehty piirilevy, joka muuttaa tiedon Raspin GPIO-pinneistä D9-liittimelle asennettiin Raspin GPIO-pinneihin. Tässä kohtaa ongelmana oli, että Raspin kosketusnäyttö oli jo varannut Raspin GPIO-pinneistä itse tehdylle piirilevylle tarvittavia pinnejä. Tämän takia piirilevyä ei pystytty suoraan asentamaan GPIO-pinneihin vaan tehtiin kuvan 23 mukainen ratkaisu. Siinä piirilevylle juotettiin lisärivi pinnejä, jolloin tarvittavat pinnit tuplaantuivat. Tämän jälkeen pystyttiin toisiin pinneihin kytkemään näytön virransyöttö sekä maadoitus ja lisäpinneihin oma piirilevy. Itse tehdylle piirilevylle tarvittiin Raspin GPIO-pinneistä pinnit 2,6,8 ja 10. Nämä olivat DC Power 5V, Ground, TXD0 ja RXD0. Oma tekemään piirilevyyn liitettiin toinen piirikortti, joka muuttaa tiedon D9-liittimeltä RJ45-liittimeen, josta pystytään kommunikoimaan uunin piirikortin kanssa.



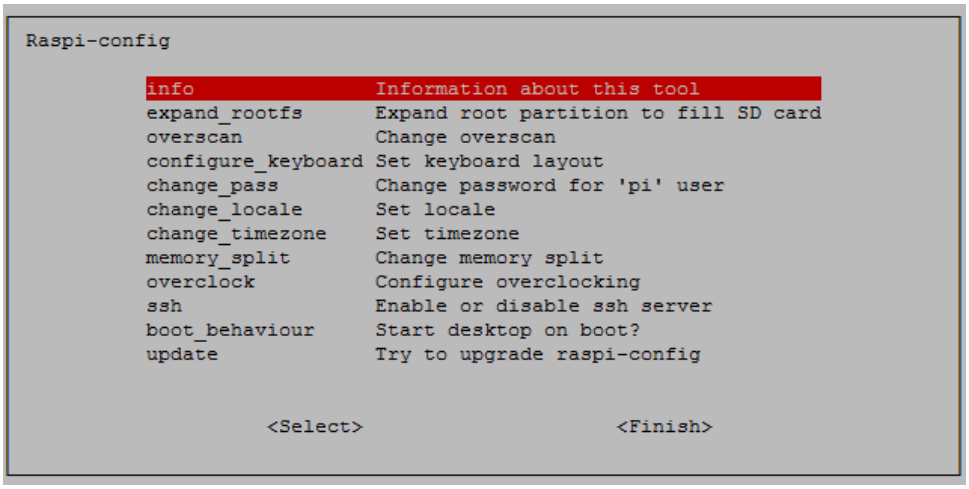
Kuva 23. Itse tehdyn piirikortin kytkeminen Raspiin

5.3.2 Raspberry Pi

Kun Raspia otettiin käyttöön, tarvitsi se toimiakseen käyttöjärjestelmän. Raspia käyttöönotettaessa ja laitteeseen käyttöjärjestelmää asentaessa tarvittiin jokin näyttö asetuksien asettamista varten. Tässä projektissa käytettiin Raspin omaa kosketusnäyttöä, joka oli myös osana tätä kehitysprojektia. Vaihtoehtoisesti tässä olisi voitu käyttää mitä vain tietokoneen näyttöä, joka olisi kytketty Raspin HDMI-porttiin. Tässä tilanteessa kosketusnäyttöä olisi pystynyt myös käyttämään Raspin ohjaamiseen sitä konfiguroitaessa, mutta helpotukseksi siihen asennettiin hiiri sekä näppäimistö. Näiden lisäksi käyttöönotossa tarvittiin muistikorttia sekä tietokonetta, jossa on muistikortinlukija.

Vaihtoehtoja käyttöjärjestelmälle oli useita ja oli täysin käyttäjän päätettävissä, mitä käyttöjärjestelmää haluttiin käyttää. Tässä kehitysprojektissa käyttöjärjestelmäksi valikoitui Rasbian Wheezy niminen käyttöjärjestelmä, koska se muistutti paljon Windows-käyttöjärjestelmää ja siksi se oli melko yksinkertainen käytettävyydeltään. Rasbian ladattiin ilmaiseksi netistä raspberrypi.org nettisivuilta. Kun se ladattiin kannettavalle tietokoneelle, tuli Win32DiskImager-ohjelma olla myös asennettuna tietokoneelle. SD-kortti laitettiin tietokoneen kortinlukijaan ja avattiin Win32DiskImager ohjelma. Tällöin aukesi ikkuna mihin haettiin tietokoneelle ladattu Rasbian Wheezy käyttöjärjestelmä. Sitten Win32DiskImager ohjelmaan valittiin Device kohtaan se levyasema, johon tietokone oli muistikortin luonut. Tämän jälkeen painettiin Write-komentoa, jolloin ohjelma loi käyttöjärjestelmän muistikortille.

Nyt muistikortti voitiin poistaa kannettavasta tietokoneesta ja asentaa Raspiin. Tässä vaiheessa oli Raspiin hyvä olla kytkettynä jo kaikki oheislaitteet kuten näyttö, hiiri ja näppäimistö USB-porttien kautta. Virtajohto kytkettiin viimeisenä Raspiin kiinni, koska se käynnisti laitteen automaattisesti, sillä Raspissa ei ole virtanappia. Seuraavaksi laite käynnistyi ja hetken kuluttua näyttöön ilmestyi Raspi-config pääikkuna (Kuva 24).



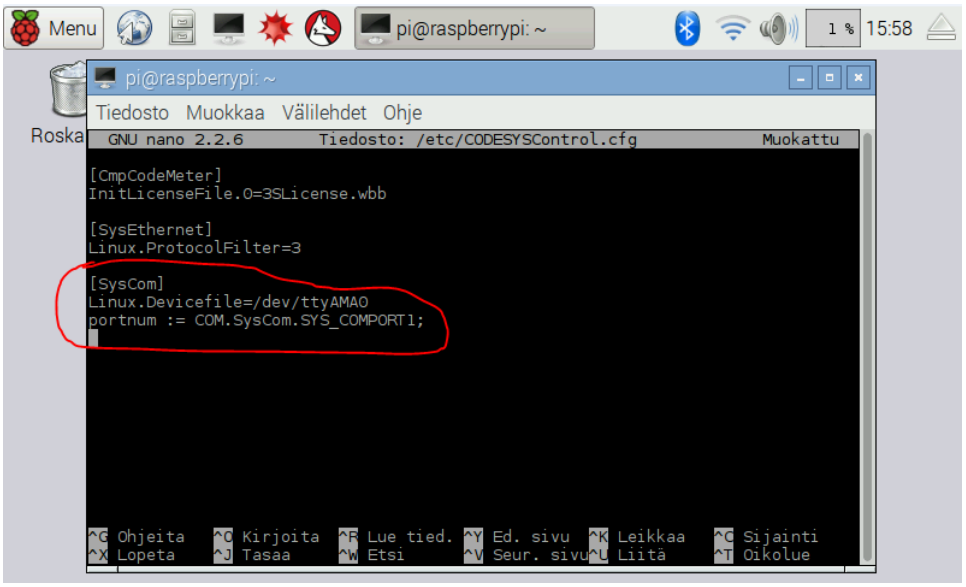
Kuva 24. Raspi-config pääikkuna

Ensimmäiseksi Raspi-config ikkunassa siirryttiin kohtaan expand_rootfs, jossa painettiin enter-näppäintä. Tämä tehtiin, jotta muistikortista saadaan käyttöön kokonaan sen muistintila. Tämän jälkeen näppäimistö asetettiin suomenkieliseksi valitsemalla Raspi-config -ikkunasta configure_keyboard. Tästä aukesi valikko, josta etsittiin projektissa käytettävä suomenkielinen näppäimistö. Tässä vaiheessa viimeinen Raspi-config -valikosta tehtävä muutos oli vaihtaa Raspin aikavyöhyke oikeaksi. Aikavyöhyke valittiin painamalla valikosta kohtaa change_timezone ja valitsemalla sieltä Europe ja Helsinki. Jotta edellä mainitut asetukset tulisivat voimaan, niin täytyi Raspberry Pi käynnistää uudelleen. Käynnistyessään uudestaan Raspia niin se pyysi asettamaan käyttäjätunnuksen ja salasanan. Jos niitä ei ollut muutettu Raspi-config:in yhteydessä niin tällöin oletuksena käyttäjätunnus oli pi ja salasana raspberry. Samaa käyttäjätunnusta ja salasanaa tarvittiin myös myöhemmin tässä projektissa, kun Raspia liitettiin CoDeSysiin. Kun Raspia käynnistettiin uudelleen, niin se otti käyttöönsä muistikortilta koko sen muistin tilan. Mikäli jotain muutoksia oli jäänyt tekemättä Raspi-config -ikkunassa ennen uudelleen käynnistystä, niin pääsi niitä muuttamaan kirjoittamalla komentoriville sudo raspi-config. Sudo komento tarkoittaa sitä, että muutoksia tehdään pääkäyttäjänä.

Kun tässä kehitysprojektissa tehtiin CoDeSysillä ohjelma Raspiin, tuli ohjelmaa tehtäessä määrittää sarjaportti, jota Raspi käyttää kommunikoimiseen uunin piirikortin kanssa. Tässä kehitysprojektissa kommunikoimiseen käytettiin Raspin COM1-sarjaliikenneporttia. Raspi ei kuitenkaan automaattisesti tiedä, mitä sarjaliikenneporttia tiedonsiirtoon halutaan käyttää ja mikä COM1-portti on. Tämän takia se tuli erikseen määritellä Raspiin.

Tämä tarkoitti sitä, että kun kommunikointiin käytettiin Raspin GPIO-pinneihin liitettyä itse tehtyä piirilevyä ja sen RS-232 -liitintä niin se tuli määritellä Raspin COM1-sarjaportiksi. COM-portin numero voi olla mikä vain, kunhan CoDeSysillä tehdyssä ohjelmassa ja Raspissa numero on sama.

Sarjaportin määrittely Raspiin tapahtui siten, että muokattiin Raspissa olevaa CODESYSControl.cfg tiedostoa, joka sijaitsee Raspin etc -tiedostoissa. Kyseinen tiedosto avattiin komentorivillä komennolla sudo nano /etc/CODESYSControl.cfg, jolloin päästiin pääkäyttäjänä muokkaamaan tekstitiedostoa. Tähän tekstitiedostoon lisättiin kuvassa 25 nähtävissä olevat tekstirivit. Sarjaportin määrittelylle käytettiin internetin keskustelupalstoilta löytyviä ohjeita, jotka olivat tarkoitettu Raspi 2. Tämä määrittelytapa paljastui kehitysprojektin edetessä virheelliseksi ja se jouduttiin muuttamaan. Uusi määrittelytapa on kuvattu tarkemmin myöhemmin tässä raportissa.



