KEMI-TORNION AMMATTIKORKEAKOULU

Modbus-mediamuunnin itsenäisellä logiikalla

Vuolasaho Hannu

Tietotekniikan koulutusohjelman opinnäytetyö
Ohjelmistotekniikka
Insinööri(AMK)
KEMI 2012

TIIVISTELMÄ

Kemi-Tornion ammattikorkeakoulu, Tekniikan ala	
Koulutusohjelma	Tietotekniikka
Opinnäytetyön tekijä	Hannu Vuolasaho
Opinnäytetyön nimi	Modbus -medianmuunnin itsenäisellä logiikalla
Työnlaji	Opinnäytetyö
Päiväys	18.5.2012
sivumäärä	44 + 42 liitesivua
Opinnäytetyön ohjaaja	DI Tapani Ruokanen, yliopettaja
Yritys	DD-control Oy
Yrityksen yhteyshenkilö/valvoja	Sähköinsinööri Kari Mikkilä Toimitusjohtaja Keijo Tuominen

Tavoitteena työssä oli kehittää ethernetin kautta hallittava ohjelmoitava logiikka Digi Ineternationalin Connect EM sulautetulle järjestelmälle. Tällä ohjelmoitavalla logiikalla tulisi olla myös toisena perustoimintona median muunto TCP:n päällä kulkevasta Modbus-liikenteestä sarjaliikenteelliseen. Modbus-kommunikointiprotokolla on määritelty TCP:n, normaalin EIA-232 ja balansoidun EIA-485 sarjaliikenteen yli kulkevaksi.

Tässä työssä median muunto tapahtuu yksisuuntaisesti. Modbus-isäntä voi lukea tietoa sarjaliikenteen puolella olevasta orjalaitteesta, mutta sarjaliikenteen puolelta TCP:n puolelle luku ei onnistu.

Mikäli medianmuuntoa ei käytetä, käyttäjä voi selaimella asettaa sisäisen logiikan ja luoda loogisia säätöketjuja. Tarjolla olevia operaatioita ei jo nyt työn suuren laajuuden vuoksi tehty suurta määrää, vaan minimalistisesti on tarjottu yksinkertaisia funktioita, joilla voi luoda lineaarisia säätöjärjestelmiä.

Asiasanat: ohjelmoitavat logiikat, sulautetut järjestelmät, Modbus.

ABSTRACT

Kemi-Tornio University of Applied Sciences, Technology

Degree Programme	Information technology
Name	Hannu Vuolasaho
Title	Modbus Medium Converter with Independent Programmable Logic
Type of Study	Bachelor's Thesis
Date	18 May 2012
Pages	44 + 42 appendixes
Instructor	Tapani Ruokanen, MSc (Tech.)
Company	DD-control Oy
Contact person/Supervisor from Company	Kari Mikkilä, BEng Keijo Tuominen, MD

In this project Modbus medium converter from Modbus-TCP to Modbus-RTU was developed. Another aspect in this project was to develop programmable logic with a web user interface. Medium conversion is unidirectional from ethernet Modbus-TCP master to Modbus-RTU slave serial port. Modbus-RTU is usually used with EA-232, EA-485 or a similar serial communication interface. Internal programmable logic can only access Modbus-RTU slaves through serial interface.

Modbus medium conversion can be executed while internal programmable logic is running or both can be used separately. This can be set from the web user interface.

This report covers design and implementation process of software for embedded Connect EM device from Digi International. Most user related tasks are documented in the user's manual so report covers technical aspects mostly.

Keywords: programmable logic, embedded systems, modbus.

SISÄLLYSLUETTELO

1. JOHDANTO

Työ aloitettiin syksyllä 2009. Työn tavoitteena oli luoda kevyt logiikka melko rajoittunein ominaisuuksin ja saada siten edullinen järjestelmä. Tärkeimmät ominaisuudet olisivat itsenäisenä logiikkana toimiminen ja median muunto ethernetistä Modbus-TCP-protokollasta sarjaväylään Modbus-RTU-protokollaan. Myös järjestelmän asetusten luonti olisi tapahduttava selaimen kautta.

Nykytilanteessa jokainen Modbus-RTU orjayksikkö viedään omalla johdotuksella keskuslogiikalle käyttäen sarjakaapelia. Osa laitteista voidaan kytkeä EA-485 yhteydellä väyläksi, mutta joka tapauksessa oma kallis ja pitkä johdotus on tarpeen.

Tarve tälle työlle oli siis saada edullinen itsenäinen logiikka ja vähentää kustannuksia. Laitteita, joissa olisi liityntä ethernetiin ja sarjaliikenteelle on olemassa, mutta ne ovat tuotevalikoimien yläpäissä. Siellä ollessa niissä on myös monia muita ominaisuuksia, joita ei tämän kaltaisessa ympäristössä tarvita. Tarve oli saada hajautettua logiikkaa ja laajentaa Modbus-verkkoa. Ethernet mediana tarjosi mahdollisuuden tähtitopologiaan ja samalla johdotuksen tarve vähenisi, koska voitaisiin käyttää mahdollisesti olemassa olevaa verkkoa. Vain orjalaitteet tuotaisiin mediamuuntimelle sarjakaapeleilla ja siitä jatkettaisiin varsinaiselle isännälle ethernetiä pitkin.

Automaatiojärjestelmien tiedonsiirtoon on monia tapoja. Tässä työssä oli käytössä Modbus-protokolla, joka on määritelty sarjaliikenteelle ja ethernetille. Kuitenkaan Modbus-ASCII-protollaa ei tueta, koska se on vanhentunut, tehoton ja melkein kadonnut maailmasta. Lisäksi Modbus protokollana on varsin yleinen ja sen rajoitettu toteuttaminen on melko helppoa.

Minulle annettiin vapaat kädet suunnitella ja toteuttaa järjestelmä oman mieleni mukaan. Vapaushan korreloi yleensä vastuun kanssa ja vastuu taas työmäärän. Suunnittelu ja suunnitelman korjaus sekä testaus osoittautuivat suurimmaksi työmääräksi. Itse toteutus oli melko nopeaa ja suoraviivaista. Virheiden etsintä, analysointi ja sen seurauksena palaaminen usein takaisin suunnitteluun hidastivat tilannetta.

Tässä raportissa keskitytään enemmän järjestelmän toimintaan ja rakenteeseen kuin järjestelmän toteuttavaan lähdekoodiin. Lisäksi jotkin järjestelmään valitut tietorakenteet ja algoritmit eivät ole optimaalisia ja kehitystä on jatkettava järjestelmän suorituskyvyn parantami-

seksi, joten niiden esittely olisi järjestelmän toiminnan kannalta epäoleellista. Laitteen käyttöäkään ei erityisemmin tässä raportissa käsitellä, koska sitä käsitellään tämän raportin liitteenä 1 olevassa käyttöohjeessa. Kyseinen dokumentaatio on tarkoitettu käyttäjälle ja peilaa laitteen toimintaa käyttäjän näkökulmasta, mutta on hyvää luettavaa myös henkilölle, joka haluaa tutustua laitteen sisäiseen toimintaan.

2. TAVOITE

Jo johdannossa mainitut itsenäisenä ohjelmoitavana logiikkana toimiminen ja median muunnon lisäksi järjestelmällä oli annettuna muitakin tavoitteita. Kaikki nämä yhdessä loivat järjestelmäsuunnittelun haastavaksi.

Median muunnolle ja itsenäisenä logiikkana toimimiselle on selkeä tarve. Ne mahdollistavat orjalaitteiden ohjaamisen kauempaa ja luovat siten hallittavuutta erilaisiin automaatioympäristöihin.

Asettamalla työssä käsiteltävä laite lähelle orjia, voidaan vähentää oman kaapelin asennuksen tarvetta. Ethernet-verkko tarvitaan laitteelle asetusten syöttämistä ja mahdollista median muuntoa varten. Kuvassa 1 kallein kaapelointi on työssä käsiteltävän valkoisen laitteen ja orjien välinen kaapelointi. Väylänä EA-485 tarvitsee erityisen impedanssisovitetun passiivisen haaroittimen, aktiivisen väylää haaroittavan jakajan tai väylä joudutaan kierrättämään jokaiselle orjalle asti ja takaisin. Ethernet-kaapelointi voidaan toteuttaa yhdellä edullisella kaapelilla ethernet-kytkimeltä laitteelle. Etäisyys ohjattaviin orjiin voidaan kasvattaa todella pitkäksi tällä laitteella. Ethernet-verkko ja Internet tarjoavat mahdollisuuden ohjata laitetta vaikka toiselta puolelta maapalloa. EA-485 väylän pituus on riippuvainen kaapelin pituudesta ja käytetystä nopeudesta ja suurin pituus voi olla sadoista metreistä kilometreihin.

Kevyimmässä tapauksessa laite toimii itsenäisenä logiikkana ja ohjaa yhden EA-485 väylällisen verran orjalaitteita. Tällöin vain asennusvaiheessa laite kytketään ethernetiin ja asetukset tehdään selaimella. Raskaimmassa käytössä on useammissa väylissä aktiivisten jakajien takana kaikki 247 laitetta, joita ohjataan ethernetin kautta keskuslogiikalta. Mikäli sulautettu järjestelmä havaitsee ohjattavan prosessin olevan karkaamassa käsistä, ohjaa järjestelmä itsenäisesti orjia esiasetettujen parametrien mukaan. Erityistä tukea tälle ei ole, mutta taitava suunnittelija voi asettaa laitteen logiikan ohjaamaan prosessia vasta jonkin mittauksen ylittäessä raja-arvon. Käytännössä koko avaruuden ohjausta ei tällä laitteella voi tehdä, koska pelkkä osoitteiden luetteleminen vie yli 64 megatavua ja laitteessa on 8 megatavua muistia. Median muunnossa voidaan ohjaus kuitenkin välittää mille tahansa osoitteelle ja kaikille laitteille.

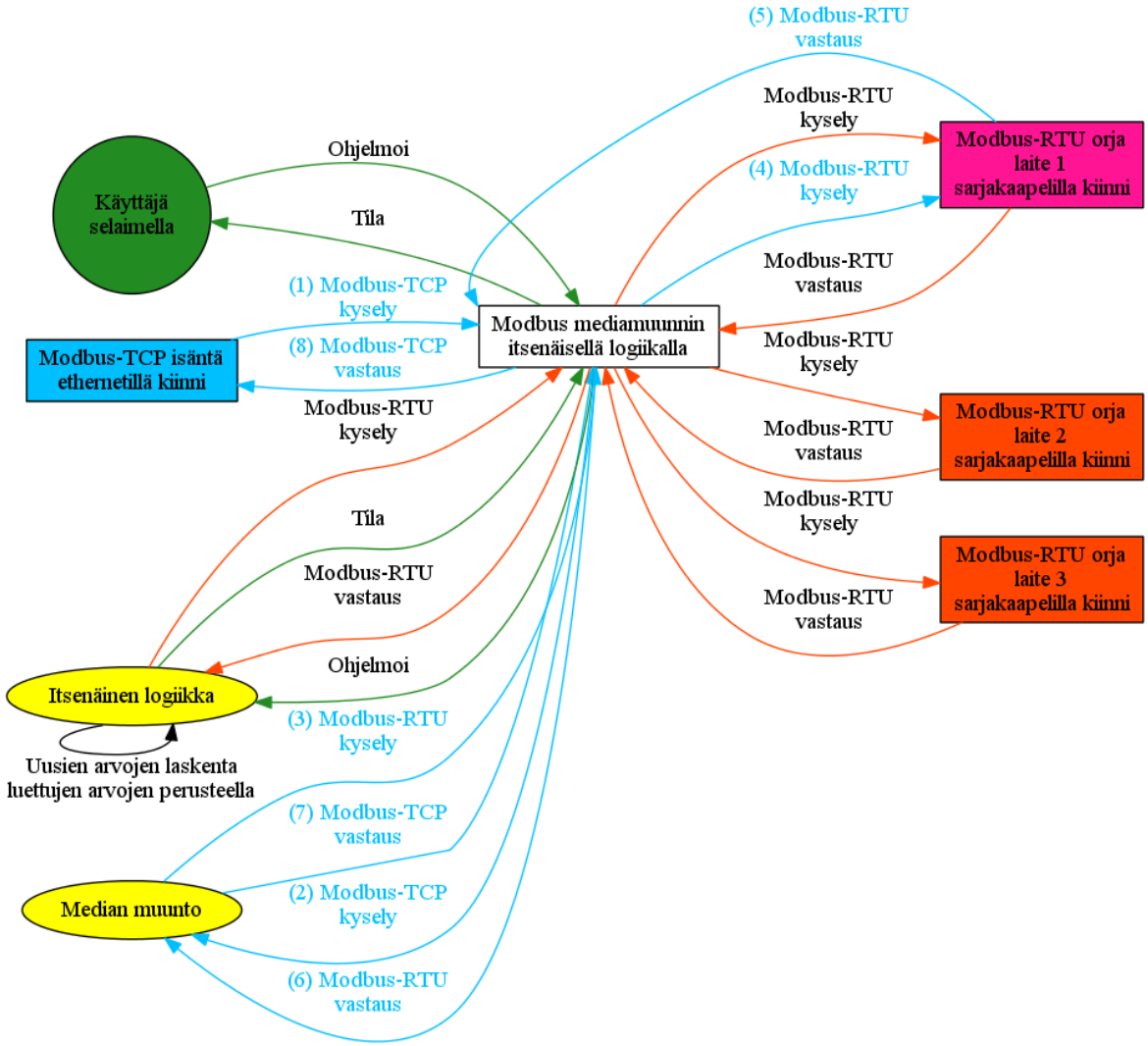
Eräs vaatimus laitteelle oli hyvin yleinen logiikan toimintakuvaus, jossa sisääntulosta luettiin data, joka ajettiin kompensaattorin lävitse ja tämän jälkeen ajettiin PI-säätimelle. Ulostulo kirjoitettiin sen jälkeen toimilaitteelle. Tämä toimi lähinnä ohjeena minulle vähimmäisvaatimuksina, minkälaisia toimintoja järjestelmän tarvitsee osata toteuttaa. Esimerkiksi mainittu PI-säädin sai mukaansa D-säädönkin, koska se oli helppo toteuttaa.

Kolmanneksi laitteen tulisi olla helppokäyttöinen. Kun laitetta asennetaan, ei voida vaatia

käyttäjää ohjelmoimaan laitetta C-koodilla tai edes muuttamaan vakioita ja ajamaan uutta ohjelmistoa järjestelmään sisään. Käyttöliittymän tulisi siis olla helppo ja selkeä, mutta silti vielä monipuolinen jokaiselle asetuksen säätämiselle.

Omaksi tavoitteekseni asetin toimivan ohjelmiston. Ohjelmointivirheet tulisi olla poissa kokonaan ja ohjelmiston siirto toiselle alustalle tulisi olla helppoa. Siirrettävyyden helpottamiseksi ohjelmiston tulisi olla modulaarinen.

Kuvassa 1 on esitetty laitteen eri ominaisuuksia. Laite täytyy saada toimimaan halutulla tavalla. Sitä varten käyttäjä ohjelmoi laitteen ja itsenäisen logiikan selaimella. Tätä varten on kirjoitettu käyttöohje, joka on liite 1. Lisäksi itsenäisen logiikan tulee pystyä ohjaamaan Modbus-RTU laitteita sarjaportin kautta. Median muunto voi myös tapahtua. Median muunto voi olla kolmessa tilassa. Vain muunto ja automaatti tilassa ethernetistä tulevat kyselyt välitetään sarjaväylälle. Median muuntamisen ollessa kokonaan pois itsenäinen logiikka ohjaa orjalaitteita. Kuvassa 1 käyttäjän toimet on kuvattu vihreällä värillä ja median muunto vaaleansinisellä. Oranssinpunainen väri kuvaa itsenäisen logiikan kommunikointia. Laitteen sisällä tapahtuvat prosessit ovat keltaisella. Modbus-RTU orjalaite 1 on kuvattu vaaleanpunaisella, koska kuvassa on sitä ohjattu itsenäisestä logiikasta ja median muunnon puolelta.



Kuva 1. Periaatekuva laitteen ominaisuuksista

3. KEHITYSYMPÄRISTÖ

Järjestelmän toimintaympäristö on sulautettu järjestelmä. Oman ohjelmiston kehitystä varten oli käytössä kehitysalusta fyysisenä laitteena ja ohjelmointiympäristö sekä ajonaikaisten tapahtumien tarkkailuun debug-järjestelmä. Debug-järjestelmä koostui ohjelmistosta ja kehitysalustan mukana toimitetusta JTAG-liityntälaitteesta.

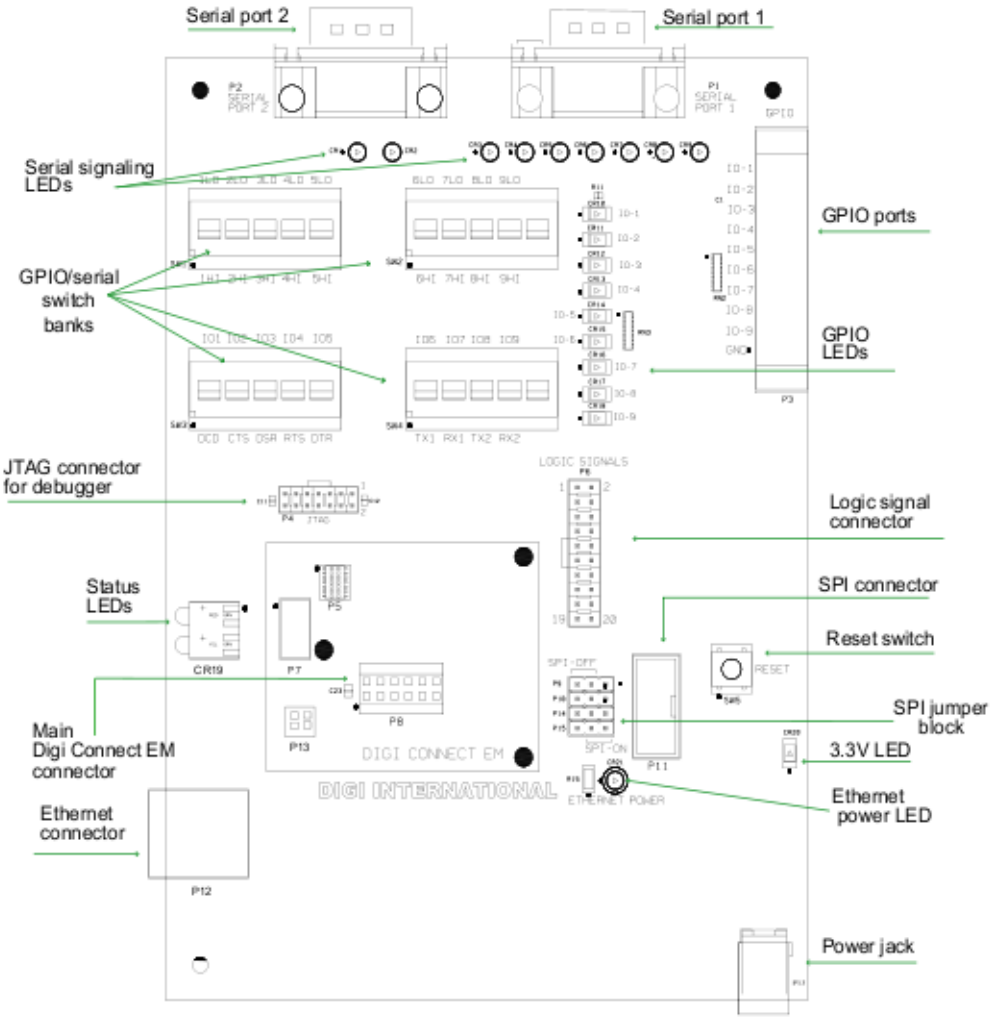
Ohjelmointiympäristö oli erityisesti Digi Internationalin laitteita varten Eclipsen päälle jatkokehitetty ympäristö. Tarjolla olivat ohjelmointiin tarvittavat työkalut ja debug-järjestelmän ohjaus. Lisäksi laitteen ja mukana toimitettujen palveluiden ja käyttöjärjestelmän dokumentaatioon pääsi helpolla käsiksi ohjelmointiympäristöstä.

3.1. Digi Connect EM

Kehitykseen käytettiin Digi Internationalin toimittamaa sulautetuille järjestelmille tarkoitettua kehitysalustaa. Kyseinen alusta on modulaarinen ja käytetty mikrokontrolleri sijaitsee kehityskortin liittimeen asetettavan piirilevyn päällä. Suoritin on ARM7-pohjainen 32-bittinen mikrokontrolleri. Flash-muistia laitteessa on 4 megatavua ja käyttömuistia on 8 megatavua. Järjestelmän kellotaajuus on 55 MHz. Kokoa mikrokontrollerin piirilevyllä on 4 cm toiseen ja 5 cm toiseen suuntaan. Mikrokontrollerimoduulin koko riippuu, paljonko erilaisia liittimiä on kiinnitetty levyyn./2/

Itse kehitysalustan koko on huomattavasti suurempi ja se tarjoaa ethernet-liittimen, yleiskäyttöisiä digitaalisia sisään- ja ulostuloja sekä muita tapoja liittyä mikrokontrolleriin. Tässä työssä tärkeimmät olivat sarjaportit ja JTAG-liityntä, jota käytettiin laitteen ohjelmointiin. Luonnollisesti verkkoa käytettiin selainta ja median muuntoa varten.

Kuva 2 antaa yleiskäsityksen kaikista liityntämahdollisuuksista. Suurimman osan kehitysalustan tilasta vie kytkimet, joilla voidaan valita sarjaporttien ja yleiskäyttöisten digitaalisten sisään- ja ulostulojen välillä. Kuvassa ne ovat GPIO/serial switch banks. Sarjaporteissa on tasomuunnin, jotta kehitysalusta voidaan kytkeä suoraan tietokoneen sarjaporttiin. Yleiskäyttöisiä digitaalisia sisään- ja ulostuloja ei tässä projektissa käytetty ja kytkimet oli asetettu sarjaporttikäyttöön.



Kuva 2. Kehitysympäristön piirilevyn sijoittelukuva/1, s. 24/

3.2. Net+OS

Net+OS on Digi Internationalin tarjoama reaaliaikakäyttöjärjestelmä. Käyttöjärjestelmä tarjoaa keskeyttävän moniajon, muistinhallinnan ja viestit säikeiden välille, sekä muita moniajoon liittyviä palveluja. Kernelinä toimii ThreadX. Net+OS ei käytä prosesseja moniajossa, vaan säikeitä. Säikeet ovat huomattavasti kevyempiä kuin prosessit ja ne käsittävät lähinnä pinon, jossa kyseinen säie toimii. Tällä saavutetaan pieni muistijälki ja keveys. Samalla menetetään muistin rajaus eri säikeiden väliltä ja huonolla toteutuksella on mahdollista käsitellä väärää paikkaa muistista. Käyttöjärjestelmän tehtävä on abstrahoida käytetyn mikrokontrollerin sisäinen toteutus helposti käytettäviksi järjestelmäkutsuiksi. Tällä saavutetaan ohjelmiston siirrettävyys, mikäli esimerkiksi mikrokontrolleri vaihtuisi. Siirroksen tekemiseen ei tarvitse kuin käyttöjärjestelmän ajureiden siirto. Tietenkin tulee olla käytettävissä käyttöjärjestelmä ja käännöstyökalut uudelle alustalle. Käyttöjärjestelmän lisäksi kehitysympäristössä on ohjelmistopalveluja käytössä. Tässä työssä käytössä oli Basic web service, joka on yksi hypertekstipalvelua tarjoavista palveluista. Muut ovat advanced ja secure web server. Ne eroavat lähinnä monimutkaisuudessa. Käyttöjärjestelmän mukana tulee myös dokumentaatiota ja kehitysympäristö. Kehitysympäristö on luotu Eclipsen osaksi joten siinä on kaikki Eclipsen tarjoamat ominaisuudet. Net+OS:n ohjelmointiin käytetään C-kieltä. Net+OS:n kääntäjä hyväksyy normaalin ANSI standardin C-kielen, joten ohjelmoinnissa ei ole ongelmia kääntäjän C-kielen erikoisuuksien suhteen. /3, 11/

BSD-tyyliset socketit ovat tarjolla monessa muussakin ympäristössä ja sarjaliikenteen rajapinta vastaa hyvin paljon UNIXeista tuttua termios-rakennetta. Näiden samankaltaisuuksien vuoksi oli järjestelmä helppo ottaa pois mustan laatikon kehityksestä ja saattaa kehitys Linuxin päälle. Hyötyhän tästä on selkeästi helpommin hallittava sarjaliikenteen tarkastelu, sovelluksen virheiden etsiminen ja muut PC:n tarjoamat palvelut kuten suuri laskentateho ja rajaton muisti sovelluksen kannalta. Kääntäminenkin oli nopeampaa, koska tietokone oli tehokkaampi. Haittoina olivat tietenkin kohdejärjestelmän puute ja mahdollisten kääntäjän tekemien virheiden löytäminen.

Käytetty versio Net+OS-reaaliaikakäyttöjärjestelmästä oli 7.1, joka on jo useita vuosia vanha. Kyseisessä versiossa on monia virheitä, jotka on korjattu myöhemmin. Viittausta muistiongelmiin en löytänyt, koska Digi Internationalin keskusteluryhmät eivät tuntuneet tarjoavat tästä versiosta paljoakaan neuvoja. Kääntäjän vikoja verkosta etsimällä pystyi löytämään muutaman mahdollisen ongelman aiheuttajan, mutta testikoodit vikojen esille tuomiseksi ei aiheuttaneet haluttua lopputulosta. Toisaalta ohjelmaa muuttamalla siten, että kääntäjän ongelmat eivät olisikaan ilmaantuneet, osa ongelmista ratkesi. Muutamissa tapauksissa kääntäjän tuottama assembleri muuttui toimivaksi vaihtamalla epäilyttävä koodirivi vastaavaksi, mutta erilaiseksi.

Kummastuttavin esimerkki oli, että lause if(osoitin == NULL) aiheutti koko järjestelmän kaatumisen, mutta if(NULL == osoitin) toimi. Syy tähän oli, että osoitin oli muutettu toisesta tietotyypistä toiseen ja kääntäjä oli tietynlaisessa tilassa käsitellessään tätä riviä. Tuota lausetta ennen pitää olla siis jotain erityistä käännösprosessin tilaan vaikuttavaa tapahtunut. Ajon aikana väärin luotu koodi hakee osoittimen osoitteesta tietoa ja yrittää suorittaa sitä ja seurauksena on virhe. Tähän ongelmaan törmäsin myös aivan toisessa yhteydessä vuodelta 2008 olevassa lähdekoodissa, jossa oli kommentti vanhempien kääntäjien luomasta rikkinäisestä ohjelmasta. Ilmeisesti se on korjattu uudemmissa kääntäjissä.

Muutenkin oli epämiellyttävää toimia vuosia vanhan järjestelmän parissa, jossa varmasti on tapahtunut kehitystä. Eclipsestä puuttui koodin kaunistaja, joka sisentäisi ja tekisi koodista helpommin luettavaa. Oli erittäin raskasta itse pitää koodi siistinä. Net+OS on varmasti hyvä käyttöjärjestelmä, jos kehitystyökalut ovat ajan tasalla ja tietää kaikki piilotetut sudenkuopat.

4. MODBUS

Modbus on yksinkertainen, yleinen ja avoin viestintäprotokolla kaupallisten logiikoiden välillä. Modbus-protokolla koostuu kahdesta kerroksesta. Ylemmässä sovelluskerroksessa, joka on määritelty OSI-mallin 7 kerrokselle, kulkee Modbus-protokollan varsinainen tieto. Alempi kerros sisältää siirtotiellä käytettävää tietoa, kuten lähettäjän ja vastaajan osoitteita ja tarkistussumman.

Alempi kerros Modbus-RTU-protokollalla on OSI-mallin kerroksilla 1 fyysinen kerros ja 2 kuljetuskerros. Modbus-TCP-protokolla on taas toisaalta kerroksilla 5 istuntokerroksella ja 6 esitystapakerroksella. Alemmat kerrokset Modbus-TCP-protokollalla kuuluvat sille tekniikalle, jonka päällimmäisenä on TCP. Modbus-RTU-protokollan OSI-malli ja Modbus-TCP-protokollan yleiskuvaus on esitetty kuvassa 3.

