KIINTEISTÖAUTOMAATION ETÄHALLINTA

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2009
Henri Karttunen

Lahden ammattikorkeakoulu
Koulutusohjelma

Henri Karttunen: Kiinteistöautomaation etähallinta

Ohjelmistotekniikan opinnäytetyö, 42 sivua

Kevät 2009

TIIVISTELMÄ

Internetyhteyksien yleistyminen kotitalouksissa on luonut mahdollisuuden ohjata kodin automaatiojärjestelmiä mistä päin maailmaa tahansa internetyhteyden välityksellä. Ohjattavia asioita voivat olla esimerkiksi huonekohtaiset lämmitykset, ovien lukitukset ja pihavalaistukset. Lämmitysten, lukitusten ja valaistusten ohjaukseen käytetään erilaisia laitteita, jotka kytketään palvelintietokoneeseen etähallintaa varten.

Tavoitteena tässä työssä oli tehdä ohjelmisto, jonka avulla kaikkia taloautomaatioon kuuluvia laitteita voidaan hallita keskitetystä yhden palvelimen ja ohjelmiston avulla internetyhteyttä käyttäen. Työssä tutustuttiin asynkronisen sarjaliikenteen toimintaan, sarjaliikennestandardeihin, sekä Modbus-protokollaan.

Asynkroninen tiedonsiirto on yksinkertainen tapa siirtää tietoa laitteesta toiseen. Sarjamuotoisessa asynkronisessa tiedonsiirrossa bitit lähetetään järjestyksessä yksi kerrallaan, ja vastaanottaja synkronoi kellonsa jokaisen vastaanotettavan tavun alussa lähettävän osapuolen lähettämän aloitusbitin avulla. Erillisen kellopulssin käyttöä ei tarvita lähetettäessä tietoa asynkronisesti. RS-232-standardi määrittelee kahden laitteen väliseen kommunikaatioon kykenevän tiedonsiirtomallin sekä fyysiset kytkennät. RS-485 ja -422 -standardit määrittelevät useiden laitteiden väliseen tiedonsiirtoon tarkoitetun tiedonsiirtoväylän.

Modbus-protokolla on yleisesti teollisuuden käytössä oleva Modiconin kehittämä tietonsiirtoprotokolla, jossa yksi laite on määritetty johtamaan laitteiden välistä keskustelua. Protokolla on alunperin kehitetty Modiconin omien ohjelmoitavien logiikkaohjainten tiedonsiirtoon ja ohjaukseen.

Työssä toteutettiin ohjelmistokokonaisuus, joka mahdollistaa kiinteistöautomaation ohjauksen internetin avulla. Työn saatiin toteutettua onnistuneesti ja tavoitteeseen päästiin suunnitelmien ja aikataulujen mukaisesti. Järjestelmä on tuotteistettu ja se on otettu käyttöön jo useissa kohteissa.

Avainsanat: RS-232, RS-485, Modbus, asynkroninen sarjaliikenne

Lahti University of Applied Sciences
Faculty of Technology

KARTTUNEN, HENRI: Remote control for real estate automation

Bachelor´s Thesis in Software Engineering, 42 pages

Spring 2009

ABSTRACT

Today almost every home has an Internet connection which makes it possible to control home automation from everywhere around the world using the internet. Things to control are for example room temperatures, door locks and outdoor lighting. Many different types of devices are used to control them and they are all connected to a server PC.

The goal of this project was to develop a software program that makes it possible to control these devices using the internet and only one server and one sofware. Asynchronous serial data transfer, serial data transfer standards and the Modbus protocol were studied during the process.

Asynchronic data transfer is a simple way to transfer data from one device to another. Using serial asynchronic data transfer, bits are sent in order one by one and the recipient synchronizes its clock in the beginning using the start bit sent by the sending device. When sending data using asynchronous transfer there is no need to use a separate clock pulse. Serial data transfer standards are used to transfer data from one device to another in serial format. The RS-232 standard defines the communication pattern and physical connections for data transfer between two peers. The RS-485 and RS-422 standards are used to create a channel for many devices to communicate together.

The Modbus protocol is a data transfer protocol widely used by industry. Modbus was developed by Modicon originally mostly for its own programmable logic controllers. In the Modbus protocol there is always one master that leads the communication and other devices are slaves.

A commercial software program was implemented in this thesis, enabling remote control for real estate automation via the internet. The goal was achieved succesfully and in schedule. The software program has been productized and is in use in many sites.

Key words: RS-232, RS-485, Modbus protocol, asynchronous data transfer

SISÄLLYS

1 JOHDANTO

Tietoverkkoyhteydet ovat kehittyneet viime vuosina paljon, ja ihmiset voivat tämän ansiosta hoitaa erilaisia asioitaan helposti mobiilin internetyhteyden välityksellä. Kiinteistöautomaation ja lämmitysjärjestelmien asetusten muuttaminen etäkäyttöiseksi on järkevää sekä taloudellisesti että luonnon kannalta. Etähallintajärjestelmän avulla esimerkiksi huoltoyritykset voivat säästää kustannuksista, kun kaikkia oven avauksia tai huoneistojen lämmityksiä ei tarvitse mennä tekemään paikan päälle.

HL-Heat on Lahden keskustassa toimiva talotekniikan ja energiaosaamisen erikoisliike. Yrityksessä työskentelee vakituisesti alle kymmenen henkeä kahdessa toimipisteessä. Lahdessa yrityksellä on myyntipiste ja toimisto ja Mikkelin toimitiloissa on tehdas, jossa valmistetaan automaatio- ja sähkökeskuksia yksityis- ja yritysasiakkaiden tarpeisiin.

Työn tavoitteena on tehdä taloautomaatiossa käytettäviä laitteita yhdistävä ohjelmisto, joka tarjoaa mahdollisuuden myös etäkäyttöön internetyhteyden välityksellä. Ohjelmiston avulla kaikkia HL-Heat Oy:n käyttämien taloautomaatiolaitteiden ohjauksia voidaan hallita keskitetysti mistä päin maailmaa tahansa.

Laitteilla voidaan ohjata esimerkiksi lämmityksiä, valaistuksia, ajastettuja toimintoja, kuten autolämmittimiä, ja hallita hälytyksiä, kuten varas- ja palohälytyksiä ja ohjata ne tarvittaessa gsm-verkon välityksellä kiinteistön omistajalle tai vartiointiyhtiölle. Säätöjärjestelmä pitää sisällään mahdollisuuden säätää tilakohtaisesti lämpötiloja Celsius-asteen kymmenyksen tarkkuudella sekä ohjauslähtöjä, joita voidaan ohjata joko ajastetusti tai käsiohjauksella. Näitä ohjauslähtöjä voidaan käyttää esimerkiksi valaistusten, autolämmittimien ja ilmanvaihtokoneiden sekä erilaisten lämpöpumppujen ohjaamiseen.

Tarve ohjelman kehittämiseen tuli vanhemman pc-hallintaohjelman jäädessä jälkeen järjestelmien ja laitteiden kehityksessä. Ohjelmiston päävaatimuksena oli HL-Heat Oy:n lämmönsäätöjärjestelmien etähallinta graafisella käyttöliittymällä. Ohjelmiston täytyy kyetä ohjaamaan kaikkia digitaaliohjauksia, kello-ohjauksia, ajastustauluja sekä tilakohtaisia lämmitysohjauksia.

Ohjelman kehittäminen aloitettiin syksyllä 2007 ja päätettiin keväällä 2008. Ohjelmaan on sen jälkeen tehty päivityksiä ja joitakin vikojen korjauksia.

Tässä opinnäytetyössä käydään läpi ohjelmaan liittyvien tärkeimpien standardien, eli RS-232, RS-485 sekä Modbus-protokollan, ominaisuuksia ja käyttötarkoituksia. Itse hallintaohjelman toimintaa tarkastellaan kuitenkin vain yleisellä tasolla.

2 ASYNKRONINEN SARJAMUOTOINEN TIEDONSIIRTO

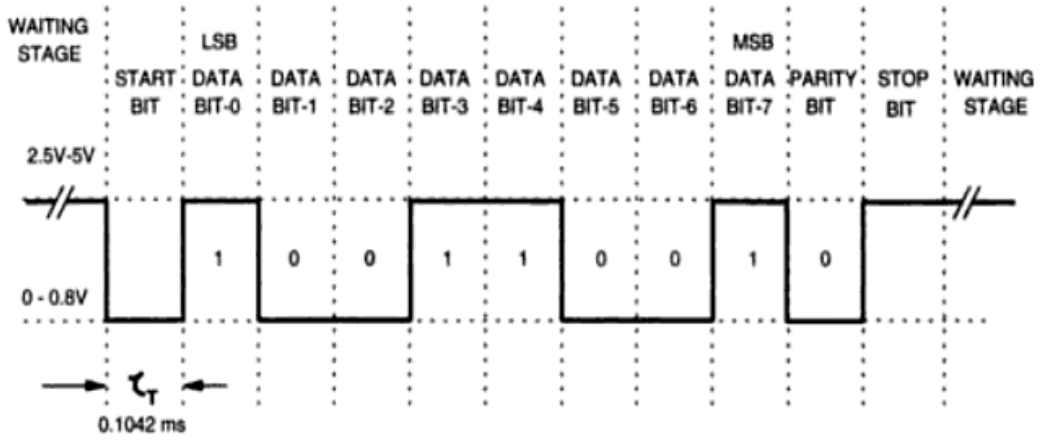
2.1 Periaatteet

Sarjamuotoisessa tiedonsiirrossa bitit siirretään peräkkäin järjetyksessä linjaa pitkin. Toinen tapa siirtää tietoa on rinnakkaissiirto, jossa tieto siirretään jokainen bitti omalla linjallaan. Tietokonemaailmassa sarjamuotoista tiedonsiirtoa käytetään muun muassa lisälaitteiden, kuten printtereiden, modeemien ja hiirien liittämiseksi laitteistoon. (An 1998, 13.)

Moni tietokoneeseen kytketyistä laitteista kommunikoi tietokoneen kanssa asynkronisesti. Asynkronista tiedonsiirtoa käytetään paljon, sillä se on helppo ja yksinkertainen tapa saada laitteet kommunikoimaan keskenään. (Asynchronous serial communication overview.)

Synkronoidussa tiedonsiirrossa lähettäjän ja vastaanottajan kelloja on synkronoitava kokoajan, jotta tieto siirtyisi oikein. Erillisen kellosignaalin käyttö on yksi tapa synkronoida vastaanottajan kello lähettäjän kanssa. Tämä toimii esimerkiksi niin, että jokaisen lähetetyn bitin kohdalla lähettäjä antaa kellosignaalin sille varatulla omalla linjalla ja vastaanottaja tietää tarkistaa linjan tilan juuri sillä hetkellä. (Stallings 1997, 141.)

Asynkronisen tiedonsiirron ideana on lähettää tietoa yksi tavu kerrallaan, jolloin vastaanottaja synkronoi kellonsa jokaisen tavun alussa aloitusbitin avulla. Kuviossa 1 näkyy 9600 bittiä sekunnissa nopeudella asynkronisesti lähetettävän tavun rakenne. Ensin lähetetään aloitusbitti, ja viimeisenä taas lopetusbitti, minkä jälkeen linja siirtyy lepotilaan. Jokaiseen lähetettävään tavuun kuuluu vähintään aloitus- ja lopetusbitit sekä viidestä kahdeksaan tavua varsinaista lähetettävää tietoa. Kun tietoa ei siirretä, linja on lepotilassa, joka on sama tila, jossa luettava bitti tulkitaan arvoksi yksi. (Stallings 1997, 141.)