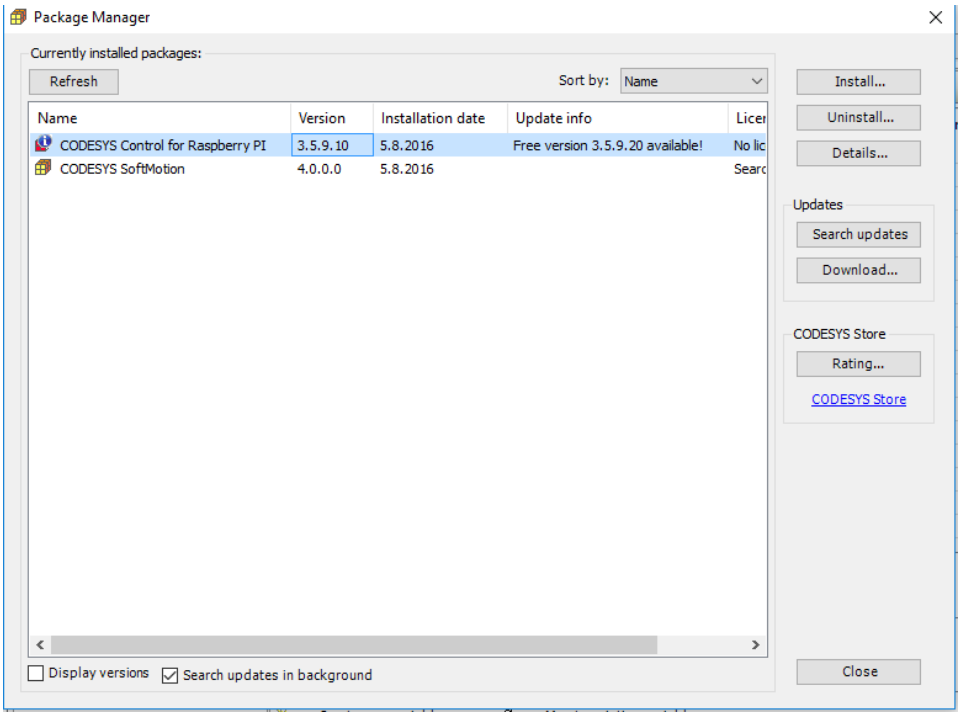
Kuva 25. COM-portin määrittely

Kun tekstitiedostoa oli muokattu kuvan 25 mukaisesti voitiin tiedosto sulkea painamalla Ctrl+X. Tämän jälkeen Raspi kysyi, että halutaanko tiedosto tallentaa, johon vastattiin kyllä painamalla K näppäintä ja se kuitattiin Enterillä. Tämän jälkeen Raspi voitiin käynnistää uudestaan ja nyt kommunikoinnin oli tarkoitus toimia.

5.3.3 CoDeSys

CoDeSys ladattiin ilmaiseksi CoDeSysin omilta nettisivuilta osoitteesta codesys.com/download. Tässä projektissa käytettiin CoDeSysin uusinta versiota, joka on CoDeSys V3.5. Lisäksi CoDeSys-ohjelmaan tuli ladata CoDeSys Control for Raspberry Pi -lisäosa, jotta se saatiin kommunikoimaan Raspin kanssa.

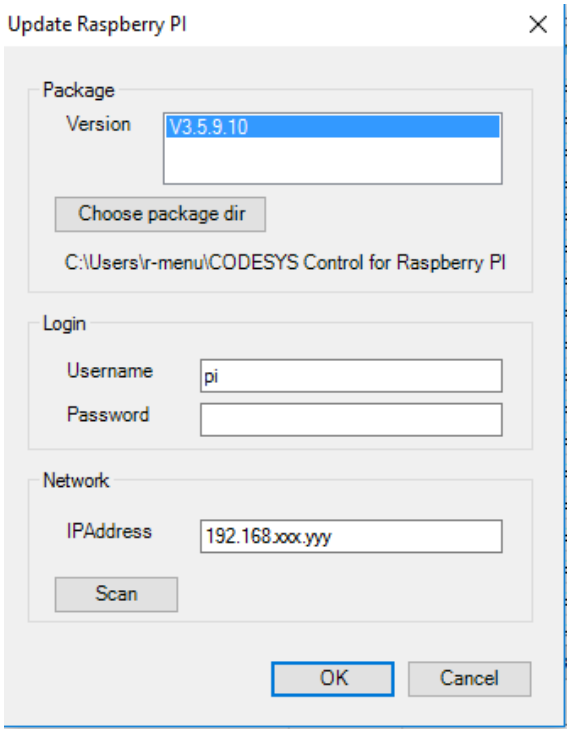
Kun tarvittavat ohjelmat oli ladattu kannettavalle tietokoneelle ja CoDeSys V3.5 oli asennettu, niin seuraavaksi CoDeSys avattiin. Tämän jälkeen avautui CoDeSysin päänäkymä ja siinä luotiin uusi projekti, joka tässä tapauksessa nimettiin Opinnaytetyo1.1 nimellä. Kun uusi projekti oli luotu, voitiin CoDeSysin lisäosa CoDeSys Control for Raspberry Pi ottaa käyttöön. Tämä lisäosa täytyi kuitenkin ensin ladata netistä CoDeSysin omilta nettisivuilta. CoDeSys Control for Raspberry Pi -lisäosan käyttöönotto tapahtui CoDeSys-ohjelman valikosta tools ja sieltä package manager. Seuraavaksi avautui kuvan 26 näköinen ikkuna, josta valittiin oikea lisäosa ja painettiin install-nappia.



Kuva 26. CoDeSys Control for Raspberry Pi käyttöönotto

Kun lisäosa oli asennettu, niin tämän jälkeen otettiin CoDeSys-ohjelmalla yhteyttä Raspiin. Tämä tapahtui CoDeSysin valikosta Tools ja kohdasta Update Raspberry Pi. Kuvassa 27 on nähtävissä valikko, jolla yhteys saatiin muodostettua Raspiin. Yhteyden muodostamisessa ilmeni ongelmia, koska siihen tuli hakea CoDeSys Control for Raspberry Pi asennuskansiosta deppäätteinen tiedosto, mutta sitä ei ensin löytynyt. Vian etsinnän jälkeen ongelma selvisi siten, että CoDeSys täytyi käynnistää tietokoneen järjestelmänvalvojan tunnuksilla. Tämä johtui siitä, että kyseessä oli yritykseltä saatu kannettava tietokone henkilökohtaisilla tunnuksilla ja käyttäjällä ei ollut oikeutta kaikkiin ladattujen ohjelmien tiedostoihin. Tämä johtui siitä, että projektissa käytettävät ohjelmat oli asennettu järjestelmänvalvojan tunnuksilla. Tämän takia kaikkia tiedostoja ei ollut nähtävissä aluksi, kun ohjelma oli käynnistetty käyttäjän tunnuksilla. Kun yhteyttä muodostettiin Raspiin, tuli kirjoittaa sen oletus käyttäjätunnus pi ja salasana raspberry, jotta yhteys muodostuisi. Network kohtaan ilmoitettiin Raspin IP-osoite ja tämä saatiin selville viemällä hiiri Raspin työpöydän oikeassa yläkulmassa

löytyvän WLAN-kuvakkeen päälle. Tässä kohtaa oli myös mahdollista painaa Scan-nappia, jolloin CoDeSys etsi automaattisesti Raspin langattomasta verkosta. Scan-nappia käytettäessä Raspin tuli olla samassa langattomassa verkossa kuin kannettavan tietokoneen.

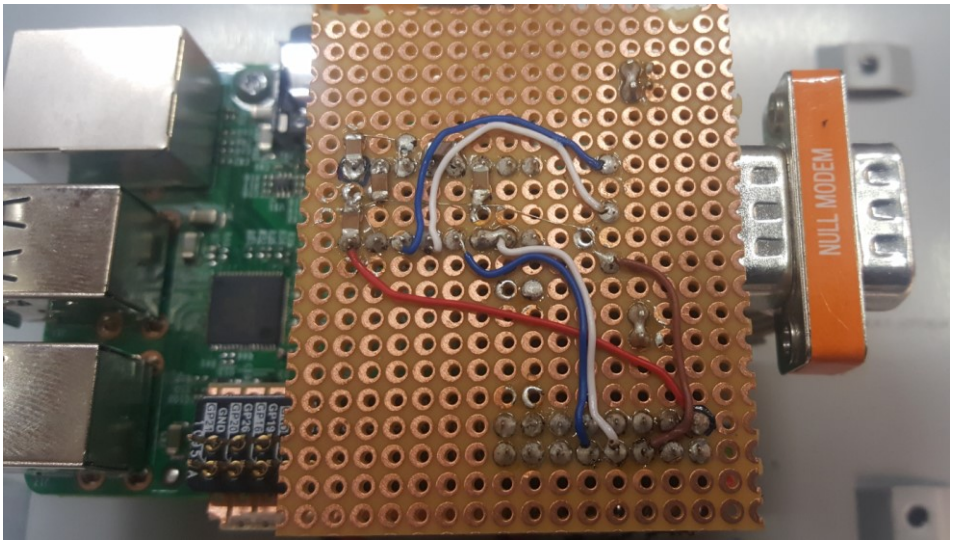


Kuva 27. Update Raspberry Pi -valikko

5.4 Kommunikaation varmistaminen

Kommunikaation varmistaminen tuli tässä kehitysprojektissa tarpeelliseksi, koska kommunikaatio ei aluksi toiminut. Kehitysprojektin alussa kommunikaatiota yritettiin saada toimimaan suoraan Raspin ja uunin piirikortin välillä, mutta se ei toiminut lukuisista yrityksistä huolimatta. Kun kommunikaatio ei toiminut niin ensimmäinen oletus viasta oli, että itse tehdyssä piirilevyssä olisi joku vika.

Niin kuin aikaisemmin tässä opinnäytetyössä on kerrottu, että itse tehdyssä piirilevyssä tapahtuu muunnos GPIO-pinneistä D9-liittimelle. Sieltä data siirretään eteenpäin toiselle piirilevylle, joka muuttaa datan D9-liittimeltä RJ45-liittimelle. Toiselta piirilevyltä data lähtee CAT6-kaapelilla uunin piirikortille. Itse tehdyssä piirikortissa tapahtuu siis muunnos Raspin GPIO-pinneistä RS-232 -muotoon. Siinä neljästä GPIO-pinnistä siirretään tietoa piirikortin D9-liittimeen. Kuvassa 28 on nähtävissä omatehdyn piirikortin kytkennät. Yksi johdoista on ground, joka on kuvassa ruskea johdin. 3,3V siirretään Raspin pinneistä piirikortille ja se on kuvassa punainen johto. Kaksi muuta johtoa ovat RXD, joka vastaanottaa tietoa Raspiin päin ja ne ovat kuvan valkoiset johdot. Viimeinen piirilevyn sininen johto on TXD ja se lähettää tietoa Raspista poispäin.



Kuva 28. Omatehdyn piirikortin kytkennät

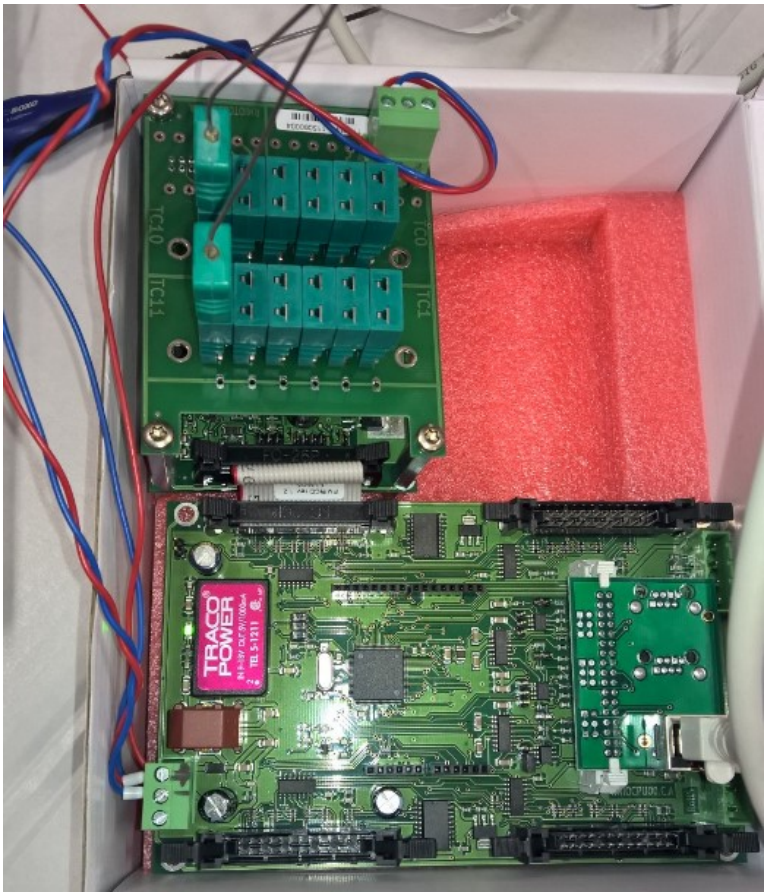
Epäily oli, että juotos on tehty siten väärin, että tietojen lähetys ja vastaanotto ns. törmäävät eli ne eivät mene ristiin jolloin ne ovat kytketty väärin. Tämä virhe pystytään pois sulkemaan siten, että kahden ylimääräisen kommunikaation toimintaan tarvittavan piirilevyn D9-liittimien väliin lisättiin nollamodeemi. Se kääntää vastaanottavan ja lähettävän signaalin ristiin. Itse tehdyssä piirilevyssä tietojen lähetys ja vastaanotto oli kytketty oikein ristiin, mutta uunin piirikortilla oleva muunnin oli myös ristissä. Tämä tarkoitti sitä, että kaksi kertaa ristiin kytketyissä piirilevyissä data menee ”suoraan”, joten nollamodeemia tuli käyttää tässä kehitysprojektissa. Se ei kuitenkaan ratkaissut ongelmaa, joten vian etsintää jatkettiin.