4.1. Modbus-protokollan kuvaus

Sarjaliikenteessä voidaan käyttää yhteyttä EA-232 tai EA-485. Näiden välinen ero on EA-485 ollessa balansoitu ja väylä, johon voi laittaa useampia laitteita, kun EA-232 tarjoaa yhteyden kahden pisteen välille sähköisesti balansoimattomana. Lisäksi EA-232 kulkiessa vain muutamia metrejä, EA-485 voidaan luoda satojakin metrejä pitkiä yhteyksiä. Tämä johtuu balansoidun yhteyden yhteismuotoisen häiriön siedosta./7/

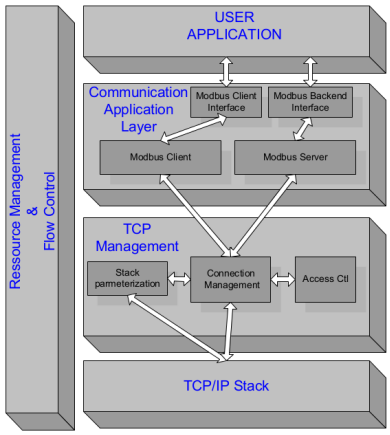


Figure 4: MODBUS Messaging Service Conceptual Architecture

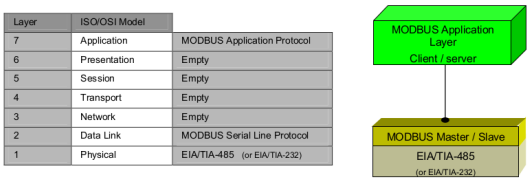


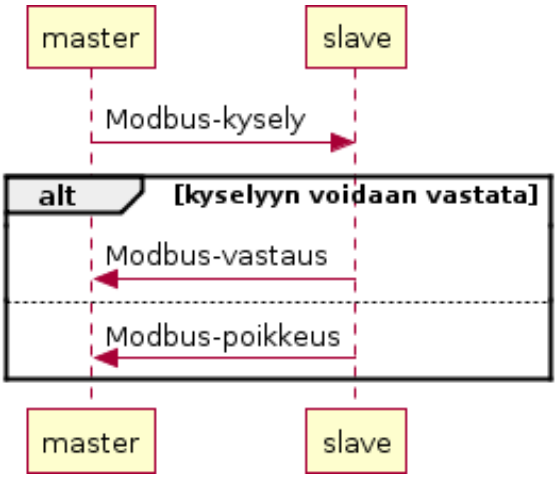
Figure 2: MODBUS Protocols and ISO/OSI Model

Kuva 3. Modbus-protokollan OSI-malli/7, 6/

Modbus-protokollalla on kaksi erilaista datan koodaustapaa sarjaliikenteessä. ASCII ja RTU. Näiden ero on lähinnä ASCII-protokollan lähettäessä tekstimuotoisia heksadesimaaliarvoja, RTU lähettää datan binäärinä. ASCII-protokolla on siis huomattavasti hitaampi samoilla linkkinopeuksilla. Esimerkiksi arvo sata on 0x64 heksadesimaalina. RTU lähettää arvon 0x64 yhdellä tavulla, kun ASCII-protokolla lähettää kaksi tavua. Merkit numero kuusi ja neljä, jotka ovat heksadesimaaleina 0x36 ja 0x34.

Modbus-protokollan muoto on yksinkertainen: isäntä kysyy ja orja vastaa. Kuva 4 antaa hyvin yksinkertaisen kuvauksen miten kyselyt menevät. Mikäli kyselyn laitenumero on sama kuin orjalla, huomioidaan kysely. Kun kysely on huomioitu, tutkitaan onko funktio tuettu ja sille annetut parametrit oikeat. Jos kysely on vastattavissa, annetaan vastaus, muutoin vastataan sopivalla poikkeuksella. Sarjaliikenteen päällä on vielä tarkistussumma, jonka ollessa virheellinen hylätään koko kysely myös. Hylkääminen aiheuttaa isännällä aikakatkaisun. Protokollina Modbus-TCP:n ja Modbus-RTU:n erot tavutasolla ovat havaittavissa kuvassa 5. Modbus-protokollan funktiot ovat määriteltyjä spesifikaatiossa./5, s. 4-11/

Yleisin rakenne Modbus-funktiolle on 8-bittinen funktiokoodi välillä 1-127, osoite ja montako tavua tietoa luetaan tai kirjoitetaan. Vastaus on sitten vastaavasti funktiokoodi, montako tavua on tulossa dataa ja itse data. Virheilmoitukset ovat funktiokoodi heksadesimaalina + 0x80 heksadesimaalina. Eli eniten merkitsevä bitti nostetaan ylös funktion koodista ja sen jälkeen tulee toinen tavu ilmoittaen virheen tyypin./5, s. 11-48/



Kuva 4. Yksinkertainen kuvaus Modbus-protokollasta

Modbus-funktiot voivat kohdistua neljään erilaiseen 16-bittiseen osoiteavaruuteen. Funktiot ovat lähinnä luku- ja kirjoitusoperaatioita. Spesifikaatio määrittää osoiteavaruuksiksi discrete input, coil, input register ja holding register, joilla on omat funktionsa lukemiseen ja kirjoitukseen. Näistä coil ja holding register avaruudet ovat luettavia ja kirjoitettavia avaruuksia, kun discrete input ja input register avaruudet vain luettavia. Discrete input ja coil ovat päällä tai pois päältä olevia binäärisiä tietoja omissa osoiteavaruuksissaan. Input register ja holding register osoiteavaruuksien osoitteet osoittavat 16 bittiin tietoa./5, s. 6-8/

Discrete input eli erillinen sisääntulo on Modbus-protokollassa yksi bitti. Reaalimaailmassa tätä vastaisi päällä tai pois oleva sisääntulo. Modbus-protokolla ei kuitenkaan ota kantaa käsiteltävän rekisterin, sisääntulon tai muun käsiteltävän osoitteen reaalimaailman olemassaoloon. Jokin erillinen sisääntulo, joka luetaan voi aivan hyvin olla ohjelmallisesti luotu ja antaa tietoa säädön tilasta. On kuitenkin helpompi ymmärtää osoiteavaruuksien ominaisuudet, mikäli reaalimaailman esimerkit annetaan. Esimerkiksi vintin lamppua ohjaavan releen tilaa voitaisiin potentiaalivapaista kärjistä lukea coil tyyppisestä avaruudesta.

Modbus-protokollan kuljetukseen käytetystä mediasta ja niiden lisäksi tuomista ominaisuuksista on kerrottu tarkemmin kohdassa 5. Lisäksi siinä osoitetaan, kuinka helposti protokollan funktio yhdessä tavussa ja data lopuissa vaihtaa mediaa.

Olen kääntänyt discrete input suomeksi sisääntulona ja erotuksena input register osoiteavaruudesta, josta käytän ilmaisua sisääntulorekisteri. Holding register olen kääntänyt pelkäksi rekisteriksi ja coil on kela, joka viittaa releen ohjauskelaan. Koska osoiteavaruus on pitkä sana, olen sen sijasta käyttänyt ilmaisua tyyppi. Lisäksi tyyppiin on tullut kahta rekisteriä tai sisääntulorekisteriä käyttävät 32-bittiset tietotyypit. Tietotyyppeinä käytössä ovat etumerkilliset ja etumerkittömät kokonaisluvut, sekä liukuluku. Yhteensä käytettävissä on kuusi erilaista 32-bittistä tietotyyppiä.

4.2. Modbus-protokollan toteutus

Tässä työssä kokeilin lähestyä Modbus-protokollaa monin tavoin. Ensimmäiset yritykset olivat tehdä se itse. Hyvinä ominaisuuksina itse tekemisestä olisi ollut sen toimivuus ympäristössä ja mahdollisuus optimointiin. Samalla se olisi voitu integroida tiukasti järjestelmään. Huono puoli olisi selkeästi työmäärä ja toimintavarmuus.

Parin yrityksen ja erehdyksen jälkeen otin käyttöön Libmodbus-kirjaston, joka on lisensoitu GNU Lesser General Purpose Licensen alla ja salli oman ohjelmiston olevan suljettua koodia. Sitä vastaan voi linkittää mistä tahansa ohjelmasta lisenssistä riippumatta. /4, kohta 4/

Kirjasto vaati kuitenkin kaksi muutosta, muistinkäsittelyn ja sarjaportin käytön muutoksen. Net+OS tarjoaa oman tavan käsitellä muistia, mikä on varsin ymmärrettävää. Perättäiset perinteiset malloc() ja free() kutsut muistin varaamiseksi ja vapauttamiseksi johtavat todennäköisesti johonkin ylivuotoon ja muistinvarauksen epäonnistumiseen. Muut mahdolliset ongelmat ovat vapaan muistin sirpaloituminen, itse käyttöjärjestelmän muistin vuotaminen tai jokin muu kummallinen ominaisuus. Tämän selvittämiseen ei käytetty ylimääräistä aikaa vaan aina ja kaikkialla muisti varataan heti alussa, ennen omien säkeiden käynnistämistä. Silloin muistin varaus onnistuu suuremmalla todennäköisyydellä.

Toinen eroavaisuus on sarjaportin asetukset määrittävässä termios rakenteessa. Kun UNIX-järjestelmissä termios sisältää literaaliset vakiot nopeuksille, Net+OS haluaa luvun. Myöskään termios-rakenteen kaikkia kenttiä ja ominaisuuksia ei ole toteutettu.

Lisäksi Libmodbus-kirjastosta puuttui erikseen toivottu Modbus-funktio, joka kirjoittaisi ja lukisi yhdellä Modbus-kyselyllä rekistereitä. Tämä tuli tähän työhön toteutettavaksi ja se liitettiin myös osaksi virallista Libmodbus-kirjastoa./9, 8/

Toinen selkeä puute oli viestin välitys TCP-protokollasta RTU-protokollaan. Tämäkin toteutettiin ja sekin liitettäneen myös viralliseen kirjastoon kattavien testien ja jatkokehityksen jälkeen. Toistaiseksi kirjaston kehitys on keskittynyt IPv6:n toiminnan parantamiseen ja useamman alustan tukemiseen. Virallisesta kirjastosta on siirretty ominaisuuksia laitteen käyttämään versioon 2011 lopulla, mutta kaikkia muutoksia ei tuotu versiomuutosten takia.

5. MEDIAN MUUNTO

Median muunnossa järjestelmä toimii orjana ethernetiin päin ja isäntänä sarjaporttiin. Modbus-TCP isäntä voi lähettää kyselyn laitteelle, joka lähettää kyselyn sarjaportin kautta ja palauttaa vastauksen.

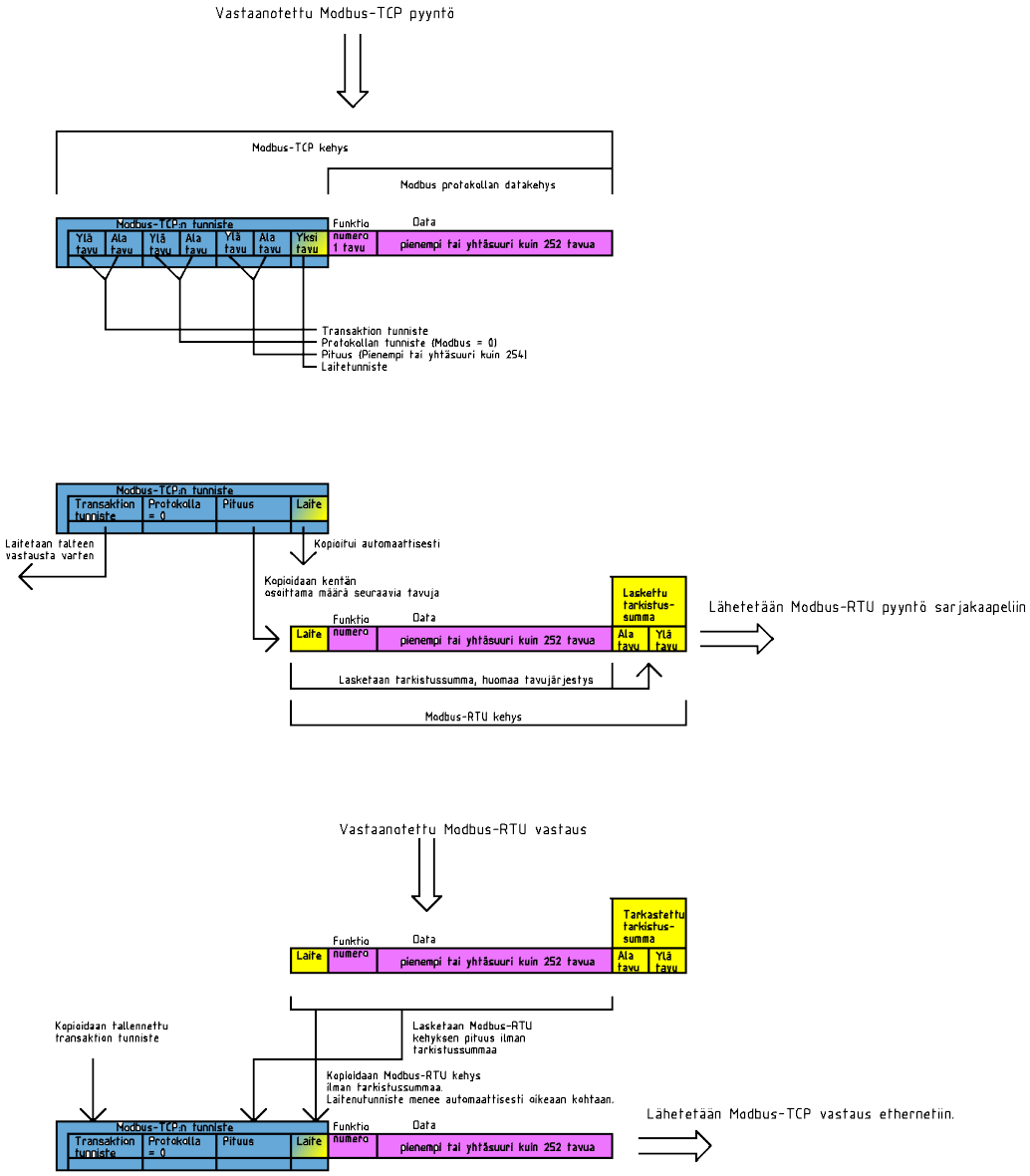
Käytännössä odotetaan ethernetin puolelta tulevaa Modbus-TCP kyselyä. Kyselyn saapuessa Modbus-TCP:llä tulkitaan välitettävän viestin koko, lasketaan välitettävän osan tarkistussumma ja välitetään kysely Modbus-RTU-protokollalla sarjaportin kautta ulos. Tämän jälkeen odotetaan vastausta sarjaportista. Vastauksesta tarkastetaan Modbus-RTU-protokollan mukainen laitenumero ja tarkistussumma. Mikäli vastaus on hyväksyttävä, välitetään vastaus ethernetiin Modbus-TCP-isännälle.

5.1. Normaali toiminta

Modbus-protokollan viestien muunto on melko yksinkertaista. Sama viestin rakenne on säilytetty kummassakin mediassa. Lisäksi kaikki data, mitä TCP:n kautta tulee, voidaan siirtää sarjaliikenteen puolelle. Ainoastaan transaktion tunnus tulee säilyttää muistissa ja käyttää sitä uudestaan palautettaessa vastaus isännälle.

Ainoa monimutkaisempi asia prosessissa on ethernetin kautta mahdollisuus tehdä useampia pyyntöjä, kuin sarjaliikenne pystyy käsittelemään ja useamman transaktion mahdollisuus. Toisaalta tämäkin saadaan hyvin hallintaan tekemällä järjestelmällisesti kyselyitä sarjaportin puolelle ja antamalla yhteyden odottaa. Lisäksi Modbus-TCP viestissä on transaktion tunnus. Se säilötään vastauksen lähetystä varten. Kuvassa 1 on lisätty numeroita vaaleansinisiin nuoliin. Niitä seuraamalla havaitaan, että ensin nuolella (1) kysely menee Modbus-TCP isännältä laitteeseen. Tämän jälkeen kysely luetaan mediaa muuntavassa säikeessä (2) ja lähetetään sarjaporttiin pienin muutoksin (3.) Laite lähettää kyselyn laitteelle sarjaportista (4) ja saa ehkä joskus vastauksen. Mikäli vastaus tulee ajallaan, laite lukee (5) sen ja mediaa muuntava prosessi käsittelee (6) sen. Tämän jälkeen vastaus muutetaan Modbus-TCP viestiksi ja tallennettu transaktiotunnus laitetaan paikoilleen. Sen jälkeen vastaus matkaa nuolella (7) laitteen käsiteltäväksi ja lopuksi saapuu (8) kyselyn tehneelle Modbus-TCP isännälle. Poikkeustilanteita voi esiintyä ja ne käsitellään kohdassa 5.2. Useamman Modbus-RTU kyselyn lähettäminen päällekkäin onnistuisi periaatteessa, mutta tällöin tulisi todennäköisemmäksi viestien törmäys ja siitä syntyvä aikakatkaisu ja uudelleenlähetys.

Kuvassa 5 on esitetty mitä datalle tapahtuu muunnoksessa. Myös tavujen määrä ja paikka on ilmaistu. Siinä vaaleansininen on Modbus-TCP:lle kuuluvaa dataa, violetti Modbus liikennettä



Kuva 5. Modbus-TCP:n muunto Modbus-RTU:ksi/5, 6/

ja keltainen Modbus-RTU protokollalle käyttämää tietoa. Väriliuku laitteessa viittaa tiedon löytyvän molemmista protokollista.

Ylimpänä kuvassa on luettu Modbus-TCP viesti. Samalla on tuotu esille Modbus-TCP:n tunnisteen kentät. Nämä 7 tavua sisältävät transaktion numeron, protokollan tyypin, joka Modbus-protokollan ollessa kyseessä on aina nolla. Tämän jälkeen kehyksessä on jäljellä olevien tavujen määrä ja laite, jolle viesti tulisi välittää. Sama laite löytyy myös Modbus-RTU protokollasta ja median muunnoksessa tämä kopioituu sinne.

TCP-RTU muunnosta aloitettaessa otetaan ensimmäiseksi talteen transaktion tunnus. Tämän jälkeen luetaan pituuskenttä ja sen jälkeen aloitetaan kopioimaan pituuden verran dataa tulleesta TCP-viestistä. Lopuksi muodostetulle viestille lasketaan tarkistussumma, joka lisä-

tään viestin loppuun. Tämän jälkeen viesti voidaan lähettää Modbus-RTU kehyksenä alkaen laitenumerosta.

Orjalaitteen vastatessa tarkastetaan Modbus-RTU kehys. Laitenumeron ja tarkistussumman tulee täsmätä. Modbus-TCP kehys kopioidaan alkuun ja transaktion tunniste kopioidaan muistista oikealle paikalle. Tämän jälkeen asetetaan pituus oikein laskemalla Modbus-RTU kehyksen laitenumerosta viimeiseen tavuun asti ennen tarkistussummaa. Sen jälkeen lähetetään vastaus takaisin Modbus-TCP isännälle.

5.2. Poikkeustilanteet

Aina median muunto ei välttämättä tapahdu kysely, uudelleenlähetys, vastaus, uudelleenlähetys kaavan mukaan. Erilaisiin poikkeustilanteisiin on varauduttu ohjelmistossa. Poikkeustilanteesta voidaan antaa virhevastaus tai antaa kyselyn tehneen Modbus-TCP isäntälaitteen aikakatkaista kysely.

Poikkeukset syntyvät lukuoperaation epäonnistumisena, saadun kyselyn jäsentymättömyydestä ja kirjoituksen epäonnistumisesta. Luku- ja kirjoitusoperaatioiden epäonnistuminen tapahtuu, kun Modbus-TCP isäntä katkaisee yhteyden tai sarjaportin asetukset ovat väärät. Jäsennysvirheessä on kyse väärästä datasta.

5.2.1. Lukuvirhe

Mikäli Modbus-TCP kehys on viallinen, eli dataa on vähemmän kuin pituus on ilmaissut, unohdetaan luettu data. Mikäli dataa on enemmän, lähetetään se mitä on saatu ja tutkitaan jäljelle jääneestä datasta, onko se Modbus-TCP viesti. Mikäli se ei ole, unohdetaan myöskin se ja toisaalta toimitaan normaalisti, mikäli kyseessä on ehjä viesti.

Modbus-RTU puolelta viestin tullessa, tarkastetaan sen tarkistussumma. Samalla lasketaan viestin pituus. Mikäli viesti on virheellinen, lähetetään saatu pyyntö uudelleen. Mikäli laite vastaa kahdesti perättäin virheellisesti, pyyntö unohdetaan ja siirrytään seuraavaan. Ethernetin puoleinen laite sitten toteaa aikakatkaisun tapahtuneen ja yrittää uudelleen.

Samoin tapahtuu myös, mikäli vastaus ei saavu ajallaan. Aikakatkaisun yhteydessä yritetään kerran uudelleen lähetystä.

5.2.2. Lähetysvirhe ethernetin puolelta

Mikäli yhteys on suljettu, unohdetaan kaikki odottavat pyynnöt. Samoin käy tuleville vastauksille. On kuitenkin huomattava, että keskeneräinen vastaus odotetaan ja vasta tämän jälkeen vapautetaan sarjaportti, jotta sisäinen logiikka voi alkaa jälleen säätä järjestelmää.

Muun kaltaisia lähetysvirheitä ei voi oikeastaan olla. Sarjaportin lähetysvirheestä ei saada tietoa. Lähinnä aikakatkaisun kautta huomataan, että joko tieto ei kulje laitteelle tai lähetystä ei ymmärretä.

5.2.3. Aikakatkaisu

Aikakatkaisu voi tapahtua kahdesta syystä. Joko aikakatkaisun arvo on liian lyhyt ja kyselyn vastausta on odotettu liian vähän aikaa tai sitten on kysely lähetetty sellaiselle laitteelle, joka ei ole väylällä kuuntelemassa. On myös olemassa kolmas mahdollisuus. Laitteet ovat broadcast-tilassa ja vastauksia ei silloin lähetetä.

Kuitenkin aikakatkaisun tapahtuessa median muunnin unohtaa transaktion tunnuksen ja jatkaa eteenpäin. Laitteessa sarjaportin puoleinen aikakatkaisu asetetaan sarjaportin asetuksista. Se vaikuttaa sisäiseen logiikkaan ja median muuntamiseen. Aikakatkaisun vaaroista on varoitettu käyttöohjeessa, joka on liite 1.

6. SUUNNITTELU

Järjestelmän suunnittelua ei toteutettu yhtenä operaationa. Suurin osa suunnittelusta tapahtui kokeilemalla ja erehtymällä. Lisäksi toteutuksessa ilmenneet ongelmat palauttivat usein suunnitteluun. Oli järkevämpää korjata toteutuksessa havaittu ongelma suunnittelemalla ongelman poistava ratkaisu, kuin yrittää korjata ongelmaa.

Suunnittelun suuret linjat olivat piirretty pienille papereille suoraan vaatimuksista ja yksityiskohtia mietin suurilla arkeilla. Kokeilemalla asioita testaamalla voiko jotain tiettyä asiaa tehdä, sain pienten asioiden tyhjiä kohtia täydennettyä. Usein vain jonkin asian kokeilu muovautui lopulliseksi toteutukseksi. Esimerkkinä mainittakoon käyttöliittymän ulkoasu.

6.1. Suunnittelun kulku

Tämä projekti lähti tilaajan tarpeesta ja ideasta ratkaista ongelma. Koska omat käsitykset olivat erittäin hatarat koko automaatiosta, oli aloitettava selvittämällä, mikä on maali johon pyritään. Suurimmat kysymykset olivat, mitä käytössä oleva alusta osaa tehdä, mitä ihan oikeasti tulisi tehdä, miten se tehdään ja voiko sen tehdä rajallisessa määrässä aikaa ja muita resursseja.

Projekti lähti eteenpäin tekemällä projektisuunnitelma ja systeemin suunnittelu hyvin yleisellä tasolla. Kenties suurin ongelma suunnitelmissa oli niiden keveys. Jokaisen yksityiskohdan raskaan suunnittelemisen jälkeen olisi ollut helpompi toteuttaa ohjelmisto.

Toisaalta työ olisi vielä toteutuksen alkuvaiheissa, mikäli olisin suunnitellut jokaisen liikahtavan bitin. Myös tiedon puute ja rajalliset kykyni olisivat aiheuttaneet hienon suunnitelman epäonnistumisen. Toteuttaessa oppi monia uusia asioita, joista useat osoittivat jo sinänsä toimivan suunnitelman mahdottomuuden.

Seuraavaksi tehtäväksi tuli kehitysympäristöön tutustuminen ja sen tuomiin palvelujen ja rajoitusten etsiminen. Tämä toteutettiin tulostamalla tuhannen sivun dokumentaatio kansioon ja lukemalla se kannesta kanteen, jotta yleiskäsitys kehitysalustan kykenevyydestä tulisi itselle selväksi.

Lisäksi oli hankittava lisää tietoa miten tämän kaltaisen järjestelmän voi tehdä. Tähän työhön kuului itselle kolme tuntematonta tekijää. Ympäristö oli uusi, automaatiosta minulla oli hyvin hatarat käsitykset ja Modbus-protokolla oli myös täysin tuntematon.

Suunnittelussa meni hyvin pitkään, koska minun piti kokeilla monia lähestymistapoja ongelmaan. Lisäksi ongelmat kehitysalustan kanssa toivat mukaan omat murheensa. Alkuperäinen suunnitelma oli huomattavasti suoraviivaisempi. Siinä oli käyttöliittymä, järjestelmäsäie ja medianmuunnosta ja päivityksistä vastaava säie. Kaikki tieto liikkui kauniisti määritellyillä tietotyypeillä ja mitään ongelmia ei ollut. Todellisuudessa ongelmia tuli ja niitä kierrettiin pilkkomalla osia eri säikeisiin, poistamalla hienosti määritelty kommunikointi ja kiertämällä lisää ongelmia.

6.2. Ratkaisut

Monien kokeilujen jälkeen järjestelmän muoto alkoi hahmottua ja neljä erilaista osaa löytyivät. Median muunto, Modbus, sisäinen logiikka ja käyttöliittymä ovat hyvinkin erotettavia osia järjestelmästä.

Näistä osista median muunto ja sisäisen logiikan ja Modbus-laitteiden päivitys sai oman säikeensä järjestelmään. Toinen säie meni asetusten hallintaan ja päivitysten ajoitukseen.

Käyttöliittymä toimii sulautetun käyttöjärjestelmän tarjoaman hypertekstipalvelimen päällä. Käyttöliittymä lähettää viestejä asetuksista vastaavalle säikeelle, joka antaa luvan sitten muuttaa asetuksia muistissa. Ratkaisu ei ole kaunis, mutta yksi käyttöjärjestelmän ongelmista osoittautui olevan viestien käsittely. Käyttöjärjestelmän ongelmat palauttivat useamminkin takaisin suunnitteluun toteutuksesta. Käyttöliittymästä tuli tehdä helppokäyttöinen. Tämä saavutettaisiin mahdollisimman kevyellä ilmeellä ja operaattoreilla, jotka olisivat yksinkertaisia. Olisi käyttäjän tehtävä rakentaa niistä monimutkaisia rakenteita. Operaattoreiden teoreettinen maksimi olisi yli 65 tuhatta kappaletta. Käytännössä muisti loppuisi ennen, mutta en törmännyt tähän missään vaiheessa.

Koska järjestelmää voi käyttää kahdella tapaa, on säädettäviin ominaisuuksiin jätetty mahdollisuus kytkeä median muunto päälle, pois tai automaatille. Päällä ollessa sisäinen logiikka ei toimi, vaikka Modbus-TCP-yhteyksiä ei olisi. Poissa ollessa mediaa ei muuteta vaikka Modbus-TCP-isäntä yrittäisi ottaa yhteyttä. Automaatilla yhteyden tullessa, keskeytetään sisäinen logiikka ja siirrytään palvelemaan Modbus-TCP-isäntää. Tällä ominaisuudella voidaan lukea ja kirjoittaa orjalaitteiden tilat ethernetin lävitse. Sisäinen logiikka palaa käyttöön, kun Modbus-transaktio on tehty.

Tavoitteessa mainittu kompensaattori jäi toteuttamatta. Sen tilalle tuli suora ja mux. Tämä perustui ajatukseen, että käyttäjä voi haluta tehdä erilaisen monimutkaisen rakenteen kuin itse olen ajatellut. Kuitenkin tarjoamalla yksinkertaiset rakennustarpeet käyttäjä voi rakentaa

tarpeeseensa niin monimutkaisen kompensoinnin kuin tarvitsee.

Suunnitelmana esitetään se, mitä on lopulliseen versioon suunniteltu. Lopullisen version toteutuksesta on vielä tuotu tapahtuneita muutoksia ja huomioita takaisin suunnitelmaan, jotta se vastaa todellisuutta.