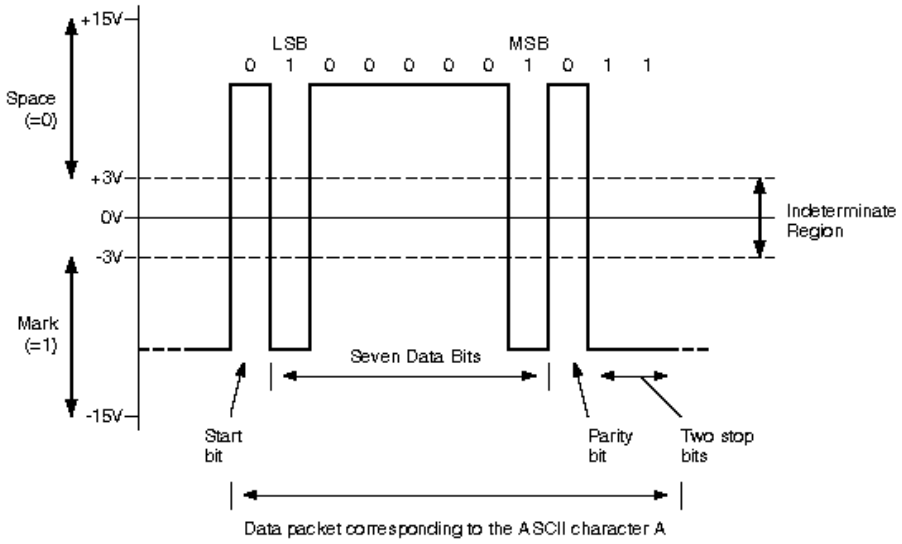
KUVIO 1. Asynkronisesti siirrettävän tavun rakenne (An 1998)

Jotta vastaanottaja tietäisi alkavasta tiedonsiirrosta ja osaisi ottaa sen vastaan oikeaan aikaan ja kokonaisuudessaan, lähettävän osapuolen on lähetettävä ensimmäisenä aloitusbitti. Lähetettävän datatavun jälkeen lähetetään myös lopetusbitti, jotta vastaanottaja tietää lähetyksen loppuvan. Aloitus- ja lopetusbitit lähetetään jokaisen siirrettävän tavun alussa ja lopussa, ja niiden on oltava eri merkkiset, jotta vastaanottaja tietää seuraavan lähetyksen alkavan tai loppuvan. (Data transmission – Wikipedia 2008.)

Esimerkiksi lähetettäessä monta kertaa ASCII merkistön mukaista 'A'-kirjainta, joka on binäärilukuna ilmaistuna 01000001, lisätään jokaiseen lähetettävään tavun alkuun bitti, jonka arvo on 0, sekä tavun loppuun bitti, jonka arvo on 1. Kolmen peräkkäisen tavun siirto näyttäisi seuraavalta:

0<u>10000010</u>10<u>10000010</u>10<u>10000010</u>1.

Alleviivatut osat ovat varsinaista siirrettävää dataa ja alleviivaamattomat nollat ja ykköset aloitus- ja lopetusbittejä. Huomioitavaa on myös bittien järjestys, sillä ensimmäisenä lähetettävä bitti on vähiten merkitsevä. (Stallings 1997, 141.)



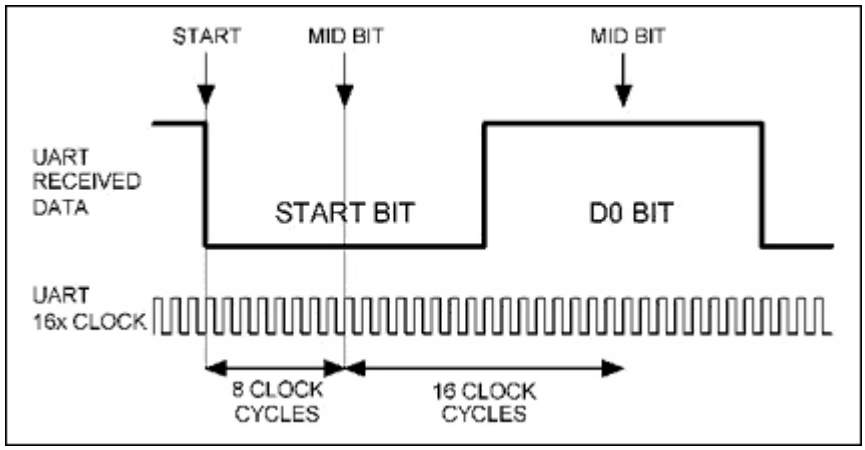
KUVIO 2. Tavun siirto RS-232 -standardin mukaisesti (Arc Electronics)

Kuviossa 2 näkyy yhden tavun siirto RS-232 -standardin mukaisesti. Alussa linja on lepotilassa, arvossa yksi. Ensimmäinen siirrettävä bitti on aloitusbitti, joka on arvoltaan nolla, ja sen jälkeen seitsemän databittiä vähiten merkitsevä ensimmäisenä. Kuvassa näkyvä tavu sisältää myös pariteetti bitin, jonka avulla voidaan tarkistaa vastaanotetun tavun oikeellisuus. Viimeisenä lähetetään kaksi lopetusbittiä, joiden arvo on sama kuin lepotilan, eli yksi. (Arc Electronics.)

2.2 Tulkinta

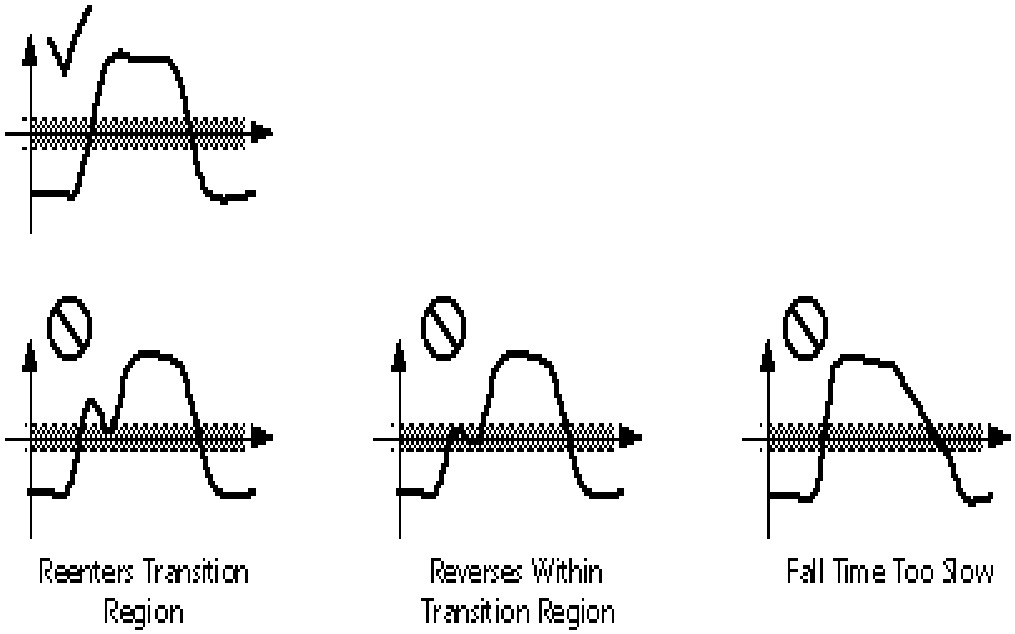
Sarjamuotoisessa tiedonsiirrossa peräkkäin vastaanottajalle saapuvat bitit tulkitaan linjan tilan mukaan. Yleisesti sarjamuotoisessa tiedonsiirrossa käytössä olevan Non Return to Zero (NRZ) -koodausmenetelmän mukaisesti lepotilaa vastaa negatiivinen jännitearvo linjassa. Nollaksi tulkittavat luvut ovat vastaavasti positiivisia jännitearvoja. Tätä menetelmää käytetään muun muassa RS-232-standardia noudattavissa liitännöissä. (Stallings 1997, 141.)

Vastaanottaja tulkitsee saapunutta dataa UART-, eli Universal Asyncronous Receiver/Transmitter -piirin avulla. Piiri ottaa vastaanotetettavasta tiedosta useita näytteitä esimerkiksi 16-kertaa vastaanottotaajuutta korkeammalla taajuudella ja pyrkii erottamaan vastaanotetut bitit ja virhetilanteet. Kuvio 3 esittää UART:n näytteenottotaajuutta verrattuna saapuvan tiedon taajuuteen. Bitin keskivälillä on paras paikka näytteenottoa varten. (Maxim 2003, 2.)



KUVIO 3. UART:n näytteenottotaajuus (Maxim 2003, 2)

Kuviossa 3 näkyy vastaanotettu aloitusbitti sekä ensimmäinen, vähiten merkitsevä databitti. UART tunnistaa aloitusbitin lepotilassa olevasta linjasta havaitsemalla tilan muutoksen. Kuviossa näkyvän aloitusbitin keskikohta, eli kahdeksas näyte, joka on se piste, mistä UART varmistaa, ettei sen havaitsema laskeva reuna ollut vain piikki tai muu lyhyt signaalihäiriö. Toinen, hiukan varmempi tapa varmistaa, että laskeva reuna on todella aloitusbitti, on tarkistaa kolme näytettä: seitsemäs, kahdeksas ja yhdeksäs. (Maxim 2003, 2.)



KUVIO 4. Signaalin tulkinta. (Strangio)

Jotta UART voisi tulkita signaalit oikein, signaalien on noudatettava seuraavia sääntöjä: signaalin on vaihdettava tila kokonaan kääntämättä tilaansa takaisin tai käymättä enää aiemmassa tilassa ja ohjaussignaalien tilanvaihtoon saa kulua korkeintaan yksi millisekunti. (Strangio.)

Näiden lisäksi ajoitukseen ja tiedonsiirtoon tarkoitetut signaalit saavat kuluttaa tilanvaihtoon korkeintaan joko yhden millisekunnin yli 25 millisekunnin signaaleihin. 125 mikrosekunnista 25 millisekuntiin kestävän signaalin kestosta signaalit saavat kuluttaa tilanvaihtoon korkeintaan neljä prosenttia tai alle viisi mikrosekuntia sitä lyhyemmissä signaaleissa. Sen lisäksi, että signaalin on oltava riittävän nopea, se ei saa olla liian nopea. Yli kolmenkymmenen voltin muutos mikrosekunnin aikana voi aiheuttaa häiriöitä linjassa. (Strangio.)

Kuviossa 4 näkyy oikein lähetetty signaali ylimmällä rivillä, sekä kolme virheellistä signaalia. Ylärivillä olevassa kuviossa signaali vaihtaa tilansa nopeasti, ja pitää tilansa loppuun asti, kunnes vaihtaa taas tilansa nopeasti. Vasemmalta lukien ensimmäisessä virheellisessä kuviossa signaali palaa nousun jälkeen alle tulkintatason, toisessa signaalin nousu muuttuu hetkeksi laskuksi ja viimeisessä

lasku on liian loiva tulkittavaksi oikein. Vain ylärivillä oleva signaali voidaan varmasti tulkita oikein. (Strangio.)

3 SARJALIIKENNESTANDARDIT

3.1 RS-232

3.1.1 Yleistä RS-232:sta

RS-232 -standardi on 1960-luvulla nykyisen Electronic Industries Associationin kehittämä sarjaliikennestandardi, joka on kehitetty kahden laitteen keskinäistä kommunikointia ja tiedonsiirtoa varten. Standardi määrittää signaalin jännitearvot, ajastuksen, funktiot, tiedonsiirtomallin ja mekaaniset liitynnät. (Strangio.)

RS-232 -standardi on mahdollistaa kaksisuuntaisen tiedonsiirron sarjamuotoisena asynkronisesti korkeintaan 20 metrin etäisyydelle. Sarjamuotoinen tiedonsiirto on yksikertaisempi kytkennältään kuin rinnakkaisliikenne, sillä se tarvitsee vain yhden linjan suuntaa kohden. Tiedonsiirto sarjaliikenneprotokollaa käyttäen on myös hitaampaa, sillä tieto siirretään bitti kerrallaan. Tietokonemaailmassa RS-232 -standardia käytetään lisälaitteiden, kuten printtereiden, modeemien ja hiirien liittämiseksi laitteistoon. (An 1998.)