Kommunikaatio-ongelmia jouduttiin selvittämään yksitellen, jotta tiedettiin mistä ongelmat johtuivat. Nämä kommunikaatiotestit tehtiin ennen kuin päästiin tekemään kehitysprojektin laitteiston toimintaa testaavaa ohjelmaa ja käyttöliittymää. Kommunikaation varmistaminen on myös tärkeää, jotta tiedetään mahdollisten virheiden ilmaantuessa valmiissa kehitysprojektissa niiden johtuvan ohjelmasta eikä fyysisistä kytkennöistä.

5.4.1 Uunin piirikortin kommunikaatio

Modbus MAT oli erinomainen ohjelma tutkimaan kommunikaation toimivuutta uunin piirikortin osalta. Tätä ohjelmaa käytettiin kannettavalta tietokoneelta ja sillä pystyttiin pakottamaan laitteen lähtöjä päälle. Modbus MATilla pystyttiin myös lukemaan laitteen lähtöjen tilaa. Se toimi Modbus masterina ja laite, jonka lähtöjä sillä ohjattiin tai luettiin, on slave-laite. Uunin piirikortti toimi testin sekä koko kehitysprojektin slave-laitteena. Uunin piirikortille ladattiin uunin käyttämä ohjelmisto, jotta Modbus MATilla pystytään ohjaamaan tai lukemaan laitteen lähtöjä. Uunin ohjelmisto ladattiin uunin piirikortille, joko Atmel Studio tai Arduino-ohjelmalla.

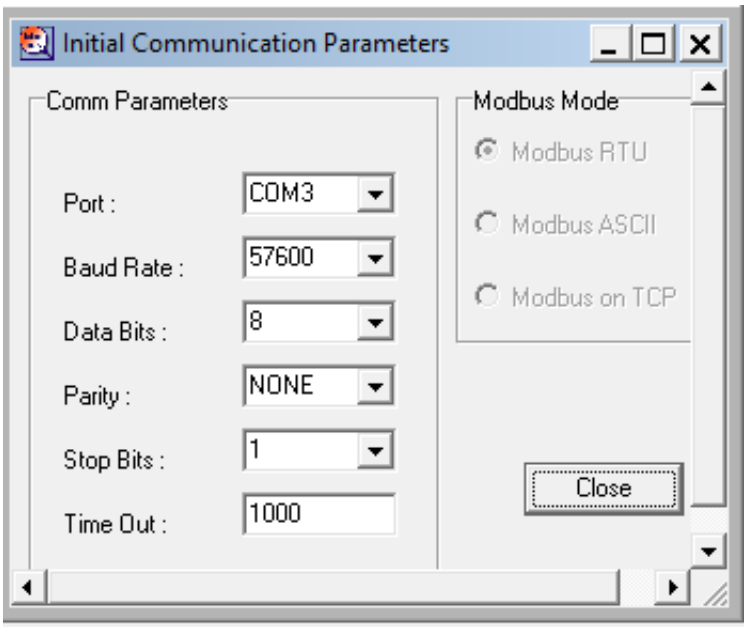
Kannettavaan tietokoneeseen kytkettiin USB-D9 -muunnin, johon kiinnitettiin muunnin joka muuttaa datan D9-liittimeltä RJ45-liittimelle. Siitä tietokone pystyttiin kytkemään uunin piirikorttiin CAT6-kaapelilla. Uunin piirikorttiin kytkettiin lattakaapelilla uunissakin käytetty termoparikortti, joka on nähtävissä kuvassa 29 ylempänä kuvan kahdesta piirikortista. Termoparikortti kerää uunin kaikkien eri toimilaitteiden ja tilojen lämpötilatietoja. Uuni pystyy tätä piirikorttia käyttämällä säätelemään toimilaitteille annettavia tehoja. Tämä piirikortti on myös kriittinen uunin toiminnan kannalta, koska se rajoittaa myös joitakin uunin toimilaitteita, jotta ne eivät lämpenisi liikaa. Uunista tulee termoparikortille yhteensä 20 eri toimilaitteen tai tilan lämpötilatietoa.



Kuva 29. Uunin piirikortti ja termoparikortti

Ensin Modbus MAT -ohjelmaan määritettiin kommunikointitestissä käytettävät parametrit, jotka ovat nähtävissä kuvassa 30. Parametreissa määritettiin, että mitä tietokoneen COM-porttia käytetään tiedonsiirrossa. Tässä tapauksessa tietokoneen asetuksista käytettäväksi otettiin sen COM3-portti. Portin valinnan lisäksi parametreissa valittiin datan siirtonopeus, joka oli tässä kommunikaation tarkastuksessa 57600 baudia. Siirtonopeus oli edellä mainittu sen takia, koska uunin piirikortilla olevan softan siirtonopeus oli myös 57600 baudia. Jos Modbus MATissa määritelty nopeus olisi eri kuin uunin piirikortilla olevan uunisoftan niin tällöin kommunikointi ei toimisi. Sama asia täytyi muistaa ohjelmaa luotaessa CoDeSysillä Raspille, koska master-laitteen ja slave-laitteen siirtonopeus tuli olla sama.

Modbus MAT kommunikointitestissä kaikki muut parametrit pidettiin vakiona, koska ne tuli myös olla samat kuin uunin piirikortilla olevassa uunisoftassa.

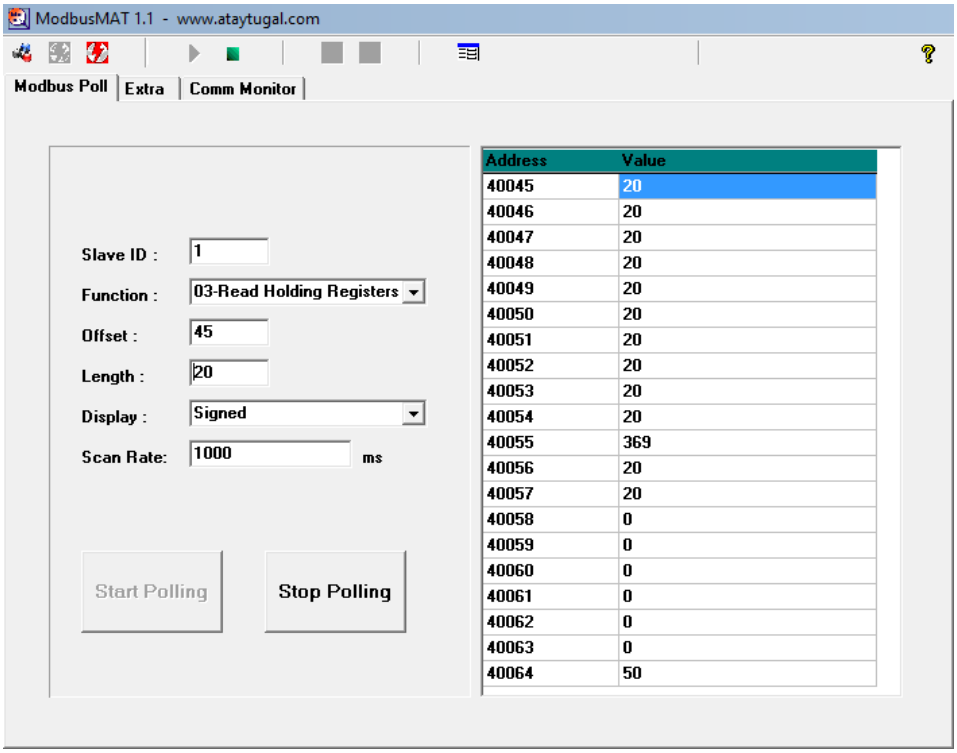


Kuva 30. Modbus MATin parametrien määrittely

Kun Modbus MAT -ohjelmassa luettiin slave-laitteen lähtöjen tilaa, tuli tietää tarkasti, mitä tietoja haluttiin lukea. Tämä oli tärkeää, jotta ohjelman asetukset pystytään määrittelemään oikein. Kuten kuvasta 31 nähdään, niin tässä kommunikaatiotestissä asetuksien Function-kohdassa on määriteltynä 03-Read Holding Registers. Se tarkoittaa, että Modbus lukee slave-laitteen rekisterejä. Tässä kohdassa on myös mahdollista määritellä Modbus MATin asetukset esimerkiksi siten, että se lukee ainoastaan bittejä. Tässä kohtaa voidaan myös pakottaa slave-laitteen lähtöjä päälle valitsemalla Function-kohtaan Write Holding Registers. Offset valinnalla päätettiin kohta, josta slave-laitteen I/O-tietoja ruvetaan lukemaan. Tässä testissä se oli 45, koska sille alueelle sijoittuu termoparikortin I/O-tiedot. Tämä alue saatiin tietoon, kun monitoimiuunia kehittäessä laitteen suunnittelijat ovat tehneet ja täydentäneet I/O-listan kaikista uunisoftaan määritellyistä uunin toimilaitteiden I/O-tiedoista. Asetuksien length kohdassa valittiin, kuinka monta sanaa halutaan samaan aikaan näkyviin Modbus MATin näytölle. Display-valinnalla oli mahdollista muuttaa saatavan tiedon muotoa. Kommunikoivan laitteen tietoja on mahdollista saada esimerkiksi heksadesimaali- tai binääri-muodossa. Tässä kommunikaatiotestissä käytettiin signed-muotoa, jotta tieto saadaan käytännöllisesti sopivana.

Parametrien ja asetusten määrittämisen jälkeen painettiin Start Polling -nappia, joka aloitti kommunikoinnin Modbus MATin slave-laitteen kanssa. Tässä testissä se oli uunin piirikortti. Kun kommunikaatio aloitettiin uunin piirikortin kanssa, niin näytölle ilmestyi eri lähtöjen tiloja. Näitä tietoja käyttäen termoparikortilta saatavia tietoja seurattiin kannettavan tietokoneen kautta. Kommunikaation toimivuus saatiin varmistettua siten, että

termoparikorttiin liitettiin lämpöanturi, jota lämmitettiin kaasupolttimella. Modbus MAT -ohjelmaa seuraamalla nähtiin, miten osoitteessa 40055 arvo muuttui 369:ksi (Kuva 31). Modbus MATissa nähtävä osoite oli 40055, koska anturi oli liitettynä termoparikortin liittimeen, jonka osoitteeksi oli määritetty kyseinen osoite uunisoftassa. Tämä arvo on suoraan Celsiusasteina ja se tarkoittaa, että kommunikointi toimi moitteettomasti uunin piirikortilta ulospäin. Modbus MATissa näkyvissä olevat muut arvot olivat 20, koska se on termoparikortin lämpötila ja se on määritettynä toimivaksi, jos muuta tietoa anturilta ei tule.