6.3. Järjestelmän kuvaus

Kuten kohdassa 6 mainittiin, oli käytössäni isoja ja pieniä papereita siitä miten järjestelmä oli ajateltu toimivan. Lisäksi olin kirjoittanut käyttöohjetta koko ajan toteuttaessani järjestelmää. Ensimmäiset vuoden 2009 syksyllä piirretyt suurten linjojen kuvat olivat aivan erilaisia verrattuna toteutukseen. Kuitenkin noista papereista oli esitettävä jonkinlainen järjestelmän kuvaus.

Käyttötapausten kuvaaminen oli helppoa. Ne vastasivat kysymykseen mitä tällä laitteella voi tehdä ulkoapäin ja minkälaisia käyttäjiä tällä laitteella on ja toteutuvatko vaatimukset käyttötapauksilla.

Tilakoneena järjestelmän mallintaminen oli haastavaa. Järjestelmän luonne rinnakkaisena prosessiryhmänä ei soveltunut helposti tilakoneena ajateltavaksi kokonaisuudeksi. Lähes kaikista tiloista olisi voinut siirtyä mihin tahansa tilaan ja kuvaus olisi ollut täysin käsittämätön.

Kuitenkin ohjelmiston lähdekoodia ja kaikkia tekemiäni suunnitelmia verratessa pystyin sanomaan mitä tapahtuu. Näistä lähdin luomaan sekvenssikaavioita ja dokumentoimaan miten järjestelmä toimi kaikkien tietorakenteidensa ympärille rakennettujen operaatioidensa kanssa.

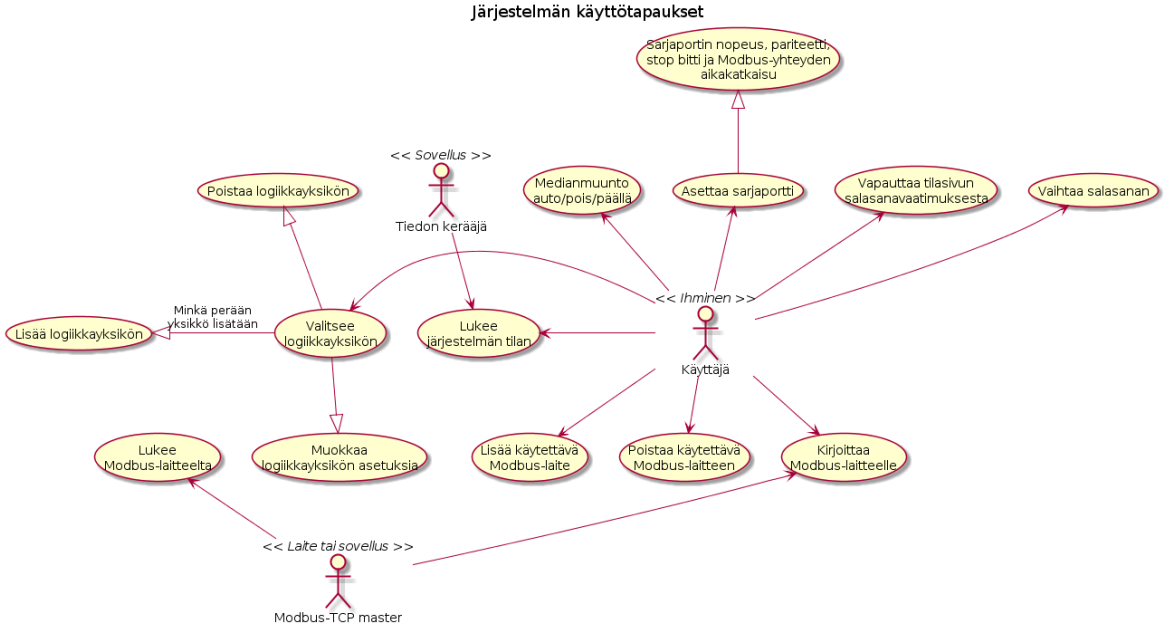
6.3.1. Käyttötapaukset

Järjestelmällä on kaksi käyttötapaa. Ne tulevat suoraan vaatimuksista. Ne ovat median muunto ja itsenäinen toiminta logiikkana sekä näiden summana automaattinen median muunto. Kuvasta 6 puuttuu kokonaan sellainen tilanne, että järjestelmä itse kirjoittaa Modbus-laitteelle. Lisäksi käyttötapauskuvassa on myös kolmantena toimijana tiedon kerääjä. Tämä toimija on syntynyt testauksen sivutuotteena. Järjestelmän koko tila tarvitsi lukea ja tämä tieto käsitellä. Oli yksinkertaista muuttaa käyttöliittymää ja sallia pääsy yhdelle sivulle ilman salasanaa. Lisäksi käyttöliittymän salasanan vaihto sivulla oleva tilasivun vapautus salasanavaatimuksesta on myös tähän liittyvä ominaisuus. Tieto järjestelmän tilasta on ihmiselle ystävällisessä

tekstimuodossa, vaikka onkin tarkoitettu automaattisesti käsiteltäväksi. Tämä ominaisuus on käsitelty liitteessä 1.

Kuva 6 käyttötapaukset eivät ole toisistaan riippumattomia. Toisten asioiden tulee tapahtua ennen kuin toisia voi tapahtua. Jos sarjaportin asetuksia ei tee, niin on turha tehdä mitään Modbus-laitteisiin liittyvää. Kommunikointi sarjaportin kanssa ei vain onnistu ja Modbus-laitteita ja logiikkaa ei päivitetä.

Myöskään sisään- ja uloskirjautumista ei ole tuohon kuvaan laitettu, koska sen lisäarvo on vähäinen ja muutenkin kuva on monimutkainen. Käyttäjän tulee siis olla kirjautuneena sisään, jotta voi tehdä laitteelle muuta kuin mahdollisesti katsoa sen tilaa.



Kuva 6. Järjestelmän käyttötapaukset

Sisäinen logiikka on rakentunut palasista. Tällainen palanen, josta käytetään nimeä logiikkayksikkö, pitää sisällään tietoa ja siihen liittyvän operaation. Logiikkayksiköt muodostavat ketjuja, jotka päivitetään järjestyksessä. Toiminnan kannalta ei ole merkitystä, onko logiikkayksikön sisääntulona toimivan toisen logiikkayksikön sijainti ennen vai jälkeen lukevaa logiikkayksikköä. Sillä on kuitenkin vaikutus saatuun lopputulokseen, koska myöhemmin olevalla logiikkayksiköllä on vielä edellisen päivityksen tiedot.

Huomion arvoinen kohta kuvassa 6 on, kun käyttäjä valitsee logiikkayksikön. Luodessa ensimmäistä logiikkayksikköä, valinta on tehty valmiiksi uudeksi ketjuksi. Muulloin logiikkayksikkö luodaan valitun jälkeiseksi logiikkayksiköksi. Toinen ominaisuus tuossa on, että suunnitelma, toteutus ja käyttöliittymä eivät aivan kohtaa. Poistettaessa logiikkaa käyttöliittymässä mennään toiselle sivulle, joka on hieman eri näköinen, kuin lisäys ja muokattavan yksikön valintasivu. Toteutus kuitenkin on lähes sama. Käyttäjän valittua logiikkayksikön, sen arvoja

voidaan tarkastella ja muokata valitsemalla lisäys ja muokkaus-sivulta katso nappi.

Modbus-TCP isännän asema on myös huomion arvoinen. Koska käyttäjä voi kirjoittaa Modbus-laitteelle käyttöliittymästä, on Modbus-TCP isännälle lisätty lukeminen. Missään muussa niitä ei eroteta ja Modbus-TCP isännän toimintoihin ei edes laite ota kantaa. Modbus-TCP isäntä voi lähettää viestin, jota ei käytetyllä Libmodbus-toteutuksella voi luoda. Median muunto on esitelty tarkemmin kohdassa 5. Käyttäjä ei voi erityisesti sanoa, että lue tietty laite, vaikka Modbus-laitteiden viimeksi luettu arvo näkyykin. Itse järjestelmä lukee nuo laitteet määräajoin. Käyttäjän kirjoitus menee kuitenkin eteenpäin heti seuraavassa päivityksessä.

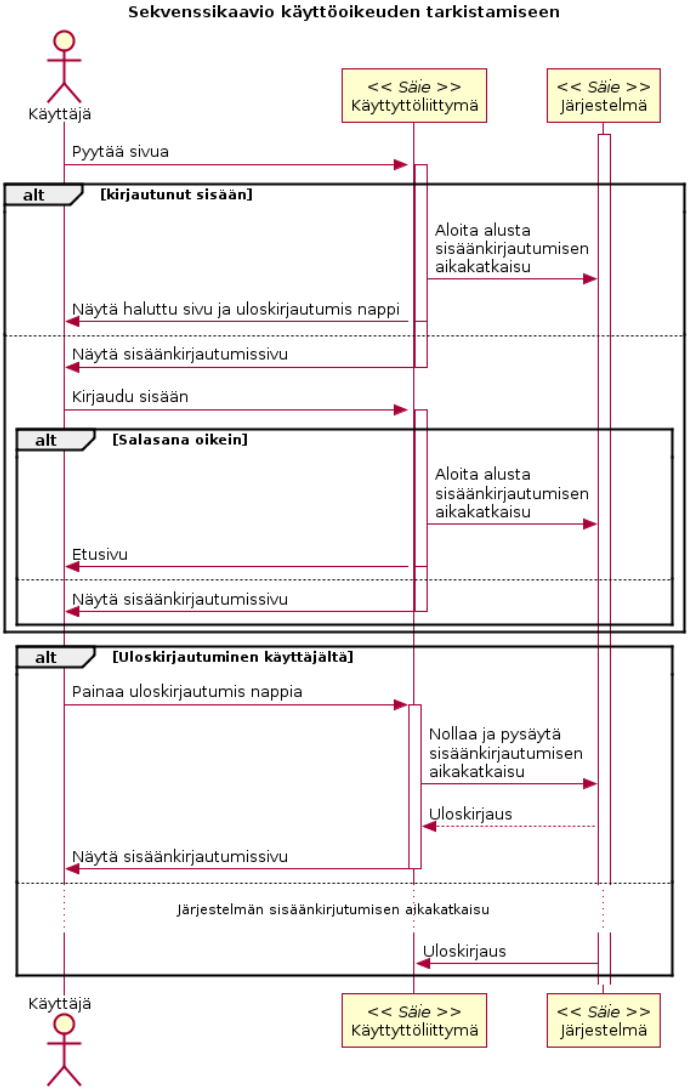
Käyttöohje toteutuksen dokumentaationa antaa hyvän käsityksen, miten eri toiminnot ovat käytettävissä ja riippuvaisia toisistaan. Lisäksi siitä löytyy kuvaukset käyttöliittymälle. Käyttöohje on liite 1.

6.3.2. Sekvenssikaaviot

Sekvenssikaavioissa kuvataan miten tapahtumat kulkevat järjestelmässä. Järjestelmässä on säikeitä. Ensimmäisissä kuvissa käsitellään käyttöliittymää ja järjestelmäsäiettä ja niiden välistä kommunikointia. Lisäksi eri tietorakenteita esiintyy kuvissa, koska ne ovat tapa välittää tietoa eri säikeiden välillä. Kommunikointi voi tapahtua viesteillä tai suorittamalla funktioita. Sekvenssikaavioissa on aktiivisuudella kuvattu yhtäjaksoista toimintaa. Esimerkiksi käyttöliittymä ei voi suorittaa mitään toimintoa ilman käyttäjän toimenpiteitä ja silloin kyseinen säie, joka on itseasiassa laitteen seittipalvelin, on pysäytettynä ja käy vain välillä tarkistamassa tilanteen uusien yhteyksien varalta. Lisäksi käyttäjä voikin päättää keskeyttää sekvenssin siirtymällä toiselle sivulle. Kohdassa jossa kyseinen toimenpide on mahdollinen on käyttöliittymän aktiivisuus poistettu. Järjestelmäsäie muodostaa synkronoinnin ja tiedon välityksen käyttöliittymän ja sisäisen logiikan välille. Järjestelmäsäie on suurimman osan ajasta odottamassa päivitysajastimen laukeamista ja viestejä tiedon tai tilan muuttamiseksi ja välittämiseksi toisille säikeille. Tuona odotusaikana muut säikeet saavat ajovuoron.

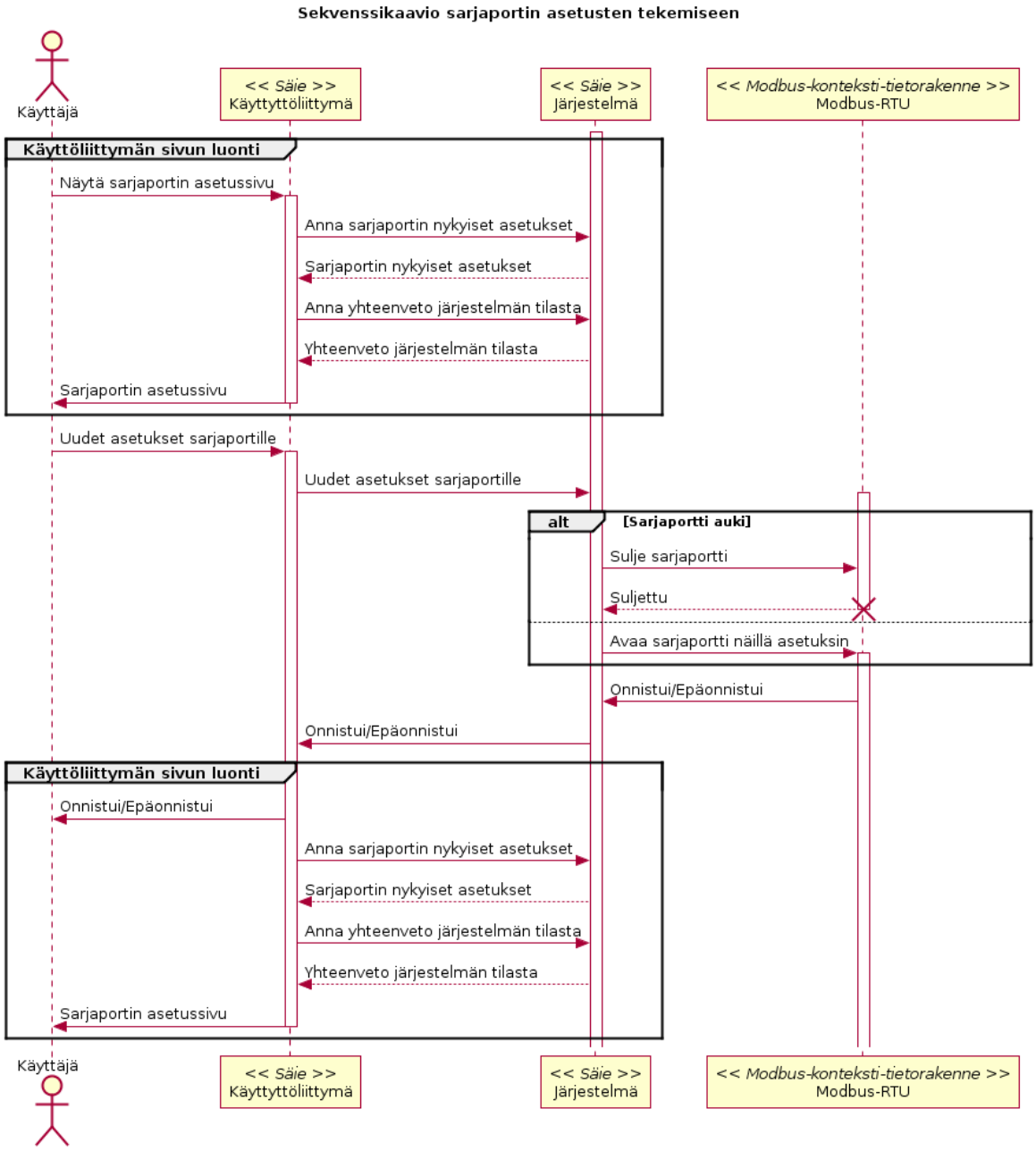
Joka kerta, kun käyttäjä pyytää järjestelmästä sivua, tapahtuu kuvan 7 mukainen tapahtuma järjestelmässä. Poikkeuksena mainittakoon mahdollisuus ohittaa tämä tarkastus logiikan tila -sivulta. Ensin tarkistetaan, onko käyttäjä sisäänkirjautunut ja jos on niin aloitetaan aikakatkaisun laskeminen alusta ja näytetään haluttu sivu. Mikäli käyttäjä ei ole kirjautunut sisään, näytetään sisäänkirjautumissivua, kunnes käyttäjä syöttää oikean salasanan. Oikean salasanan syötön jälkeen näytetään etusivu. Mikäli sisäänkirjautuminen vanhenee, järjestelmä ilmoittaa tästä käyttöliittymälle. Käyttäjä voi myös itse kirjautua ulos ja viestiksi onnistuneesta tapahtumasta saa sisäänkirjautumissivun. Käyttöliittymä tietää, onko käyttäjä kirjautunut sisään, mutta koska säikeenä sitä suoritetaan vain kun käyttäjä pyytää sivuja

tai lähettää tietoa. Tämän ongelman ratkaisemiseksi jatkuvasti suoritettava järjestelmäsäie muuttaa aikakatkaisussa käyttöliittymän tilaa ja pitää huolen ajasta.



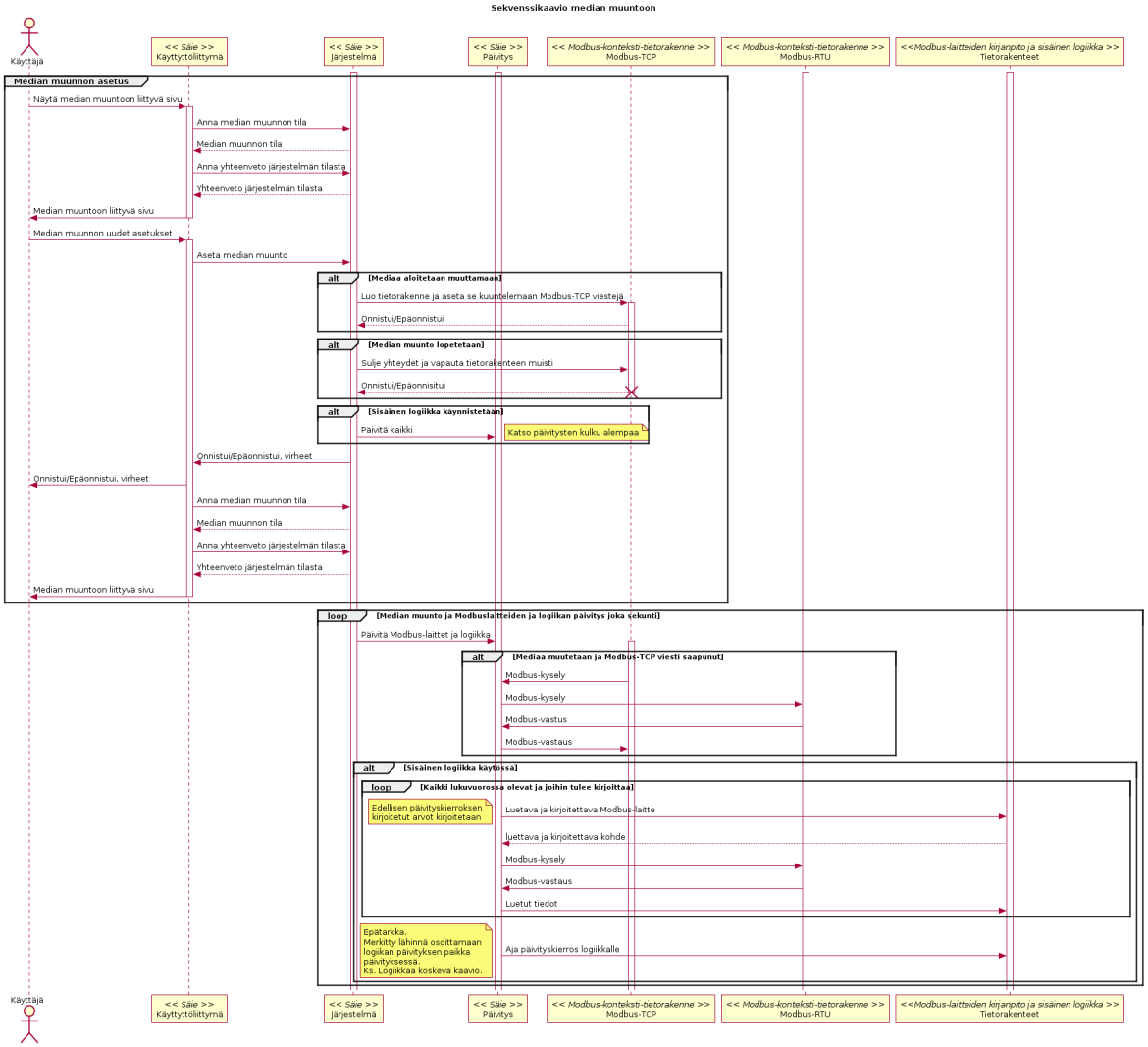
Kuva 7. Sisäänkirjauksen sekvenssikaavio

Erittäin tärkeä käyttötapaus on sarjaportin asetusten määrittäminen. Kuvassa 8 on esitelty toiminnan tapahtuminen. Käyttöliittymä kyselee tietoa järjestelmästä ja järjestelmä palauttaa tiedot sarjaportin tilasta. Tämän jälkeen käyttäjä syöttää uudet tiedot ja käyttöliittymä lähettää ne järjestelmälle. Järjestelmä sulkee sarjaportin käyttäen Modbus-kontekstia. Tämän jälkeen kyseinen konteksti tuhotaan ja muisti vapautetaan. Uusi konteksti luodaan annetuilla sarjaportin arvoilla ja portti avataan. Modbus-konteksti on tietorakenne, jota käytetään kaikkeen Modbus-liikenteeseen ja se on osa järjestelmää sisältäen käytettävät portin, aikakatkaisun arvon ja toimii tilakoneena sekä osin liityntänä Modbus-protokollan toteutukseen, jota on käsitelty kohdassa 4.2. Modbus-konteksti on myös yhteinen tietorakenne järjestelmäsäikeen ja päivityssäikeen kanssa.



Kuva 8. Sarjaportin asetusten sekvenssikaavio

Median muunnon asettaminen ja päivitystapahtumat on kuvattu kuvassa 9. Asetusten syöttö on hyvin saman kaltainen, kuin sarjaportin asetusten syöttö. Siinä luotava ja tuhottava kohde on Modbus-TCP konteksti, joka vastaa ethernetin puolelta tulevasta Modbus-kyselyistä. Lisäksi, jos siirrytään tilasta vain muunto johonkin toiseen tilaan, kaikki olemassa olevat Modbus-laitteet päivitetään. Muulloin päivitys tapahtuu siten, että tutkitaan, onko saapunut Modbus-TCP-viestiä ja sen jälkeen se välitetään ja siihen vastataan. Modbus-laitteilla on tietty päivitysväli. Päivityssekvenssi kuitenkin suoritetaan joka sekunti. Tästä seuraa pieni viive kirjoituksessa. Mikäli esimerkiksi sekunnilla 5 on luettu Modbus-laite ja päivitetty logiikka, joka on kirjoittanut tähän laitteeseen, kirjoitetaan se vasta sekunnilla 6 sarjaväylän

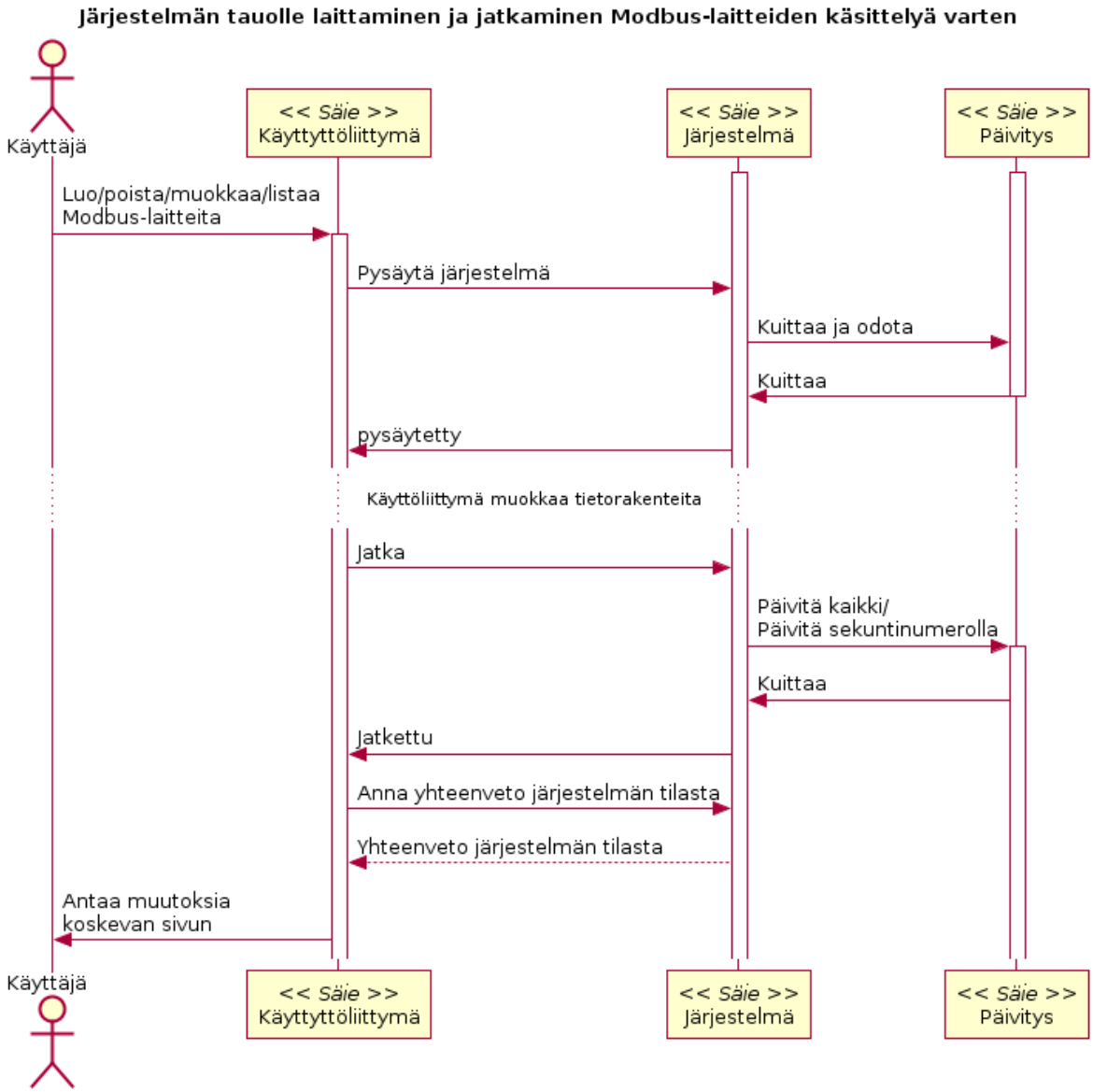


Kuva 9. Median muunnon ja päivityksen sekvenssikuva

ylitse. Tämä skaalautuu sitten päivitysvälin monikertoihin. $N * päivitysväli = lukuaika$ ja $N * päivitysväli + 1 = kirjoitusaika$

Käyttötapaukset vaihtaa salasanan ja vapauttaa tilasivun salasanavaatimuksesta muuttavat käyttöliittymän sisäistä tilaa ja toimintaa. Niistä ei ole tarpeen piirtää sekvenssikaaviota. Ainoa kommunikaatio järjestelmään on moneen kertaan esitetty järjestelmän tilan yhteenvedon hakeminen.