RS-232 -standardin mukainen siirrettävä tavu koostuu aloitusbitistä, joka on aina nolla; databiteistä, joita on joko viisi, kuusi, seitsemän, kahdeksan tai yhdeksän; yhdestä pariteetti bitistä ja yhdestä tai puolestatoista loppubitistä, joka on aina asetettu. Kun dataa ei lähetetä, linja on aina asetetussa tilassa. Tiedonsiirto aloitetaan lähettämällä aloitustavu eli nollaamalla linjan tila. Tietoa vastaanottaa ja lähettää siihen tarkoitukseen kehitetty UART-piiri, eli Universal Asynchronous Receiver/Transmitter, joka huolehtii lukemisen ja kirjoittamisen ajoittamisesta. (An 1998.)

Standardin mukaisesti tietoa voidaan siirtää erilaisilla baud-nopeuksilla, joista käytetyimmät ovat 110, 150, 300, 600, 1200, 2400, 4800, 9600 ja 19200 bittiä

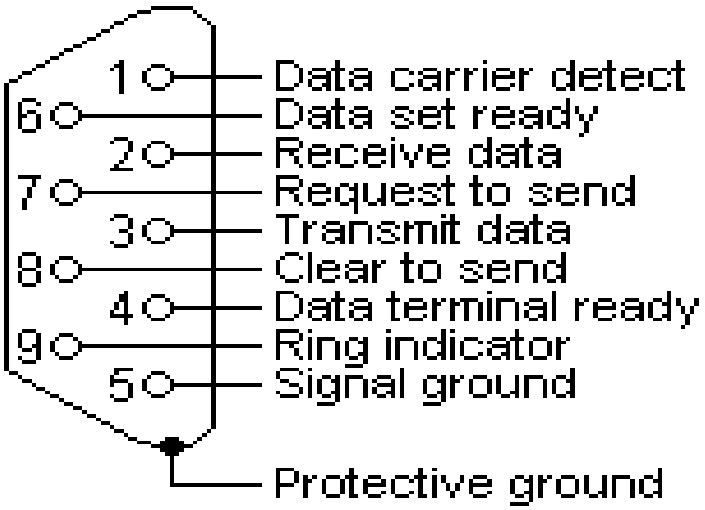
sekunnissa. Baudinopeus tarkoittaa sekunnissa siirrettyjen bittien lukumäärää. Kun lasketaan, montako bittiä jokaista tavua varten tarvitsee siirtää, voidaan laskea, montako tavua sekunnissa pystytään siirtämään. Esimerkiksi kahdeksan databittiä, yksi aloitus- ja lopetusbitti sekä pariteettibitti ovat yhteensä 11 bittiä tavua kohden. Jos tällaisia tavuja siirretään nopeudella 19200 bittiä sekunnissa, tarkoittaa se noin 1,75 kilotavua sekunnissa, jolloin yhden tavun lähetys kestää noin viisi millisekuntia. (An 1998.)

RS-232 -standardi tukee yksinkertaista bittitason virheentarkistusta pariteettibitin avulla. Pariteettibitin ideana on, että lasketaan kaikki asetetut bitit lähetettävän tavun dataosasta ja summataan ne. Paritonta pariteettia käytettäessä pariteettibitti nollataan, jos tulos on pariton. Jos tulos on parillinen pariteettibitti asetetaan. Pariteettibitin avulla voidaan tunnistaa osa virhetilanteista, mutta esimerkiksi useiden bittien väärä tulkinta jää huomaamatta. Pariteettitarkistus voidaan jättää tekemättä asettamalla pariteetin tarkistus linjan molemmissa päissä pois päältä. (An 1998.)

3.1.2 Signaalit

Tiedonsiirtoa RS-232-standardin mukaisessa sarjaliikenteessä ohjataan signaaleilla, joille on jokaiselle omistettu oma nasta sarjaporttiliittimestä. Nämä signaalit voidaan jakaa kuuteen eri pääryhmään: signaalimaa ja suojamaa, ensisijainen kommunikointikanava, toissijainen kommunikointikanava, modeemin tila ja ohjaussignaalit, lähetyksen ja vastaanoton ajoitussignaalit, sekä kanavan testisignaalit. Kuviossa 5 näkyy pienemmän, yhdeksän nastan, liittimen kytkentä ja signaalit. (Strangio.)

Liittimiä RS-232 -standardin mukaisille sarjaliikenneväylille on kahdenlaisia: pienempi yhdeksän nastan liitin, joka on yleisesti käytössä esimerkiksi pc-laitteissa sarjaporttiliittimenä sekä suurempi 25-nastan liitin. Pienemmässä liittimessä ei ole lainkaan toissijaista kommunikointikanavaa, ja siitä on karsittu pois suurin osa signaaleista sekä yhteyden testaamiseen tarkoitetut signaalinastat. (Bies 2008a.)



KUVIO 5. Yhdeksän nastan RS-232 liitin (D-SUB9) (Bies, 2008a)

Ensimmäiseen nastaan kytketään Data Carrier Detect (DCD) -signaali, joka on ilmoitus linjalla havaitusta kantoaallosta. Toiseen nastaan kytketään sisään saapuvaa dataa kuljettava johdin ja kolmanteen lähetettävän datan johdin. Neljäs, eli Data Terminal Ready (DTR) -signaali on kättelysignaali, joka kertoo päätelaitteen olevan valmis tiedonsiirtoon. (Bies 2008a.)

Kuudenteen nastaan kytkettävä DSR, eli Data Set Ready, -signaali on ilmoitus valmiudesta vastaanottaa dataa ja komentoja. Linjaan kytketty päätelaite voi pyytää lupaa tiedon lähettämiseen, joka tapahtuu seitsemänteen nastaan kytkettävän RTS, Request To Send, -signaalin avulla. CTS, eli Clear To Send -signaalin avulla päätelaite voi varmistaa esimerkiksi modeemin valmiuden lähettää tietoa. (Bies 2008a.)

Koska RS-232 -standardissa signaalit tulkitaan erotuksena maasignaalista, täytyy laitteiden olla kytkettynä samaan nollatasoon. Viidenteen nastaan kytketään signaalimaa, joka toimii signaalien tulkinnassa nollareferenssinä. Suojamaa

kytketään liittimen kotelointiin ja sitä kautta päätelaitteen runkoon ja myös signaalimaa saatetaan kytkeä päätelaitteen maahan. (Bies 2008a.)

3.2 RS-485 ja RS-422

Kuten RS-232, myös RS-485 on erilaisten laitteiden väliseen tiedonsiirtoon tarkoitettu sarjaliikennestandardi. RS-485-standardi mahdollistaa kuitenkin asioita, joita RS-232 ei salli. RS-485 sallii useita laitteita samaan verkkoon yhtäaikaisesti, toisin kuin RS-232, joka on tarkoitettu vain kahdelle laitteelle. RS-485 mahdollistaa myös pitemmän matkan laitteiden välille kuin RS-232:n standardin tarjoama 20 metrin välimatka, sillä RS-485-standardin mukaisessa väylässä voi laitteiden välistä matkaa olla jopa 1200 metriä. (Bies 2008b.)

Lähes kaikkien tietokonelaitteiden suoraan tukema RS-232 -standardi kykenee kommunikoimaan lyhyellä välimatkalla laitteiden kesken korkeintaan kahdenkymmenen kilobaudin sekuntinopeudella, kun taas RS-485:n avulla voidaan siirtää 20 metrin etäisyydellä jopa kymmeniä megabaudeja sekunnissa. (Bies 2008b.)

RS-485-väylä voidaan kytkeä joko puolittain kaksisuuntaisesti kaksijohtimisena kytkentänä, tai RS422-standardin mukaisesti aidosti kaksisuuntaiseksi käyttämällä neljää johdinta. (Bies 2008b.)

RS-485-standardin määritykset, toisin kuin RS-232:n määritykset, rajautuvat vain verkon fyysiseen kerrokseen. Tiedonsiirrossa RS-485-standardin mukaisesti käytetään differentiaalista signalointia, joka yhdistettynä kierrettyyn parikaapeliin auttaa myös vähentämään häiriöitä tiedonsiirrossa, koska häiriö kohdistuu tällöin molempiin johtimiin ja jännite-ero pysyy samana. (RS-485 – Wikipedia 2008.)

RS-422 on hyvin samantapainen standardi, kuin RS-485, sillä se on nopea ja siihen voidaan kytkeä useita laitteita samaan väylään. RS-422 on kuitenkin hieman rajoittuneempi, sillä yhdessä väylässä voi olla vain yksi lähettävä laite,

mutta useita vastaanottajia. Verrattuna RS-485-standardi, RS-422:n korkein mahdollinen siirtonopeus, 10 megabaudia sekunnissa, on myös hitaampi. (Bies 2008b.)

Käytettäessä pitkän välimatkan kytkentöjä tai tiedonsiirtonopeuksia, jotka ovat yli 200 kilobaudia sekunnissa, on RS-485- ja RS-422 -protokollan mukaisissa väylissä käytettävä linjan terminointia linjan molemmissa päissä. (Soltero, Zhang & Cockrill 2002, 15.)

RS-485 -standardin tärkeimmät johdinmerkinnät ovat A ja B. A tarkoittaa negatiivista, eli invertoitua ja B positiivista invertoimatonta jännitettä. A voidaan merkitä myös - -merkillä ja B + -merkillä. Jos linjassa käytetään kolmatta johdinta, sen merkintä on C, joka tarkoittaa yhteistä signaalimaata. (EIA-485 – Wikipedia 2009.)

RS-485- ja RS-422 -standardissa väylästä luettava tieto tulkitaan jännitearvojen polariteetista. Oikea polariteetti, jossa A-johtimeen tulee negatiivinen jännite ja B-johtimeen positiivinen, tulkitaan arvoksi 1. Kun jännitearvot käännetään, väylän signaali tulkitaan nollaksi. Standardi ei määrittele sen tarkemmin, kuinka nollia ja ykkösiä tulisi tulkita. (Smethurst 2007.)

RS-485 -väylä, joka sallii useita lähettäviä laitteita, sallii oletusarvoisesti kahdentoista kilo-ohmin vastaanottovastuksella varustettuna 32 laitetta samaan verkkoon. Kuitenkin saatavilla on korkean impedanssin RS-485 -laitteita, joilla laitteiden suurin sallittu määrä voidaan nostaa 256:een. Käyttämällä toistimia voidaan laitteiden määrä nostaa jopa tuhansiin ja laitteiden väliset matkat useisiin kilometreihin. Tästä syystä RS-485 -standardi on suosittu tietokoneiden, ohjelmoitavien logiikkakontrollereiden ja erilaisten tieteellisten ja teknisten sovellusten käytössä. (Bies 2008b.)