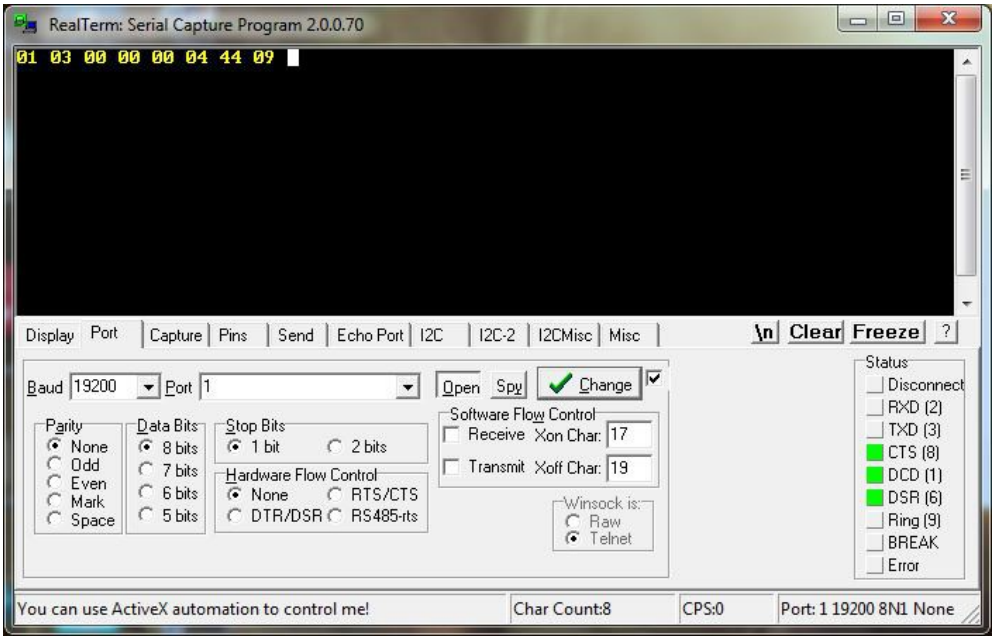
Kuva 31. Modbus MAT -näkymä lämpöanturia lämmitettäessä.

5.4.2 Raspberry Pi:n kommunikaatio

Raspin kommunikaation toiminta varmistettiin siten, että laitteen toimiessa se lähettää Modbus request -signaalin Modbus-väylän kautta. Modbus request on viestikehys, joka lähetetään aina, kun master-laite ottaa yhteyttä Modbus-väylän kautta. Tämä viestikehys sisältää slave-laitteen osoitteen, Function, aloitus-osoitteen, bittien määrän ja CRC-virheentarkistuksen, kun luetaan Holding Registers -arvoja. Tässä kehitysprojektissa Raspi kommunikoi uunin piirikortin kanssa, jolle sen tuli lähettää näitä Modbus request -signaaleja.

Tämän toiminta saatiin varmistettua siten, että Raspi kytkettiin Modbus-väylän kautta kannettavaan tietokoneeseen. Kannettavaan tietokoneeseen ladatun ohjelman avulla luettiin Raspin lähettämiä Modbus request -signaaleja. Jos kannettavalla tietokoneella saatiin vastaanotettua Raspin

lähettämä oikeanlainen Modbus request -signaali, niin tällöin Raspin kommunikointi toimisi moitteettomasti. Tämä tarkastettiin siten, että ensin tietokoneeseen ladattiin RealTerm-ilmaisohjelma, jonka avulla pystyttiin kommunikoimaan sarjaliikenneportin avulla. Samalla ohjelmalla pystyttiin myös lukemaan dataa, joka tuli sarjaliikenneportin kautta kannettavalle tietokoneelle. Kun ohjelma oli ladattu kannettavaan tietokoneeseen, niin tämän jälkeen se yhdistettiin toiseen kannettavaan tietokoneeseen sarjaliikenneväylän kautta. Toinen tietokone lähetti Modbus MAT -ohjelmalla pyynnön lukea neljä ensimmäistä Holding Register -arvoa. Tällöin tietokoneella, jossa RealTerm ohjelma oli avoinna, saatiin esille Modbus request -viestikehys, joka on nähtävissä kuvassa 32.



Kuva 32. Modbus request -viestikehys

Tämän jälkeen Raspiin ladattiin CoDeSysillä tehty pieni ohjelma, johon oli määritettynä muutama muuttuja. Muuttujilla ja ohjelmalla ei tässä kohtaa ollut merkitystä, kunhan ohjelma lukisi neljä ensimmäistä uunisoftan Holding Register -arvoa, kun ohjelma käynnistettiin. Tästä CoDeSysillä tehdystä ohjelmasta tehtiin boot application. Boot application tarkoittaa sitä, että kun PLC eli tässä tapauksessa Raspi alkoi suorittaa sinne ladattua ohjelmaa heti, kun laite tai ohjelma käynnistettiin.

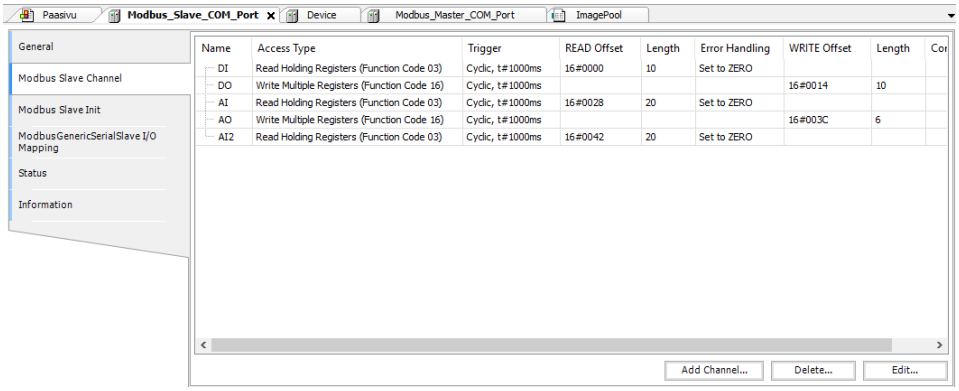
Kun ohjelma oli ladattu Raspiin, kytkettiin tämän jälkeen kannettava tietokone Raspiin kiinni Modbus-väylän kautta. Tietokoneeseen avattiin RealTerm -ohjelma ja sitten Raspi käynnistettiin uudestaan. Kun Raspi käynnistettiin uudestaan, pystyttiin näkemään RealTerm-ohjelmasta, kun Raspi lähetti samanlaisen Modbus request -viestikehyksen. Aikaisemmin oltiin samanlainen viestikehys saatu esille tietokoneiden välisellä kommunikoinnilla. Modbus request -viestikehyksiä ilmestyi 1000ms välein, koska se oli asetettu sille CoDeSysillä tehdyssä ohjelmassa. Tämän testin avulla varmistui, että Raspin kommunikaatio toimi.

5.4.3 Uunin piirikortin ja Raspberry Pi:n kommunikointi

Kun jokaisen kehitysprojektissa käytettävän laitteen kommunikaation toiminta oli varmistettu yksitellen, perehdyttiin seuraavaksi uunin piirikortin ja kehitysprojektin PLC eli Raspin väliseen kommunikaation. Tämä tehtiin siten, että jo aikaisemmin kappaleessa 5.3.1 tarkemmin esitelty lämpötilatesti suoritettiin nyt Raspilla. Tässä lämpötilatestissä uunin piirikorttiin liitettiin termoparikortti (Kuva 29.) samalla tavalla, kuin aikaisemmassakin testissä. Aikaisempi lämpötilatesti tehtiin Modbus MATilla, jolla pystyttiin lukemaan termoparikortille kytkettyjen antureiden lämpötilatietoja kannettavan tietokoneen kautta. Tässä testissä antureiden lämpötiloja luettiin Raspilla, jotta saadaan varmistettua uunin piirikortin ja Raspin välisen kommunikaation toimivuus.

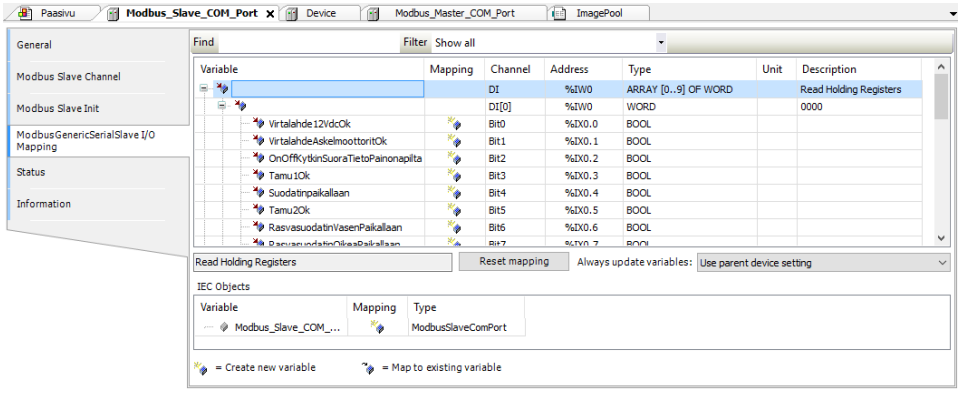
Tässä kommunikaatiotestissä uunin piirikortti liitettiin Modbus RTU -sarjaliikenneväylällä Raspiin. Ennen kuin kommunikaatiotestiä pystyttiin tekemään tuli CoDeSysiin määritellä kehitysprojektissakin tarvittavat uunin I/O-tiedot. Ensin CoDeSysiin rakennettiin oikeanlainen ohjelmarunko (Liite 1.) siten, että Device kohdan päällä painettiin hiiren oikealla ja valittiin Add Object. Sitä kautta saatiin perustettua ohjelmarunkoon kohta Modbus_COM, jonka päällä painettiin hiiren oikealla painikkeella ja valittiin Insert Device. Tällä tavalla perustettiin Modbus_COM:in alapuolelle Modbus_Master_COM_Port kohta, joka tarkoittaa tässä kehitysprojektissa Raspin laitteistoa. Seuraavaksi painettiin hiiren oikealla napilla Modbus_Master_COM_Port kohdan päällä ja valittiin Insert Device. Sieltä saatiin perustettua tämän kohdan alle Modbus_Slave_COM_Port, joka tarkoittaa kehitysprojektissa uunin piirikorttia.

Jotta kommunikaatiotesti saatiin toimimaan, tuli uunin piirikortin I/O-tiedot määritellä CoDeSysiin. Tämä tapahtui siten, että tupla klikattiin äsken perustettua kohtaa Modbus_Slave_COM_Port. Sieltä valittiin kohta Modbus Slave Channel, johon määriteltiin tarvittavien I/O-tietojen kanavat. Uunin piirikortin I/O-tiedot sisältävät neljää erilaista tyyppiä joita ovat DI, DO, AI ja AO. Kanaviin tuli myös määritellä niille oikeanlaiset asetukset. Asetuksien määrittelyt ovat nähtävissä kuvassa 33. Kanavien asetuksien määrittelyssä READ Offset ja WRITE Offset tarkoittaa, että mistä osoitteesta käytettyjen I/O-tietojen määrittelyt alkavat. Lenght tarkoittaa pituutta, jonka yksi kanava varaa sille määritetystä osoitteesta eteenpäin.