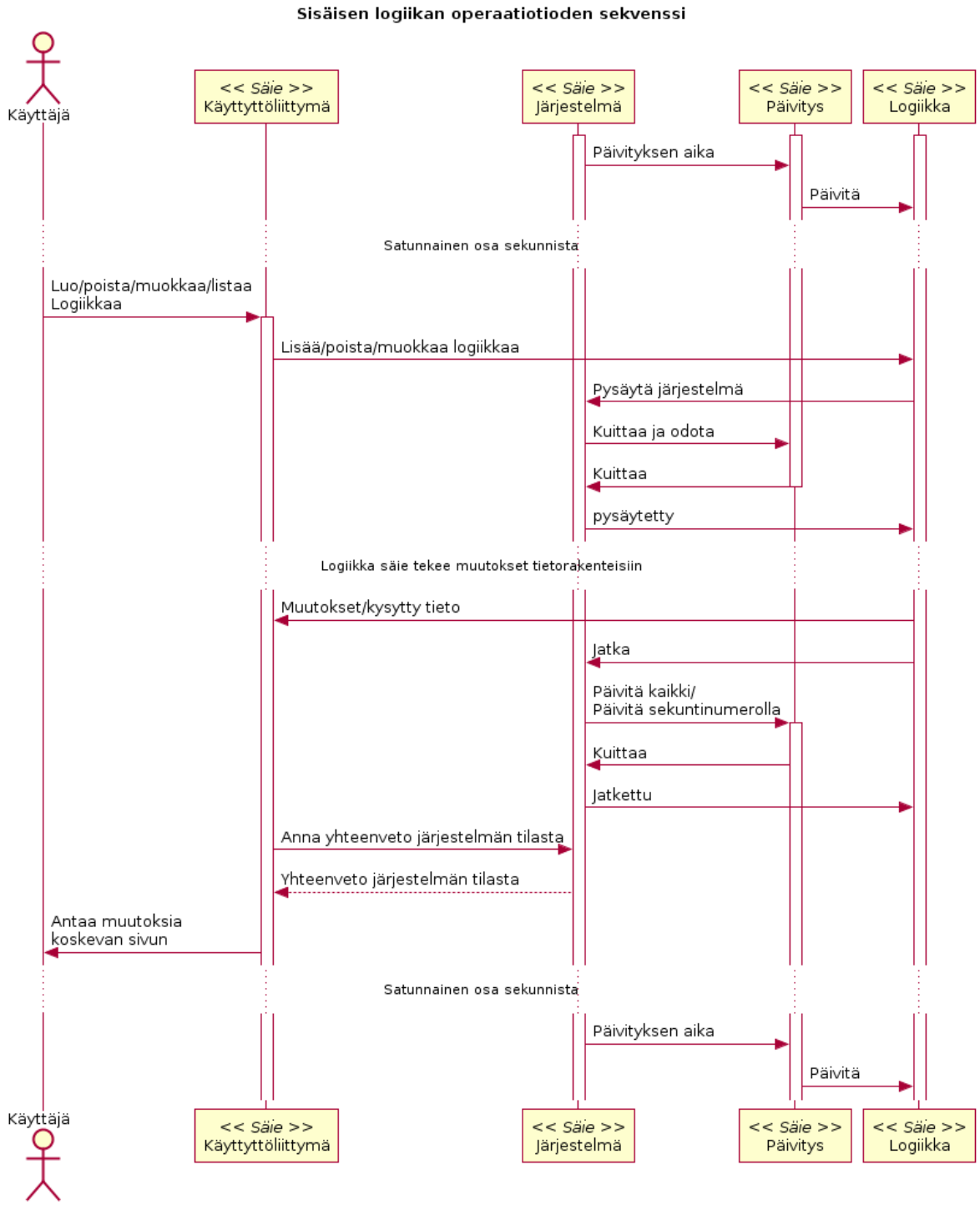
Koska viestien välityksen luotettava toteuttaminen on haastavaa käytetyssä alustassa, on otettu käyttöön kuvassa 10 esitelty pysäytys Modbus-laitteiden käsittelyä varten. Tämän kanssa päästään yhdellä viestillä järjestelmään ja kuittauksella viestin perille menosta. Ongelman on jokin kummallisuus joko muistinhallinnassa tai ohjelmointivirhe viestien välityksessä. Kuva 10 osoittaa, kuinka järjestelmä pysähtyy ja antaa käyttöliittymäsäikeelle mahdollisuuden muokata Modbus-laitteiden tietorakenteita. Lupa saadaan, kun päivitys on pysähtynyt. Käyttöohje, joka on liitteenä 1 varoittaa perustellusti miten tämä ominaisuus voi saattaa koko järjestel-



Kuva 10. Järjestelmän keskeytys Modbus-laitteiden käsittelyä varten

män vastaamattomaan tilaan. Tuo mahdollinen jumiutuminen ei ole helposti ratkaistavissa, sillä aina ja joka kerta seurauksena olisi muistiongelma. Odottamatta päivitystä voitaisiin vapauttaa käytössä olevaa muistia. Pysäyttämällä väkisin päivityssäikeen ja tuhoamalla sen, päivitys pysähtyisi, mutta samalla se voisi tapahtua kesken kirjoituksen ja taas muisti olisi sekaisin. Paras tapa on siis kertoa käyttäjälle, että ole varovainen ja tee oikein kerralla.

Vielä ei ole käsitelty logiikkaa juurikaan. Logiikka on yksi säie, joka hallinnoi niitä tietorakenteita, joilla toteutetaan ohjelmoitava logiikka. Kuvassa 9 ollut huomautus selvennetään kuvassa 11. Tämän syy on hyvin yksinkertainen. Logiikka on kehitetty PC:n päällä ja se on käytännössä ohjelmointivirheetön.



Kuva 11. Logiikan sekvenssikaavio

6.3.3. Käyttöliittymä

Käyttöliittymä tarjoaa selaimelle sivut ja tulkitsee käyttäjän selaimen kautta antamat syötteet. Itse palvelu on käyttöjärjestelmän tarjoama. Sivujen tarkoitus on olla mahdollisimman kevyet. Ne ovat pelkkää tekstiä. Käyttöliittymä ja sen tekemät suoritetaan hypertekstipalvelun säikeen alla. Selaimessa ajetaan hyvin vähän Javascriptiä.

Käyttöliittymä koostuu kehyksistä ja sen yhdessä sivussa olevista taulukoista tai tiedoista. Itsenäinen logiikka ja Modbus-laitteet esitetään taulukossa. Median muunto, salasanan vaihto ja sarjaportin asetusten asettaminen ovat lähinnä tekstisivuja. Kuvia ja kuvauksia käyttöliittymästä löytyy liitteestä 1.

Vapaita tekstikenttiä käyttöliittymässä on kirjautumissivulla, salasanan vaihdossa, laitteiden nimissä ja Modbus-laitteiden osoitteena. Näin on saatu karsittua tekstin jäsentämisoperaatioita minimiin.

Käyttöliittymä ei kuitenkaan pysy pelkästään omalla tontilla. Kohdassa 7.2 mainitun viestijärjestelmän ongelman kanssa takia ei riskiä otettu ja sen sijaan, että käyttöliittymä välittäisi viesteillä järjestelmä säikeelle esimerkiksi Modbus-laitteiden lisäykset, se pysäyttää järjestelmän yhdellä viestillä kuten kuvassa 10 on esitetty. Vastauksen käyttöliittymä saa lippuryhmästä ja kuittaamalla lipun järjestelmä jatkaa toimintaa. Neljästä viestistä tuli siis yksi.

Lähes aina käyttöliittymä toimii ensin antaen sivun, jolle käyttäjä syöttää tietonsa. Sivun luontiin on voitu vaatia logiikan ja median muuntamisen pysäytystä. Tulostuksen päätyttyä, annetaan logiikan päivitysten ja muunnoksen jatkua. Kun käyttäjä lähettää tietoja, ensin mahdollinen Javascript-koodi tarkastaa syötetyt arvot. Tämän jälkeen sivun syötteitä käsittelevä funktio lukee annetut arvot ja tutkii, ovatko ne laillisia. Mikäli ne eivät ole, tulostetaan virheilmoitus ja syöttösivu uudelleen. Jos annetut arvot ovat hyväksyttäviä ja aiheuttavat muutoksia, lähetetään järjestelmälle pysäytyskomento Modbus-laitteiden muutoksissa tai sarjaportin asetusten tai median muunnoksen kyseessä ollessa uudet arvot järjestelmälle. Logiikan muutoksissa syötetyistä tiedoista luodaan jäsennetty viesti ja se lähetetään logiikalle, joka sitten pysäyttää järjestelmän, tarkastaa arvoja vielä tarkemmin, kuten esimerkiksi olemassa olevaa logiikkaa vastaan ja tekee muutokset, tulostaa käyttäjälle tietoa ja vapauttaa järjestelmän takaisin toimintaan.

6.3.4. Itsenäisen logiikan Modbus-laitteiden kirjanpito

Itsenäisen logiikan alle kuuluu kaksi osaa. Ensimmäinen on Modbus-laitteiden kirjanpito ja toinen on varsinainen logiikka. Modbus-laitteiden kirjanpidossa pidetään yllä käytettävistä Modbus-laitteista laite, osoite, tyyppi, päivitysväli, vain luettavuus, kirjoitus ja luku funktio tuki ja käyttäjälle tunnisteena nimi tietoja. Modbus-laitteiden kirjanpito on yksinkertainen linkitetty lista. Siinä on hyvin vähän mahdollisuuksia muokkaukseen ja sen tärkein ominaisuus on järjestys. Lista tulee olla järjestetty laitteen, tyypin, päivitysvälin ja osoitteen mukaan jossa laite on eniten merkitsevä. Tästä kirjanpidosta sitten luodaan ne taulukot, joita Libmodbus käyttää lukiessaan ja kirjoittaessaan. Lisäksi kirjanpidon tulee pitää yllä tietoja 32-bittisistä tiedoista. Ne vaativat kaksi osoitetta ja päällekkäisyyksien estämiseksi tarvitsee tarkistaa lista edeltävältä 32-bittselta tyypiltä ja 32-bittisen tyypin ollessa kyseessä myös seuraava osoite tulee varata käyttöön.

Kirjanpidossa päivityksen tapahtuessa, kirjoitetaan ensin logiikan kirjoittamat Modbus-laitteet uusilla arvoilla ja sitten luetaan lukuvuorossa olevat Modbus-laitteet. Mikäli jokin kirjanpidossa oleva lukuryhmä, joka siis koostuu samasta laitteesta, tyypistä, päivitysvälistä ja peräkkäisistä, korkeintaan 125 rekisteristä tai 1968 kelasta, sisältäisi kirjoittamattomia tietoja, kirjoitettaisiin ne ensin ja sen jälkeen luettaisiin. Rekisterien tapauksessa olisi mahdollista käyttää Modbus-protokollan kirjoita ja lue rekistereitä Modbus-funktiota.

Kirjoita ja lue rekistereitä Modbus-funktiota käytetään lähinnä ainoastaan käyttöliittymästä kirjoittamisessa. Sisäisen logiikan kirjoittaessa tietoja Modbus-laitteille, ne kirjoitetaan vasta seuraavalla Modbus-laitteiden kirjanpidon päivitysyrityksellä noin sekunnin kuluttua. Kirjoita ja lue rekistereitä-funktion käyttö voisi tapahtua, mikäli orjalaiteille kirjoitettaisiin vain lukutapahtuman yhteydessä. Lisäksi kirjoitettavien rekistereiden tulisi olla eri rekisterit kuin luettavat, koska kirjoitus tapahtuu ensin ja vasta sitten luku.

Modbus-protokollan kirjoita ja lue rekistereitä Modbus-funktiota toivottiin toteutettavaksi erityisesti. Ongelma on vain sen käyttö. Mikäli se toimisi toisin päin eli lukisi ja sitten kirjoittaisi, olisi sen käyttö helpompaan. Logiikka haluaa toimiakseen sisään tietoa. Tieto käsitellään ja se tulee ulos logiikasta. Logiikan päivitysrytmi on sama kuin Modbus-laitteiden. Ratkaisuna tähän olisi kirjoittaa myös samaan aikaan, kun luetaan. Nyt kuitenkin kirjoitus tapahtuu heti seuraavalla sekunnilla. Tällä on haluttu tasata kuormaa. Tarkemmin päivitysprosessi on käsitelty liitteessä 1. /5/

Modbus-laitteiden kirjanpito pitää siis yllä tietoa muutetuista, muttei kirjoitetuista osoitteista. Myös kirjoitussuojauksen tieto sijaitsee tässä kirjanpidossa. Kaikilla samaan laitteeseen kuuluvilla Modbus-laitteilla menee tieto 0x17 tuesta, jonka voi kytkeä pois päältä, kuten käyttöohje kertoo liitteessä 1. Lisäksi kirjanpidossa on tietona arvo 32-bittisenä, päivitysväli ja

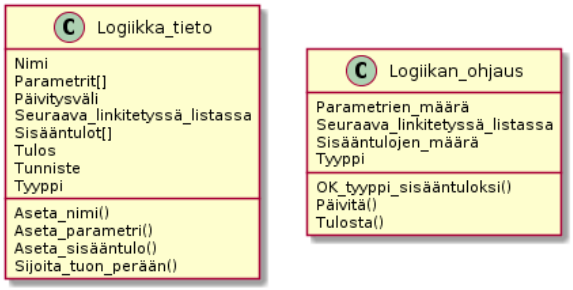
nimi. 32-bitillä voidaan myös ilmaista helposti 8-bittinen laitenumero, 16-bittinen osoite ja lopulla 8 bitillä tyyppi. Näistä luodaan tunniste jonka, jälkeen listan järjestäminen on helppoa. Yhdellä kirjanpidon elementillä tulee olla myös tieto siitä, mihin lukuryhmään kyseinen elementti kuuluu, jotta sitä ei tarvitse laskea joka kerta päivityksessä. Lisäksi jo mainittu 32-bittiä löytyy Modbus-laitteen arvosta. Modbus-laite voi kirjoittaa ja lukea 32-bittisiä kokonaislukuja ja liukulukuja yksinkertaisella tarkkuudella. Tämä tarkoittaa sitä, että sisäinen logiikka laskee kaksinkertaisella tarkkuudella verrattuna Modbus-laitteille välitettyä tarkkuutta.

6.3.5. Itsenäinen logiikka

Itsenäinen logiikka on perinteisen tikapuu- ja sarjamuotoisen ohjelmoinnin sekoitus. Itsenäisessä logiikassa ei ole selkeitä rinnakkaisia polkuja, jotka toteutettaisiin suoraviivaisesti jokaisella päivityskerralla. On kuitenkin mahdollista luoda tällaisia ohjelmointeja logiikalle. Logiikan päivitys alkaa ensimmäiseltä riviltä kulkien vasemmalta oikealle. Kuitenkaan ohjelman ei tarvitse olla mitenkään kiinni tästä järjestyksestä. Ensimmäinen logiikkayksikkö, joka koostuu tiedosta ja jostain määritellystä operaatiosta, voi ottaa sisääntulokseen vaikka viimeisen logiikkayksikön. Suurta huolellisuutta tulee kuitenkin käyttää tämän voimakkaan mahdollisuuden kanssa. On mahdollista luoda ohjelmalle logiikka, joka ensimmäiseksi kirjoittaa ulos koko ketjunsa tiedot ja ottaa tietoa operaatioihinsa sisään vasta viimeisen yksikön päivityksessä. Tällöin tapahtuu edellisen päivityksen kirjoitus, päivitys ajaa itsensä edellisen päivityksen tiedoilla ja vasta sitten lukee uudet tiedot. Vaihtamalla luvun ja kirjoituksen paikat, logiikan vasteaika menee kolmanteen osaan. Tätä ajatusmallia on käsitelty käyttöohjeessa liitteessä 1.

Tämän kaltainen ongelma oli mitä selkeimmin ratkaistavissa olio-ohjelmoinnilla. Ratkaisuna olisi tietoluokka ja ohjausluokka. C-kielessä ei ole luokkia, joten tilannetta tarvitsi lähestyä toiselta suunnalta. Ratkaisuksi tuli lopuksi struct tietorakenne, funktio-osoittimet ja käännösvaiheessa tapahtuva makrojen käyttö. Itsenäisellä logiikalla on kaksi tärkeää tietorakennetta. Ensimmäinen on itse tiedon rakenne ja toinen on sitä ohjaava rakenne. Yksinkertaisena esimerkkinä käänteisellä puolalaisella merkintätavalla a b 2 ++ voidaan ajatella kolmen yhteenlaskettavan yhteenlasku operaattorin, jollaista itsenäisellä logiikalla ei ole, olevan ohjaava rakenne ja laskussa olevat muuttujat kuvaavat tietoa sisältäviä rakenteita ja ovat siis sisääntuloina yhdessä tietoa kuvaavassa rakenteessa. Tulokseksi tuleva a+b+2 menee tietoa kuvaavaan rakenteeseen kohtaan tulos. Muuttujat on valittu esimerkkiin juuri siksi, että ne ovat juurikin toisia tietoja kuvaavia rakenteita. Numero 2 taas on kiinteä numero, joka on tarkastelemamme tietoa sisältävän tietorakenteen parametreissa. Kuva 12 esittää luokkakaavioina nämä tietorakenteet. Kuitenkin C-kieli ei tue luokkia ja toteutuksessa on luokkien metodit muutettu tavallisiksi funktioiksi ja attribuutit structin jäseniksi. Lisäksi luokista puuttuu jäseniä, joten kuvaa on tulkittava lähinnä suuntaa antavana suunnittelukuvana.

Nämä tietorakenteet ovat sijoitettuna omiin linkitettyihin listoihinsa, joista voidaan sitten hakea haluttu jäsen aina käsiteltäväksi. Logiikan_ohjaus tyypin päivitä ja tulosta metodit ovat määritelty funktio-osoittimilla tuohon tietorakenteeseen. Molemmilla tietorakenteilla on yhtenevä attribuutti tyyppi. Se sitoo logiikan datan ja ohjaavan tietorakenteen yhteen. Samalla tarvittu muistin määrä vähenee, kun tiedon tyyppistä riippuvaa tietoa ei kopioida muistiin itse tiedon kanssa.



Kuva 12. Luokkakaavio mahdollisesta logiikan toteutuksesta

Logiikan tietorakenteella on parametrit taulukkona. Niitä käytetään päivityksessä, joka tapahtuu päivitysvälin väliajoin. Logiikan sisääntulot ovat myös taulukossa. Taulukon koko on määritelty käännösvaiheessa ja siihen vaikuttaa, minkä tyyppinen logiikka luodaan. Sisääntulot logiikalle ovat osoittimia toisiin samanlaisiin tietorakenteisiin. Logiikan_tieto luokassa on myös määritelty tunniste. Se kertoo, missä kohtaa muiden joukossa kyseinen olio on.

On selvää, että ilman liimalogiikkaa ei tämän kaltainen järjestelmä voi toimia. Mukaan tarvitaan luontiin poistoon ja manipulointiin funktioita, jotka saavat logiikan toimimaan. Sisääntulon lisäämiseen tarvitaan esimerkiksi OK_tyyppi_sisääntuloksi() funktio. Virheen tarkastelu ja järjestely linkitettyyn listaan on myös tehtävä, joka vaatii näiden luokkien ulkoista ohjelmointia.

Sisäinen logiikka, joka perustuu näihin kahteen tietorakenteeseen, ei ole ottanut mitenkään kantaa Modbus-toteutukseen. Operaattoreina löytyvät sisääntulo ja ulostulo vain hakevat ja vievät tiedon Modbus-toteutuksen kirjanpidon välillä. Syy on hyvin selkeä. On helpompi pitää kirjaa perättäisistä luettavista ja kirjoitettavista osoitteista. Käyttöohjeen lopussa, mikä on liite 1, on selitetty, kuinka tämä vaikuttaa tehokkuuteen.

Pelkillä tietorakenteilla ei sisäinen logiikka pelkästään toimi. Jo kuvassa 12 oli metodeja. Nämä luokkakaavion metodit ovat kuitenkin joukko erilaisia funktioita, jotka luovat, poistavat, muokkaavat, päivittävät tai tulostavat sisäisen logiikan tilaa.

Luotaessa itsenäiseen logiikkaan logiikkayksikkö, tarkistetaan arvojen järkevyys ja sen jälkeen varataan muistialtaasta tila sille. Tämän jälkeen uudelle yksikölle annetaan käyttäjän syöttämä nimi, päivitysväli ja tyyppi. Tyypin perusteella ohjaavasta tietorakenteesta haetaan

oletusarvot. Samalla varataan muistit sisääntulojen osoittimille ja parametritaulukolle. Kun uusi logiikkayksikkö on luotu, asetetaan se linkitettyyn listaan omalle paikalleen.

Logiikkayksikön poistaminen tapahtuu tarkistamalla käytetäänkö kyseistä logiikkayksikköä sisääntulona jollekin toiselle logiikkayksikölle. Mikäli kyseinen logiikkayksikkö ei ole minkään sisääntulona, irrotetaan se linkitetystä listasta ja vapautetaan sen käyttämät muistialueet.

Muokatessa logiikkayksikköä ensin tarkastetaan käyttäjän antamat parametrit. Parametreilla voi olla rajoitteita, kuten virhe ala pitää olla pienempi kuin varoitus ala. Lisäksi ohjaavaan tietorakenteeseen voidaan määritellä erityinen parametrin päivitysfunktio, joka ajetaan tässä vaiheessa. Oletuksena parametrit vain kirjoitetaan oikeaan kohtaan. Myös sisääntulojen tietoja muokattaessa kaikki lähtee tarkastuksista. Järjestelmä tutkii onko sisääntuloksi ehdotettua logiikkayksikköä olemassa ja onko se kelvollinen muokattavalle logiikkayksikölle. Ohjaavasta rakenteesta tutkitaan onko edes halutussa indeksissä sisääntuloa. Kahden sisääntulon logiikkayksikköön ei voi laittaa kolmannen sisääntulon kohtaan mitään. Silloin tapahtuisi kauheita. Tarkastamisen jälkeen muutetaan sisääntulotaulukon osoitin haluttuun uuteen sisääntuloon. Virhetilanteessa tulostetaan virheilmoitus ja palautetaan ajo käyttöliittymälle.

Päivitys tapahtuu joka sekunti. Mikäli päivitysaika on jaollinen logiikkayksikön päivitysvälin kanssa, päivitetään se. Logiikkayksiköt käydään lävitse järjestyksessä ja tutkitaan tuota päivitysaikaa. Läpikäyntijärjestys on vasemmalta oikealle käyttöliittymän logiikkayksiköiden listauksessa tai riveittäin ylhäältä alas käyttöliittymän logiikan tila sivulta saadun logic.csv tiedostossa. Joka tapauksessa läpikäynti tapahtuu tietyssä invariantissa järjestyksessä. Käyttöohjeessa on tarkemmin käsitelty käyttäjän kannalta oleellisia asioita päivityksestä. Käyttöohje on tämän raportin liite 1. Kun logiikkayksikköä päivitetään, haetaan ohjaavasta tietorakenteesta funktio-osoitin päivitysfunktioon. Päivitysfunktio lukee sisääntulot, käyttää parametreja ja asettaa tilan ja tuloksen. Lisäksi se voi muuttaa logiikkayksikön tulkintaa. Mikäli päivitysfunktio epäonnistuu, logiikkayksikön tila asetetaan kriittiseksi. Kun päivitysfunktio on palannut onnistuneesti, tutkitaan logiikkayksiköstä sisääntuloja ja asynkronisuuden sallimista. Mikäli asynkronisuus ei ole sallittu ja tila on OK, asetetaan tilaksi varoitus hitaammasta tai nopeammasta sisääntulosta.

Tulostus on eräs monimutkaisimmista logiikan tehtävistä. Tulostus on myös raskain ja aikaa vievin. Tulostuksessa voidaan luoda kolme asiaa: Luonti ja muokkaus taulukkosivu, logiikkayksikön muokkaussivu tai logiikkayksiköiden poistotaulukkosivu. Näistä molemmat taulukkosivut ovat toteutettu lähes samalla koodilla.

Luonti- ja muokkaussivulla käydään lävitse logiikan tietoa sisältävien rakenteiden linkitettyä listaa. Aluksi tulostetaan ei valintaa, uusi ketju rivit lisää yksikkö ketjun alkuun solu. Jokaisen listan jäsenen kohdalla tulostetaan ensin jäsenen tilasta riippuen solun taustaväri. Sitten

tulostetaan tuloksen tulkintaa tarkoittava kirjain, logiikkayksikön nimi ja tuloksen arvo tulkittuna. Tämän jälkeen etsitään ohjaavista rakenteista tyypin nimi ja tulostetaan. Valinta piste lisätään loppuun. Tämän valintapisteen arvo luodaan logiikkayksikön paikasta riippuen linkitetyssä listassa. Nämä operaatiot tehdään jokaiselle logiikkayksikölle. Lopuksi logiikkayksiköiden taulukon alapuolelle tulostetaan pieni luontitaulukko ja palautetaan ajo takaisin käyttöliittymälle. Liitteen 1 sivulla 33 on kuva tulostetusta sivusta.

Mikäli on taulukkosivuna on tulostettu poistotaulukko, ohitetaan ei valintaa ja alkuun valintaruudukkojen tulostus. Myös uusien logiikkayksiköiden luontitaulukko jää pois. Lisäksi käyttöliittymässä syötetyt tiedot käsitellään eri tavalla. Listauksessa ovat vain ne logiikkayksiköt, jotka eivät ole minkään toisen logiikkayksikön sisääntulona. Tämän jälkeen valittu logiikkayksikkö poistetaan järjestelmästä ja sen varaama muisti vapautetaan.

Logiikkayksikön muokkaussivu on monivaiheinen tapahtuma. Kun käyttöliittymä on antanut viestin tehtävistä muutoksista, tulostetaan tulkintojen selitykset ja yleistä tietoa logiikasta. Mikäli ei ole erikseen määritelty otsikoiden tulostusfunktiota ohjaavassa tietorakenteessa, tulostetaan kaikille yhteiset otsikot eli tulokseen asti. Sen jälkeen tulostus kyselee ohjaavalta rakenteelta sisääntulojen ja parametrien määriä ja nimiä. Ne tulevat otsikoiksi.

Otsikko rivin jälkeen tutkitaan jälleen erityisen tietojen tulostusfunktion olemassa oloa. Siinä tulostetaan arvot taulukkoon. Esimerkiksi muuttujalla on tyypin valinta. Sillä voidaan antaa muuttaa tulkinta tyyppi ilman yhden parametrin lisäämistä parametritaulukkoon. Muita omaa tulostusfunktiota käyttäviä ovat sisään ja ulostulot ja input2param operaattorit. Ne ovat dokumentoitu käyttöohjeessa tarkemmin, joka on liite 1. Lopuksi ajetaan jälkitulostus funktio. Siinä taulukko suljetaan, painikkeet luodaan ja jälkikirjoitukset tulostetaan. Matemaattiset operaattorit käyttävät tätä ominaisuutta luodessaan toimintotaulukkonsa.

Näitä toimintalinjoja noudattaen on luotu useita erilaisia operaattoreita. Ne ovat dokumentoitu käyttöohjeessa tämän raportin liitteessä 1. Operaattorien rungot ovat automaattisesti luotuja, joten alusta on kohtuu toimiva. Kehittäjän tulee vain antaa moduulille nimi, parametrien ja sisääntulojen määrä ja yksinkertaisessa tapauksessa toteuttaa päivitysfunktio. Kaikki muu tapahtuu automaken avulla.

7. TOTEUTUS

Toteutusvaiheessa kohtasivat järjestelmän uudelleensuunnittelu, toteuttaminen ja testaaminen. Uudelleensuunnitteluun jouduin turvautumaan uusien ennen tuntemattomien rajoitusten tullessa esille tai valittu algoritmi ei ollut mielestäni tarpeeksi tehokas tai sillä oli ikäviä sivuvaikutuksia.

Toteuttaminen oli melko suoraviivaista. Ensin tein ohjelman, joka antoi hyviä ja huonoja syötteitä ja joka tarkasti tapahtuneen muutoksen. Tämän jälkeen toteutin itse ohjelman osan, jota olin toteuttamassa. Toiminnallinen testaus tapahtui lähes kokonaan Ficl ohjelmaa apuna käyttäen. Luonnollisesti samalla etsittiin muistivuotoja ja varmistettiin testin kattavuus koko testattavalle ohjelman osalle.

7.1. Toteutuksen kulku

Toteutuksen tekeminen aloitettiin tutustumalla Libmodbus-kirjastoon ja kehittämällä siihen kirjoita ja lue rekisterit-funktio. Lisäksi kyseistä kirjastoa korjattiin ja tehtiin hieman vakaammaksi PC:n päällä. Sen jälkeen se siirrettiin kehitysympäristöön. PC:n päällä jatkettiin kehittämistä median muuntoa varten. Kun se oli valmis ja luetettava voitiin sekin siirtää kehitysympäristöön. Seittipalvelimen sivut valmistuivat ja niihinkin alkoi tulla sisältöä ja ominaisuuksia. HTML-koodin standardinmukaisuus aiheuttivat pientä pään vaivaa. Käyttöliittymän käyttäessä kehyksiä ja niistä karkaaminen toivat eteen uuden ongelman. Javascript oli pakko ottaa mukaan, jotta ulos kirjautuessa kehykset saatiin tiputettua pois. Samalla myös eräs ärsyttävä ominaisuus voitiin ratkaista. Jättämällä esimerkiksi uuden logiikan sijoituspaikan tyhjäksi, sivulle ilmaantui virheilmoitus ja annetut arvot olivat syötettävä uudelleen. Pienellä Javascript-koodilla tässä tilanteessa tuli ikkuna eteen, joka kehotti valitsemaan paikan uudelle logiikkayksikölle.