4 MODBUS-PROTOKOLLA

4.1 Yleistä

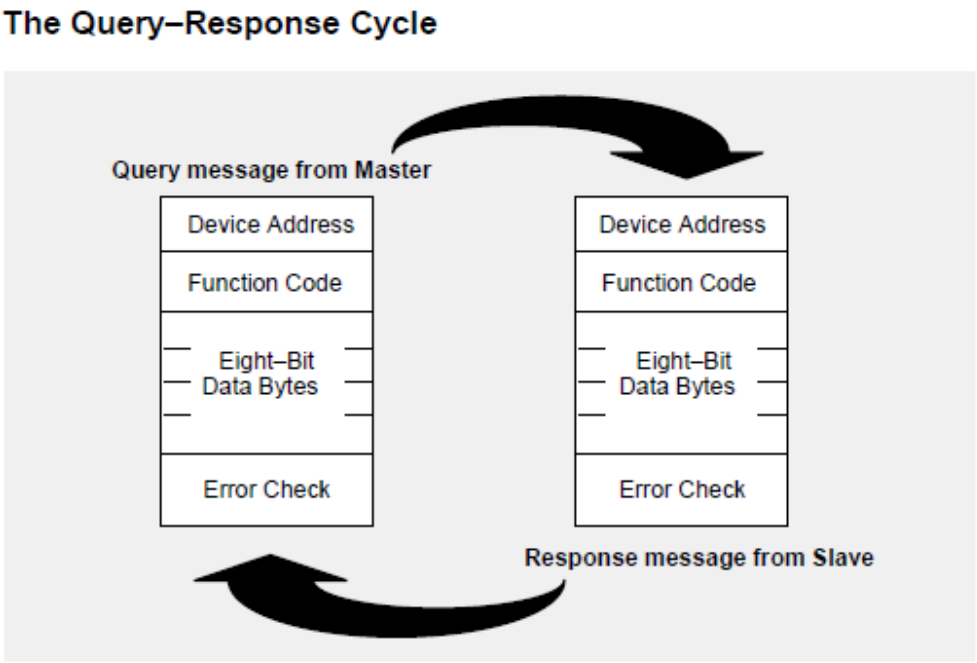
Vuonna 1979 Modicon kehitti Modbus-nimisen sarjaliikenneprotokollan Modiconin omien ohjelmoitavien logiikkalaitteiden väliseen tiedonsiirtoon. Modbus-protokollan käyttö on sittemmin levinnyt laajalle, ja sitä käytetään hyvin paljon erityisesti ohjelmoitavien logiikoiden kommunikaatiossa muiden laitteiden kanssa. (RS-232 – Wikipedia 2008.)

Modbus-verkossa laitteet kommunikoivat keskenään master-slave -tekniikalla niin, että yksi laite on määritetty masteriksi ja muut ovat slaveja. Master johtaa keskustelua lähettämällä slaveille kyselyitä. Kyselyt voidaan osoittaa joko vain yhdelle vastaanottajalle tai broadcast-tyylisesti kaikille vastaanottajille. Yksittäiselle vastaanottajalle osoitettuihin viesteihin vastaanottajat vastaavat, mutta kaikille lähetettäessä slave-laitteet eivät vastaa. (Modicon 1996, 13.)

Slaveja voi yhdessä Modbus-verkossa olla korkeintaan 247, sillä hyväksyttäviä arvoja slaven osoitteeksi ovat 1-247. Osoitetta nolla käytetään tiedon lähettämiseen masterilta jokaisella slavelle, jotka ovat verkkoon kytkettyinä. Vastatessaan kyselyihin slave kertoo myös oman osoitteensa paluuviestissä, jotta isäntälaite tietäisi, mikä slave vastaa kyselyihin. (Modicon 1996, 19.)

Masterin kyselyviestit muodostetaan asettamalla ensin vastaanottaja-slaven numero tai broadcast, sen jälkeen funktiokoodi, mahdollinen lähetettävä data ja virheentarkituskenttä. Muodostaessaan vastausviestiä slave vahvistaa toteutetun tehtävän asettamalla vastausviestiin funktiokoodiksi saman koodin, jota master pyysi tehtäväksi, sekä lisäämällä viestiin mahdolliset palautusarvot ja virheentarkistuskentän. Jos slave havaitsee ongelman virheentarkistuksessa tai ei

kykene toteuttamaan masterin pyyntöä, se palauttaa virheilmoituksen. Kuviosta kuusi näkyy kyselyn ja vastauksen kierto. Kehys on rakenteeltaan aina samanlainen riippumatta siitä, mitä tietoa siirretään tai kuka sitä lähettää. (Modicon 1996, 13.)



KUVIO 6. Kysely ja vastaus Modbus-protokollan mukaisesti (Modicon 1996, 14)

Funktiokoodi viestissä kertoo slavelle, mitä isäntälaite tahtoo sen tekevän. Datakentässä voidaan kuljettaa lisätietoa, kuten tallennettavaa dataa liittyen masterin sille antamaan tehtävään. Tehtävä voi olla esimerkiksi lukea laitteen muistista tietoa ja lähettää se masterille. Tällaisessa tehtävässä masterin täytyy lähettää slavelle myös tiedot siitä, mistä muistiosoitteesta alkaen tietoa luetaan ja montako rekisteriarvoa halutaan luettavan. (Modicon 1996, 14.)

Sarjamuotoisessa tiedonsiirrossa Modbus-protokollaa voidaan käyttää tiedonsiirtoon kahdessa eri formaatissa: ASCII ja RTU. Jokaisen laitteen Modbus-verkossa on käytettävä samaa tiedonsiirtomuotoa ja samoja sarjaporttiasetuksia, kuten siirtonopeutta ja pariteettitarkistusta. Sen lisäksi, että formaatit vaikuttavat

tiedon muotoon, ne vaikuttavat myös siihen, minkälaisessa muodossa jokainen tavu lähetetään ja minkälaista virheentarkistusta käytetään. (Modicon 1996, 15.)

4.2 ASCII

ASCII-muodossa siirrettävä tieto kulkee tavuina niin, että jokainen tavu lähetetään kahtena ASCII-merkkinä, jotka ovat heksadesimaalijärjestelmän lukuja. Heksadesimaalijärjestelmän lukuja ovat luvut nollasta yhdeksään sekä a, b, c, d, e ja f. ASCII-formaatin yksi eduista on se, että jokaisen merkin välillä saa olla jopa yhden sekunnin väli, ennen kuin viive lasketaan virhetilanteeksi. Jokainen tavu ASCII-muotoisessa Modbus-paketissa sisältää yhden aloitusbitin, seitsemän databittiä, yhden pariteetti bitin, jos pariteetti käytössä, sekä yhden pysäytysbitin, jos pariteetti käytössä, muutoin kaksi. (Modicon 1996, 15.)

Virheentarkistuksessa käytetään LRC-, eli Longitudinal Redundancy Check, tarkistusta. Alla, kuviossa 7, esimerkki LRC-tarkitussumman laskemisesta C-kielellä. Esimerkkifunktiossa on parametreinä auchMsg sekä usDataLen. Merkkijono-osoitin auchMsg on osoitin viestiin, jonka tarkistussumma funktion avulla lasketaan. Viestin pituus annetaan funktiolle usDataLen-muuttujassa, joka on etumerkitön kokonaisluku. (Modicon 1996, 15.)

Esimerkkifunktiossa sanoman jokaisen tavun arvo lisätään rivillä viisi alustettuun uchLRC-muuttujaan. Funktio palauttaa tavujen yhteenlasketun arvon etumerkittömässä char-muuttujassa muutettuna negatiiviseksi. (Modicon 1996, 15.)

```
01  static unsigned char LRC(auchMsg, usDataLen)
02  unsigned char *auchMsg ; /* message to calculate LRC upon */
03  unsigned short usDataLen ; /* quantity of bytes in message */
04  {
05      unsigned char uchLRC = 0 ; /* LRC char initialized */
06      while (usDataLen--) /* pass through message buffer */
07      uchLRC += *auchMsg++ ; /* add buffer byte without carry */
08      return ((unsigned char)(-((char)uchLRC))) ; /* return twos complement */
09  }
```

KUVIO 7. LRC-tarkistussumman laskeminen C++-kielellä (Modicon 1996, 117)

Tiedon lähettäminen Modbusin ASCII-muotoisessa tiedonsiirrossa aloitetaan lähettämällä :-merkki. Sen jälkeen sanomassa on yhden tavun kokoinen osoite, yhden tavun funktiokoodi, datakenttä, jos lähetettävässä sanomassa on dataa, ja tarkistustavu. Viimeisenä lähetetään 2 merkkiä, heksadesimaaliluvut 0x0D ja 0x0A, jotka ovat tarkoittavat telanpalautusta ja rivinvaihtoa. Kuvio 8 esittää oikein muodostettua Modbus ASCII-datakehystä. (Modicon 1996, 117.)

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	*n* CHARS	2 CHARS	2 CHARS CRLF

Figure 3 ASCII Message Frame

KUVIO 8. Modbus ASCII datakehys (Modicon 1996, 17)

4.3 RTU

RTU-muotoisessa tiedonsiirrossa data lähetetään kahdeksanbittisinä binäärimerkkeinä. Tämä nopeuttaa tiedonsiirtoa verrattuna ASCII-muotoon pienempään tilaan mahtuvan esitysmuotonsa takia. RTU-formaatilla muodostettu paketti koostuu osoitekentästä, funktiokoodista, mahdollisesta datakentästä sekä 16-bittisestä crc-tarkistussummasta. Kuvio 9 esittää oikein muodostettua RTU-pakettia. (Modicon 1996, 18.)

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1–T2–T3–T4

Figure 4 RTU Message Frame

KUVIO 9. Modbus RTU datakehys (Modicon 1996, 18)

```
01  unsigned int Plc::calcCRC(int len, unsigned char *buffer)
02  {
03    int crc=0xFFFF, LSB=0;
04    for(int i=0;i<len-1;i++)
05    {
06      crc=((crc^buffer[i]) | 0xFF00) & (crc | 0x00FF);
07      for(int u=0;u<8;u++)
08      {
09        LSB=(crc & 0x0001);
10        crc/=2;
11        if(LSB) crc^=0xA001;
12      }
13    }
14    return crc;
15  }
```

KUVIO 10. CRC-tarkistussumman laskeminen C++-kielellä (Camille Bauer AG 2003, 5.)

Kuviossa 10 esitetylle funktiolle annetaan parametreinä käsiteltävän viestin pituus sekä osoitin merkkijonoon, jossa on viesti, jonka tarkistussumma halutaan laskea. Jokainen tavu käsitellään funktiossa bitti kerrallaan. (Camille Bauer AG 2003, 5.)

4.4 Funktiokoodit

Modbus-protokollan mukaisesti master kertoo slavelle funktiokoodin avulla tehtävän, jonka haluaa sen toteuttavan. Funktiokoodi on yhden tavun kokoinen kokonaisluku välillä 1 – 255. Osa koodeista on etukäteen määritetty toimimaan samalla tavalla kaikissa Modiconin laitteissa, kun taas osa toimii vain tietyissä laitteissa tai on jätetty varatuksi tulevaisuuden varalle. (Modicon 1996, 19.)

Funktiokoodit voivat esimerkiksi käskeä slaven lukemaan jonkin muistiosoitteen tai tulon tila. Tällöin slave lukee halutun muistiosoitteen tai tulon tilan ja palauttaa sen arvon masterille data-kentässä. Tässä työssä on käytetty seuraavassa esitettyjä funktiokoodeja.

0x01: Master lukee binääriarvoja slaven digitaalisista lähdöistä. Kyselyssä määritellään mistä rekisteriosoitteesta tai lähdöstä lukeminen aloitetaan ja kuinka monta bittiä luetaan. Vastauksessa jokainen arvo on yhden bitin pituinen. Jos vastausviestissä jää tyhjiä bittejä, ne täytetään nollilla. Funktiota ei voida käyttää broadcast-osoituksella. (Modicon 1996, 24.)

0x02: Master lukee binääriarvoja slaven yhdestä tai useammasta digitaalisesta tulosta. Kyselyssä määritellään mistä rekisteriosoitteesta tai lähdöstä lukeminen aloitetaan ja kuinka monta bittiä luetaan. Vastauksessa jokainen arvo on yhden bitin pituinen. Jos vastausviestissä jää tyhjiä bittejä, ne täytetään nollilla. Funktiota ei voida käyttää broadcast-osoituksella. (Modicon 1996, 26.)

Kuvio 11 esittää masterin lähettämää kyselyä sekä slaven vastausta kyselyyn. Esimerkkikyselyssä slaven osoite on 0x11, ja funktion 0x02 avulla luetaan 22:n, eli heksadesimaalilukuna 0x16:n, binärilähdön tilat. Vastausviestissä data-kentän

pituus kolme tavua, koska 22 bitin lähettämiseen tarvitaan kolme kahdeksan bitin tavua. (Modicon 1996, 26.)