Kuva 33. Modbus Slave Channel määrittelyt

Kun kanavat ja niiden asetukset oli määritelty, tuli seuraavaksi I/O-tiedot lisätä CoDeSysiin yksitellen. Uunin piirikortin I/O-tiedot lisättiin painamalla kohdasta ModbusGenericSerialSlave I/O Mapping. Tästä avautui näkymä, joka oli vastaava kuin äsken kanavia ja niiden asetuksia määriteltäessä. Tähän näkymään pystytään kuitenkin määrittämään I/O-tiedot, josta osa on nähtävissä kuvassa 34. Kun kaikki uunin piirikortin I/O-tiedot oli määritelty niin ne tallennettiin, koska niitä käytettiin kommunikaatiotestin lisäksi myös kehitysprojektissa. Kommunikaatiotestissä ja kehitysprojektissa ei ollut tarvetta uunin jokaiselle I/O-tiedolle, mutta oli järkevää määritellä jo tässä vaiheessa kaikki I/O-tiedot, jotta laitteiston laajentaminen olisi työn tilaajalle mahdollisimman helppoa.



Kuva 34. I/O-tietojen määrittely

Tässä kommunikaatiotestissä Raspin tulisi pystyä lukemaan kahden termoparikorttiin liitetyn anturin lämpötilatietoja. Testissä Raspi luki lämpötilatietoja uunin piirikortilta ja sen tarkoituksena oli näyttää ne Raspin kosketusnäytöllä. Raspin kosketusnäytölle tehtiin CoDeSysillä yksinkertainen grafiikka, jossa oli pelkästään kaksi lämpötilapalkkia. Jos näytön lämpötilapalkit reagoisivat lämpötilan muutoksiin, niin tällöin kommunikaatio toimi myös uunin piirikortin ja Raspin välillä. Lämpötilojen esittämiseen tehty grafiikka ja ohjelma olivat yksinkertaisia eikä niitä käytetty muuten tässä kehitysprojektissa.

Kun uunin piirikortin I/O-tiedot ja yksinkertainen grafiikka ladattiin Raspiin niin huomattiin, että Raspi ei kommunikoinut uunin piirikortin kanssa. Koska kaikki muut kehitysprojektissa käytetyt laitteet oli jo testattu erikseen, niin pystyttiin päättelemään, että vika on CoDeSysillä tehdyssä ohjelmassa tai sen asetuksissa. Monitoimiuunin pääsuunnittelijan mukaan uunia kehitettäessä oli sen ohjelmisto ollut tarkoitus tehdä CoDeSysillä, mutta se oli vaihdettu erilaisista syistä johtuen. Silloin kuitenkin oli tullut vastaan ongelmia, että CoDeSysillä tehdyssä ohjelmassa datan siirtonopeus oli ollut liian suuri niin isolle ohjelmalle. Käytännössä tämä tarkoittaa sitä, että CoDeSys ei ollut pystynyt siirtämään haluttua dataa niin nopeasti kuin vaadittiin. Tässä kehitysprojektissa esiintyvää ongelmaa lähdettiin ensin korjaamaan siten, että datan siirtonopeutta laskettiin 57600 baudista 19200 baudiin. Kun CoDeSysiin muutettiin datan siirtonopeutta, jouduttiin sama muutos tekemään myös uuniohjelmistoon. Tämä johtui siitä, että Raspin ja uunin piirikortin ohjelmien datan siirtonopeus täytyi olla sama. Tämä ei kuitenkaan ratkaissut ongelmaa, joten vian etsintää jatkettiin.

Kun tämän kehitysprojektin laitteistoa koottiin ja käyttöönotettiin, käytettiin tarvittaessa internetistä saatavilla olevia ohjeita. Ongelmana kuitenkin oli, että tässä kehitysprojektissa käytettävä Raspi 3 on niin uusi, että sitä ei ennen ole käytetty tämmöisessä ympäristössä. Oletuksena oli, että Raspi 3 mallin käyttäminen ei aiheuttaisi ongelmia kehitysprojektin laitteiston toimivuudessa vaan se toimisi samalla tavalla kuin Raspi 2. Pitkän vian etsinnän jälkeen löytyi kuitenkin eroavaisuuksia edellisiin Raspi malleihin verrattuna.

Kappaleessa 5.3.2 on kerrottu, miten Raspin GPIO-pinneihin liitetty piirikortti määriteltiin toimivan Raspin COM1-porttina. Se tehtiin muokkaamalla CODESYSControl.cfg tekstitiedostoa. Raspin tekstitiedostoon määriteltiin, että sen GPIO-sarjaportti, joka on raspin tiedostoissa nimellä ttyAMA0, toimii COM1 sarjaportin nimellä. Vian etsinnän seurauksena selvisi, että tällä tavalla tehty määrittely toimii vain Raspi 2:ssa, mutta ei uusimassa, tässä kehitysprojektissa käytetyssä Raspi 3:ssa. Raspi 3:seen on tullut uutena ominaisuutena sisäänrakennettu Bluetooth. Raspi 2 sarjaportin määrittely ohjeet opastavat, että ttyAMA0 tarkoittaa GPIO-sarjaporttia, mutta uusimmassa Raspi 3 se tarkoittaakin bluetooth sarjaporttia. Näin ollen, Raspi oli aikaisemmin määritelty käyttämään kommunikointiin Bluetoothia, vaikka tarkoitus oli käyttää GPIO-sarjaporttia. Raspi 3 /etc/CODESYSControl.cfg tiedostoon tuli määrittää Raspi 3:ssa ttyAMA0 sijaan ttyS. Sen perään tulee numero sen mukaan, mikä laite siihen on kytketty. Raspi määrittää numeron automaattisesti sen mukaan mikä laite siihen on kytkettynä. Esimerkiksi jos ttyS eli GPIO-sarjaporttiin kytketään laite se saa nimen ttyS0 ja seuraava laite saa taas nimen ttyS1 jne. Tämä numero on instanssinumero ja sitä ei näy, jos Raspiin ei ole kytkettynä yhtään fyysistä laitetta.

Näiden muutosten jälkeen saatiin kommunikaatio toimimaan osittain, mutta lämpötilat pätkivät vielä grafiikassa. Tästä pystyttiin päättelemään,

että muutokset olivat auttaneet kommunikointi-ongelmaan, mutta jossain CoDeSysin asetuksissa oli vielä ongelmia. Vian etsinnän jälkeen huomattiin, että aikaisemmin CoDeSysiin määritettyjen kanavien asetuksissa oli virhe. Kanavia määriteltäessä tulee aloitus osoitteet olla heksadesimaalimuodossa eikä binäärimuodossa, joita ensin käytettiin. Kun osoitteiden alut korjattiin heksadesimaaliksi ja ladattiin Raspiin, saatiin grafiikka reagoimaan moitteettomasti uunin piirikortin kanssa. Kun kommunikointi saatiin toimimaan, voitiin aiemmin muutettu datan siirtonopeus muuttaa takaisin 57600 baudiin, koska vika ei ollut siinä. Tämän jälkeen aloitettiin CoDeSysillä tekemään pieni muotoista testiohjelmaa ja käyttöliittymää Raspiin.

5.5 **Ohjelmointi**

Kun Raspiin suunniteltiin käyttöliittymää ja tehtiin ohjelmaa, oli tarkoituksena luoda yksinkertainen testiohjelma, jolla pystytään todentamaan kehitysprojektin laitteiston toiminta. Tavoitteena olikin tehdä ohjelmasta ja käyttöliittymästä yksinkertainen ja helposti ymmärrettävä, jotta työn tilaajan olisi sitä helppo jatkossa laajentaa.

5.5.1 Käyttöliittymän suunnittelu

Ennen kuin käyttöliittymää tehtiin, suunniteltiin sen kokonaisuus huolellisesti, jotta sitä olisi jatkossa helppo laajentaa. Laajennettavuus oli tärkein asia käyttöliittymän tekemisessä, koska tämän kehitysprojektin tavoite oli koota laitteisto, johon automaattisia testejä voidaan jatkossa perustaa. Tähän käyttöliittymään, jolla varmistettiin kehitysprojektin laitteiston toiminta suunniteltiin pääsivu, josta pääsi kolmelle muulle välilehdelle. Yksi välilehdistä sisälsi uunin I/O-tietoja, josta voitiin tarkistaa toimilaitteiden tiloja ilman niiden käyttämistä. Lisäksi käyttöliittymässä oli välilehti, jossa oli nähtävillä uunin toimilaitteiden lämpötiloja. Viimeisellä välilehdellä pystyttiin käynnistämään testiohjelma, jonka testilaitteisto suoritti uunille.

Jotta käyttöliittymästä saatiin tehtyä helposti laajennettava ja käytettävyydeltään hyvä kokonaisuus, tutustuttiin aihetta käsitteleviin standardeihin. Standardi SFS-EN ISO 9241-11 määrittelee käytettävyyden seuraavalla tavalla:

> *”Mitta, miten hyvin määrätyt käyttäjät voivat käyttää tuotetta määrätyssä käyttötilanteessa saavuttaakseen määritetyt tavoitteet tuloksellisesti, tehokkaasti ja miellyttävästi”.* (SFS 9241/1998, 6.)

Lisäksi standardissa SFS-EN ISO 9241-11 kerrottu käytettävyyden perusteista ja hyödyistä seuraavasti:

> *”Käytettävyys on tuotteiden suunnittelussa tärkeä tekijä, koska siinä otetaan huomioon, miten tuloksekkaasti, tehokkaasti ja tyytyväisenä tuotteiden käyttäjät voivat työskennellä. Tuotteen käytettävyyden taso voi vaihdella merkittävästi, kun sitä käytetään eri käyttötilanteissa.”* (SFS 9241/1998, 8.)

Käyttöliittymän suunnittelussa tutustuttiin myös standardiin ISO 9241-210, jossa kerrotaan ihmiskeskeisen suunnittelun periaatteita. Nämä periaatteet tuli pitää mielessä käytettävyyden lisäksi, kun käyttöliittymää suunniteltiin.

> *”Tuotteet, järjestelmät ja palvelut olisi suunniteltava ottaen huomioon sekä niitä käyttävät henkilöt, että muut sidosryhmät, mukaan lukien ne, joihin järjestelmän käyttö voi (suoraan tai epäsuorasti) vaikuttaa. Sen vuoksi olisi tunnistettava kaikki oleelliset käyttäjä- ja sidosryhmät. Yksi merkittävimmistä syistä järjestelmien epäonnistumiseen on se, että niitä kehitetään perustuen väärään ja vajaaseen käyttäjätarpeiden ymmärtämiseen.”* (SFS 9241/2010, 20.)

Näiden standardien pohjalta toteutettu suunnittelu mahdollisti helposti laajennettavan ja käyttäjälle käytettävyydeltään hyvän käyttöliittymän. Käyttöliittymän suunnittelussa otettiin huomioon myös laitteiston käyttäjien näkemyksiä ja toiveita.

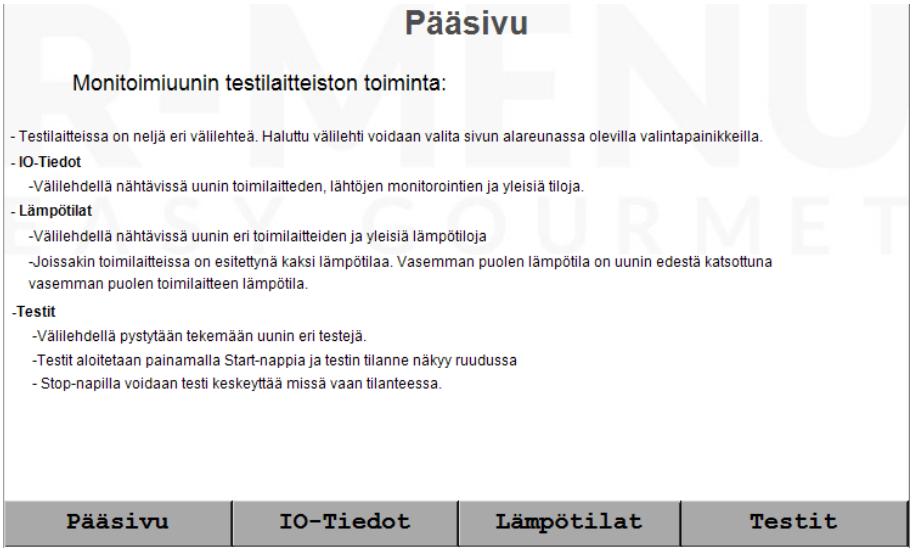
5.5.2 Ohjelman ja käyttöliittymän luominen

Testilaitteiston ohjelma sekä käyttöliittymä luotiin CoDeSys V3.5-ohjelmistolla. Kun ohjelmaa ja käyttöliittymää tehtiin, oli tärkeää tietää ja hahmottaa, mikä olisi kehitysprojektin toivottu lopputulos. Ennen kuin testiohjelma tehtiin laitteiston toiminnan varmistamiseksi, tehtiin käyttöliittymä Raspin kosketusnäytölle. Perusteellisen suunnittelun jälkeen aloitettiin käyttöliittymän rakentaminen CoDeSySiin.