Kun Modbus-kommunikointi ja sen käyttöliittymä toimi, aloin kehittelemään logiikan toteutusta. Erilaisia lähestymistapoja oli monia. Etsiessäni jotain testipenkkiä, jossa testata tulevaa toteutusta, törmäsin Ficl nimiseen ohjelmaan. Se on Forth kääntäjä ja tulkki, jolle voi antaa C-kielen funktioita ajettavaksi. Se ratkaisi ongelman testipenkistä ja tutustuessani sen ominaisuuksiin ja opetellessani Forthia huomasin hyvin saman kaltaisen Ficlissä tietorakenteen, josta tuli sitten logiikkani toteuttavat tietorakenteet./10/

Piirsin hyvin paljon kuvaa 12 muistuttavan kuvan paperin reunaan ja toteutin lisäyksen, poiston ja järjestämisen. Koko kehityksen ajan testasin mahdollisia muistivuotoja ja väärää laskentaa erilaisilla pienillä testiohjelmilla. Tämän jälkeen tein ohjaavan rakenteen. Kaksi tes-

tauksessa käytettävää operaatiota, joista toinen laittoi sisääntulon neliön ja toinen teki jotain muuta. Tässä vaiheessa havaitsin tekeväni toistuvia asioita, jotka olisi syytä antaa tietokoneen tehtäväksi. Kääntäessäni testattavaa järjestelmää olin jo ottanut Automaken käyttööni. Kirjoitin siis pohjan jokaiselle operaattorille ja pienen määrän erilaisia makroja esikääntäjälle. Tämän jälkeen pääsin tilanteeseen, että piti antaa operaattorille nimi, sisääntulojen ja parametrien määrä sekä lopuksi toteuttaa päivitysfunktio. Lisäksi piti kirjoittaa käyttöohjeeseen kuvaus operaattorista. Tuossa automaatiossa oli vielä lisämausteena valmis tyyppien rekisteröinti, joka logiikan alustuksessa kutsuttiin.

Jossain vaiheessa havahduin todellisuuteen ja muistin, että maailmassa on muitakin kuin 16-bittiset etumerkittömät kokonaisluvut. Logiikan olisi kyettävä tulkitsemaan eri tavalla reaalilukuja ja totuusarvoja ja etumerkillisiä lukuja. Tässä vaiheessa näiden ominaisuuksien lisääminen oli helppoa. Logiikan tietoa sisältävään rakenteeseen muutin päivityksestä saadun tuloksen reaaliluvuksi ja lisäsin tiedon miten tulosta tulisi tulkita. Logiikkaa ohjaavaan rakenteeseen lisäsin kentän, jonka perusteella pystyi määrittelemään sisääntulon tuloksen tulkintatyypin. Tämä muutos poisti esimerkiksi ongelman miten määritellä luvun ja epätoden jakolasku, koska mahdollisuus sellaiselle poistui.

Tarvittiin yksi kenttä lisää tietoa sisältävään rakenteeseen kertomaan, miten tulee logiikan tulos tulkitaan ja vähän jokaiselle operaattorille muutoksia valikoimaan mitkä tulkintatyypeistä olisivat sopivia.

Tulostusta virittelin pitkään ja hartaudella. Taulukkorakenteet käyttöliittymässä olivat helposti rikkoontuvia ja aikaa kului paljon standardinmukaisen ja nätin näköisen tulostuksen luontiin. Kaikille operaattoreille ei kelvannut aivan yleinen tulostus tai parametrien asettelu funktiot, joten siihenkin oli tehtävä pieniä muutoksia ja luoda mahdollisuuksia vaikuttaa tulostamiseen.

Kun olin mielestäni tyytyväinen järjestelmän toiminnallisuuteen, kirjoitin Forthilla testiohjelman. Se teki kaikki samat asiat kuin testattava ohjelmakin. Lisäksi se kirjoitti tuloksensa toiselle säikeelle jota vastaan sitten testattavan järjestelmän tuloksia verrattiin. Testauksessa ajettiin kaikki yhdistelmät mahdollisista sisääntuloista, sekä yli ja alivuodot laskennassa. Yksi testiajo kesti noin 95 tuntia pöytäkoneella. Kuukauden kuluttua, testi ei tuottanut virheitä ja järjestelmä oli mielestäni vakaa, siirsin sen kehitysalustalle. Kirjoitin käyttöliittymään yhden napin, joka loi toivomani Modbus-laitteet ja logiikan. Tämän jälkeen pöytäkoneella pyörivä testausohjelma ja vertasi sarjaportista saatua tietoa omiin tuloksiinsa ja havaitsin ohjelman toimivan samalla tavalla molemmissa ympäristöissä. Lisäksi aiheutin testiohjelmasta häiriöitä ja tein median muuntoa samanaikaisesti.

Jossain vaiheessa testausta halusin luoda itselleni visuaalisen käsityksen tapahtumista ja sil-

loin syntyi käyttöliittymään sivu logiikan tila. Sille sivulle olin ajatellut kaikilla sivuilla näkyvän taulukon ja jotain muuta pientä. Kuitenkin päätin kirjoittaa sinne shelli-skriptin joka taltio tiedon. Siitä sitten loin kuvaajia. Totesin sen olevan hyödyllinen ominaisuus ja jätin sen lopulliseenkin versioon. Tämän sivun toiminta on selitetty liitteessä 1.

Viimeinen lisäys oli varoitus asynkronisesta päivitystahdista. Testatessani jotain ominaisuutta käsin onnistuin luomaan itselleni todella kummallisia tuloksia. Syynä oli siis eri aikoina tapahtuvat päivitykset. Ominaisuuden lisättyäni totesin haluavani kytkeä pois moisen automatiikan häiritsemästä, koska asynkronisuus aiheutti varoituksen ja sen erottaminen jostain oikeasta varoituksesta oli mahdotonta nopealla tutkimisella.

7.2. Toteutuksen ongelmat

Toteutuksen ensimmäinen ongelma oli Modbus-laitteita tehdessä. Se struct jossa oli myös union rakenne ei ollut kääntäjän mielestä tunnetun kokoinen, vaikka kaikki tiedot olivat sillä. Tämän ongelman kiertämiseksi Modbus-laitteiden toteutuksesta tuli hieman rajoittunut ja teknisesti ruma. Vasta myöhemmin löysin sen kääntäjän toimintaan vaikuttavan ominaisuuden, jolla alkuperäinen rakenne olisi saatu toimimaan. Vahinko oli tapahtunut ja toteutukseksi jäi tuo ruma versio. Toisaalta ruma toteutus toimi virheettömästi ja teknisesti paremman toteutuksen käyttöön ottaminen olisi vaatinut kaikkien käsittelyfunktioiden uudelleentoteutuksen.

Toinen ei niin kovasti luottamusta herättänyt ongelma oli viestien pallottelu. Järjestelmän mukana toimitetusta esimerkkikoodista sovellettu viestien lähettely sai koko laitteen kaatumaan. Edestakaisin kulkevat viestit saivat aikaan lukualueen ylivuodon väärässä paikassa ja sen jälkeen debuggerista sai lukea rekisterilistausta koko järjestelmän kaatuessa.

Malloc() ja ikuinen silmukka oli mielenkiintoinen ongelma. Kaksi prosessia luotuna ja ne itse luovat oman muistialtaansa käynnistyessään. Ensin ajetaan malloc() kutsu, jolla varataan C-kielellä muisti. Tämän jälkeen siihen muistiin luodaan käyttöjärjestelmän tarjoama muistiallas. Malloc() ei koskaan palannut, ja väitteen ei koskaan on määritelty kahdeksan tunnin yöuniksi. Jumiutuminen liittyi jotenkin Net+OS käyttämän ThreadX kernelin tarjoaman mutexin saamiseen. Tämäkään jumiutuminen ei tapahtunut tietenkään joka kerta. Siksi kaikki säikeet tarvitsivat alustusfunktion, joka varasi malloc()-kutsulla muistin ja loi käytettävät muistialtaat. Ratkaisu oli helppo, mutta toi hieman lisää monimutkaisuutta.

Myös aiemmin mainittu kääntäjän vanhuus toi ongelmia koko kehitystyön aikana. Sen kaikki kummallisuudet ja vielä oudommat virheilmoitukset kuluttivat aikaani paljon. Jossain vaihees-

sa aloin jo epäilemään järkeänikin, kun aivan laillinen tapa antaa merkkijono lainausmerkkien sisällä ei toiminutkaan järkevällä nopeudella. Muuttamalla suurin osa sellaisista esiintymisistä vakiomuuttujiksi nopeus palautui siedettävälle tasolle.

Hypertekstipalvelun autentikointi oli jo minulle liikaa. Jollain kerralla pääsin sisään, toisella kerralla en. Jokin ei ilmeisesti mennyt oikein. Sen takia tein teknisesti ruman virityksen, jolla sisäänkirjaus suoritetaan.

8. YHTEENVETO

Lähtiessäni tutkimaan opinnäytetyöni suunnittelua ja toteutusta, olin hyvin optimistinen. Tämän tekeminen olisi helppoa ja nopeaa. Myöskään alusta ei vaikuttanut mitenkään erityisen haastavalta. Perusidea oli hyvin helposti toteutettavissa ja suoraviivainen. Alustan ongelmat, jotka vaikuttivat ohjelmointivirheiltä, mutta eivät varmasti niitä olleet hidastivat ja tuhosivat innon tehdä työtä. Kuitenkin ongelmat olivat ratkaistavissa tai ainakin kierrettävissä ja tulosta alkoi syntyä. Samalla kun näki mitä syntyy, näki puutteita. Pieniä parannuksia tarvitsi tehdä. Esimerkiksi laittaa logiikan käyttöliittymässä, että yksi logiikkayksikkö ei voi olla itselleen sisääntulo, koska se on erityisen typerää.

Kuitenkin pystyin luomaan toimivat ratkaisun ja saavuttamaan siihen sellaisen tason, että voin sanoa sen olevan demonstraatiota parempi. Lisäksi se täyttää vaatimuksetkin. Suurin epäilykseni on tikapuuohjelmointia osaavaa henkilöä kohtaan. Pystyykö tällainen henkilö käyttämään tehokkaasti luomaani järjestelmää.

Paljon on kuitenkin kehitettävä lisää. Lähdekoodi on mielestäni rumaa. Se tulisi monilta osin uudelleen kirjoittaa ja järkeistää. Miettiä mitkä funktiot tulisi poistaa ja siirtää koodi ajettavaksi käytettävään kohtaan. Kirjoita ja lue rekisteri Modbus-funktiota pyydettiin erityisesti. Nyt se on toteutettuna, mutta siitä ei ole paljoa iloa. Kirjoituksen lukutapahtumaan yhdistävä ominaisuus voisi olla ratkaisu tähän.

Jos nyt alkaisin kehittää tätä, tekisin mahdollisimman yksinkertaisen kommunikointitavan järjestelmälle. Sen jälkeen pieni Java-sovellus, joka tarjoaisi Forth-tyylisen ympäristön. Testipenkkiä tehdessäni olin erityisen hyvällä mielellä, kun käyttöliittymästä vaikeiden asioiden tekeminen tapahtui näppäimistön kirjoitusnopeudella. Käyttöliittymä on mielestäni todella epämiellyttävä. Toisaalta tämän kaltaisen laitteen voisi toteuttaa paljon paremmin esimerkiksi edullisella tietokoneella tai sitten aivan tätä varten tarkoitetulla sulautetulla järjestelmällä, jossa ei ole kaikkea sitä ylimääräistä mitä Net+OS tarjoaa.

Olen kuitenkin oppinut C-kääntäjästä, Forthista, shelliohjelmoinnista ja vielä Latexistakin dokumentaatiota tehdessäni melko paljon. Lisäksi Graphviz kuvien luomisessa ja PlantUML-ohjelma UML-kaavioiden tekemisessä tulivat tutuiksi. HTML ja Javascript säilyivät onneksi käsittämättöminä pahoina tekniikan ilmentyminä.

9. LÄHDELUETTELO

/1/ Digi International, Digi Connect EM and Digi Connect Wi-EM Hardware Reference, [PDF-dokumentti], http://www.digi.com/pdf/prd_ds_digiconnectem_hwman.pdf, 20.4.2012.

/2/ Digi International, Digi Connect EM®Family, [PDF-dokumentti], http://www.digi.com/pdf/prd_ds_digiconnectem.pdf, 20.4.2012.

/3/ Digi International, Feature Spec NET+OS®7 Integrated Real-Time OS Development Platform, [PDF-dokumentti], http://www.digi.com/pdf/fs_netos7.pdf, 20.4.2012.

/4/ Free Software Foundation, GNU LESSER GENERAL PUBLIC LICENSE, [WWW-dokumentti], http://www.gnu.org/licenses/lgpl.html, 20.4.2012.

/5/ Modbus-IDA, MODBUS Application Protocol Specification V1.1b, [PDF-dokumentti], http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf, 20.4.2012.

/6/ Modbus-IDA, MODBUS Messaging on TCP/IP Implementation Guide V1.0b, PDF-dokumentti, http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

/7/ Modbus-IDA, MODBUS over serial line specification and implementation guide V1.02, PDF-dokumentti, http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf.

/8/ Raimbault, Stephane, Libmodus, [www-dokumentti], http://libmodbus.org, 20.12.2012.

/9/ Raimbault, Stephane, New read and write registers function, [www-dokumentti], https://github.com/stephane/libmodbus/commit/9a2733e7ecc2c622704753fae7fbf14521de10ea, 20.4.2012.

/10/ Sadler, John, Ficl, [WWW-sivusto], http://ficl.sourceforge.net/, 20.04.2012.

/11/ The Eclipse Foundation, Eclipse, [www-dokumentti], http://eclipse.org/, 20.4.2012.

10. LIITELUETTELO

Liite 1 Käyttöohje: Modbus-muunnin itsenäisellä logiikalla

Käyttöohje: Modbus-muunnin itsenäisellä logiikalla

Hannu Vuolasaho

18. toukokuuta 2012

Sisältö

Osa I.
Käyttö

1. Aloitus

1.1. Aluksi

Onneksi olkoon. Olet saanut käyttöösi Modmus-muuntimen itsenäisellä logiikalla. Tämä laite on tehty opinnäytetyönä Kemi-Tornion Ammattikorkeakoululle. Tällä laitteella on kaksi ominaisuutta. Se voi toimia mediamuuntimena Modbus-verkkojen välillä ja välittää ethernetistä tulevat Modbus-pyynnöt sarjaporttiin. Toinen ominaisuus on toimia itsenäisenä logiikkana. Laitteen konfigurointi suoritetaan selaimella ethernetin puolelta. Selaimen on tuettava kehyksiä (frames) ja Javascript on hyvä olla päällä. Ilman Javascriptiä käyttöliittymän käyttäminen voi olla hankalaa.

Laite ei kirjoita mitään pysyvään muistiinsa ja siksi laite on paristovarmennettava, jotta asetukset säilyvät sähkökatkojen aikana. Tämä myös mahdollistaa laitteen alkutilaan asettamisen käynnistämällä se uudelleen. Mikäli laitteen salasana on unohtunut, se palautuu oletukseen myös uudelleenkäynnistyksessä.

Laite ei ota kantaa pöljiin asioihin. Estämällä pöljiä asioita, estetään myös haluttuja asioita. Lisäksi laite on tarkka omista asioistaan. Se ei lue eikä kirjoita sisäisestä logiikasta kuin omiin kohtiinsa. Toisaalta median muunnossa lävitse menee melkein mikä vain. Lisäksi kirjoitus tapahtuu joka sekunti, kun taas lukeminen määräajoin.

Käyttöliittymä on tarkoituksella karu. Mikäli vilkkuvilla kuvilla ja hienoilla toteutuksilla saavutettaisiin jotain todellista hyötyä, niitä olisi käytetty. Esimerkiksi käyttöliittymän Javascriptiä tarvitaan oikeasti vain uloskirjauksessa, kun kehykset poistetaan. Kuitenkin sillä on voitu tuoda lisäarvoa tekemällä lomakkeiden arvojen esitarkastelua. Jos arvojen tarkastuksessa laitteessa törmätään virheeseen, unohdetaan arvot ja annetaan virheilmoitus. Siksi Javascript estää lomakkeen lähetyksen ja antaa virheilmoituksen, jotta virheellinen kohta voidaan korjata.

Tutustu tähän käyttöohjeeseen huolella. Tämän laitteen tulisi toimia ja maailmassa ei pitäisi olla sotia. Juuri kummallekaan asialle ei voida tehdä mitään.

1.2. Pikaohje kiireisille

1. Ota web-yhteys laitteeseen. Oletuksena käyttäjätunnus on *admin* ja salasana *default*
2. Muuta salasana
3. Tee sopivat sarjaportin asetukset.
4. Määritä median muunto.
5. Luo Modbus-laitteet.
6. Luo logiikkayksiköt.
7. Säädä logiikkayksiköt.
8. Kiroa laite ja ohjelmiston tekijä, kun mikään ei onnistunut.
9. Lue nämä käyttöohjeet.
10. Palaa kohtaan yksi.

1.3. Käyttöönotto

Kytke ethernetkaapeli verkkoporttiin ja Modbus-laitteet sarjaportti 2:n. Sarjaportit ovat RS-232-porttej joten mahdollinen RS-232 - RS-485-muunnin voidaan tarvita väliin. Sarjaportti 1 voidaan kytkeä tietokoneeseen ja nähdä käynnistysviestit. Tämän jälkeen kytke virta laitteeseen päälle. TCP/IP verkossa tulee olla DHCP palvelin, joka antaa laitteelle IP-osoitteen. Vain IPv4 on tuettu. Käynnistysviesteissä näkyy laitteen IP-osoite.

Ongelman, jonka kohtaat on tässä vaiheessa laitteen etsiminen. Laitteen käyttöön tarvitaan joko sen osoite tai nimi. Helpoin ratkaisu on tosiaan lukea se sarjaportti 1 käynnistysviesteistä.

Mikäli ethernet-yhteyttä käytetään vain asetusten syöttämiseen, useimmille käyttöjärjestelmille on saatavana DHCP-palvelu. Palvelu on DHCP-protokollan se osapuoli, joka antaa osoitteen ja asiakas se joka ottaa osoitteen. Kytke laite suoraan tietokoneen verkkoliityntään ja aja DHCP-palvelua. Lokitiedostoista näet laitteen saaman IP-osoitteen. Muista pysäyttää DHCP-palvelu verkoissa joissa sitä ei tarvita. Toinen vaihtoehto on olla samassa verkossa ja vain etsiä se laite kokeilemalla.

Verkkoon pysyvästi kytkettävä laite kannattaa asettaa saamaan aina sama osoite. Silloin muiden Modbus-TCP-laitteiden ja laitteen konfigurointi on paljon helpompaa.

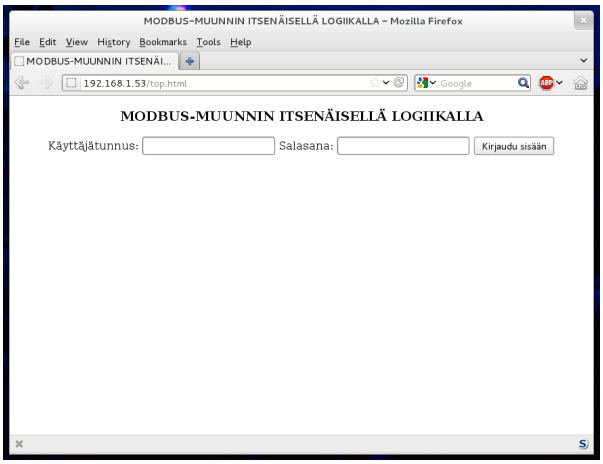
Ota selaimella yhteys laitteeseen ja sivulla pitäisi näkyä käyttäjätunnuksen ja salasanan kyselykaavake.

Oletuskäyttäjätunnus on *admin* ja salasana *default*

1.4. Käyttöliittymä

1.4.1. Ensimmäinen sivu

Ottaessa yhteyttä laitteeseen eteen aukeaa kirjautumissivu, joka on esitetty kuvassa 1. Käytännössä sitä käytetään kirjoittamalla tunnus ja salasana. Tämän jälkeen painetaan Kirjaudu sisään nappia. Mikäli salasanaa ei ole vaihdettu laitteen käynnistymisen jälkeen, se on *default* oletuksena.



Kuva 1: Kirjautumissivu

1.4.2. Yleiskuvaus

Käyttöliittymässä on neljä osaa. Ylimpänä on otsikko ja uloskirjautumisnappi. Uloskirjautuminen voi tapahtua joko käyttäen tätä nappia tai automaattisesti kymmenen minuutin kuluttua. Kuvassa 2 seuraavalla sivulla se on merkitty vihreällä pohjalla havainnollistamiseksi.

Vasemmalla on navigointilinkit. Niistä pääsee asettamaan median muunnon, sarjaportin asetukset, luomaan ja poistamaan Modbus-laitteita, jotka luetaan ja kirjoitetaan, sekä konfiguroimaan logiikan. Lisäksi on linkki salasanan vaihdolle. Tältä alueelta löytyy hieno, ensimmäinen, viimeinen ja ainoa multimediaominaisuus. W3C:n linkitetty validointileima kertoo, että tämä laite tuottaa validia HTML:ää. Se voi olla virheellinen kuva, mikäli selaimellasi ei ole pääsyä Internetiin. Tämä alue on väritetty vaalean keltaisen värillä.

Navigointilinkkien oikealla puolella on sivu, johon syötetään arvoja ja näkyy tietoja laitteen tilasta. Tämän alueen sisältö riippuu valitusta sivusta. Etusivulla on yleisiä jorinoita. Tämän alueen yläosaan kuitenkin tulostuu virheilmoitukset mikäli jokin yritetty toiminto ei onnistunut ja teksti onnistumisestakin.

Neljäs osa on kaikilla tietosivulla. Se on tietosivun alaosassa oleva taulukko, jossa näkyy yhteenveto laitteen tilasta. Taulukon solun taustaväri kertoo laitteen olevan virhetilassa punaisena, keltaisena ominaisuus on itsessään kunnossa, mutta jokin toinen asia estää sen toimimisen tai virhe ei ole niin vakava, että ainakin osa siitä toimii. Vihreänä oleva solu kertoo toimivasta ominaisuudesta. Taulukon tiedot nousevat vasemmalta oikealle monimutkaisuudessa. Ensin on vain sarjaportti ja median muunto. Sen jälkeen on sarjaväylän päällä oleva protokolla ja lopuksi monimutkaisin asia eli sisäinen logiikka.

Kuva 2: Etusivu

1.4.3. Salasanan vaihto

Ensimmäinen tehtävä on vaihtaa salasana johonkin parempaan. Vaihdettaessa salasanaa järjestelmä kysyy vanhan salasanan ja uuden kahdesti. Uuden salasanan tulee olla sama molemmissa kentissä ja isot ja pienet kirjaimet ovat eri merkkejä. Salasanan tulee täyttää ilmoitetut kriteerit, jotta se voidaan vaihtaa. Onnistuneen vaihdon jälkeen tietosivulla lukee isolla salasanan vaihto onnistui. Uusi salasana on käytössä seuraavalla kirjautumiskerralla.

Salasanan vaihtosivulta voidaan valita, tarvitaanko logiikan tila-sivulle salasana. Mikäli salasana tarvitaan, tulee käyttäjän olla kirjautuneena sisälle. Silloin myös koko muu järjestelmä on avoin mistä vain käytettäväksi, joka pääsee kiinni laitteeseen verkosta. Tämä johtuu siitä, että joka kerta tilasivua lukiessa ajastin nollataan ja sisäänkirjautuminen pysyy voimassa. Toisaalta, jos logiikan tila-sivu ei vaadi salasanaa, se voidaan lukea mistä tahansa edelleen, mutta muihin sivuihin ei pääse käsiksi.

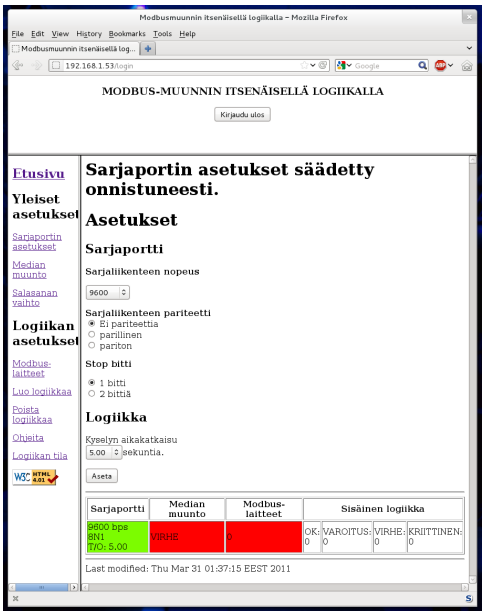
Salasana halutaan kytkeä pois tilasivulta lähinnä siksi, että jollain toisella laitteella luetaan tietoja laitteesta. Muutosten tekeminen ei onnistu tältä laitteesta.

1.4.4. Sarjaportin asetukset

Sarjaportin asetussivulta asetetaan parametrit sarjaliikenteen puoleiselle Modbus-väylälle. Aikakatkaisu vaikuttaa Modbus-kyselyn aikakatkaisuun. Mikäli tässä ajassa ei saada vastausta, unohdetaan kyseinen kysely ja yritetään myöhemmin uudelleen.

Aikakatkaisun asettaminen liian pitkäksi voi saattaa järjestelmän jumiutumisen kaltaiseen tilaan. Mikäli jotain Modbus-laitetta luetaan nopeampaan tahtiin kuin aikakatkaisu on asetettu, järjestelmä odottaa vain kuulumatonta vastausta Modbus-väylältä ja verkkokäyttöliittymä voi muuttua hitaaksi ja lopuksi yhteys selaimeenkin aikakatkastaan. Tässä on laitteen heikko kohta.

Sarjaportin asetukset on tehtävä ennen median muunnon kytkemistä. Median muunto voidaan asettaa, mutta ilman sarjaportin asetuksia sarjaväylälle kirjoittaminen tai lukeminen on mahdotonta. Onnistuessaan sarjaportti muuttuu vihreätaustaiseksi tilataulukossa ja luvut kertovat nopeuden, databittien määrän, joka on aina 8, pariteetin ja stop-bittien määrän. 8N1 tulkitaan 8 databittiä, ei pariteettia (*None*) ja yksi stop-bitti. Muut mahdolliset merkit pariteetin kuvaamiseen on parittoman O, kuten *odd*, ja parillisen E sanasta *even*. Lisäksi aikakatkaisun arvo on ilmoitettu T/O: merkinnän jälkeen. Onnistuneen asetusten antamisen jälkeen käyttöliittymä näyttää saman kaltaiselta kuin kuvassa 3



Kuva 3: Sarjaportin asettaminen

1.4.5. Median muunto

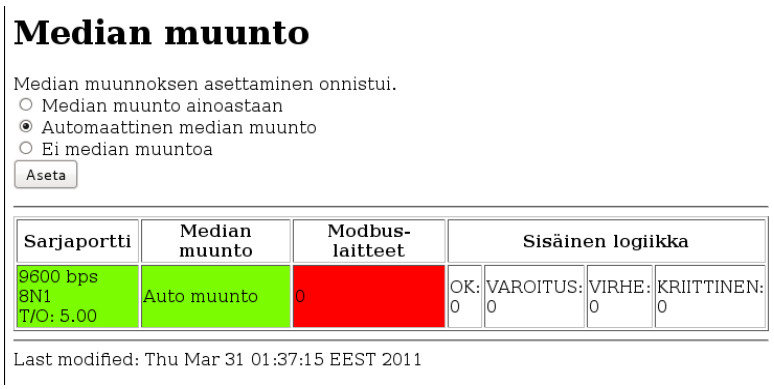
Median muunnossa otetaan laitteeseen yhteys TCP:llä porttiin 502 Modbus-protokollalla ja viesti välitetään sarjaliikenteen puolelle. Modbus-TCP-protokollasssa on käytettävä laitenumeroita, jotta tiedetään mille laitteelle lähetetään. Modbus-TCP-protokollan oletus 255 ei välity minnekään. Toisaalta riippuen kuinka tarkasti sarjaväylään kytketyt laitteet seuraavat Modbus-protokollan määritelmiä, myöskään laitetta numero 0 ei kannata käyttää. Laite nolla on yleislaite, jota kaikkien laitteiden tulisi kuunnella. Kuitenkin osa Modbus-toteutuksista vastaa tähän ja se on väärin. Toisaalta Modbus-muunnin itsenäisellä logiikalla välittää tämän vastauksen eteenpäin ja ainoastaan yksi vastaus pääsee läpi. Muuten laitteen ei tulisi ottaa mitään kantaa mitä Modbus-viesti sisältää.

Median muunto sivulla on kolme vaihtoehtoa. Sivu on esitelty kuvassa 4. Vain median muunto on tila, jossa sisäinen logiikka on kytketty pois päältä. Mahdollisesti konfiguroitu sisäinen logiikka ei vain päivity. Se on kuitenkin laitteen käyttömuistissa käyttövalmiina.