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	02
Starting Address Hi	00
Starting Address Lo	C4
No. of Points Hi	00
No. of Points Lo	16
Error Check (LRC or CRC)	—

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	02
Byte Count	03
Data (Inputs 10204–10197)	AC
Data (Inputs 10212–10205)	DB
Data (Inputs 10218–10213)	35
Error Check (LRC or CRC)	—

KUVIO 11. Funktion 0x02 kysely- ja vastausviestit (Modicon 1996, 27)

0x03: Luetaan slaven yhden tai useamman analogisen lähdön arvon. Kyselyssä määritellään, mistä lähdöstä lukeminen aloitetaan ja kuinka monta sanaa luetaan. Vastausviestissä jokaiselle luettavalle arvolle on varattu kaksi tavua tilaa. Jokaisessa rekisterissä ylimmät bitit ovat ensimmäisessä tavussa ja alimmat toisessa. Funktiota ei voida käyttää broadcast-osoituksella. (Modicon 1996, 28.)

0x04: Luetaan slaven analogisten tulojen tiloja. Kyselyssä määritellään, mistä rekisteriosoitteesta tai lähdöstä lukeminen aloitetaan ja kuinka monta sanaa luetaan. Yhdellä kyselyllä voidaan lukea yksi tai useampi arvo. Vastausviestissä jokaisen tulon tila ilmoitetaan kahdessa tavussa. Ylimmät tavut ovat ensimmäisessä ja alemmat toisessa tavussa. Funktiota ei voida käyttää broadcast-osoituksella. (Modicon 1996, 30.)

0x05: Asetetaan slaven digitaalinen lähtö datakentässä määritettyyn tilaan. Käsky pakottaa slaven käskyssä määritellyn digitaalisen lähdön annettuun tilaan huolimatta siitä, missä tilassa se on ennen käskyn vastaanottamista ollut. Käytettäessä broadcast-viestiä, joka lähetetään verkon jokaiselle slavelle, asetetaan lähtö jokaisessa laitetteessa käskyssä annettuun tilaan. Käskyllä voidaan myös asettaa binäärimuistien tiloja. Vastausviesti on sama kuin masterin lähettämä asetuskäsky. (Modicon 1996, 32.)

0x06: Master asettaa slaven analogisen lähdön arvon. Käsky pakottaa määritellyn lähdön arvon annettuun tilaan huolimatta siitä, missä tilassa se on ennen käskyn vastaanottamista ollut. Vastausviesti on sama kuin masterin lähettämä asetuskäsky. Käytettäessä broadcast-viestiä, joka lähetetään verkon jokaiselle slavelle, asetetaan lähtö jokaisessa laitetteessa käskyssä annettuun tilaan. (Modicon 1996, 34.)

0x0F: Master asettaa useita slaven digitaalisia lähtöjä. Käskyssä määritellään asetettavien lähtöjen lukumäärä, aloitusosoite sekä data-kentässä asetettavat tilat järjestyksessä. Data-kentän ensimmäisessä tavussa on vähiten merkitsevät bitit ja viimeisessä eniten merkitsevät. Vastausviestissä slaven numero, funktiokoodi, aloitusosoite sekä asetettujen lähtöjen tai muistien lukumäärä. (Modicon 1996, 44.)

Kuviossa 12 näkyy useita digitaalilähtöjä asettavan 0x0F -funktiokoodin esimerkki. Esimerkkifunktiossa slaven numero on 0x11, ja sille lähetetään käsky asettaa kymmenen lähtöä alkaen numerosta 0x13 niihin tiloihin, jotka datakentässä annetaan. Vastausviesti on muuten sama viesti kuin masterin lähettämä, mutta siitä on jätetty data-kenttä pois. (Modicon 1996, 45.)

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Lo	13
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Byte Count	02
Force Data Hi (Coils 27-20)	CD
Force Data Lo (Coils 29-28)	01
Error Check (LRC or CRC)	—

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Lo	13
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Error Check (LRC or CRC)	—

KUVIO 12. Funktiokoodin 0x0F kysely- ja vastausviestit (Modicon 1996, 45)

0x10: Asetetaan useita slaven analogisten lähtöjen arvoja. Käskyssä määritellään asetettavien lähtöjen lukumäärä, aloitusosoite sekä data-kentässä asetettavat tilat järjestyksessä. Jokaisen kirjotettavan arvon koko on kaksi tavua. Vastausviestissä ovat slaven numero, funktiokoodi, aloitusosoite sekä asetettujen lähtöjen lukumäärä. (Modicon 1996, 46.)

5 TOIMINTAYMPÄRISTÖ

5.1 Taloautomaation etähallintajärjestelmä

Käyttäen HL-Heat Oy:n lämmönsäätöjärjestelmiä, jokaiselle huoneelle voidaan säätää Celsius-asteen kymmenyksen tarkkuudella erikseen sekä lattia-, että kattolämmityksen asetusarvot. Järjestelmän avulla voidaan myös säätää ja valvoa sekä paikallisesti että etäkäyttöisesti muitakin talotekniikan laitteita, kuten ilmanvaihtokoneita ja lämpöpumppuja.

Lisäksi sekä paikallisesti, internetin välityksellä, että tekstiviestein voidaan ohjata esimerkiksi ovien lukituksia tai pihavalaistusta. Kiinteistön hälytykset, kuten palo-, murto-, ja sähkökatkohälytykset, voidaan ohjata omistajalle tai kiinteistön huoltoyhtiölle tekstiviestinä. Tämän lisäksi erityisesti mökeissä käytetään ominaisuutta, jonka avulla sähkölämmitystä voidaan ohjata tekstiviestillä niin, että mökki voidaan lämmittää valmiiksi, ennen kuin paikalle saapuu asukkaita.

Jotta kohteeseen voidaan asentaa HL-Heatin internetetähallintajärjestelmä, sinne täytyy saada internetyhteys ja HL-Heatin sähkökeskukseen asennettava minipalvelin, joka hoitaa tiedonsiirron niin, että käyttäjä voi ohjata kiinteistönsä automaatiolaitteistoa nettiselaimen välityksellä. Lämmitysten hallintaan voidaan käyttää usein kiinteistössä jo valmiiksi olevia lämmönsäätöjärjestelmiä, mutta HL-Heatin omat järjestelmät takaavat varmatoimisen yhteyden laitteiston eri osien välillä.

Laitteiston kytkennän jälkeen käyttäjä rekisteröidään HL-Heatin käyttäjä-tietokantaan, ja käyttäjälle annetaan salasana ja käyttäjätunnus sekä ohjeet järjestelmään kirjautumiseksi. Etäohjauksen käyttöliittymä on tehty mahdollisimman helpoksi ja suunniteltu mukailemaan paikalliskäyttöön

tarkoitetun ohjaimen käyttöliittymää, jotta sopeutuminen uuteen ohjelmistoon olisi mahdollisimman helppoa.

5.2 Laitteet

Työssä käytettävä laitekokonaisuus koostuu erilaisista talo- ja automaatiotekniikan laitteista, joista osa on uusinta tekniikkaa ja osa taas hieman vanhempia, mikrokontrolleripohjaisia laitteita, jotka ovat osoittaneet toimivuutensa vuosien varrella. Yhteistä kaikissa laitteissa on kuitenkin sarjamuotoinen tietoliikenne, joka käyttää joko RS-485-, tai RS-232-standardia.

Laitteiston tärkein elementti on lämmönsäätöjärjestelmän power output-, eli pop-kortti. Pop-kortti on järjestelmän vanhin elementti, mutta se on kokenut ohjelmisto- ja laitteistopäivityksiä vuosien varrella. Se sisältää Motorolan mikrokontrollerin, joka ohjaa siihen kytkettyjä laitteita säätöjärjestelmäohjelman käskyjen mukaan. Korttiin kytkettävät laitteet voivat olla esimerkiksi lämmityslaitteita tai valaistusohjaimia. Laitteessa on RS-485-sarjaväylä, johon voidaan kytkeä käytännössä rajattomasti lisää ohjauskortteja ja muita laitteita.

Säätöjärjestelmän pop-kortteihin voidaan kytkeä paikalliskäyttöä varten näppäimistölaitteita, joita voi olla useita, ja niillä voi olla esimerkiksi jokaisella oma tilansa ohjattavana. Näppäimistöön kuuluvat alumiinikoteloon pakattu numeronäppämistö valinta- ja peruutusnäppäimillä sekä LCD-näyttö. Yhdellä näppäimistöllä voidaan hallita enimmillään 32 lämmitystä ja kuutta ajastettua ohjausta. Laitteessa on RS485-liitäntä, jonka avulla se voidaan kytkeä samaan väylään muiden laitteiden kanssa.

Säätöjärjestelmä voi toimia yksistään vain pelkän pop-kortin avulla, mutta jotta sitä voisi ohjata joko paikallisesti tai etäkäyttönä, on tarpeellista liittää siihen joko edellä mainittu näppäimistö tai tietokone. Tietokoneen etuna on mahdollisuus ohjata laitteita internetin kautta, kun taas näppäimistö on vain paikalliseen käyttöön tarkoitettu.

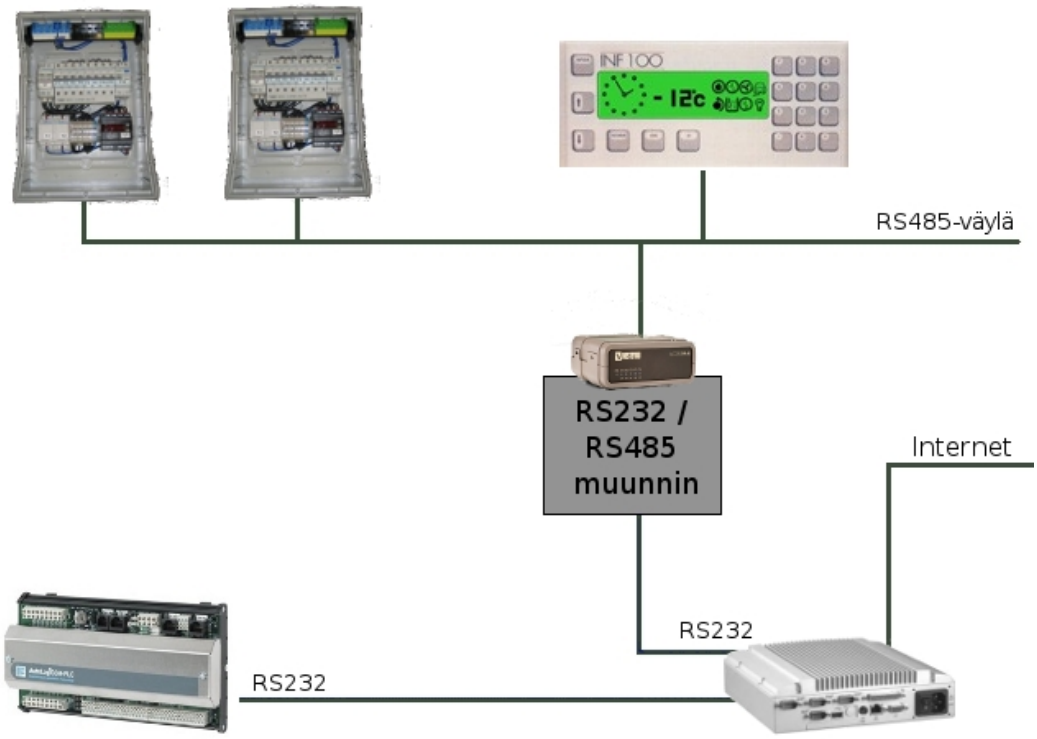
Jossain tapauksissa, kuten esimerkiksi erämökeillä, on vaikea saada järjestettyä riittävän nopeaa ja luotettavaa internetyhteyttä siten, että taloautomaation ohjaus voitaisiin rakentaa sen varaan. Kohteen ollessa kaukana on pelkästään paikallinen ohjauskin mahdoton ratkaisu. Tällaisissa tapauksissa kohdetta voidaan ohjata GSM-liittymän kautta esimerkiksi tekstiviestien avulla käyttäen gsm-yhteyttä tukevaa ohjelmoitavaa logiikkakontrolleria.