Kun käyttöliittymään tulevat sivut oli suunniteltu paperille, luotiin ne seuraavaksi CoDeSySiin. Tämä tapahtui siten, että CoDeSys-ohjelmassa nähtävään Opinnäytetyö1.1 ohjelmarungon (Liite 1.) application-kohdan päällä painettiin oikealla hiiren painikkeella ja valittiin add object kautta visualization. Ensimmäinen visualization nimettiin paasivuksi, joka toimi käyttöliittymän pääsivuna. Kun käyttöliittymän välilehtiä luotiin, niin yhden visualization tiedoston luominen tarkoitti yhden välilehden luontia käyttöliittymään. Tässä kehitysprojektin ohjelmassa ja käyttöliittymässä käytettiin neljää eri välilehteä, jolloin tehtiin neljä visualizationia. Loppuja välilehtiä ei kuitenkaan kannattanut luoda yksitellen vaan ensin pääsivulle lisättiin käyttöliittymän taustakuva, otsikko ja painonapit, jolla liikuttiin välilehdillä.

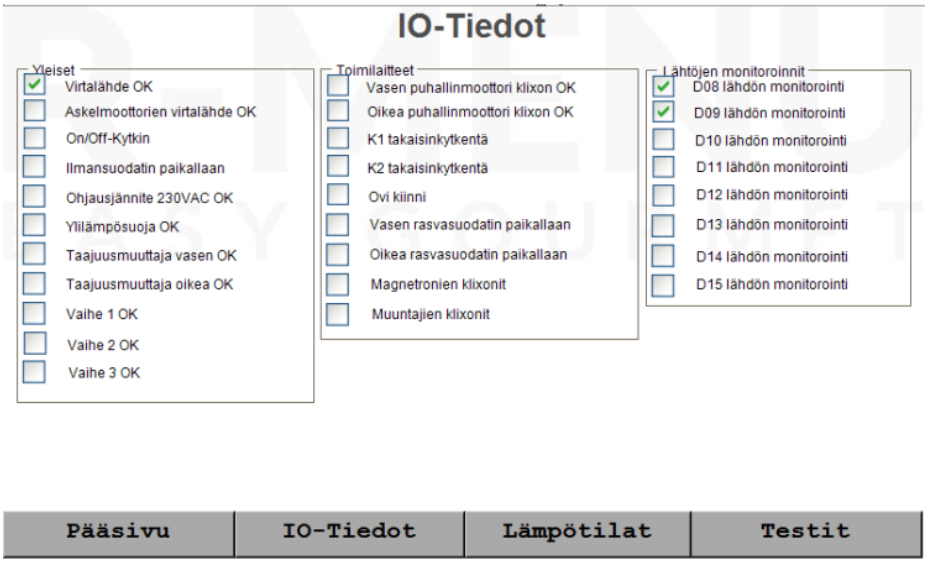
Lisäksi pääsivulle määritettiin oikea resoluutio, jotta käyttöliittymä sopii Raspin kosketusnäytölle. Tämän jälkeen pääsivu visualization kopioitiin neljäksi ja nimet muutettiin välilehtien otsikoiden mukaiseksi. Tällä tavalla jokaisella välilehdellä painonapit, otsikot ja asetukset olivat automaattisesti samoja ja niitä ei tarvinnut yksitellen määritellä.

Kuvassa 35 on nähtävissä käyttöliittymän pääsivu, joka aukesi, kun ohjelma käynnistettiin Raspissa. Pääsivu sisälsi ohjeet monitoimiuunin testilaitteiston toiminnasta. Pääsivun ohjeiden mukaan, kuka vain uunin toiminnan tunteva käyttäjä kykenee suoriutumaan sähköisistä testeistä tässä työssä kehitetyn laitteiston avulla. Käyttöliittymän jokaisella välilehdellä oli sivun alareunassa valinta painikkeet, millä pystyttiin siirtymään välilehdillä. Näille painonapeille oli määriteltynä niiden asetuksien variable kohtaan sen visualizationin nimi, mihin painonapilla haluttiin mennä.



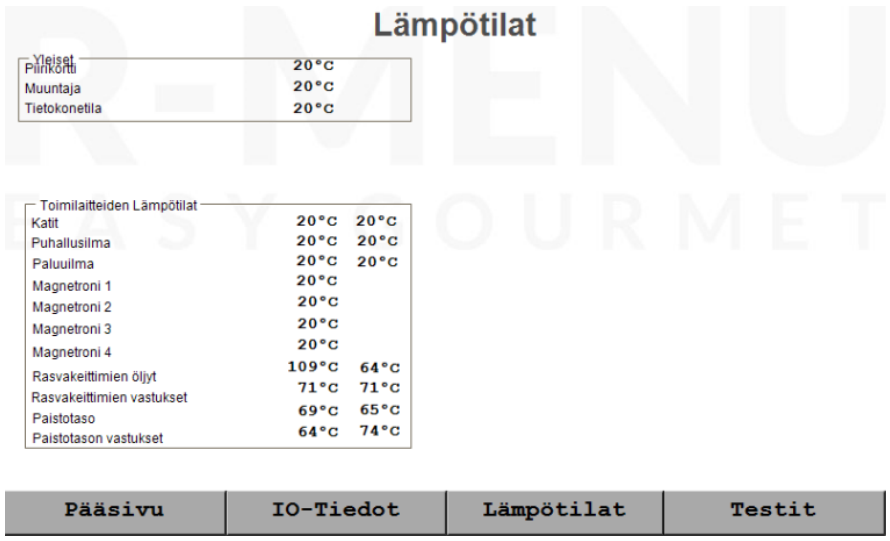
Kuva 35. Raspberry Pi:n käyttöliittymän pääsivu

Kuvassa 36 nähtävän käyttöliittymän toisen välilehden otsikko oli I/O-Tiedot. Tältä välilehdeltä pystyttiin näkemään monitoimiuunin eri I/O-tietoja, jotka olivat jaettuna kolmeen eri kategoriaan. Yleiset- ja toimilaitteet-osio sisälsi uunin toimilaitteisiin vaikuttavia I/O-tietoja. Lähtöjen monitoroinnit tarkoittivat sitä, että jos esimerkiksi DO8 oli pois päältä, niin silloin lähtöjen monitoroinnissa kyseinen kohta näytti ”1”. Kuvassa 36 kaksi ensimmäistä lähtöjen monitorointia olivat päällä, koska niihin oli kiinnitettynä kaksi led valoa, mutta niitä ei kuitenkaan ohjattu päälle. Kun kehitysprojektin kommunikaatiota varmistettiin, niin CoDeSysiin määritettiin uunin toimilaitteiden I/O-tiedot. Käyttöliittymän I/O-tiedot -välilehden merkit olivat linkitettynä aikaisemmin määriteltyihin I/O-tietoihin, josta niiden tila saatiin selville.



Kuva 36. I/O-Tiedot -välilehti

Kolmas välilehti sisälsi uunin yleisiä sekä eri toimilaitteiden lämpötilatietoja. Uunissa oli samoja toimilaitteita kummallakin puolella uunia. Niiden lämpötilat ovat esitetty siten, että käyttöliittymässä nähdyn vasemmanpuoleisen lämpötilan arvo vastasi monitoimiuunin vasemmalla puolella sijainnutta toimilaitteen lämpötilaa. Kuvassa 37 on nähtävissä lämpötilat-välilehti. Jotta lämpötilat saatiin näkyviin, täytyi ne linkittää välilehdelle oikeisiin kohtiin. Ensin välilehdelle luotiin rivit ja niille tekstit, jotka vastasivat lämpötiloja, joita haluttiin esittää. Seuraavaksi riveille linkitettiin lämpötilojen I/O-tiedot. Jotta lämpötilat näkyivät niille halutuissa kentissä ja lämpötilan perässä oli oikea yksikkö, tuli tekstikentän asetuksiin text kohtaan määritellä %i°C. Tällöin CoDeSys tiesi esitettävän luvun olevan integer-muotoinen ja sen yksikkö oli °C.



Kuva 37. Lämpötilat-välilehti

Käyttöliittymän neljännellä välilehdellä (Kuva 38.) tehtiin uunin sähköisiä testejä. Tälle käyttöliittymän välilehdelle luotiin testin aloittava start-nappi sekä stop-nappi, jolla testi voitiin keskeyttää. Lisäksi välilehdellä oli kenttä, josta voitiin seurata testin edistymistä. Jotta ohjelmaan määritetyt testin edistymistä kuvaavat tekstit tulisivat näkyviin, tuli ne määritellä siihen varattuun kenttään. Tekstikentän asetuksiin määritettiin %s, jolloin ”Testin edistyminen” -kenttä automaattisesti tiesi sen olevan string-muotoista ja tällöin näytti tekstin. Tälle välilehdelle linkitettiin CoDeSysillä ohjelmoitu testiohjelma, jolla voitiin varmistaa, että sähköiset testit toimisivat. Kun työn tilaaja lisää testauslaitteistoon automaattisia testejä niin ne linkitetään tälle välilehdelle. Silloin välilehdelle lisätään alasvetovalikko, josta valitaan testi, jota halutaan tehdä ja painetaan start-nappia jolloin laite suorittaa testin. Testit-välilehti jätettiin mahdollisimman yksinkertaiseksi, koska sinne tehdään jatkossa suurimpia lisäyksiä samalla kun automaattisia testejä lisätään. Välilehti kasattiin kuitenkin siten, että sen kautta oli helppo varmistaa laitteiston toiminta.

Kuva 38. Testit-välilehti

Kun välilehtien ulkoasut oli viimeistelty ja tarvittavat I/O-tiedot linkitetty välilehdille siirryttiin seuraavaksi ohjelmointiin. Tämän kehitysprojektin ohjelmointi tehtiin käyttäen kappaleessa 4.3.4 esiteltyä ST-kieltä. Tässä kohtaa selvisi, että lämpötilat ja I/O-tiedot eivät tulisi näkymään käyttöliittymän välilehdillä, ellei niitä ollut määriteltynä CoDeSysillä tehtyyn ohjelmaan. Tämän takia CoDeSysin PLC_PRG:iin tehtiin määrittely IF (lämpötila1<lämpötila2) AND (lämpötila3<lämpötila4) AND ja sitä jatkettiin niin pitkään, kunnes samassa määrittelyssä oli otettu huomioon kaikki välilehdillä esiintyvät I/O-tiedot. Määrittelyllä ei muuten ollut merkitystä ja se ei koskaan tulisi toteutumaan, mutta siinä oli lueteltuna uunin toimilaitteiden kaikki muuttujat, jolloin ne näkyivät käyttöliittymässä.