Automaattinen median muunnossa ethernetistä tuleva Modbus-viesti välittyy sarjaväylään. TCP pyyntö käsitellään, kun sisäisen logiikan päivitys on ohitse. Se ei keskeytä sisäistä logiikkaa vaan sisäinen logiikka antaa tilaa ja välitettävä viesti käsitellään. Tämän jälkeen jatketaan sisäisellä logiikalla.

Vain sisäisessä logiikassa median muuntoa ei suoriteta, vaan sisäinen logiikka on ainoa lähde Modbus-viesteille sarjaliikenteen puolelle.

Tilataulukon arvot ovat *vain muunto*, *auto muunto* ja *ei muuntoa*. Näistä vain muunto on varoitustila, sillä se voi aiheuttaa kummastusta miksi sisäinen logiikka ei päivitä Modbus-laitteita.



Kuva 4: Median muunnon asettaminen

1.4.6. Laitteiden lisäys ja poisto

Navigointialueen Modbus-laitteet linkin kautta pääsee sivulle, josta voi luoda, poistaa ja muokata Modbus-laitteita. Kuvassa 5 sivulla 10 on esitelty Modbus-laitteiden käsittelyyn tarkoitettu käyttöliittymä. Jokaisella taulukon rivillä on yksi laite. Mikäli ainoatakaan Modbus-laitetta ei ole luotu, näkyy vain taulukon alin rivi eli luomiseen käytettävä rivi.

Ensimmäinen syötettävä sarake, eli *nimi,* on kenttä johon voi kirjoittaa laitteelle nimen. Sen tulee olla kuvaava, sillä logiikassa tätä käytetään tunnistamaan eri laitteet. Nimien ei tarvitse olla yksilöllisiä, koska järjestelmä tunnistaa laitteet laitenumeron, tyypin ja osoitteen mukaan. Käyttäjälle on kuitenkin mukavampi, mikäli nimi on tunnistettava ja yksilöllinen.

Laite-valikosta valitaan käytettävä Modbus-laitteen numero. Tämä numero on siis Modbus-RTU-protokollan mukainen laitenumero. Ne menevät yhdestä 247.

Osoite on myös Modbus-protokollan määrittelemä tieto. Se voi olla nollasta 65535 eli eli tuttavallisesti 16-bittinen etumerkitön kokonaisluku.

Tyyppi tarjoaa mahdollisuuden valita erilaisista tietotyypeistä, joita Modbus-protokolla tarjoaa. Perustyyppejä on neljä. Erillinen sisääntulo, kela, sisääntulo ja rekisteri. Näistä erillinen sisääntulo ja sisääntulo ovat vain luettavia. Lisäksi on yleisesti käytössä 32-bittiset kokonaisluvut ja 32-bittinen reaaliluku. 32-bittisistä luvuista on huomioitava, että esimerkiksi kuvassa oleva kokonaisluku, joka on heksadesimaalina 0xFEDCBA98, tallentaa rekisteriin 2 arvon 0xFEDC ja rekisteri 3 saa 0xBA98. Lukujen järjestyksen rekistereissä 32-bittisillä luvuilla lisäksi huomattakoon seuraava pieni, mutta tärkeä asia. Yksinkertaisen tarkkuuden omaava liukuluku on erittäin epätarkka. Kuvassakin näkyvä luku on syötetty järjestelmään arvona 1.2345.

Päivitysväli sarakkeessa on kahden lukupäivityksen välillä kuluva aika. Lyhin on 5 sekuntia ja pisin on 11 tuntia.

Viimeinen sarake, jota käytetään on *vain luku*-sarake. Laittamalla ruksin tähän kohtaan estät rekisteriin tai kelaan kirjoittamisen. Sisääntuloilla ja sisääntulorekistereillä tätä ei voi luonnollisestikaan poistaa käytöstä.

Muilla riveillä, kuin luontiriveillä näkyy myös muuta tietoa. *Tila*-sarake kertoo virheistä. Se voi antaa virheen Modbus-protokollan mukaisesta poikkeuksesta tai mikäli itse ohjelmistossa sattuu jotain todella pahaa ja kummallista niin myös silloin. Ohjelmiston sisäisen virheen ja laitteen vastaamattomuuden välistä eroa on hankala erottaa, johtuen käytetystä Modbus-protokollan toteutuksesta. Molemmista saadaan tilaksi järjestelmävirhe. *Vain luku*-sarake vastaa luontirivin raksikohtaa eli laitteeseen ei voi kirjoittaa. Mikäli kirjoituksen esto kytketään päälle, muuttuu solun tausta keltaiseksi.

Toiseksi viimeinen sarake *kirjoitus ja luku (0x17)* tuki koskee koko laitetta aina. 0x17 viittaa Modbus-protokollan kirjoita ja lue funktioon. Siinä kirjoitetaan ensin ja sitten luetaan rekisterit. Sen käytön etuna on yhden vastausviestin poisjäänti.

Viimeinen sarake on *poista* sarake. Raksi siihen ja Modbus-laite poistetaan painettaessa *lähetä*. Mikäli logiikka käyttää sisääntulonaan kyseistä laitetta, aiheuttaa se kriittisen *sisääntulo puuttuu* tilan kyseisen logiikkayksikön seuraavassa päivityksessä.

Tietotaulukon alla on kirjoitus mahdollisuus. Valikosta valitaan kirjoituskohde. Vaikka valikko onkin nimien mukaan luotu, voi kaksi saman nimistä laitetta olla siinä. Ne vain esiintyvät peräkkäin ja niiden erotteleminen on jätetty kotiläksyksi.

Seuraavana sivulla on 0x17 tuki. Se voidaan kytkeä laitteittain tästä taulukosta päälle ja pois. Tuen tila näkyy tietotaulukossa ja myös tuen säätämistaulukon oletuksessa.

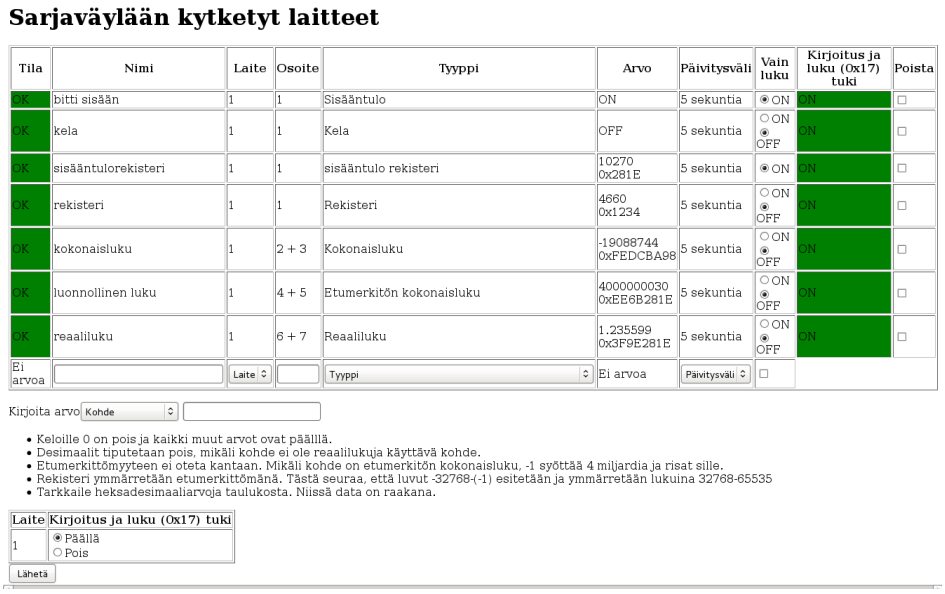
Lopuksi on *lähetä*-nappi. Se lähettää tiedot järjestelmään käsiteltäväksi. Yhtä aikaa voi luoda, poistaa, kirjoittaa ja muuttaa laitteita.

Muistutettakoon, että mikäli Modbus-laitteen nimeä, laitenumeroa, osoitetta, tyyppiä tai päivitysväliä haluaa muuttaa, niin se on ensin poistettava ja sitten luotava uudelleen. Tästä seuraa myös, että laite katoaa logiikasta ja sitä käyttänyt sisäänmeno tai ulostulo menee kriittiseksi kyseisen logiikkayksikön seuraavassa päivityksessä. Pisimmillään 11 tuntia.

2. Logiikka

2.1. Yleistä

Jotta tämän laitteen nimi olisi Modbus-muunnin itsenäisellä logiikalla, tulee laitteessa olla itsenäinen logiikka. Logiikka toimii ottamalla määritetyn Modbus-laitteen sisääntulokseen, tekee sille laskutoimituksia ja laittaa saadun tuloksen ulostuloon, joka on myös määritetty Modbus-laite. Logiikalla on viisi erilaista aritmeettista operaatiota ja kaksi ehtotyyppistä toi-



Kuva 5: Modbus-laitteet

mintoa. Lisäksi on keskiarvo, suora ja PID-säädin. Näiden lisäksi ovat jos, muuttuja ja MUX. Lisäksi on hyvin voimakas työkalu input2param, joka syöttää sisääntulon parametriksi jollekin toiselle logiikkayksikölle. Sillä pystyy tekemään hienoja asioita, mutta myös hirmutekoja. Aivan kuten moottorisahalla voi veistää karhun tai pätkiä aarniometsän hakkuuaukoksi.

Erityisen tärkeää on huomata, että aritmeettisten operaatioiden toiminta riippuu sisään tulevista tietotyypeistä. Niitä logiikalla on neljä. Ne johtuvat osaksi Modbus-protokollan tietotyypeistä ja osaksi ne tuovat näppäryyttä logiikan ohjelmointiin.

Logiikkayksiköt on asetettu ketjuihin. Tämän sivuvaikutuksena päivitykset tapahtuvat ketjuittain ylimmästä ketjusta alimpaan. Ketjun sisällä päivitykset etenevät vasemmalta oikealle. Logiikkayksikön päivitys tapahtuu vain, jos järjestelmän päivitysaika on jaollinen päivitysvälillä. Järjestelmän päivitysaikaa ei voi tietää, mutta siihen lisätään yksi aina noin[1] sekunnin väliajoin.

Käyttöliittymät eri logiikkaoperaattoreille on kuvaustaulukkoina alkaa kohdassa 2.4 sivulla 13, mutta ne eivät ole kuvia. Taulukoista puuttuvat syöttöalueet ja niiden ulkonäkö voi olla hieman erilainen. Ne ovat kehitysversiosta tuotuja.

Tiedon ei tarvitse kulkea logiikan ketjussa suoraan alusta loppuun. Jokainen logiikkayksikkö voi ottaa minkä tahansa logiikkayksikön itselleen sisääntuloksi. Varoittavana esimerkkinä todettakoon kuvassa 6 seuraavalla sivulla esitetyn logiikan toimivan eri tavalla kuin olettaisi.

Päivityksessä tapahtuu seuraavaa mikäli kaikilla on sama päivitysaika.

1. Erotus lukee ketjussa toisena olevan summan tuloksen ja kolmannen muuttujan tuloksen. Tämän jälkeen se vähentää summasta muuttujan ja kirjoittaa tuloksensa. Todennäköisesti tulos on -1

2. Summa lukee ensimmäisenä olevan erotuksen tuloksen ja muuttujan tuloksen. Laskee ne yhteen ja kirjoittaa tuloksen. Todennäköisesti 0.

3. Muuttuja lukee parametrinsa ja kirjoittaa sen tuloksekseen. Oikeasti mikään ei muutu.

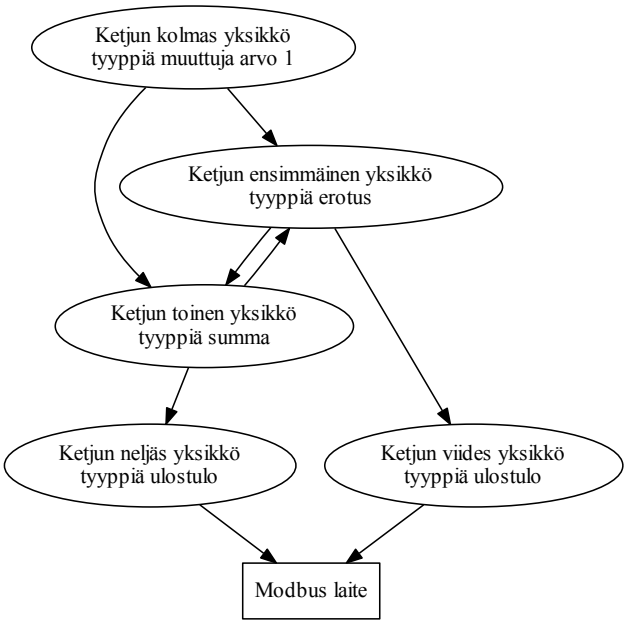
[1] Kello on havaittu käyvän 3560 - 3650 sekuntia tunnissa. Eli noin minuutin virhe tunnissa tai ±2%. Virheeseen vaikuttavat lähes kaikki asiat kuun vaiheesta alkaen.

4. Ulostulo lukee summan arvon ja antaa sen Modbus-laitteelle kirjoitettavaksi. Samalla kirjoittaa sen tuloksekseen.

5. Ulostulo lukee erotuksen tuloksen ja päivittää. Modbus-laitteen arvo tuli juuri yli kirjoitetuksi.

6. Modbus-laitteen arvo kirjoitetaan sarjaväylää pitkin orjalaitteelle.

Sen sijaan, että kuvassa olisi ollut hieno värähtelijä, onkin -1 kirjoittava turhan monimutkainen härveli. Samaan olisi päässyt yhdellä muuttujalla ja ulostulolla. Operaattoreita käsitellään kohdassa 2.4 sivulla 13 ja siitä eteenpäin.



Kuva 6: Esimerkki huonosti toimivasta logiikasta

2.2. Logiikan tietotyypit

2.2.1. Totuusarvo

Totuusarvo on otettu logiikkaan mukaan, jotta on helppo luoda logiikka, joka käsittelee keloja tai sisääntuloja. Totuusarvojen muunto muista tietotyypeistä on määritelty siten, että 0 on OFF tila, eli epätosi, ja kaikki muut arvot on ON tilaa.

Käyttöliittymässä totuusarvoon viitataan *B:* merkinnällä. Siitä muistutetaan miltei jokaisella logiikan tietosivulla. Merkintä tulee Boolen algebrasta.

2.2.2. Luonnollinen luku

Luonnollisia lukuja ovat ne luvut, joita voidaan laskea rajattomalla määrällä sormia ja varpaita. Nolla kuuluu myös luonnollisiin lukuihin. Tietenkin tietty raja tulee vastaan ja suurin mahdollinen luku on reipas neljä miljardia.

Käyttöliittymässä luonnolliseen lukuun viitataan N: merkinnällä. Matematiikan $\mathbb{N}$ on toiminut pohjana merkinnälle.

2.2.3. Kokonaisluku

Perinteisesti kokonaisluvut on luotu kertomalla lapsille $x + (-1 * x) = 0$ yhtälöstä. Tietotekniikassa kaunis matematiikka ei aivan onnistu vaan pitää todeta, että kokonaisluvut alkavat negatiivisen kahden miljardin alta ja päätyvät positiiviseen kahden miljardin päälle. Desimaali tai murto-osia kokonaisluvuilla ei ole.

Käyttöliittymässä kokonaislukuun viitataan Z: merkinnällä. Matematiikan $\mathbb{Z}$ on toiminut pohjana merkinnälle.

2.2.4. Reaaliluku

Villellä oli viisi omenaa. Matti ja Teppo halusivat myös. Kuka sai madon? Ja silloin joku keksi desimaaliluvut. Reaalilukuihin kuuluvat rationaaliluvut ja irrationaaliluvut. Tietotekniikka on vain kovin huono esittämään mm. tarkkaa piin arvoa tai arvoa $\frac{1}{3}$. Siksi käytetään desimaalilukuja. Likiarvoina sisäisesti tarkkuus on noin 16 desimaalia, mutta sarjaväylällä käytetään 32-bittistä esitystä, joka on vain noin 7 desimaalia tarkka. Käyttöliittymässä reaalilukuun viitataan R: merkinnällä. Matematiikan $\mathbb{R}$ on toiminut pohjana merkinnälle.

2.3. Logiikan tilat

Sen lisäksi, että logiikkaketjun yksi lenkki voi olla kunnossa, hyvinvoipa ja elämäänsä tyytyväinen tilassa OK, on myös muita tiloja. Yksittäisen logiikkayksikön tila ei yleensä vaikuta koko ketjun toimintaan. Mikäli päivitystä ei voida tehdä, pidetään ulostulo samassa arvossa. MUX kuitenkin tekee valintansa sisääntulontilan mukaan, josta lisää kohdassa sivulla 18.

2.3.1. Kriittiset

Kriittiset tilat ovat logiikkayksikölle tilanteita, joissa päivitystä ei voitu suorittaa.

Sisääntulo puuttuu tilassa logiikkayksikköä yritettiin päivittää, mutta sisääntuloa ei oltu määritetty ja se oli tyhjä.

Sisääntulo virheellinen logiikkayksikön sisääntulo on vääränlainen. Tämä voi olla mahdollista, mikäli sisääntuloksi on määritetty muuttuja, jos tai MUX. Näiden ulostulojen muuttuminen on mahdollista ilman käyttäjän toimia. Myös käyttäjän toimesta voidaan tämä virhe saada aikaan. Esimerkiksi yhteenlasku ottaa kaikki tyypit sisään. Mikäli yhteenlasku on jakojäännöksen sisääntulona, niin muuttamalla ensimmäinen yhteenlaskettava reaaliluvuksi, saadaan epäyhteensopivuus aikaan.

Ei päivitetty tila on yleensä nollalla jakamisen yrittämistä. Jokin seikka siis sisääntulo tai parametritiedoissa esti päivityksen.

Sisäinen virhe tarkoittaa järjestelmän sotkeutuneensa omiin solmuihinsa. Kurja kohtalo tämän näkijällä.

Muu kriittinen virhe on erikseen määrittelemätön kriittinen virhe.

2.3.2. Virheet

Virheet ovat tilanteita, joissa päivitys on todennut, että voitaisiin jatkaa, mutta siinä ei ole mitään järkeä ja toimitaan nyt näin niin tiedetään missä mennään.

Huipussa virhe tapahtuu, kun saavutetaan joko lukualueen yläraja tai virheen määritetty yläraja-arvo.

Pohjassa virhe on huipussa virheen vastine alarajalla.
Muu virhe on erikseen määrittelemätön virhe.

2.3.3. Varoitukset

Varoitukset ovat tilanteita, jotka antavat ymmärtää tilanteessa olevan jotain sellaista, jota käyttäjä ei ole tarkoittanut. Päivitys tapahtuu edelleen normaalisti.

Oletuksessa on lähinnä ilmoitusluonteinen varoitus, että kaikki arvot oletuksessa ja sisääntuloja ei ole määrätty. Tämä tila on yleensä käytössä ainoastaan heti logiikkayksikön luomisen jälkeen.

Alhainen osoittaa varoitus ala arvon saavuttamista.

Korkea osoittaa ylemmän varoitus arvon saavuttamista.

Sisääntulo hitaampi ilmaisee logiikkayksikön sisääntulon päivittyvän hitaammin kuin itse yksikön.

Sisääntulo nopeampi taas ilmaisee toisin päin olevaa tilannetta. Nämä varoitukset voidaan estää kyseisen logiikkayksikön muokkaussivulta ja valitsemalla *Salli hitaampi* tai *nopeampi tila*-sarakkeesta.

Muu varoitus on erikseen määrittelemätön varoitus.

2.4. Logiikan operaattorit

2.4.1. Rajat

Useimmilla operaattorilla on rajoja. Virhe- ja hälytysrajat on kehitetty turvaamaan logiikan toiminta. Varoitusrajoista voi mennä ylitse, mutta tila muuttuu keltaiseksi ja tila asetetaan varoitustilaan. Virhe raja on raja, jota ei voi ylittää. Jos tulos on rajoitettu näillä rajoilla ja tulos menisi rajan ylitse, se pysäytetään siihen. Rajat oikein määrittämällä voidaan logiikan toimintaa järkeistää. Esimerkiksi prosessi, joka käyttäytyy hyvin jollain alueella voidaan rajata OK alueeksi. Kun mennään epävarmalle alueelle siirrytään varoitustilaan. Jos esimerkiksi[2] laitetaan Volume yksikön varoitukseksi 10, jotta havaitaan kaiuttimien särkymisriski ja virheeksi 11 niin sitä nuppia ei voi vääntää kohtaan 12.

2.4.2. Päivitys

Päivityksessä logiikkayksikkö lukee sisääntulojen arvot, käyttää tallennettuja parametreja ja suorittaa päivitys toimintonsa. Tällöin ulostulon arvo voi muuttua. Sana päivitys esiintyy myös tarkoittamassa sitä aikaa, joka kuluu kahden päivitysoperaation välillä. Päivitystaajuutta voi muuttaa logiikasta, mutta ei Modbus-laitteesta. Sisääntulona olevan logiikkayksikön ja sitä lukevan päivityksen erisuuruus voi aiheuttaa varoitustilan. Tämä kuitenkin voidaan ohittaa, kuten kohdassa 2.3.3 osoitettiin.

[2] Katso This is Spinal Tap elokuva.

2.4.3. Muuttuja

Muuttuja on operaattori mahdollistaa kiinteiden arvojen syöttämisen logiikkaketjuun. Muuttujan tulos määräytyy sen ensimmäisestä parametrista, joka on *Arvo* sarakkeessa. Arvo siirtyy tulokseen vasta päivityksen myötä. Se millä tavalla tulos tulkitaan, riippuu alla olevasta vetolaatikon arvosta. Vetolaatikon vaihtoehto *Ei muutosta* jättää tietotyypin ennalleen. Kuva 10 sivulla 33 on muuttujan käyttöliittymästä. Kirjoitusvirhe on korjattu. Huomioitavaa on, että tila on vielä oletuksessa, tulos nollilla ja tyyppi on reaaliluku. Kuva on siis otettu ennen päivitystä, jossa syötetty arvo olisi siirtynyt tulokseksi.

Nimen ja sisääntulojen yhteydessä esiintyvät kirjaimet kuvaavat ulostulon *tyyppi* ä.
B on *totuusarvo* ja se saa arvoikseen *ON* ja *OFF* . *OFF* voidaan tulkita nollaksi ja *ON* kaikiksi muiksi arvoiksi.
N on *luonnollinen luku* ja sen arvot ovat kaikki positiiviset *kokonaisluvut* ja nolla.
Z on *kokonaisluku*
R on *reaaliluku*
Järjestelmä suorittaa laskutoimitukset *reaaliluvuilla* mikäli mahdollista.
Jotkin toiminnot eivät kuitenkaan ole mahdollisia niillä.
Pyöristystä ei tehdä. Desimaalit vain katkaistaan pois.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Arvo
OK	**Z:** Int var	Muuttuja	5	6	6.280000

Ulostulo on asetettu arvo. Asettamalla uuden arvon **Arvo** kenttään
ja valitsemalla **Muuta** muuttuu ulostulon arvo.
Ulostulon *tyyppiä* voidaan muuttaa valikosta ja valitsemalla **Muuta**

2.4.4. Pienempi kuin ja suurempi kuin

Vertailuoperaattorit vertaavat sisääntulojaan tai ensimmäistä sisääntuloa ja asetettua arvoa. Mikäli *vertaa sisääntulot* on ON tilassa, vertailu tapahtuu sisääntulojen välillä. Tulos on totuusarvo. Tämä siksi, että aritmeettiset operaattorit summa 2.4.13 sivulla 21, erotus 2.4.14 sivulla 23 ja kertolasku 2.4.17 sivulla 29 tarjoavat mielenkiintoisia mahdollisuuksia rakentaa logiikka. Lisäksi on jos 2.4.9 sivulla 18 operaattori käytössä. Käyttöliittymät molemmilla vertailuoperaattoreilla ovat hyvin saman kaltaiset.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Input 1	Input 2	Vertaa sisääntulot	Vertailuarvo
OK	**B:** Vertaus 1	Pienempi kuin	5	OFF	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	OFF	2.700000

Pienempi kuin antaa bool tyyppisen arvon mikäli *sisääntulo 1* on pienempi kuin *sisääntulo 2* mikäli *vertaa sisääntulot* on *ON* .
vertaa sisääntulot ollessa *OFF* vertailu kohdistuu *Sisääntulo 1* ja *vertailuarvoon* .

Tila	Nimi	Tyyppi	Päivitys	Tulos	Sisääntulo 1	Sisääntulo 2	Vertaa sisääntulot	Vertailuarvo
OK	**B:** Vertaus 2	Suurempi kuin	5	OFF	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	ON	0.000000

Suurempi kuin antaa bool tyyppisen arvon mikäli *sisääntulo 1* on suurempi kuin *sisääntulo 2* mikäli *vertaa sisääntulot* on *ON* .
vertaa sisääntulot ollessa *OFF* vertailu kohdistuu *Sisääntulo 1* ja *vertailuarvoon* .

2.4.5. Keskiarvo

Keskiarvo toimii liukuvana keskiarvona ja alipäästösuotimena. Se laskee 7 viimeisimmän sisääntulon perusteella tuloksensa. Vaikka kohdassa 2.4.5 onkin kuvattu vain kuusi vanhaa. Tämä johtuu siitä, että uusi sisääntulo otetaan mukaan ja vasta sitten lasketaan niistä seitsemästä arvosta keskiarvo. Tämän jälkeen vanhoja tallennettuja arvoja siirretään kohti vanhempaa ja vanhin unohdetaan. Lisäksi sisääntulon tulos kopioidaan tuorein arvo kohtaan. Ulostulo seuraa sisääntulon tyyppiä ja totuusarvot ovat kiellettyjä. Edelliset arvot on tuotu käyttäjän näkyville, mutta niitä ei voi muuttaa. Luotaessa *keskiarvo* vanhat arvot saavan arvon 0.

Keskiarvo vaimentaa värähtelyn $\frac{1}{7}$ osaan. Toisaalta askel tyyppinen muutos pyöristyy. Tällä operaattorilla ei voi laskea keskiarvoa useammasta lähteestä.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Sisääntulo	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
OK	**Z:** KA	Keskiarvo	5	0	**Z:** Int var 6 Muuttuja	-25.30	0.000	214.0000	214.000

Jatkoa perästä

Tuorein arvo	2. tuorein	3. tuorein	4. tuorein	5. tuorein	Vanhin
6.00	0.00	0.00	0.00	0.00	0.00

Keskiarvo ottaa viimeiset seitsemän sisääntulon arvoa
ja laskee niille keskiarvon ja tallentaa tuloksen ulostuloon.
Lisäksi tarjolla on ulostulon rajaus.

2.4.6. Sisääntulo

Sisääntulo voi aloittaa loogisesti tiedon kulun, mutta sen ei tarvitse olla ketjun ensimmäinen. Se voi sijaita missä tahansa kohtaa ketjussa. Pitää vain ottaa huomioon päivitysjärjestys, mikäli aikaisempi logiikkayksikkö ottaa sisäänsä myöhäisemmän sisääntulon. Silloin luetaan edellisen päivityskierroksen tietoa.

Sisääntulon käyttöliittymän sarake *Modbus-laite* ilmoittaa luettavan laitteen nimen. Valikosta voi vaihtaa laitteen. Sisääntulo lukee laitteen päivityksen yhteydessä, mutta se ei tarkoita, että fyysinen laite olisi siinä välissä oikeasti luettu. Laitteiden lukemisen päivitysväli on määritettävä Modbus-laitteita luotaessa. Sisääntulon tietotyyppi määräytyy valitun Modbus-laitteen perusteella. Rekisteristä ja kokonaisluvusta tulee kokonaisluku, etumerkittömästä kokonaisluvusta tulee luonnollinen luku ja reaaliluku pysyy reaalilukuna. Mikäli Modbus-laite poistetaan, tulee *sisääntulo puuttuu kriittinen* tila ja tulos säilyy viimeksi luetussa arvossa.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Modbus-laite
OK	**Z:** Sisään	Sisääntulo	5	-1	MB

2.4.7. Ulostulo

Ulostulo on logiikan portti ulospäin ja sen *Modbus-laite* sarakkeessa oleva Modbus-laitteen nimi on se kohde jonne tietoa kirjoitetaan. Tulos osoittaa minkälaisen arvon ulostulo kirjoittaa. Käyttöliittymäkuvauksessa esimerkiksi Modbus-laite on joko rekisteri tai kokonaisluku.