Logiikkaohjain voidaan liittää säätöjärjestelmään pc:n kautta tai sitä voidaan käyttää erillään muista laitteista omana kokonaisuutenaan. Myös logiikan toimintoja voidaan ohjata internetkäytön avulla tietokoneeseen liitettynä. Gsm-verkkoa ja internetiä hyödyntääkseen logiikka tarvitsee gsm-modeemin. Jotta logiikkaohjain voitaisiin liittää internetetähallintaohjelmistoon, se täytyy myös liittää tietokoneeseen, jossa on internetyhteys. Laitteessa itsessään on RS-232-sarjaliitäntä sekä laajennuskorttipaikka RS-232 - RS-485-muunninta varten, jos halutaan rakentaa useita logiikkamoduuleita käyttävä väylä.

Tietokoneohjausta varten tarvitaan tietokone sekä RS-232 - RS-485 -adapteri. Uusissa tietokoneissa, joissa ei välttämättä ole enää tavallista com-porttia, joudutaan käyttämään myös usb- tai pcmcia-porttiin liitettävää sarjaportti-muunninta, jota käytetään kuten tavallista sarjaporttia.

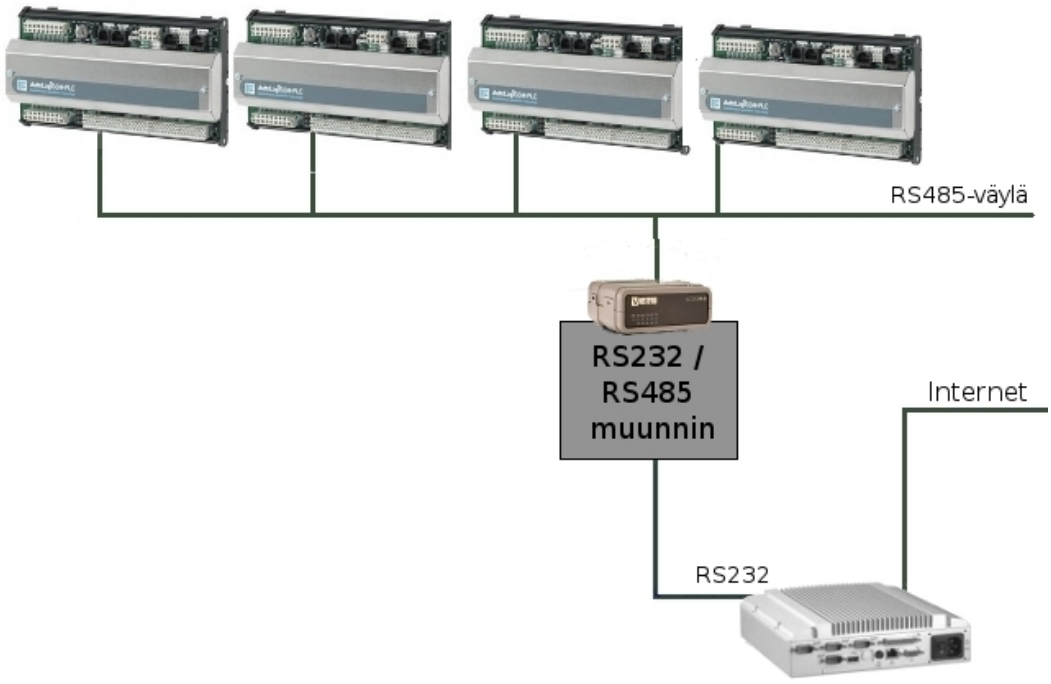
5.3 Kytkennät

Kuviossa 13 näkyy, kuinka säätöjärjestelmän komponentit kytketään toisiinsa RS-232- ja RS-485-protokollien mukaisten sarjaliikenneväylien avulla. Tietokoneen normaali COM-portti tukee kahden datajohtimen RS-232-standardia, eikä säätöjärjestelmän tarvitsemaa RS-485-standardia. Siksi sen ja säätöjärjestelmän dataväylän väliin tarvitsee kytkeä RS-232 - RS-485-muunnin, joka muuntaa dataliikenteen molempiin suuntiin oikeaan muotoon. Internetyhteys kytketään palvelin PC:hen.



KUVIO 13. Säätöjärjestelmän komponenttien kytkentä

Logiikka voidaan kytkeä suoraan tietokoneeseen, joka keskustelee sen kanssa Modbus-protokollan avulla sarjaportin kautta. Jotta logiikoita voitaisiin kytkeä enemmän järjestelmään, tarvitsee logiikoihin asentaa RS-485 -laajennuskortti ja tehdä muutoksia tietokoneen kytkentään. Tietokone tarvitsee siinä tapauksessa väliin RS-232 - RS-485 -muuntimen ja kytkentä ei tapahdu suoraan logiikkaan, vaan logiikoita yhdistävään väylään, kuten kuviossa 14 näkyy.



KUVIO 14. Useita logiikkayksiköitä kytkettynä tietokoneeseen

5.4 Ohjelmat

5.4.1 Yleistä

Ohjelmiston tärkein vaatimus oli olla internetin kautta mistä päin maailmaa tahansa hallittava kodintekniikan ohjauskeskus. Suunnittelun ja eri vaihtoehtojen vertailemisen jälkeen päädyttiin ratkaisuun, jossa asiakkaalle asennetaan palvelinkone sähkökeskuksen yhteyteen ja palvelimelle joko kiinteä ip-osoite tai dynaaminen dns-palvelu.

Ohjelmisto päätettiin toteuttaa kolmessa osassa: säätöjärjestelmäkomponentteja ohjaava komentorivisovellus, logiikkaohjainta ohjaava komentorivisovellus ja web-sovellus, joka käskyttää edellä mainittuja komentorivisovelluksia.

Komentorivisovellukset toteutettiin C++-ohjelmointikielellä ja web-sovellus käyttäen html-, php- ja css-tekniikoita ja Mysql-tietokantaa.

Web-käyttöliittymä kutsuu joko logiikan tai säätöjärjestelmän kanssa keskustelevan komentorivisovelluksen riippuen siitä, kumpaa tarvitaan käyttäjän pyytämän tehtävän suorittamiseksi.

Logiikan kanssa keskusteleva ohjelma tarkistaa, onko siitä olemassa jo tausta-ajossa instanssi vai ladataanko se. Tausta-ajossa ajettava, niin kutsuttu daemon-sovellus, keskustelee logiikan kanssa ja pitää logiikan tärkeimpiä muistipaikkoja valmiiksi päivitettynä ja ladattuna keskusmuistissa toiminnan nopeuttamiseksi. Säätöjärjestelmän kanssa keskusteleva ohjelma käyttää suoraan sarjaporttia.

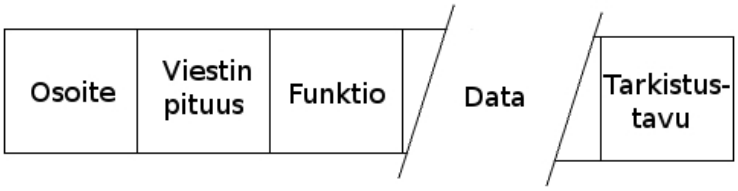
Käyttöliittymän avulla voidaan lukea arvoja, kuten lämpötilatietoja, digitaali- ja analogialähtöjen ja -tulojen asetuksia tai kellotauluja sekä asettaa lähtöjen tiloja ja muuttaa kellotaulujen tietoja.

5.4.2 Pop-ohjaus

Säätöjärjestelmän Pop-kortin ohjaus tietokoneella tapahtuu sarjaportin avulla. Sarjaporttiin kirjoitetaan viesti, joka sisältää ohjattavan laitteen numeron ja kertoo sitä kautta ohjattavalle laitteelle, mitä sen tahdotaan tekevän. Laite vastaa tähän viestiin ja siten voidaan varmistaa, että viesti meni perille. Jos laitteelta haetaan tietoa, niin vastaanotettava viesti on huomattavasti normaalia pidempi, sillä se sisältää kaiken kysytyn tiedon.

Lähetettävä ohjauskäsky muodostuu yhden tavun kokoisista paloista. Jokainen käsky sisältää vähintäänkin kohdelaitteen numeron, joka on kokonaislukuarvo välillä 0-255, funktion numeron, viestin pituuden tavuina sekä XOR-tarkistussumman, josta vähennetään heksadesimaaliarvo 80h. Viestin pituuden ilmaisevan tavun ja tarkistussumman sisältävän tavun välissä sijaitsee kaikki ne tiedot, joilla arvoja asetetaan tai luetaan. Tämän datalohkon koko on vähintään

yksi tavu ja korkeintaan 123 tavua. Kokoa rajoittaa viestin maksimikoko 127 tavua ja siitä vähennetään neljä pakollista tavua. Viestin kokonaispituus on 127 tavua siksi, että viestin pituutta ilmaiseva tavu lasketaan lisäämällä siihen heksadesimaaliarvo 0x80. Kuvio 15 esittää säätöjärjestelmän viestirakennetta.



KUVIO 15. Säätöjärjestelmän viestirakenne

Viestin muodostettuaan ohjelma lähettää viestin ja alkaa odottaa vastausta. Jos vastausta ei kuulu 200 ms kuluessa, ohjelma lopettaa odottamisen ja lähettää viestin uudelleen. Tätä toistetaan maksimissaan 20 kertaa, minkä jälkeen ohjelma luovuttaa ja palauttaa virheilmoituksen. Uudelleenyrityksiä pitää siksi tehdä niin monta, että jos samassa väylässä on useita laitteita, voivat viestit mennä vahingossa päällekäin, sillä esimerkiksi näppäimistöohjain kyselee laitteilta koko ajan niiden tilaa ja päivittää omia tietojaan.

Viestien sotkeutumista toisiinsa on pyritty vähentämään siten, että niitä ei lähetetä sarjaväylään heti pyynnön tultua, vaan ensin varmistetaan, että linjalla on hiljaista vähintään 10 ms ajan. Jos linjalla on ruuhkaa, odotetaan hetki ja yritetään uudelleen. Tällä tavalla sotketaan mahdollisimman vähän muiden laitteiden keskinäistä liikennettä ja saadaan mahdollisimman paljon oikeaa dataa omalle ohjelmalle.

Jokainen tietokoneelle saapunut viesti tarkistetaan lukemalla ensin viestin lähettäjä. Jos lähettäjä on se, jolta vastausta on odotettu, siirrytään tarkistussumman laskemiseen. XOR-tarkistussumma lasketaan niin, että otetaan ensimmäinen tavu ja lasketaan siihen jokainen tavu erikseen XOR-operaatiolla,

kunnes tulee viimeinen tavu, johon tätä arvoa verrataan. Lasketun arvon tulee olla heksadesimaalijärjestelmän 0x80, eli 128 kymmenjärjestemässä pienempi kuin viestin viimeinen tavu. Jos luku täsmää, voidaan olettaa, että viesti on oikea. Saatuja lämpötiloja-arvoja myös tarkistetaan ja varmistetaan, että niiden arvot ovat järkeviä. Myös tällä tavoin karsitaan mahdollisia virheitä.

Alla kuviossa 16 esitetyn funktion avulla tarkistetaan saapuneen viestin oikeellisuus. Funktio ei ota mitään tietoja argumentteina, vaan se lukee luokassaan privaattina jäsenenä olevan pbuffer-merkkijono-osoittimen sisältämää, juuri vastaanotettua, tietoa. Funktio palauttaa nollan, jos luetun tiedon puskurissa ensimmäisenä oleva vastaanotettu viesti pitää sisällään oikean tarkitussumman. Muussa tapauksessa palautusarvo on negatiivinen.