Kun testiohjelma tehtiin ohjelmarungon application kohdan alle, niin sille luotiin sequences niminen kansio. Tähän kansioon tehtiin testiohjelma, joka nimettiin Ledtest nimellä. Kun työn tilaaja tekee testauslaitteistoon uunin sähköisien testien automatisointeja, niin ne tehdään tähän kansioon. Yksi ohjelma sisältää aina yhden uunin sähköisen testin. Nyt opinnäytetyön ohjelmarunko näytti CoDeSysissä liitteen 1 mukaiselta.

Tämän jälkeen aloitettiin tekemään käyttöliittymän testit-välilehdelle ohjelmaa, jolla voidaan todeta kehitysprojektin kokonaisuuden toimiminen. Ohjelma tehtiin siten, että sillä voitiin ohjata uunin piirikortille lattakaapelilla kiinnitetyn I/O-kortin DO-liittimeen kytkettyä led valoa. Tämä testilaitteisto on esiteltynä liitteessä 2. Ensin ohjelmaan määritettiin siinä käytetyt muuttujat, jotka ovat nähtävissä kuvassa 40. Kuvassa 40 nähdään myös muuttuja StateTex1, joka oli kuitenkin turha tässä kehitysprojektissa, koska sitä ei käytetty testiohjelmassa.

```
1  PROGRAM Ledtest
2  VAR
3      State: INT;
4      StateText: STRING[200];
5      StateTex1: STRING[200];
6      Timer: TON;
7      IN: INT;
8      PT: TIME;
9  END_VAR
10 VAR_INPUT
11     Start: BOOL;
12     Stop: BOOL;
13 END_VAR
14
```

Kuva 39. Ledtest-ohjelman muuttujien määrittely

Ledtest-ohjelma tehtiin käyttäen case-rakennetta, jossa ohjelma ei siirry seuraavaan State-kohtaan, ellei edeltävän State-kohdan määrittelyt toteudu. Ohjelman alussa määritettiin testit-välilehdellä olevan stop-napin toiminnot. Jos Stop-nappia painettiin niin ohjelma palautti kaikki määrittelyt nollaan ja palasi ohjelman alkuun. State 0 sisälsi määritellyn, jossa start-nappia painettaessa siirryttiin kohtaan State 1. Siinä led asetettiin päälle ja siirryttiin ohjelmassa eteenpäin. Jos State-kohdassa olevat määrittelyt eivät toteutuneet niin ohjelma ei siirtynyt seuraavaan

State-kohtaan vaan jäi odottamaan ehtojen täyttymistä. Kohdassa State 2 laitteisto varmisti ledin toimintaa. Ledin toiminnan varmistaminen toteutettiin ohjelmassa käyttämällä lähdön monitorointia, joka menee nollaksi, jos siinä oleva laite oli päällä. Jokaisessa State-kohdassa oli viiden sekunnin ajastin, jotta käyttäjä ehti lukemaan ”Testin edistymisen” -kentässä olevan tekstin. Jokaisessa State-kohdassa, jossa ajastinta käytettiin tuli se nollata State-kohdan lopussa, jotta sitä pystyttiin käyttämään uudestaan ohjelman muissa kohdissa. Kuvassa 41 on nähtävissä ohjelman stop-napin määrittely sekä kolme ensimmäistä State-kohtaa.

```
1  IF Stop THEN
2      Led:=0;
3      Timer(IN:=FALSE, PT:=T#0S);
4      State:=0;
5  END_IF
6
7  CASE state OF
8
9  0: (*StartSequence*)
10 StateText:='Käynnistä testi painamalla Start-nappia.';
11
12 (*State change*)
13 IF Start =1 THEN
14     State:=1;
15 END_IF
16
17 1: (*Start sequence*)
18
19 Led:=1;
20 State:=2;
21
22 2: (*Timer*)
23 Statetext:='Laitteisto varmistaa led:in toimintaa.';
24
25 Timer(IN := NOT DO8LahdonMonitorointi, PT:=T#5S);
26
27 IF Timer.Q THEN
28     State:=3;
29     Timer(IN := FALSE, PT:=T#0S);
30 END_IF
```

Kuva 40. Ledtest-ohjelman osa 1.

Ohjelman kolmannessa State-kohdassa led valo sammutettiin ja varmistettiin sen sammuminen käyttämällä lähdön monitoroinnin I/O-tietoa. Viimeisessä State-kohdassa ohjelma ilmoitti, jos led oli kunnossa. Jokaisessa ohjelman State-kohdassa nähtävät StateText-määrittelyt näkyivät käyttöliittymän testit-välilehden osiossa ”Testin edistyminen” riippuen siitä, missä kohtaa ohjelma eteni. Kaksi ohjelman viimeistä kohtaa on nähtävissä kuvassa 42.

```
31
32    3: (*Shutdown Led*)
33    Statetext:='Ledin sammumisen varmistaminen.';
34
35    Led:=0;
36    Timer(IN := DO8LahdonMonitorointi, PT:=T#5S);
37
38    IF Timer.Q THEN
39        State:=4;
40        Timer(IN := FALSE, PT:=T#0S);
41    END_IF
42
43    4: (*Shutdown Led*)
44    Statetext:='Led on kunnossa!';
45
46    Timer(IN := DO8LahdonMonitorointi, PT:=T#5S);
47
48    IF Timer.Q THEN
49        State:=0;
50        Timer(IN := FALSE, PT:=T#0S);
51    END_IF
52
53
54    END_CASE
55
```

Kuva 41. Ledtest-ohjelman osa 2.

5.6 **Laitteiston toiminnan testaaminen**

Tämän kehitysprojektin laitteiston toiminta tarkistettiin, jotta työn tilaaja pystyisi tekemään siihen sähköisten testien automatisointeja. Laitteiston kommunikaation toiminta laitteiden välillä varmistettiin aikaisemmin tämän kehitysprojektin aikana, joten itse tehdyn ohjelman ja käyttöliittymän toiminta tuli vielä varmistaa.

Kappaleessa 5.5.2 esitettyä CoDeSysillä tehtyä käyttöliittymää ja Ledtest-ohjelmaa käytettiin Raspilla WebVisualization ikkunassa. Tämä tarkoittaa sitä, että Raspilla mentiin osoitteeseen http://localhost:8080/webvisu.htm, jossa localhost kohtaan kirjoitettiin Raspin IP-osoite. Kun ohjelma ja käyttöliittymä ladattiin CoDeSysillä niin ne ilmestyvät automaattisesti yllä olevaan verkko-osoitteeseen.

Ohjelmaa ja käyttöliittymää ladattaessa, tuli CoDeSys ja Raspin välinen yhteys varmistaa. Tämä varmistettiin painamalla CoDeSysin valikosta Tools ja sieltä valittiin Update Raspberry Pi. Tätä kautta luotiin yhteys Raspiin, jos sitä ei ole vielä aikaisemmin oltu luotu. Tätä ei tarvitse kuitenkaan tehdä kuin ainoastaan ensimmäistä kertaa laitteistoja käynnistettäessä. Kun yhteys oli kunnossa niin tehdyt ohjelmat ja käyttöliittymä ladattiin painamalla valikosta Online ja sieltä kohtaa Login. Tällöin CoDeSys käänsi automaattisesti ohjelman Raspille sopivaksi ja otti samalla yhteyttä Raspiin. Jos

ohjelma haluttiin kääntää, mutta yhteyttä ei haluttu vielä muodostaa voitiin tämä tehdä valikon build, kohdasta build. Kun nämä oli tehty, ilmoitti CoDeSys mahdollisista virheistä ja jos virheitä ei ollut niin voitiin ohjelma käynnistää painamalla valikon debug kohdasta start. Jos virheitä jostain syystä ilmestyi, tuli ne korjata ennen kuin ohjelmaa voitiin ladata ja käyttää Raspissa. Webvisualization avattiin menemällä Raspilla aikaisemmin mainitulle nettisivulle. Jos ohjelmaan haluttiin tehdä muutoksia ei ohjelmaa tarvitse pysäyttää, kunhan CoDeSysistä painaa logout.

Ledtest-ohjelman tarkoituksena oli saada varmistettua kehitysprojektin laitteiston toiminta. Projektissa kehitetyn laitteiston ideana on pakottaa ohjelmallisesti uunin piirikortin lähtöjä päälle Modbus RTU -sarjaliikenneportin kautta. Kun ohjelma on pakottanut toimilaitteita päälle niin se varmistaa laitteiden toiminnan jonkun muun laitteesta saatavan I/O-tiedon perusteella. Esimerkiksi jos ohjelma pakottaa uunin rasvakeittimen lämpiämään niin se pystyy varmistamaan laitteen toiminnan sieltä saatavan lämpötilatiedon perusteella. Ledtest-ohjelman ja käyttöliittymän muiden välilehtien toiminta varmistettiin siten, että uunin piirikorttiin liitettiin termoparikortin lisäksi I/O-kortti (Liite2.). Tähän I/O-korttiin liitettiin kaksi led valoa, joista kuitenkin vain toista ohjattiin CoDeSysillä tehdyllä ohjelmalla. Led valo kytkettiin kortin digital output -liittimiin ja niiden osoitteet määritettiin I/O-listaan. Ledtest-ohjelma tarkistaa led valon toiminnan sytyttämällä sen ja varmistamalla sen syttymisen DO8LahdonMonitoroinnin kautta. Testissä voitaisiin testata mitä vain uunin yksinkertaista toimilaitetta, sillä ohjelma toimisi samalla periaatteella. Yksinkertaisella toimilaitteella tarkoitetaan esimerkiksi kontaktorin toiminnan varmistamista.

Lämpötilat-välilehden toiminta varmistettiin liittämällä uunin piirikorttiin kiinnitettyyn termoparikorttiin lämpöanturi. Tätä anturia lämmitettiin kaasupolttimella ja nähtiin kuinka Raspin käyttöliittymä reagoi lämpötilaan. Uunin kahdelle termoparikortille uunin toimilaitteilta ja tiloista tulevat anturit ovat numeroitu ja siitä tiedetään mistä lämpötilatiedosta on kyse. Niin kuin kuvasta 37 nähdään, moni lämpötiloista on 20°C, koska niihin ei ole liitettynä anturia ja ne ottavat lämpötilatietonsa automaattisesti termoparikortilla sijaitsevasta anturista.

I/O-tiedot -välilehden toiminta varmistettiin, kun led valoja käytettiin testiohjelmalla niin välilehden D08lahdonmonitorointi kohta reagoi sen toimintaan. Välilehden toiminta varmistettiin myös toisella testillä. Aikaisemmin määritellyistä I/O-tiedoista nähtiin, että I/O-kortin ensimmäinen liittimen osoite oli I0.0 ja sen nimi oli VirtalahdeOK. I/O-kortin liittimeen kytkettiin normaali painonappi, jota painamalla saatiin I/O-tiedot -välilehden ensimmäinen kohta reagoimaan.

6 YHTEENVETO

Kehitysprojektin alussa monitoimiuunin toiminnan tuntemus oli tärkeää työn onnistumisen kannalta. Kokemus monitoimiuunien parissa työskentelystä helpotti tämän työn lopputulokseen pääsemistä. Kehitysprojektissa käytetyn laitteiston kokoaminen sujui ongelmitta ja aikataulun mukaisesti. Projektissa ei kuitenkaan täysin pysytty aikataulussa, koska laitteiston kommunikaation kanssa ilmeni ongelmia.

Kommunikaatio-ongelmia selvitettäessä tutustuttiin erilaisiin ohjelmiin, jolla kommunikaation toiminta saatiin varmistettua. Ongelmien selvittelyyn tarvittavat ohjelmat eivät olleet entuudestaan tuttuja ja tästä syystä niiden käyttöä täytyi opiskella. Ongelmat syntyivät, koska projektissa käytettyjä laitteistoja ei kukaan ole aikaisemmin käyttänyt yhdessä ja tästä syystä ohjeiden löytäminen projektin toteutukseen oli hankalaa.