Jos rekisteriin, joka on 16-bittinen luku, yrittää kirjoittaa liian suurta tai liian pientä, tulee tilaksi virhe ja Modbuslaitteelle kirjoitetaan raja-arvo. Jos luvun siirto Modbus-järjestelmään epäonnistuu, tulee tilaksi kriittinen *ei päivitetty*. Mikäli Modbus-laitteen tyyppi ja sisääntulon tyyppi eroavat liikaa, ulostulo yrittää parhaansa mukaan kirjoittaa, mutta onnistumisesta ei ole minkäänlaista varmuutta. Tilaksi tulee silloin *virheellinen sisääntulo*. Mikäli Modbus-laite poistetaan, tulee *sisääntulo puuttuu kriittinen tila* ja tulos säilyy viimeksi luetussa arvossa.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Modbus-laite
OK	**Z:** Ulos	Ulostulo	5	-1	MB

2.4.8. Suora

Suora viritetään sisääntulo 1 ulostulo 1 ja sisääntulo 2 ulostulo 2 väliin. Mikäli sisääntulo 1 on 0, ja ulostulo 1 on 1 ja sisääntulo 2 on arvo 1 ja ulostulo 2 on 2 niin suora toteuttaa yhtälön $y = 1.0 * x + 1$ sisääntulo ala, sisääntulo ylä välisellä alueelle. Eli silloin saadaan jana. Mikäli sisääntulo ei ole tällä alueella on tuloksena virhetila, eikä päivitystä tulokseen tehdä. Toisaalta ulostulokin on rajoitettu. Tämä mahdollistaa suoran käytön niin, että näillä sisääntuloarvoilla suora on voimassa ja ulostulo pysyy kurissa.

Mikäli Sisääntulo 1 ja Sisääntulo 2, jotka edustavat x-akselia ja sen arvoja ovat yhtä suuret on tuloksena kriittinen tila. Tämä johtuu suoraan matematiikan kaavasta $Tulos = \frac{Ulostulo_2 - Ulostulo_1}{Sisääntulo_2 - Sisääntulo_1} * x + Sisääntulo_1 - \frac{Sisääntulo_1 * (Ulostulo_2 - Ulostulo_1)}{Sisääntulo_2 - Sisääntulo_1}$, joka taas johtaisi nollalla jakamiseen. Kaavassa x kuvaa kyseisen logiikkayksikön sisääntulon arvoa.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Sisääntulo	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
Korkea	**Z:** Jana	Suora	5	1	**Z:** Int var 6 Muuttuja	-25.300000	-25.300000	-5.300000	21474.00

Jatkoa perästä

Sisääntulo ala	Sisääntulo ylä	Sisääntulo 1	Ulostulo 1	Sisääntulo 2	Ulostulo 2
-2147488.00	214748.0000	-25.300000	-5.300000	0.000000	0.000000

Suora viritetään kahden annetun pisteen väliin.

On huomioitava, että suora rajoitetaan ulostulon sekä sisääntulon kanssa.

Sisääntulon rajoittaessa on aina seurauksena virhetila.

2.4.9. Jos

Jos tarjoaa mahdollisuuden päättää yhdellä totuusarvo sisääntulolla, kumpi kahdesta muusta sisääntulosta kirjoitetaan ulostuloon. Myös tietotyyppi muuttuu valitun sisääntulo tietotyypiksi. Kuten käyttöliittymän kuvauksestakin näkee, on helppo sanoa, että *jos ehto* ollessa ON (totta), tapahtuu *silloin* ja mikäli ehto on OFF lausutaan ääneen pahaenteisesti "tai *muuten...*" Lisäksi sisääntulon tila välittyy *jos* operaattorin tilaksi.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Ehto	Silloin	Muuten
Alhainen	**Z:** Valinta	Jos	5	-13	**B:** Bool var Off Muuttuja	**Z:** Int var 19 Muuttuja	**Z:** Int var -13 Muuttuja

2.4.10. MUX

MUX tulee sanasta multiplekser, joka kuvaa sen toimintaa ehkä parhaiten. Päivityksessä, MUX käy kaikkien neljän sisääntulon tilan lävitse pienimmästä suurimpaan. Mikäli se löytää OK tilassa olevan sisääntulon, valitaan se. Mikäli OK tilassa olevaa ei löytynyt valitaan ensimmäinen *varoitus* tilassa oleva. Jos *varoitus* tilassa olevaa sisääntuloa ei löytynyt, yritetään vielä kerran *virhe* tilassa olevia. Tämän jälkeen mitä tahansa, joka on olemassa. Tila, tulos ja tietotyyppi asetetaan valitun sisääntulon mukaiseksi. Jos sisääntuloja ei ole, tulee tilaksi *sisääntulo puuttuu*. On huomioitava, että MUX toimii myös vähemmällä kuin neljällä sisääntulolla. Valittu sisääntulo näkyy omassa sarakkeessaan ja ulostulo on rajoitettu varoituksella ja virheellä. Mikäli sisääntulo olisi OK ja ulostulon rajoitus rajaa sisääntulon arvoa niin silloin MUX saa tilakseen varoituksen tai virheen.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Sisääntulo 1	Sisääntulo 2	Sisääntulo 3	Sisääntulo 4
OK	**Z:** Muksi	Mux	5	6	**Z:**Osamäärä 6 Jakojäännös	**Z:** Tulo 114 Kertolasku	**Z:** Säätö -13 PID	**Z:** Miinus -13 Erotus

Jatkoa perästä

Valittu sisääntulo	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
1	-214748364.00	-214748364.00	21474836.00	21474836.000

2.4.11. Input2Param

Input2Param on jotain, joka on jäänyt reliikiksi kehitystyöstä. Sen tarkoitus on mennä kyljestä sisään, eli ottaa sisääntulonsa ja muuttaa parametria toisessa logiikkayksikössä. Sillä voi tehdä hienoja asioita, mutta samalla on mahdollista särkeä monia kohtia logiikan suunnittelusta. Lisäksi tämä operaatio ei tarkastele kohteitaan kovin tarkasti estäen virheellistä toimintaa.

Asetusten syöttäminen on tehtävä kahdessa osassa. Ensin on laitettava *luettava* ja *kohde*. Vasta tämän jälkeen kun *kohde* on määritetty tulee *parametri*-valikkoon vaihtoehtoja. Kohdetta vaihtaessa voi myös käydä sellainen vahinko, että entinen valittu parametri jää kummittelemaan. Hyvässä tapauksessa parametrin asetus ei onnistu ja Input2Param menee virhetilaan. Huonossa tapauksessa päivitetään väärää paikkaa. Tämä ei ole virhetoiminta vaan suunniteltu ominaisuus. Suojauksen rakentaminen olisi haitannut enemmän kuin hyödyttänyt.

Tiloikseen Input2Param saa *OK, Sisääntulo puuttuu* ja *Sisääntulo virheellinen.* Lisäksi tietotyyppi asetetaan luettavan sisääntulon mukaan.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Luettava	Kohde	Parametri
OK	**Z:** Kyljestä sisään	Input2Param	5	25	**Z:** 25 Summa	**Z:** -13 PID	I kerroin

Sisääntulon arvo kopioidaan toiseen parametriksi.
TÄMÄ ON VAARALLINEN JA TEHOKAS.
Mikäli luettu arvo on epäkelpo parametriksi niin tulee virheeksi epäsopiva sisääntulo.
Tätä voi käyttää, kun

- sisääntulo ei kasva tai vähene liikaa rajoituksia vastaan
- tyyppi ei muutu parametrille epäsopivaksi.

2.4.12. PID-säädin

PID-säädin on logiikan monimutkaisin operaattori. Sen toimintaa on havainnollistettu kuvassa sivulla 21.

PID-säädin tarvitsee toimiakseen kaksi sisääntuloa. Asetus ja mittaus ovat nämä, jotka on piirretty kuvaan nelikulmion sisään. Ympyrät ovat käyttäjän syöttämiä arvoja ja ovaalit ovat joko välituloksia tai arvoja joita käyttäjä voi nähdä. Lisäksi ulostulon maksimit ja minimit on niputettu yhteen arvoon. Tulos on myös laatikko.

PID-säädin toteuttaa funktion, joka on esitetty kaavassa 2 seuraavalla sivullakokonaisuutena. Tätä tulee kuitenkin ensin tarkastella osina, jotta funktion toiminta selkeytyisi. Suppeassa muodossa funktio voidaan kirjoittaa $Tulos = P_{termi} + I_{termi} - D_{termi}$, jotka voidaan pilkkoa pienemmiksi funktioksi. $P_{termi} = P_{kerroin} * virhe$, missä $virhe = asetetus - mittaus$. P termin arvo on siis suoraan verrannollinen asetuksen ja mittauksen erotukseen. Tämä vastaa P:n määritelmää proportional. *I termi* on integraatio. Jotta integraatio voi toimia tarvitaan tietoa

edellisestä arvosta. Tämä on tallennettu PID-säätimen parametriksi *I tila*. $I_{tila} = I_{edellinen} + virhe$ Ja silloin $I_{termi} = I_{kerroin} * I_{tila}$. D tila voidaan sanoa olevan edellinen virhe ja D termiksi voidaan näin kirjoittaa $D_{termi} = D_{kerroin} * virhe_{edellinen}$.

PID-säätimen eräs ongelma on *I tilan* kasvaminen tai romahtaminen liikaa, mikäli säätö ei pure tarpeeksi tehokkaasti. Siksi sen saamat arvot on rajoitettu siten, että *I termi* ei voi yksistään ajaa tulosta varoitusalueelle. Mikäli kuitenkin asetusarvo on asetettu *varoitus* ja *virhe* alueen väliin, voi *I termi* ajaa ulostuloa virherajojen sisällä. Yksistään *I termi* ei kuitenkaan tässä tilanteessa aja virhe alueelle. Tätä prosessia on kuvattu I tilan maksimi ja minimi ovaalilla kuvassa seuraavalla sivulla. Kaavassa 1 on esitetty tämän raja-arvon laskenta. Laskenta suoritetaan molemmille raja-arvoille. Mikäli *I tilan* laskenta menee raja-arvon ylitse, rajoitetaan *I tila* laskettuun raja-arvoonsa. Tämä ei aiheuta mitään virhettä.

$$I_{raja} = \frac{Ulostulo_{raja} + D_{termi} - P_{termi}}{I_{kerroin}} \tag{1}$$

$$Tulos = P_{kerroin} * (asetus - mittaus) + I_{kerroin} * \sum(asetus - mittaus) - D_{kerroin} * (asetus - mittaus)_{edellinen} \tag{2}$$

PID-säädin saa tiloikseen normaalit virheelliset ja puuttuvat sisääntulot Lisäksi ulostulon rajat on määritelty varoituksin ja virhe alueisiin. Sisääntuloksi PID-säätimelle kelpaa kaikki muut tietotyypit paitsi totuusarvot.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Asetus	Mittaus
Alhainen	**Z:** Säädin	PID	5	-13	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja

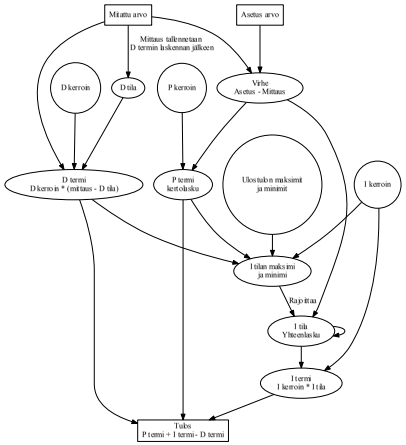
Jatkoa perästä

Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä	P kerroin	I kerroin	D kerroin
-25.3000	0.0000	21474836.00	21474836.00	1.000000	25.00000	0.00000

Edelleen jatkoa perästä arvoja joita käyttäjä ei voi muuttaa.

I termi ala	I termi ylä	I tila	D tila	P termi	I termi	D termi
0.000000	0.000000	-13.000000	19.000000	-13.000000	-0.000000	0.000000

PID Säädin säätää ulostuloa P:n I: ja D:n kertoimien kohti asetusta.
Normaalit virhe ja varoitusrajat on käytössä. Lisäksi integraattorin kerääntyminen on pysäytetty säätöalueelle.
Sallitut sisääntulotyypit **N, Z** ja **R** .



Kuva 7: Pid-säätimen toiminta

2.4.13. Summa

Summa on ulostulorajoitettu operaatio, jonka toiminta muuttuu sisääntulojen tyyppien mukaan. Tosin kuin algebran summa, yhteenlaskettavat eivät ole sivuvaikutusten takia yleisesti päittäin vaihdettavia. Ulostulon tietotyyppi asetetaan ensimmäisen yhteenlaskettavan mukaan.

Summalla on erityisominaisuutena boolen algebran mukainen OR operaatio ensimmäisen yhteenlaskettavan ollessa totuusarvo tietotyyppiä. OR operaatio voidaan esittää lauseella *mikäli kumpikaan sisääntuloista ei ole ON, on ulostulo OFF muutoin ON*, ja kääntäen *mikäli toinen tai molemmat sisääntulot ovat ON, niin ulostulo ON muutoin OFF.* Jos toinen sisääntulo ei ole totuusarvo tietotyyppiä, toimitaan kuten kohdassa 2.2.1 sivulla 11 on määritelty.

Toinen erityisominaisuus on summalla, kun ensimmäinen yhteenlaskettava ei ole ole totuusarvoinen tietotyyppi ja toinen on. Silloin sisääntulo lasketaan yhteen edellisen tuloksen kanssa jos toinen sisääntuloista on ON tilassa. Mikäli toinen sisääntuloista on OFF tilassa, siirtyy ensimmäisen sisääntulon arvo suoraan tulokseksi.

Muilla tietotyypeillä summa toimii normaalin yhteenlaskun tapaan. Kokonaislukuun voi lisätä reaaliluvun, mutta desimaalit katkaistaan pois. Lisäksi ulostulojen rajaus voi rajoittaa arvot virhe arvoihin tai lukualueen rajoihin. Valittu toimintamuoto näkyy käyttöliittymän taulukossa vihreällä taustavärillä. Käyttöliittymä on kuvattu seuraavalla sivulla ilman värejä.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Yhteen-laskettava	Yhteen-laskettava	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
OK	**Z:** Plus	Summa	5	25	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	-25.30	0.0000	214.000	214.00

Ulostulon *tyyppi* määräytyy ensimmäisen sisääntulon mukaan.

Funktio	Ensimmäisen sisääntulon tyyppi	Toisen sisääntulon tyyppi	Ulostulon tyyppi	Ulostulon arvo
Summa	**B**	**B, N, Z** tai **R**	**B**	Ensimmäinen sisääntulo *OR* Toinen sisääntulo
Summa	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo + entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Summa	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Ensimmäinen sisääntulo + Toinen sisääntulo
Erotus	**B**	**B, N, Z** tai **R**	**B**	Jos Vähentäjä *ON* Ulostulo *OFF* Muuten Vähenevä
Erotus	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Vähentäjä *ON* Vähenevä - 1 Jos Vähentäjä *OFF* Vähenevä
Erotus	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Vähenevä - Vähentäjä
Kertolasku	**B**	**B, N, Z** tai **R**	**B**	Kerrottava *AND* Kertoja
Kertolasku	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo * entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Kertolasku	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Ensimmäinen sisääntulo * Toinen sisääntulo
Jakolasku	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Jaettava / Jakaja
Jakojäännös	**N** tai **Z**	**N** tai **Z**	**N** tai **Z**	Ensimmäinen sisääntulo *MODULO* Toinen sisääntulo

Huomioitava on, että kaikki tyypit eivät kelpaa kaikille funktioille. Lisäksi nollalla jakaminen aiheuttaa virheen. Jakojäännös toimii vain *luonnollisilla* ja *kokonaisluvuilla* ja jakolasku ei toimi *totuusarvoilla*.

2.4.14. Erotus

Erotus on ulostulorajoitettu operaatio, jonka toiminta muuttuu sisääntulojen tyyppien mukaan. Ulostulon tietotyyppi asetetaan vähenevän mukaan.

Erotuksella on erityisominaisuuksina ehdollinen nollaus operaatio vähenevän ollessa totuusarvo tietotyyppiä. Ehdollinen nollausoperaatio voidaan esittää lauseella *mikäli vähentäjä on ON, on ulostulo OFF muutoin vähenevän arvo*. Jos vähentäjä ei ole totuusarvo tietotyyppiä, toimitaan kuten kohdassa 2.2.1 sivulla 11 on määritelty.

Toinen erityisominaisuus on erotuksella, kun vähenevä ei ole totuusarvo tietotyyppinen ja vähentäjä on. Silloin sisääntulosta vähennetään yksi jos vähentäjä on ON tilassa. Mikäli vähentäjä on OFF tilassa, siirtyy vähentäjän arvo suoraan tulokseksi.

Muilla tietotyypeillä erotus toimii normaalin vähennyslaskun tapaan. Kokonaislukuun voi lisätä reaaliluvun, mutta desimaalit katkaistaan pois. Lisäksi ulostulojen rajaus voi rajoittaa arvot virhe arvoihin tai lukualueen rajoihin. Valittu toimintamuoto näkyy käyttöliittymän taulukossa vihreällä taustavärillä. Käyttöliittymä on kuvattu seuraavalla sivulla ilman värejä.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Vähenevä	Vähentäjä	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
Alhainen	**Z:**Miinus	Erotus	5	-13	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	-25.300	0.000000	2147484.00	2147483647.00

Ulostulon *tyyppi* määräytyy ensimmäisen sisääntulon mukaan.

Funktio	Ensimmäisen sisääntulon tyyppi	Toisen sisääntulon tyyppi	Ulostulon tyyppi	Ulostulon arvo
Summa	**B**	**B**, **N**, **Z** tai **R**	**B**	Ensimmäinen sisääntulo *OR* Toinen sisääntulo
Summa	**N**, **Z** tai **R**	**B**	**N**, **Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo + entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Summa	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Ensimmäinen sisääntulo + Toinen sisääntulo
Erotus	**B**	**B**, **N**, **Z** tai **R**	**B**	Jos Vähentäjä *ON* Ulostulo *OFF* Muuten Vähenevä
Erotus	**N**, **Z** tai **R**	**B**	**N**, **Z** tai **R**	Jos Vähentäjä *ON* Vähenevä - 1 Jos Vähentäjä *OFF* Vähenevä
Erotus	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Vähenevä - Vähentäjä
Kertolasku	**B**	**B**, **N**, **Z** tai **R**	**B**	Kerrottava *AND* Kertoja
Kertolasku	**N**, **Z** tai **R**	**B**	**N**, **Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo * entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Kertolasku	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Ensimmäinen sisääntulo * Toinen sisääntulo
Jakolasku	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Jaettava / Jakaja
Jakojäännös	**N** tai **Z**	**N** tai **Z**	**N** tai **Z**	Ensimmäinen sisääntulo *MODULO* Toinen sisääntulo

Huomioitava on, että kaikki tyypit eivät kelpaa kaikille funktioille. Lisäksi nollalla jakaminen aiheuttaa virheen.
Jakojäännös toimii vain *luonnollisilla* ja *kokonaisluvuilla* ja jakolasku ei toimi *totuusarvoilla*.

2.4.15. Jakolasku

Jakolasku on määritelty luonnollisille, kokonais- ja reaaliluvuille. Lisäksi sen ulostulossa on rajoitukset. Jakolaskussa muilla kuin reaaliluvuilla tiputetaan desimaalit pois ja niitä ei pyöristetä. Esimerkiksi luonnollisilla ja kokonaisluvuilla laskutoimitus olisi $\frac{7}{4} = 1.75 \approx 1$, kun reaaliluvuilla saadaan tulokseksi juurikin 1.75.

Rajoittavat ylä- ja alavaroitukset sekä virheet ovat olemassa. Nollalla jakaminen aiheuttaa kriittisen *ei päivitystä* tilan. Lisäksi puuttuvat sisääntulot aiheuttavat kriittisen tilan.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Jaettava	Jakaja	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
OK	**Z:** Osamäärä	Jakolasku	5	0	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	-25.300	0.0000	2147.000	2147.000

Ulostulon *tyyppi* määräytyy ensimmäisen sisääntulon mukaan.

Funktio	Ensimmäisen sisääntulon tyyppi	Toisen sisääntulon tyyppi	Ulostulon tyyppi	Ulostulon arvo
Summa	**B**	**B, N, Z** tai **R**	**B**	Ensimmäinen sisääntulo *OR* Toinen sisääntulo
Summa	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo + entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Summa	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Ensimmäinen sisääntulo + Toinen sisääntulo
Erotus	**B**	**B, N, Z** tai **R**	**B**	Jos Vähentäjä *ON* Ulostulo *OFF* Muuten Vähenevä
Erotus	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Vähentäjä *ON* Vähenevä - 1 Jos Vähentäjä *OFF* Vähenevä
Erotus	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Vähenevä - Vähentäjä
Kertolasku	**B**	**B, N, Z** tai **R**	**B**	Kerrottava *AND* Kertoja
Kertolasku	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo * entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Kertolasku	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Ensimmäinen sisääntulo * Toinen sisääntulo
Jakolasku	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Jaettava / Jakaja
Jakojäännös	**N** tai **Z**	**N** tai **Z**	**N** tai **Z**	Ensimmäinen sisääntulo *MODULO* Toinen sisääntulo

Huomioitava on, että kaikki tyypit eivät kelpaa kaikille funktioille. Lisäksi nollalla jakaminen aiheuttaa virheen.
Jakojäännös toimii vain *luonnollisilla* ja *kokonaisluvuilla* ja jakolasku ei toimi *totuusarvoilla*.

2.4.16. Jakojäännös

Jakojäännös laskee väärin. Seuraavassa listassa on esitetty jakolaskun 7/4 etumerkkien vaikutusta ja miten jakojäännös laskee väärin

- $7 mod(4) = 3$ Oikein.
- $-7 mod(-4) = -3$ Oikein
- $-7 mod(4) = -3$ Väärin
- $7 mod(-4) = 3$ Väärin

Lukijalle jätettäköön harjoitustehtäväksi kirjoittaa oikeat arvot virheellisten perään. Jakojäännöksen tulos on jaettavan etumerkki tuloksen edessä ja jakojäännös lasketaan jaettavan ja jakajan itseisarvoilla. Oikea toimintaa ei ole mahdollista toteuttaa tietotekniikan rajoittuneisuudesta johtuen helposti.

Jakojäännös on määritelty kokonaisluvuille ja luonnollisille luvuille. Muut tietotyypit aiheuttavat *virheellinen sisääntulo* kriittisen tilan. Sisääntulon puuttuessa tulee kriittinen *sisääntulo puuttuu* tila. Myöskään nollalla ei tässä saa jakaa. Ulostulo on rajoitettu virhearvojen väliin ja varoitusarvot ovat toiminnassa.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Jaettava	Jakaja	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
OK	**Z:** Modulo	Jakojäännös	5	6	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	-25.300	0.00000	21474.00	21474.00

Ulostulon *tyyppi* määräytyy ensimmäisen sisääntulon mukaan.

Funktio	Ensimmäisen sisääntulon tyyppi	Toisen sisääntulon tyyppi	Ulostulon tyyppi	Ulostulon arvo
Summa	**B**	**B, N, Z** tai **R**	**B**	Ensimmäinen sisääntulo *OR* Toinen sisääntulo
Summa	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo + entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Summa	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Ensimmäinen sisääntulo + Toinen sisääntulo
Erotus	**B**	**B, N, Z** tai **R**	**B**	Jos Vähentäjä *ON* Ulostulo *OFF* Muuten Vähenevä
Erotus	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Vähentäjä *ON* Vähenevä - 1 Jos Vähentäjä *OFF* Vähenevä
Erotus	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Vähenevä - Vähentäjä
Kertolasku	**B**	**B, N, Z** tai **R**	**B**	Kerrottava *AND* Kertoja
Kertolasku	**N, Z** tai **R**	**B**	**N, Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo * entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Kertolasku	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Ensimmäinen sisääntulo * Toinen sisääntulo
Jakolasku	**N, Z** tai **R**	**N, Z** tai **R**	**N, Z** tai **R**	Jaettava / Jakaja
Jakojäännös	**N** tai **Z**	**N** tai **Z**	**N** tai **Z**	Ensimmäinen sisääntulo *MODULO* Toinen sisääntulo

Huomioitava on, että kaikki tyypit eivät kelpaa kaikille funktioille. Lisäksi nollalla jakaminen aiheuttaa virheen. Jakojäännös toimii vain *luonnollisilla* ja *kokonaisluvuilla* ja jakolasku ei toimi *totuusarvoilla.*

2.4.17. Kertolasku

Kertolasku on hyvin saman kaltainen kuin summa kohdassa 2.4.13 sivulla 21. Kerrottavan ollessa totuusarvo tyyppinen on looginen operaatio AND. Silloin pitää molempien kerrottavan ja kertojan olla ON tilassa, jotta tulos on ON. Muutoin tulokseksi saadaan OFF.

Sisääntulon ja edellisen arvon kertolasku on kertolasku, kun summassa se oli yhteenlasku. Tämä tapahtuu jos kertoja on totuusarvo tietotyypiltään. Muuten kertolaskussa on virherajat ja muilla kuin reaaliluvuilla kerrottaessa desimaalit pudotetaan pois. Ulostulon tyyppi määräytyy kerrottavan tietotyypistä. Näistä siis seuraa, että $5 * 1.75 = 8.75 \approx 8$ ja $1.75 * 5 = 8.75$, koska kerrottava määrää tuloksen tyypin.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Kerrotta-va	Kertoja	Virhe ala	Varoitus ala	Varoitus ylä	Virhe ylä
OK	**Z:** Tulo	Kertolasku	5	114	**Z:** Int var 6 Muuttuja	**Z:** Int var 19 Muuttuja	-25.300	0.0000	214748.00	2147483.00

Ulostulon *tyyppi* määräytyy ensimmäisen sisääntulon mukaan.

Funktio	Ensimmäisen sisääntulon tyyppi	Toisen sisääntulon tyyppi	Ulostulon tyyppi	Ulostulon arvo
Summa	**B**	**B**, **N**, **Z** tai **R**	**B**	Ensimmäinen sisääntulo *OR* Toinen sisääntulo
Summa	**N**, **Z** tai **R**	**B**	**N**, **Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo + entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Summa	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Ensimmäinen sisääntulo + Toinen sisääntulo
Erotus	**B**	**B**, **N**, **Z** tai **R**	**B**	Jos Vähentäjä *ON* Ulostulo *OFF* Muuten Vähenevä
Erotus	**N**, **Z** tai **R**	**B**	**N**, **Z** tai **R**	Jos Vähentäjä *ON* Vähenevä - 1 Jos Vähentäjä *OFF* Vähenevä
Erotus	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Vähenevä - Vähentäjä
Kertolasku	**B**	**B**, **N**, **Z** tai **R**	**B**	Kerrottava *AND* Kertoja
Kertolasku	**N**, **Z** tai **R**	**B**	**N**, **Z** tai **R**	Jos Toinen sisääntulo *ON* Ensimmäinen sisääntulo * entinen tulos. Jos Toinen sisääntulo *OFF* Ensimmäinen sisääntulo
Kertolasku	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Ensimmäinen sisääntulo * Toinen sisääntulo
Jakolasku	**N**, **Z** tai **R**	**N**, **Z** tai **R**	**N**, **Z** tai **R**	Jaettava / Jakaja
Jakojäännös	**N** tai **Z**	**N** tai **Z**	**N** tai **Z**	Ensimmäinen sisääntulo *MODULO* Toinen sisääntulo

Huomioitava on, että kaikki tyypit eivät kelpaa kaikille funktioille. Lisäksi nollalla jakaminen aiheuttaa virheen.
Jakojäännös toimii vain *luonnollisilla* ja *kokonaisluvuilla* ja jakolasku ei toimi *totuusarvoilla*.

2.5. Logiikan luonti, muokkaus ja poisto

Logiikka toimii ketjumaisesti. Ketjun lenkki voi ottaa sisään kaikista muista lenkeistä, jotka ovat sille sopivia. Ulos se antaa yhden arvon. Tätä on ilmennetty kuvassa seuraavalla sivulla, jonka esimerkin kautta käydään läpi logiikan luonti ja asetusten tekeminen. Tämä on annettu harjoitustehtäväksi kohdassa 3.2 sivulla 39. Siinä yksi ulostulo menee moneen paikkaan. Mikäli olisi haluttu rakentaa enemmän logiikkaa kuin pelkät *pienempi kuin* viittaukset, ne olisivat laskettu myös joka kerta logiikkaa päivitettäessä. Eli laskenta ei voi haarautua, mutta se voi valita. Toisin sanoen ei voida valita tuloksen perusteella, että tämä sisääntulon arvo näyttää nyt sellaiselta, että laskenta menee toista polku pitkin. Voidaan silti kuitenkin laskea kaksi eri polkua ja valita jossain kohdassa toisen polun tulos johonkin sisääntuloksi. Tähän tarkoitukseen sopivat Jos ja MUX tyyppiset operaatiot, jotka ovat käsitelty sivulla 18 ja sivulla 18. Esimerkiksi voidaan luoda murtoviiva *suora*-operaatioilla ja asettaa sisääntulon rajat peräkkäin niin, että viiva on jatkuva. Tämän jälkeen suorista valitaan MUXilla oikea, eli se mikä on OK tilassa. Suoran sivuttaissiirtoja voi tehdä summalla. Summa suorien edessä siirtää X-suunnassa suoraa toisen yhteenlaskettavan verran ja summa suoran jälkeen siirtää Y-suunnassa. Y-suuntainen siirto voi kuitenkin johtaa murtoviivan epäjatkuvuuteen, joten parempi paikka olisi MUXin jälkeen. Silloin koko murtoviiva liikkuu ylös ja alas. Suoraan voi tutustua kohdassa 2.4.8 sivulla 17 ja summaan kohdassa 2.4.13 sivulla 21.