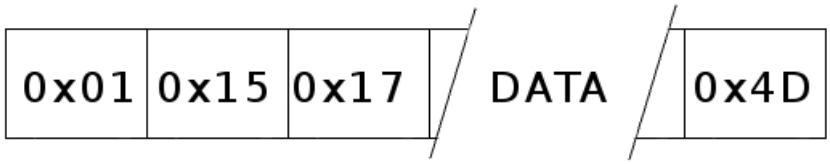
```
1  int Pop::verifyXOR(void) {
2      int ret=-1;
3      int xor=pbuffer[0];
4      for(int i=0;i<pbuffer[SIZE_BYTE]-0x81;i++) {
5          xor^=pbuffer[i];
6      }
7      xor+=0x80;
8      if(xor==pbuffer[SIZE_BYTE-0x81]) ret=0;
9      return ret;
10 }
```

KUVIO 16. XOR-tarkistussumman laskeminen C++-kielellä

Kun viesti on saapunut perille ja on varmistettu, että se on tullut ehjänä perille, sen sisältö täytyy analysoida. Jos lähetetty viesti on ollut asetuskäsky, takaisin tuleva viesti kertoo, että lähetetty viesti on otettu vastaan ja että sen sisältämä tieto on ollut hyväksyttävää. Jos taas viesti on ollut pyyntö saada dataa säätöjärjestelmältä, luetaan saapuneesta viestistä data, muutetaan se oikeaan muotoon ja palautetaan ohjelmaa kutsuneelle web-sovellukselle.

Asetuskäskyn vastaus sisältää laitteen numeron, viestin pituuden, asetusfunktion numeron sekä tarkistussumman. Kyselyn paluuviesti on muuten samanlainen, mutta asetusfunktion numeron jälkeen tulee data-kenttä ja sen jälkeen vasta

tarkistussumma. Kuviossa 17 on esitetty malli säätöjärjestelmän vastausviestin rakenteesta.



KUVIO 17. Esimerkki säätöjärjestelmän vastausviestistä

Kun web-käyttöliittymä pyytää jotakin tietoa, se tietää montako lukua komentoriviohjelman kuuluu palauttaa. Tämä tieto on esiohjelmoitu, sillä tietomäärä jokaisella kysely- ja asetustyypillä on vakio. Jos tietoa on jostakin syystä vähemmän saatavilla, kuin mitä web-sovellus odottaa, komentoriviohjelma palauttaa nollia merkkijonon seassa niiden arvojen kohdalla, jotka jäävät puuttumaan. Tällaista tilannetta ei kuitenkaan pitäisi päästä tapahtumaan.

Paluuarvo tulee merkkijonona, jossa kaikki arvot on erotettu toisistaan puolipisteillä. Jos web-sovelluksen kautta halutaan asettaa jotakin arvoja, komentoriviohjelman palautusarvona on teksti ”OK”, jos asetus onnistui.

Virhetilanteessa paluurivi on tyhjä, jos kyseessä on arvon asetus. Jos säätöjärjestelmään ei saada minkäänlaista yhteyttä, vastauksena on ”Virhe laiteyhteydessä”. Kysyttäessä järjestelmältä arvoja, kuten lämpötiloja, ohjelma tulostaa paluuviestiin puolipisteiden väliin, johon normaalisti tulostetaan tietoja järjestelmästä, kuten lämpötiloja, ilmoituksen virheestä, jolloin se tulostuu myös käyttöliittymälle, kuten kuviossa 18 näkyy.

KUVIO 18. Virheilmoitus näkyvissä käyttöliittymässä

5.4.3 Logiikkaohjaus

Kuten säätöjärjestelmän tietokoneohjaus, myös ohjelmoitavaa logiikkaohjainta ohjataan sarjaportin kautta. Tiedonsiirtoon käytetään Modbus-protokollaa, joka helpottaa datan siirtämistä, sillä tieto on aina järjestetty viesteihin samalla tavalla. Tietokoneen kanssa samaan väylään kytketty logiikkaohjain toimii masterina ja tietokone slavena. Jokaiseen lähetettyyn viestiin tulee protokollan mukainen vastaus. Logiikka-ohjainten väliseen RS485 Modbus-väylään asetetaan yksi master-ohjain, joka rytmittää datakyselyjä niin, ettei tiedonsiirtoon tule ongelmia päällekkäisten datakyselyjen takia. Kun tietokone kytketään modbus-väylään, se asetetaan myöskin kuuntelemaan masterin käskyjä, ja sille annetaan numero, jolle lähetettyihin viesteihin se vastaa.

Tietokoneohjelmistoa ei ole ohjelmoitu käsittelemään broadcast-viestejä, jotka master lähettää kaikille verkossa oleville laitteille. Tämä siksi, että kyseisellä ohjelmistolla varustettuja tietokoneita on tarkoitus kytkeä verkkoon vain yksi

kerrallaan, eikä kannata reagoida logiikkaohjaimille tai virtamittareille tarkoitettuihin viesteihin.

Koska master käskee mitä kysytään, pitää sekä tietokoneohjelman että logiikkaohjaimen tietää, mitä tietoja mahdollisesti halutaan asettaa sekä mitä tietoja saatetaan haluta lukea. Tätä varten logiikkaohjaimeen tehdään esiasetetut kyselyt ja asetukset, joita se lähettää jokaisella lukukierroksella huolimatta siitä, kuunteleeko tietokone sitä. Jos tietokone ei vastaa sen kyselyihin, mitään asetuksia ei tehdä. Kuviossa 19 on katkelma ohjelman lähdekoodin kohdasta jossa tarkistetaan, minkälainen tieto puskurissa on seuraavaksi.

```
int Plc::Read(int func) {
  rbytes = read(fd, buffer, BUFLEN);
  if(!checkAddr() && !checkCRC()) {
    if(func==READ_BINARY) {
        unsigned char s_buffer[]={
            address,                // addr
            READ_BINARY             // function
            0x01,                   // bytes
            rb,                     // data
            0x00,                   // CRC hi
            0x00                    // CRC low
        };
        s_buffer[4]=0x00FF&calcCRC(5, s_buffer);
        s_buffer[5]=calcCRC(5, s_buffer)>>8;

        tbytes=write(fd, s_buffer, 6);
    }

    else if(func==WRITE_BINARY) {
        unsigned char s_buffer[]={
            address,                // addr
            WRITE_BINARY,           // function
            buffer[2],              // reg hi
            buffer[3],              // reg low
            buffer[4],              // regs hi  (number of registers)
            buffer[5],              // regs low (offset)
            0x00,                   // CRC hi
            0x00                    // CRC low
        };
        s_buffer[6]=0x00FF&calcCRC(7, s_buffer);
        s_buffer[7]=calcCRC(7, s_buffer)>>8;

        tbytes=write(fd, s_buffer, 8);
    }
```

KUVIO 19. Modbus-viestien käsittely

Vastauskoodi muodostetaan siis tietyn kaavan mukaan käyttäen mahdollisimman paljon master-laitteelta saatuja tietoja. Viestin käsittelyssä ja vastauksen muodostamisessa kuluu sen verran aikaa, ettei erityistä viestin loppumisen ja alkamisen merkiksi tehtävää taukoa kannata järjestää.

Yllä olevassa koodissa binäärilähtöjen ja -tulojen lisäksi eritellään myös analogisten lähtöjen ja tulojen tilojen asettamiset ja lukemiset. Tämä toimii täysin samalla tavalla kuin binääristen arvojen kanssa.

Yhdessä väylässä voi olla useita laitteita, joille jokaiselle voi olla useita kyselyjä ja sen takia Modbus-kyselyjen lista saattaa paisua satoihin kyselyihin per kierros. Tällaisessa tapauksessa kyselyaika saattaa muuttua pitkäksi ja säätöohjelmiston käyttäminen todella hitaaksi.

Hitauden välttämiseksi ohjelma luo itsestään kopioksi toisen prosessin, joka jää järjestelmään daemonina esilukemaan kymmenen minuutin ajaksi tietoja sarjaväylästä. Tämän avulla lukuoperaatiot nopeutuvat, kun luettavat tiedot ovat jo keskusmuistissa. Kirjoitettava data annetaan prosessille komentoriviohjelman kautta. Kun tulee kyseisen datan kirjoitusvuoro, ohjelma kirjoittaa datat logiikkaohjaimelle sarjaväylään ja varmistaa kirjoituksen onnistumisen jälkeenpäin lukuoperaation avulla. Kuviossa 20 on koodiesimerkki daemonin luomisesta.

Esimerkkifunktiossa luodaan fork()-funktion avulla uusi prosessi. Prosessille aloitetaan uusi istunto, ja sen käsketään olemaan välittämättä prosessin lopettamiseen johtavista signaaleista. Daemoniksi muunnettu prosessi jakaa logiikan tietoja sisältävän struktuurin sitä kutsuneen ohjelman sekä muiden siltä tietoja hakevien prosessien kanssa jaetulla muistialueella. Muistialue alustetaan luomisen jälkeen memset()-funktion avulla arvoon -1, jotta voidaan helposti selvittää, mitä arvoja ei ole saatu vielä luettua logiikkaohjaimelta.

Ennen funktion kutsua määritellään jaettava muistialue sekä tarkistetaan mahdollisesti jo olemassaolevan daemon-prosessin olemassaolo. Jos daemon on

jo ladattu palvelimen muistiin, kutsutaan funktiota, joka lukee jaetusta muistista kysytyn alueen ja palauttaa sen sisällön.

Prosessi asetetaan lukemaan logiikkaohjaimelta saatavia käskyjä ja kyselyitä sekä vastaamaan niihin rivillä 21 käytetyn Read()-metodin avulla. Metodi lukee logiikalta käskyn tai kyselyn ja muodostaa siihen sopivan vastausviestin. Parametrinä kyseiselle metodille annetaan logiikan tahtomat tiedot sisältävän struktuurin osoitin.

```
1  int start_server() {
2      int lfp, i;
3      if(!fork()) {
4          mode=0;
5          setsid();
6          for (i=getdtablesize();i>=0;--i) close(i);
7          signal(SIGCHLD,SIG_IGN);
8          signal(SIGTSTP,SIG_IGN);
9
10         if ((plcdata = (plc_data *)shmat(shmid, NULL, 0))<0) {
11             perror("shmat");
12             return 1;
13         }
14         memset(plcdata, -1, sizeof(plc_data));
15         time(&plcdata->update);
16         Plc plc;
17         time_t start, end;
18         time(&start);
19         for(int i=0;;i++) {
20             plcdata->WM[0]=i;
21             plc.Read(plcdata);
22             time(&end);
23             if(difftime(end, start)>(float)TIME) break;
24         }
25         usleep(200);
26         plcdata->readyflg=0;
27         shmctl(shmid, IPC_RMID, NULL);
28     }
29     else readwrite();
30     return 0;
31 }
```

KUVIO 20. Taustalla ajettavan daemon-prosessin käynnistys

Tiedot siitä, mitä ohjelman kuuluu tehdä, annetaan ohjelmalle komentoriviparametreinä. Tehtävänä voi olla esimerkiksi tulostaa logiikan

lähettämiä digitaalisten ulostulojen tiloja. Käskyjä voi antaa maksimissaan yhden jokaista ajokertaa kohden. Paluuarvot tulevat merkkijono-muotoisena tulosteena, jossa mahdollinen taulukoitu data on eroteltu alkio kerrallaan puolipisteellä toisistaan.

Jos ohjelman daemon-osa on jo valmiiksi ladattu muistiin, ohjelma palauttaa tiedot välittömästi muististansa, mutta muussa tapauksessa ohjelma lataa daemonin muistiin ja palauttaa kysytyt tiedot vasta, kun se on saanut ne luettua logiikalta.

5.4.4 Sarjaporttitunnistus

Sekä logiikkaohjainten että säätöjärjestelmäkorttien ohjaukseen tarkoitetut ohjelmat käyttävät sarjaporttia laitteiden kanssa kommunikointiin. Koska uudemmissa tietokoneissa on harvemmin com-porttia RS-232 -sarjaliikennettä varten, joudutaan käyttämään usb-sarjaporttimuuntimia. Tällöin ei voida etukäteen tietää, mihin porttiin käyttöjärjestelmä minkäkin sarjaporttilaitteen liittää.