Kun laitteiston kommunikaatio saatiin toimimaan, niin laitteiston ohjelman ja käyttöliittymän teko sujui hyvin. Kokonaisuudessaan tässä työssä päästiin haluttuihin lopputuloksiin ja se mahdollistaa työn tilaajan toiveet tuotannon kehittämisestä. Opinnäytetyössä kehitettyyn laitteistoon aloitettiin työn tilaajan toimesta lisäämään uunin sähköisiä testejä ja laitteisto otetaan käyttöön tuotannossa heti testien valmistuttua. Tulevaisuudessa työn tilaajalla onkin toiveena, että laitetta voitaisiin käyttää myös huoltojen yhteydessä esimerkiksi vianhaussa. Lisäksi laitetta tullaan kehittämään siten, että se lähettää automaattisesti testeistä saatavan datan testauspöytäkirjana sähköisesti yrityksen tietokantaan.

Opinnäytetyön valmistumisen jälkeen jäin kehittämään testauslaitteistoon lisättäviä monitoimiuunin automatisoituja sähköisiä testejä. Tätä opinnäytetyötä tehdessä motivaationi pysyi korkealla, koska tiesin laitteiston tulevan yrityksen käyttöön. Tiesin myös, kuinka laitteen toiminta vaikuttaisi tuotannon kehittymiseen. Henkilökohtaisesti opin tämän kehitysprojektin aikana paljon projektien läpiviemisestä sekä työssä käytettävistä laitteista, kuten Raspista, Modbusista ja CoDeSysistä. Kun laitteisto luovutettiin työn tilaajalle, sain paljon positiivista palautetta, jolloin tiesin onnistuneeni tässä kehitysprojektissa.

LÄHTEET

Alander, J. (2016). Haettu 18.9.2016 osoitteesta http://lipas.uwasa.fi/~TAU/AUTO1010/slides.php?Mode=Printer&File=9050PLC.txt

Beckhoff (n.d.). CFC - Continuous Function Chart – Language. Haettu 23.9.2016 osoitteesta https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/136018571.html&id=

Beckhoff (n.d.). Instruction List. Haettu 18.9.2016 osoitteesta https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages.htm&id=96109762213157887610

Beckhoff (n.d.). Structured Text. Haettu 21.9.2016 osoitteesta https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/TcPlcCtrl_Languages%20ST.htm&id=

Brookes, T. (2012). Raspberry Pi – A Credit-Card Sized ARM Computer – Yours For Only $25. Haettu 12.9.2016 osoitteesta http://www.makeuseof.com/tag/raspberry-pi-creditcard-sized-arm-computer-25/

EasyIoT (n.d.). Internet Controlled switch. Haettu 16.9.2016 osoitteesta http://iot-playground.com/blog/2-uncategorised/12-internet-controlled-switch

element14 (2015). Raspberry Pi 7” Touchscreen Display. Haettu 29.9.2016 osoitteesta https://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen-display

Fläkt Woods (2016). Modbus-yleistä. Haettu 26.9.2016 osoitteesta http://resources.flaktwoods.com/Perfion/File.aspx?id=5ae5d3e3-2af6-46c6-9244-f7d3e1304f54

Harris, R. (2016). Raspberry Pi 3 Model B Offers Built-in Wireless LAN and Bluetooth. Haettu 18.9.2016 osoitteesta https://appdevelopermagazine.com/3688/2016/2/29/Raspberry-Pi-3-Model-B-Offers-Built-in-Wireless-LAN-and-Bluetooth/

John, K. & Tiegelkamp, M. (2001). IEC 61131-3: Programming Industrial Automation Systems. Haettu 20.9.2016 osoitteesta http://www.dee.ufrj.br/controle_automatico/cursos/IEC61131-3_Programming_Industrial_Automation_Systems.pdf

Lessons In Electric Circuits (n.d.). Ladder Logic. Haettu 18.9.2016 osoitteesta http://www.machinetoolhelp.com/Learn/ladderlogic.html

Merrychef (n.d.). Merrychef e4 -turbouuni. Haettu 15.9.2016 osoitteesta http://www.merrychef.com/Product/fam_fpwehi/eikon-e4

Modbus Organization (n.d.). Modbus application protocol specification V1.1b3. Haettu 22.9.2016 osoitteesta http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Mullins, R (2012). Raspberry Pi, University Of Cambridge. Haettu 13.9.2016 osoitteesta http://www.cl.cam.ac.uk/projects/raspberrypi/

Omron (n.d.). SFC Introduction Guide. Haettu 24.9.2016 osoitteesta https://www.fa.omron.com.cn/data_pdf/mnu/r149-e1-03_sfc.pdf?id=1605

PLC Academy (2015). Structured Text Tutorial to Expand Your PLC Programming Skills. Haettu 21.9.2016 osoitteesta http://www.plcacademy.com/structured-text-tutorial/

PLC Manual (n.d.). CoDeSys. Haettu 17.9.2016 osoitteesta http://www.plcmanual.com/codesys

ProSoft Technology (2011). What's the difference between Modbus ASCII and Modbus RTU?. Haettu 26.9.2016 osoitteesta http://www.prosoft-technology.com/kb/article.php?id=382

Raspberry Pi (n.d.). FAQS. Haettu 12.9.2016 osoitteesta https://www.raspberrypi.org/help/faqs/#introWhatIs

Raspberry Pi (n.d.). Raspberry Pi 3 Model B. Haettu 16.9.2016 osoitteesta https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

R-Menu Oy (n.d.). Tietoja yrityksestä. Haettu 12.9.2016 osoitteesta http://www.r-menu.fi/

SFS 9241 (2010). Ihmisen ja järjestelmän vuorovaikutuksen ergonomia. Osa 210: Vuorovaikutteisten järjestelmien käyttäjäkeskeinen suunnittelu. SFS Online. Haettu 12.10.2016 osoitteesta https://online.sfs.fi

SFS 9241 (1998). Näyttöpäätteillä tehtävän toimistotyön ergonomiset vaatimukset. Osa 11: Käytettävyyden määrittely ja arviointi. SFS Online. Haettu 12.10.2016 osoitteesta https://online.sfs.fi

Simply Modbus (2015). Frequently Asked Questions. Haettu 26.9.2016 osoitteesta http://www.simplymodbus.ca/FAQ.htm#Modbus

Simply Modbus (2015). Modbus TCP/IP. Haettu 26.9.2016 osoitteesta http://www.simplymodbus.ca/TCP.htm

SKS Group Oy (2016). CoDeSys. Haettu 17.9.2016 osoitteesta http://www.sks.fi/www/_Codesys&id=CODESYS

Sundström, J. (2015). *Ravintolauunin tuotekehitys*. Opinnäytetyö. Automaatiotekniikan koulutusohjelma. Hämeen ammattikorkeakoulu.

Toshiba Corporation (2008). Programming Instructions (LD/FBD/SFC/ST). Haettu 24.9.2016 osoitteesta http://www.toshitec.pt/e107_files/downloads/toshiba/PLC/NV_series/6F8C1226.pdf

Upton, E. (2012). Model B now ships with 512MB of RAM. Blogijulkaisu 15.10.2012. Haettu 15.9.2016 osoitteesta https://www.raspberrypi.org/blog/model-b-now-ships-with-512mb-of-ram/

Upton, E. (2016). Raspberry Pi 3 on sale now at $35. Blogijulkaisu 29.2.2016. Haettu 13.9.2016 osoitteesta https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/

Vainionpää, J. 2016. Uusi monitoimiuuni konsepti. PowerPoint-esitys.

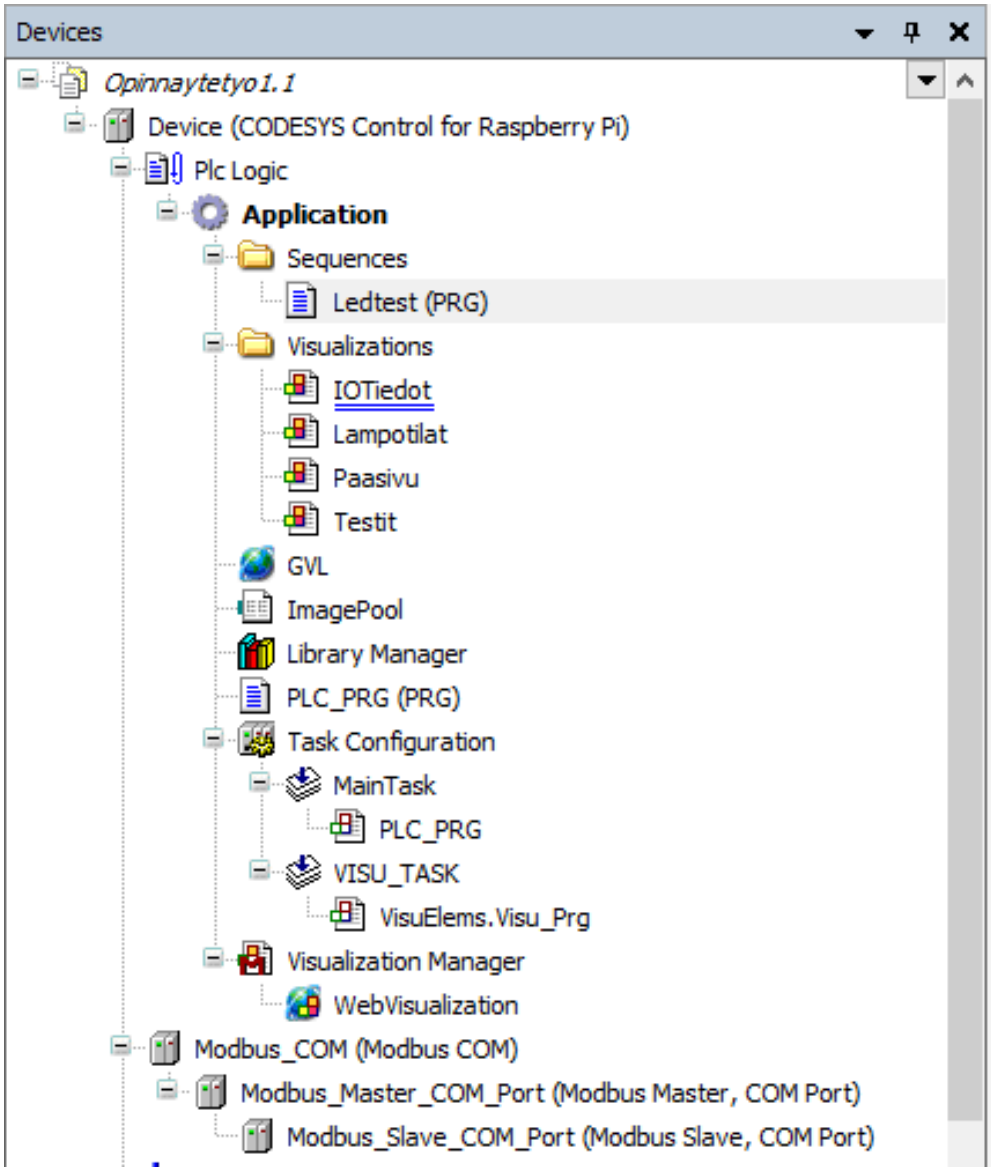
Vilches, J. (2012). Interview with Raspberry´s Founder Eben Upton. Haettu 12.9.2016 osoitteesta http://www.techspot.com/article/531-eben-upton-interview/

Witthoeft, J. (2013). MODBUS in a Nutshell. Haettu 21.9.2016 osoitteesta http://gridconnect.com/blog/tag/modbus-tutorial/

3S-Smart Software Solutions GmbH (2016). CODESYS - The comprehensive software suite for automation technology. Haettu 16.9.2016 osoitteesta https://www.codesys.com/the-system.html

Liite 1

CODESYS-OHJELMARUNKO



Liite 2

LED VALON TOIMINNAN TESTAUS