Ongelman ratkaisemiseksi tehtiin myös kolmas ohjelma, jonka tehtävänä on lähettää tietty viesti, johon odotetaan säätöjärjestelmän vastaavan ensin kaikkiin usb-sarjaportteihin. Mahdolliset vastaukset tarkistestaan, jotta tiedetään vastauksen tulleen juuri säätöjärjestelmältä. Jokaisesta oikeasta vastauksesta laitetaan merkintä asetustiedostoon, jotta ohjelmat tietäisivät, mistä porteista löytyy säätöjärjestelmälaitteistoa.

Kun kaikki säätöjärjestelmät on löydetty, käydään sama läpi niin, että niitä portteja, joista ei löytynyt säätöjärjestelmää, kuunnellaan viisi sekuntia. Näin löydetään ne sarjaportit, jotka on kytketty sellaiseen väylään, jossa on modbus-master. Jos portista tulee jokin modbus-viesti, kirjataan portti ylös asetustiedostoon. Myös modbus-viestit tarkistetaan, jotta tiedetään kyseessä olevan oikeasti modbus-protokollan mukainen viesti.

Käynnistyessään molemmat ohjelmat lukevat asetustiedoston ja lukevat sieltä, mihin porttiin tarvittava laite on kytketty. Asetustiedosto päivitetään tietyin väliajoin ajastetulla toiminnolla sekä jokaisen uudelleenkäynnistyksen yhteydessä. Porttien numerot eivät muutu, jos palvelinta ei käynnistetä uudelleen tai sen johtoja irroitella. Tätä ei kuitenkaan pitäisi normaalikäytössä tapahtua.

5.5 Tietoturva

Kun tietokoneet ovat koko ajan käynnissä ja kytkettyinä internetiin, on tärkeää pitää huoli tietoturvasta. Etähallintapalvelimen tietoturvaa on parannettu asentamalla siihen vain tarvittavat ohjelmistot sekä sulkemalla oletuksena kaikki portit ja avaamalla vain välttämättömät. Myös kaikki välttämätön tietoliikenne on siirretty sellaisiin portteihin, joihin niitä ei oletusarvoisesti aseteta. Palvelin on asetettu käyttämään automaattisia tietoturvapäivityksiä, jotka se noutaa käyttöjärjestelmän jakelijalta käyttäen internetyhteyttä.

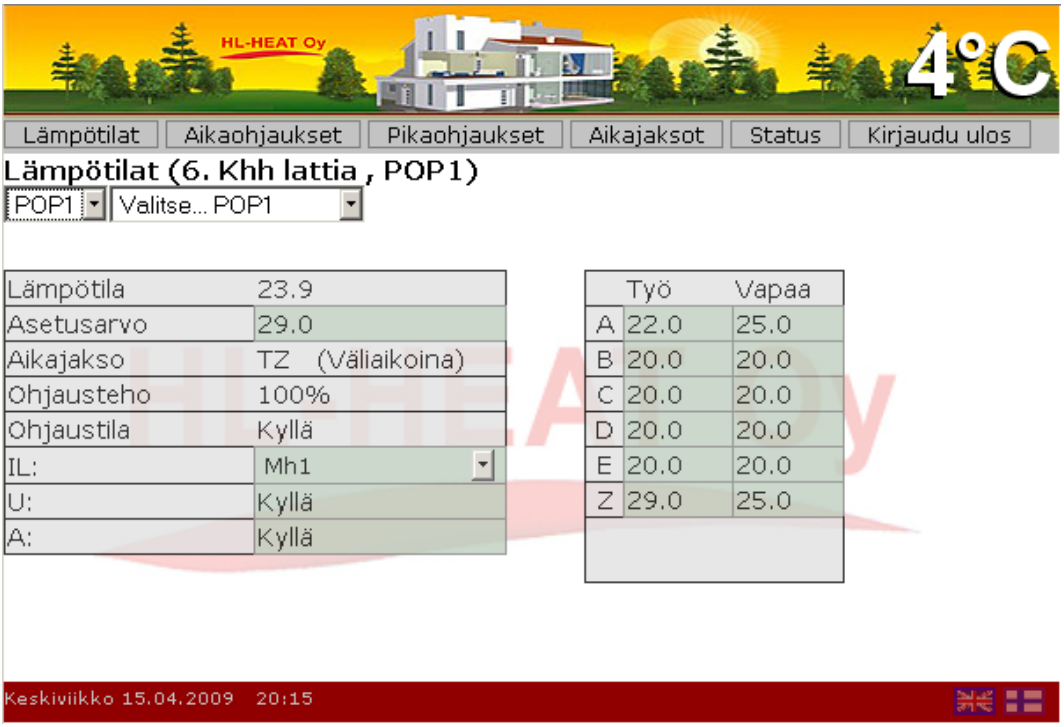
Palomuuriohjelmistona palvelimessa toimii Iptables, jonka pääasiallinen tehtävä on estää kaikkien paitsi haluttujen yhteyksien luominen palvelimeen. Päästäkseen sisään etähallintajärjestelmään, täytyy myös suorittaa sisäänkirjautuminen. Tämä tapahtuu HL-Heatin antamalla käyttäjänimellä ja salasanalla. Käyttäjä voi käyttöliittymän kautta itse muuttaa salasanoja, lisätä käyttäjiä sekä määrittää salasanat näille käyttäjille.

Salasanat on tallennettu tietokantaan tiivistesummina. Sisäänkirjautumisen yhteydessä salasanaan yhdistetään palvelimen lähettämän satunnaisen merkkijonon tiivistesumma. Varsinaista salasanaa tai pelkän salasanan tiivistettä ei koskaan kuljeteta verkon yli. Selaimen päässä tiivistesumma lasketaan aina ennen lähetystä javascript-funktion avulla.

Sisäänkirjautumisen jälkeen asiakkaan tila tarkistetaan php:n istuntojen avulla. Asiakas aloittaa istunnon aina kirjautuessaan sisään. Jokaisen sivulatauksen yhteydessä tarkistetaan, että asiakas on kirjautunut sisään ja istunto on aktiivinen .

Jos tulee epäselviä tilanteita tai asiakas yrittää ladata jotakin tietoa ilman sisäänkirjautumista, yhteys katkaistaan ennen kuin mitään tietoa lähetetään. Myös vastaanotettavan tiedon yhteydessä tarkastetaan, että tietoa lähettävä asiakas on kirjautunut sisään palveluun.

Istunnon voi lopettaa kirjautumalla palvelusta ulos tai lopettamalla sivuston käytön. Selaimen sulkeminen ei varsinaisesti lopeta istuntoa, mutta jos istunto on tarpeeksi kauan käyttämättä, sitä ei voida enää käyttää, vaan on kirjauduttava uudestaan sisään palveluun. Kuviossa 21 on esitetty käyttöliittymän näkymä lämpötila-asetus valikosta.



KUVIO 21. Käyttöliittymän lämpötila-asetusvalikko

6 YHTEENVETO

Työn tavoitteena oli tehdä olemassaolevaan taloautomaatiojärjestelmään mahdollisuus etähallintaan. Suunnitelmissa päädyttiin sähkökeskukseen asennettavaan erilliseen Linux-minipalvelimeen, jossa on sisälle asennettuna web-palvelin.

Tavoite saavutettiin, ja järjestelmä on todettu toimivaksi ratkaisuksi. Kehitystyössä tuli vain vähän ongelmatilanteita. Heti projektin alussa syntynyt ongelmatilanne oli ensimmäisenä käytetyn usb-sarjaporttimuuntimen yhteensopimattomuus laitteiston kanssa. Ongelma ratkaistiin vaihtamalla yhteensopimaton komponentti sellaiseen, joka sopi käytettyyn laitteistoon.

Toinen ongelma tuli vastaan tietokoneeseen kytkettävien sarjaporttimuuntimien käytössä. Jos tietokoneeseen liitetään useampi kuin yksi sarjaporttimuunnin ja tietokone käynnistetään uudelleen tai muuntimet irroitetaan ja kytketään uudelleen, niiden numerot järjestelmässä saattavat muuttua. Tämän takia ohjelman asetustiedosto pitäisi joka kerta muuttaa vastaamaan uutta tilannetta. Ongelma ratkaistiin kolmannella ohjelmalla, joka tunnistaa sarjaportteihin liitetyt laitteet ja päivittää asetustiedoston aina käynnistyksen yhteydessä sekä päivittäin ajoitetusti varmuuden vuoksi.

Jos järjestelmä pitäisi kehittää nyt alusta uudestaan, siihen varmastikin tulisi muutoksia sisäänkirjautumisen osalta. Nyt sisäänkirjautuminen tehdään jokaiselle palvelimelle erikseen, mutta olisi luultavasti järkevämpää rakentaa keskitetty kirjautumissivusto, jonka kautta tieto aina siirrettäisiin. Tämän avulla saataisiin tietoturvaa parannettua.

Käyttäjät ovat olleet tuotteeseen tyytyväisiä ja ohjelman kehitystä jatketaan tällä hetkellä ja tulevaisuudessa muun muassa laajentaen ohjelman toimintoja sekä kehittämällä siihen uusia ominaisuuksia.

LÄHTEET

An, P. 1998. PC interfacing: practical guide to Centronic, RS232 and game ports. Newnes.

Asynchronous serial communication overview. [Viitattu 5.4.2009]. Saatavissa: http://www.quatech.com/support/comm-over-asyncserial.php.

Stallings, W. 1997. Data and computer communications. 5. Painos. Prentice-Hall.

ARC Electronics. RS232 Tutorial on Data Interface and Cables. [Viitattu 5.4.2009]. Saatavissa: http://www.arcelect.com/rs232.htm.

Maxim 2003. Determining Clock Accuracy Requirements for UART Communications. Maxim.

Strangio C. The RS232 Standard. [Viitattu 5.4.2009]. Saatavissa: http://www.camiresearch.com/Data_Com_Basics/RS232_standard.htm.

2008. RS-232 – Wikipedia. Wikipedia, Vapaa tietosanakirja [viitattu 5.4.2009]. Saatavissa: http://fi.wikipedia.org/wiki/RS-232.

Modicon, Inc 1996. Modicon Modbus Protocol Reference Guide. Modicon.

Camille Bauer AG 2003. MODBUS interface definition DME401 / 440. Camille Bauer AG.

FF-Automation OY 2005. SMS Programming Protocol for GSM-20, V1.49. FF-Automation.

Bies, L. 2008a. RS232 serial cable pinout information. [Viitattu 5.4.2009]. Saatavissa: http://www.lammertbies.nl/comm/cable/RS-232.html.

Bies, L. 2008b. RS485, specifications and in depth tutorial. [Viitattu 5.4.2009]. Saatavissa: http://www.lammertbies.nl/comm/cable/RS-485.html.

2008. RS-485 – Wikipedia. Wikipedia, Vapaa tietosanakirja [viitattu 5.4.2009]. Saatavissa: http://fi.wikipedia.org/wiki/RS-485.

Soltero, M., Zhang J., & Cockrill C. 2002. 422 and 485 Standards Overview and System Configurations. Texas Instruments.

Smethurst, S. 2007. What is RS485, EIA-485. [Viitattu 5.4.2009]. Saatavissa: http://www.chipkin.com/articles/what-is-rs485-eia-485.

2009. EIA-485 – Wikipedia. Wikipedia, Vapaa tietosanakirja [viitattu 5.4.2009]. Saatavissa: http://en.wikipedia.org/wiki/EIA-485.

2009. Data transmission – Wikipedia. Wikipedia, Vapaa tietosanakirja [viitattu 5.4.2009]. Saatavissa: http://en.wikipedia.org/wiki/Data_transmission.