2.5.1. Luonti

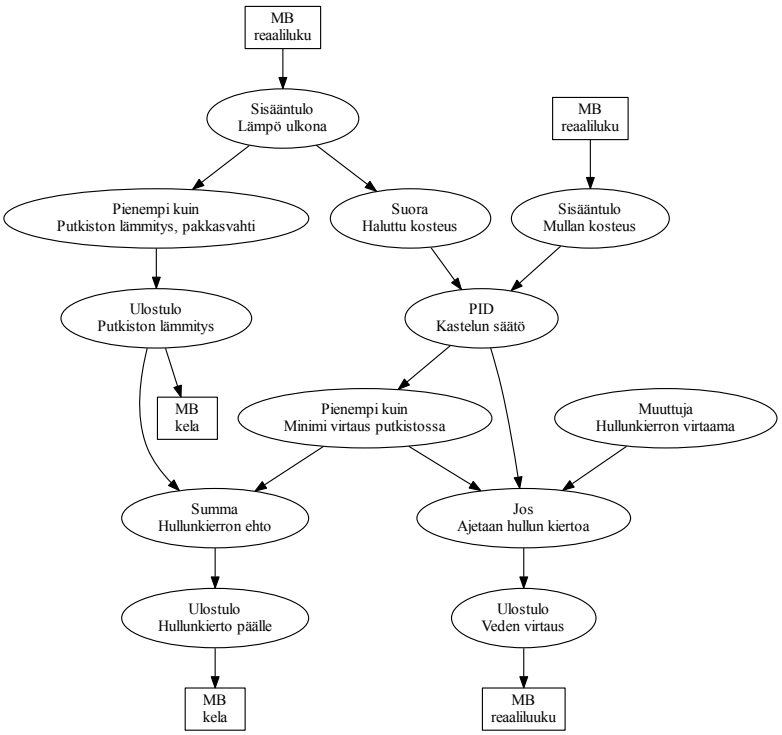
Järkevin ketjutusjärjestys olisi tehdä kolme ketjua. Luominen onnistuu valitsemalla *luo logiikkaa* sivu. Linkin paikka on esitelty kohdassa 1.4.2 sivulla 5. Luominen tapahtuu kuvan 9 sivulla 33 kaltaiselta sivulta valitsemalla kohdan, minkä perään uusi lenkki ketjussa tulee. Alkuun kohdan valitseminen asettaa uuden logiikkayksikön ketjun alkuun. Uusi ketju voidaan luoda vain alimmaksi ketjuksi. Tyyppi pudotusvalikosta valitaan luotavan logiikkayksikön tyyppi. Päivitysväli valitaan lopuksi toisesta pudotusvalikosta. Ensimmäistä logiikkayksikköä luotaessa ei logiikkayksiköiden taulukkoa näytetä.

Luodaan ensin Ulostulot pumppaukselle, hullunkierrolle ja lämmitykselle. *Veden virtaus* ulostulon eteen luodaan Jos, Pienempi kuin ja suurempi kuin, Muuttuja, PID-säädin, Sisääntulo kosteudelle, Suora ja Sisääntulo lämpötilalle. Eli kiivetään käänteisessä järjestyksessä logiikkaa ja käyttöliittymästä laitetaan ketjun *alkuun* pallukka. Keskimmäiseen hullunkierron ketjuun luodaan Ulostulon eteen Summa tyyppinen logiikkayksikkö ja alimpaan luodaan pakkasvahti Pienempi kuin ja suurempi kuin lenkki lämmityksen Ulostulon eteen.

2.5.2. Muokkaus

Logiikkayksiköiden säätäminen kannattaa tehdä sisääntulosta ulospäin. Säätösivulle pääsee, kun ensin valitsee *luo logiikka* sivulta logiikkayksikön ja painaa nappia *katso*. Kuvassa 9 sivulla 33, joka on *luo logiikkaa* sivulta löytyvät pallukoiden paikat ja *katso* nappi. Kuvassa 10 sivulla 33 näkyy muuttujan muokkaussivu. Jokaisen operaattorin muokkaus sivu on omanlaisensa ja tarkoitukseensa kehitetty. Niiden kuvaukset ja toiminta alkavat kohdasta 2.4 sivulla 13. *Muuta* nappi päivittää parametreja, mutta ei aiheuta päivitystä. Tämä tarkoittaa, että kyseisen logiikkayksikön tulos tai tila ei päivity painettaessa *muuta* nappia. *Takaisin* nappi vie takaisin *luo logiikkaa* sivulle.

Ylimmässä ketjussa *sisääntulo*ihin laitetaan Modbus-laitteet, *suora* ja *PID* viritetään ja *pienempi kuin* saa raja-arvonsa. *Jos* saa ehdokseen *pienempi kuin* minimi virtaus putkistossa niin *silloin* kohtaan tulee *PID,* sekä tai *muuten* saa arvokseen *muuttuja*n, joka pitää sisällään hullunkierron minimivirtauksen. Lopuksi *ulostulo* saa sisääntulokseen *Jos* lenkin ja Modbus-laite asetetaan osoittamaan oikeaan laitteeseen.



Kuva 8: Kasvihuoneen lämmitetty kastelujärjestelmä

Ylin ketju on siinä. Huomioitavaa on, että minimivirtausta vertaava pienempi kuin lenkin raja-arvo ei ole esimerkiksi hullunkierron muuttuja. On aivan mahdollista, että muuttujan arvo olisi toinen ja mikäli ohjaus alittaa arvon, niin virtaus hyppäisi tai romahtaisi. Toisaalta tämäkin voitaisiin laittaa toiseksi vertailuarvoksi ja kytkemällä vertaa sisääntulot päälle.

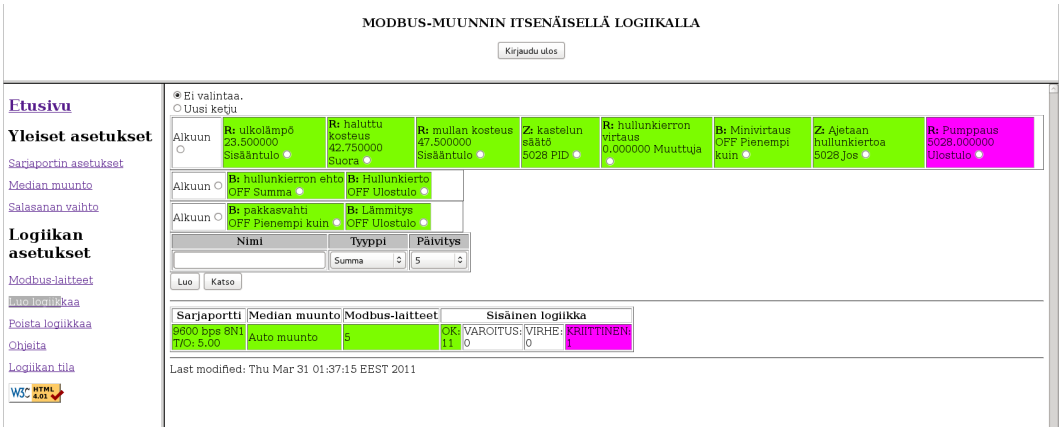
Keskimmäisen ketjun *Summa*, joka tulee toimimaan loogisena OR-operaattorina, saa sisääntuloikseen kolmannen lämmitysketjun *ulostulo*n ja ensimmäisen pumppausketjun *Pienempi kuin* lenkin.

Alimman eli lämmitysketjun *pienempi kuin* lenkki saa sisääntulokseen ensimmäisessä ketjussa olevan lämpötilan *sisääntulo*n ja raja-arvo *muuttujan*.

Päivityksen tapahtuessa, luetaan ensin sisääntulojen arvot, lasketaan pumppaus ja kirjoitetaan se. Sen jälkeen päätetään hullunkierrosta ja lopuksi lämmityksestä. Mikäli kaikilla on sama päivitysväli, kaikki toimii juuri näin. Erilaisia ratkaisuja pystytään luomaan päivitysväliä muokkaamalla.

Esimerkiksi mikäli ensimmäisen ketjun pienempi kuin lenkin päivitysväliä pidennetään, pumppaus ohjautuu hitaammin hullun kierrolle. Laittamalla PIDin ja pienempi kuin väliin vielä keskiarvon saadaan aikaan hystereesiä.

Ratkaisuna on esitetty kuvakaappaus 9 seuraavalla sivulla luo logiikka sivulta. Kriittisen tilan aiheuttanut ohjelmistovika on korjattu.



Kuva 9: Kasvihuoneen lämmitys toteutettuna

Nimen ja sisääntulojen yhteydessä esiintyvät kirjaimet kuvaavat ulostulon *tyyppi*ä.
B on *totuusarvo* ja se saa arvoikseen *ON* ja *OFF*. *OFF* voidaan tulkita nollaksi ja *ON* kaikiksi muiksi arvoiksi.
N on *luonnollinen luku* ja sen arvot ovat kaikki positiiviset *kokonaisluvut* ja nolla.
Z on *kokonaisluku*
R on *reaaliluku*

Järjestelmä suorittaa laskutoimitukset *reaaliluvuilla* mikäli mahdollista.
Jotkin toiminnot eivät kuitenkaan ole mahdollisia niillä.
Pyöristystä ei tehdä. Desimaalit vain katkaistaan pois.

Tila	Nimi	Tyyppi	Päivitys	Tulos	Arvo
Oletuksessa	**R:** hullunkierron virtaus	Muuttuja	5 5	0.000000	25.720000 25.720000 Ei muutosta

Muuta Takaisin

Ulostulo on asetettu arvo. Asettamalla uuden arvon **Arvo** kenttään
ja valitsemalla **Muuta** muuttuu ulostulon arvo.
Ulostulon *tyyppiÃ¤* voidaan muuttaa valikosta ja valitsemalla **Muuta**

Kuva 10: Muuttujan muokkaus

2.5.3. Poisto

Poisto onnistuu valitsemalla *Poista logiikkaa* sivu. Linkin paikka on esitelty kohdassa 1.4.2 sivulla 5. Silloin eteen avautuu taulukko, joihin on listattu ne logiikkayksiköt, jotka eivät ole minkään sisääntulona.

Mikäli sisääntulona oleva logiikka poistettaisiin, olisi seurauksena sitä käyttävän logiikkayksikön kriittinen tila. Lisäksi koko logiikan rakentelu voisi mennä pieleen, kun poistaisi väärän, aiheuttaisi kaaoksen logiikkaan ja ei muistaisi mitä pitikään olla arvoissa uudelleen luotaessa samaa logiikkaa tilalle.

Poisto tapahtuu valitsemalla yhden logiikkayksikön ja painamalla poista. Pois poisto sivulta pääsee valitsemalla jonkin toisen linkin.

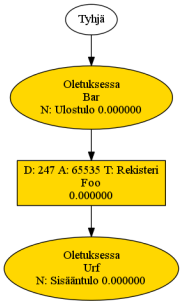
2.6. Logiikan tila

2.6.1. Tietojen kuvaus

Logiikan tila on koneluettavaksi tarkoitettu sivu. Se on Bash[3] komentoriville kirjoitettu ohjelma. Tässä esimerkkinä käytettävä ohjelma on listauksessa 4 sivulla 37. Tämä ohjelma luo Modbus-laitteista ja logiikasta CSV-tiedostot. Lisäksi koko järjestelmästä luodaan system.dot tiedosto. Dot tiedosto on tekstiä ja sisältää kuvauksen, josta Graphviz ohjelmisto[4] voi luoda graafisia kuvaajia. Tiedostot ovat tekstiä ja niitä voi ihminen kuitenkin lukea ja ymmärtää. Kuvassa 11 on kuvattu keskelle yksi Modbus-laite, joka on laitteessa 247, osoitteessa 65535, tyypiltään rekisteri ja nimeltään Foo. Lopuksi laitteen arvo on esitetty muotoilematta. Modbus-laitteet esitetään nelikulmioilla ja logiikka ovaaleilla.

Bar niminen ulostulo on keskeneräinen. Sen sisääntulo on tyhjä ja tiedon kulku tyhjästä on kuvattu nuolelle. Ulostulo kirjoittaa Foo nimiseen Modbus-laitteeseen. Toisaalta Sisääntulo Urf lukee juurikin saman Foon. Logiikan pallukoiden ylimmällä rivillä on tila, joka on käsitelty kohdassa 2.3 sivulla 12. Tämän jälkeen on nimi keskimmäisellä rivillä ja alimmalla rivillä on ulostulon tyyppi, joka käsiteltiin kohdassa 2.2 sivulla 11. Lisäksi alimmalla rivillä on kerrottu logiikkayksikön tyyppi ja arvo.

Nuolet kuvassa kuvaavat tiedon kulkua ja pallukoiden ja suorakaiteiden taustavärillä on myös merkitystä. Vihreä on OK väri, keltainen varoitus ja punainen on virhe tai kriittinen tila. Mikäli näkee punaista, mielenkiinto herää kuitenkin, joten ei ole syytä erotella logiikan kriittisiä ja virheellisiä tiloja.



Kuva 11: Yksinkertainen ja keskeneräinen systeemikuva

CSV[5] tiedostot sisältävät laitteen tilan. Tiedostossa on kentät erotettu puolipisteellä (;) ja

[3] `http://www.gnu.org/software/bash/bash.html`
[4] `http://www.graphviz.org`
[5] CSV tulee sanoista Comma separated values. Erottimena on vain tällä kertaa puolipiste.

mahdollisissa tekstikentissä on lainausmerkit (“ “) ympärillä. Ensimmäinen rivi on otsikkorivi ja viimeinen rivi on aina #END OF MB CSV. Listauksessa 1 on esitetty edellä kuvatun systeemikuvan Modbus-laitteiden tiedot. Tila esitetään tekstinä ja numeerisesti.

Tyypistä on annettu ainoastaan numeerinen ilmaisu. Sallitut arvot ovat alla olevassa listassa.

1. Sisääntulo
2. Kela
3. Sisääntulo rekisteri
4. Rekisteri
5. Sisääntulo kokonaisluku
6. Kokonaisluku
7. Etumerkitön sisääntulo kokonaisluku
8. Etumerkitön kokonaisluku
9. Reaaliluku
10. Sisääntulo reaaliluku

Vain luku kenttä saa arvot 0 ja 2, jossa 0 sallii kirjoituksen. Vastaavasti 0x17 tuki saa arvot 0 ja 4, jossa 4 kytkee tuen päälle. Alin rivi on tunniste, josta voidaan periaatteessa katkaista tuo CSV tieto. Tunnisteiden mukaan katkaiseminen on käsitelty tarkemmin kohdassa 2.6.2 seuraavalla sivulla.

Listaus 1: Esimerkin modbus.csv tiedoston sisältö

```
1 "Tilan nimi"; "Tila"; "Nimi"; "Laite"; "Osoite"; "Tyyppi"; "Arvo"; "Päivitysväli"; "Vain luku"; "0
      x17"
2 "Lukematta";   -1; "Foo";   247;  65535;       4; 0.0000;            5;          0;    4;
3 #END OF MB CSV
```

Logiikasta saadaan myös CSV tiedosto. Sen rakenne on sama, kuin Modbus-laitteilla. Ensimmäinen on otsikkorivi ja viimeinen on #END OF LOGIC CSV

Tulkitse sarake käyttää samoja tuttuja tietotyyppien lyhenteitä ja kummallisissa tilanteissa kysymysmerkkiä (?).

Tyyppi on ilmoitettu tekstinä, koska numeerisella arvolla ei ole eri käynnistyskertojen välillä mitään johdonmukaisuutta.

Asynkroni voidaan selittää siten, että 1 sallii hitaamman sisääntulon ja 2 nopeamman. Laskemalla ne yhteen saadaan 3, joka sallii molemmat tai sitten 0 joka vaatii saman päivitystahdin tai tilaksi tulee varoitus.

Mikäli logiikalla ei ole jotain sisääntuloa, on sen tilalla tyhjä kenttä. Mikäli sillä on sisääntulo, mutta sisääntulolla ei ole nimeä näkyvät lainausmerkit (“”) tai jos se on välilyönti, niin välilyönti näkyy lainausmerkeissä (“ “). Normaalisti oikea nimi niin sisääntulon kohdalla ja näkyy (“nimi”). Listauksessa 2 näkyy useimmat ominaisuudet. Lisäksi on huomattava, että otsikoita ei ole kaikille parametreille ilmoitettu.

Listaus 2: Logiikan CSV tiedosto

```
1 "Tilan nimi"; "Tila"; "Nimi"; "Tulkitse"; "Tyyppi"; "Arvo"; "Päivitysväli"; "Asynkroni"; "Sisääntulo
      1"; "Sisääntulo 2";"Sisääntulo 3"; "Sisääntulo 4"; "Parametrit 1-n"
2 "Oletuksessa"; 6; "Urf"; "N"; "Sisääntulo"; 0.000000; 5; 0; ; ; ; ; 4144300031.000000; ; ; ; ; ; ; ;
      ; ; ; ;
3 "Oletuksessa"; 6; "Bar"; "N"; "Ulostulo"; 0.000000; 5; 3;  ; ; ; ; 4144300031.000000; ; ; ; ; ; ; ;
      ; ; ; ;
4 #END OF LOGIC CSV
```

DOT tiedosto on myös tekstiä ja sekin on ihmiselle luettavaa. Itse asiassa useimmat kuvat kuten 8 sivulla 32 ovat käsin kirjoitettu DOT tiedostona ja siitä on luotu kuva. Listaus 3 esittelee kuvan 11 sivulla 34 luontiin käytetyn tiedoston. Tiedoston ensimmäinen rivi kertoo, että halutaan kuva, jossa suuntia ja sen nimi on system. Toinen rivi määrittää käytetyn merkistön. Koska on laitteen puolelta helppoa tuottaa ISO-8859-1 eli Latin1 merkistöä, sitä käytetään. Graphviz-ohjelmisto käyttää oletuksena UTF-8 merkistöä ja ilman merkistön erikseen asettamista, Graphviz ei toimisi. Rivi kolme kertoo pallukoiden ja suorakaiteiden olevan mustalla täytettyjä. Neljäs rivi luo "Tyhjä" pallukan. Huomioitavaa on, että mikäli fillcolor = transparent puuttuisi, koko pallukka olisi musta. Tähän saakka system.dot tiedosto on aina samanlainen. Numerolla alkava rivi ja sen perässä määritteet hakasuluissa kuvaa Modbus-laitteen. Lisäksi määritteissä on shape = box. L-kirjain luku alaviiva luku kuvaa logiikkaoperaation. Lisäksi kuvauksessa käytetään nuolta ->, jolla piirretään kuvan nuolet.

Listaus 3: Esimerkin system.dot tiedosto

```
1  digraph system{
2  graph [charset="latin1"];
3  node [style = filled, color = black];
4  empty_logic [label = "Tyhjä" fillcolor = transparent];
5  4144300031 [shape = box, fillcolor = gold, label = "D: 247 A: 65535 T: Rekisteri\nFoo\n 0.000000"];
6  L1_1 [ fillcolor = gold, label = "Oletuksessa\nUrf\nN: Sisääntulo 0.000000"];
7  4144300031 -> L1_1;
8  L1_2 [ fillcolor = gold, label = "Oletuksessa\nBar\nN: Ulostulo 0.000000"];
9  empty_logic -> L1_2;
10 L1_2 -> 4144300031;
11 }
12 #END OF SYSTEM DOT FILE
```

Eräs huomattava asia on tiedoissa. Päivitysjärjestystä ei näistä saa ulos tai milloin on viimeksi päivitetty. Tämä tarkoittaa myös esimerkissämme sitä, että ei voi tietää lukeeko Urf Foon ennen Barin kirjoitusta ilman suurempaa miettimistä. Tässä tapauksessa vastaus on kyllä. Urf on ennen Baria ja Barilla on vähintään yhtä suuri päivitysväli, joka näkyy logic.csv tiedostosta.

2.6.2. Bash ohjelman toiminta

Listauksessa 4 seuraavalla sivulla on koko ohjelma, jolla luodaan kohdassa 2.6.1 sivulla 34 käsitellyt tiedostot. Ensimmäinen rivi on taikarivi, joka kertoo järjestelmälle, että tämä on bashille tarkoitettu ohjelma. Toinen rivi on kommentti. # merkin jälkeen tuleva teksti rivin loppuun saakka ymmärretään kommentiksi. Tämän jälkeen tulee HASHFUNCTION muuttujan käsittely. Mikäli komento which ja md5sum löytyvät järjestelmästä tiivisteen luomiseen käytetään md5sum komentoa. HASHFUNCTION muuttujan käsittelyn jälkeen luodaan tiedoston loppumerkki muuttujaan EOFMARKER. Loppumerkki on joko merkkijonon *d3b07384d113edec49eaa6238ad5* tai *foobarendoffile-23123-1333650127*. Täämä siksi, että mikäli kävisi niin onnettomasti, että joku antaa jollekin logiikkayksikölle nimeksi *foobarendoffile\nrm -rf / –no-preserve-root* ei tapahdu ikäviä asioita. Ensin pitää arvata satunnaisluku, joka on tuossa 23123 ja sen jälkeen päättää ajankohta sekunnilleen jolloin tuo arvattu satunnaisluku osuu kohdalleen ja ohjelma ajetaan. Kuitenkin pahimmassa tapauksessa aikaleimaa ei voida tulostaa ja $RANDOM ei toimi.

Tätä varten on otettu käyttöön md5sum. Tiiviste ajetaan ohjelmaa itseään vastaan ja silloin ilkeämielisen lopetusmerkkiä muistuttavan logiikkayksikön nimen muuttaminen muuttaa tarkistetta ja ilkeämielisyys ei ainakaan tässä kohdassa toimi. Toisaalta kurja nimeäminen voi onnistua esimerkiksi *ilkeää"];12345*, joka rikkoo graphvizia ja CSV tiedostojakin. Yhtä ilkeämielistä nimeämistä ei ole estetty. Logiikkayksikön nimeksi voi aivan hyvin kirjoittaa HTML:ää ja sillä rikkoa tämän sivun.

Rivit 12-34 käsittelevät eri muuttujia ja niiden toiminta on esitelty kohdassa 2.6.3 seuraavalla sivulla. Rivi 35 on mielenkiintoinen niille, jotka keräävät tältä sivulta tietoa ajamatta

ohjelmaa. *cat >modbus.csv<<$EOFMARKER #BEGIN OF MB CSV* sisältää kommentin #BEGIN OF MB CSV ja vastaavasti tuon tiedoston lopussa on #END OF MB CSV. Näiden välistä voi sitten lukea Modbus-laitteiden CSV tiedot, kunhan huomioi mitä on sanottu ilkeämielisyydestä ylempänä.

Listaus 4: Logiikan tila tuottama ohjelma

```
#!/bin/bash
# Generate End of file marker for here documents.
if [ -z "$HASHFUNCTION" ]; then
which md5sum>/dev/null
if [ $? = 0 ]; then
HASHFUNCTION=`which md5sum`
else
HASHFUNCTION="echo foobarendoffile blaah"
fi
fi
EOFMARKER=`$HASHFUNCTION $0|awk '{print $1}'`-$RANDOM-`date +%s`
#Tests for file processess
if [ -z "$PREMBCSVSCRIPT" ]; then
echo "No Modbus preCSV script defined"
else source "$PREMBCSVSCRIPT"
fi
if [ -z "$PRELOGICCSVSCRIPT" ]; then
echo "No logic preCSV script defined"
else source "$PRELOGICCSVSCRIPT"
fi
if [ -z "$PREDOTSCRIPT" ]; then
echo "No pre dot script defined"
else source "$PREDOTSCRIPT"
fi if [ -z "$POSTMBCSVSCRIPT" ]; then
POSTMBCSVSCRIPT="/dev/null"
echo "No Modbus postCSV script defined"
fi if [ -z "$POSTLOGICCSVSCRIPT" ]; then
POSTLOGICCSVSCRIPT="/dev/null"
echo "No logic postCSV script defined"
fi
if [ -z "$POSTDOTSCRIPT" ]; then
POSTDOTSCRIPT="/dev/null"
echo "No post dot script defined"
fi
cat >modbus.csv<<$EOFMARKER #BEGIN OF MB CSV
"Tilan nimi"; "Tila"; "Nimi"; "Laite"; "Osoite"; "Tyyppi"; "Arvo"; "Päivitysväli"; "Vain luku"; "0
    x17"
"Lukematta";   -1; "Foo";   247;  65535;       4; 0.0000;            5;          0;    4
#END OF MB CSV
$EOFMARKER
source $POSTMBCSVSCRIPT
cat >logic.csv<<$EOFMARKER #BEGIN OF LOGIC CSV
"Tilan nimi"; "Tila"; "Nimi"; "Tulkitse"; "Tyyppi"; "Arvo"; "Päivitysväli"; "Asynkroni"; "Sisääntulo
    1"; "Sisääntulo 2";"Sisääntulo 3"; "Sisääntulo 4"; "Parametrit 1-n"
"Oletuksessa"; 6; "Urf"; "N"; "Sisääntulo"; 0.000000; 5; 0; ; ; ; ; 4144300031.000000; ; ; ; ; ; ; ;
    ; ; ; ; ;
"Oletuksessa"; 6; "Bar"; "N"; "Ulostulo"; 0.000000; 5; 3;  ; ; ; ; 4144300031.000000; ; ; ; ; ; ; ;
    ; ; ; ; ;
#END OF LOGIC CSV
$EOFMARKER
source $POSTLOGICCSVSCRIPT
cat >system.dot<<$EOFMARKER #BEGIN OF SYSTEM DOT FILE
digraph system{
graph [charset="latin1"];
node [style = filled, color = black];
empty_logic [label = "Tyhjä" fillcolor = transparent];
4144300031 [shape = box, fillcolor = gold, label = "D: 247 A: 65535 T: Rekisteri\nFoo\n 0.000000"];
L1_1 [ fillcolor = gold, label = "Oletuksessa\nUrf\nN: Sisääntulo 0.000000"];
4144300031 -> L1_1;
L1_2 [ fillcolor = gold, label = "Oletuksessa\nBar\nN: Ulostulo 0.000000"];
empty_logic -> L1_2;
L1_2 -> 4144300031;
}
#END OF SYSTEM DOT FILE
$EOFMARKER
source $POSTDOTSCRIPT
```

2.6.3. Muuttujat ohjelmassa

Muuttujat tilasivulla mahdollistavat tilasivulta tulleen ohjelman luomien tiedostojen jatkokäytön. Niiden pitää olla Bashille tarkoitettuja komentojono-ohjelmia. Listauksessa 5 seuraavalla sivulla tulee READHOST muuttuja asettaa oikeaksi IP-osoitteeksi tai nimeksi, jotta URL täydentyy. Esimerkissä on käytetty Links[6] selainta, jolla tuo tiedosto luetaan tiedostoksi status.sh. Olisi myös mahdollista, että tiedosto luettaisiin esimerkiksi wget ohjelmalla ja

[6] `http://links.twibright.com/`

erotettaisiin HTML-tiedostosta kaikki <PRE> ja </PRE> tagien välistä.

Listaus 5: Esimerkki jolla voi noutaa tilasivun

```
#!/bin/bash
READHOST=192.168.1.2
# Get the status page in latin1 and remove leading and trailing space.
links -codepage iso-8859-1 -dump http://$READHOST/status_logic.html|sed 's/^[ ]*//;s/[ ]*$//' >status.sh

# Set timestamp and place where postdot.sh puts its data.
OUTPUTDIR="data"
# Set timestamp
TIMESTAMP=date +%s

# I feel paranoid. Here is my hash function
HASHFUNCTION=./myhash.sh

## Set the helper scripts. Uncomment and set the scripts filename.
## This script file is run before Modbus CSV generation
#PREMBCSVSCRIPT=premb.sh"
## This script file is run after Modbus CSV generation
#POSTMBCSVSCRIPT="postmb.sh"
## This script file is run before logic CSV generation
#PRELOGICCSVSCRIPT="prelogic.sh"
## This script file is run after logic CSV generation
#POSTLOGICCSVSCRIPT="postlogic.sh"
## This script file is run before DOT file generation
#PREDOTSCRIPT="predot.sh"
## This script file is run after DOT file generation
POSTDOTSCRIPT="postdot.sh"
source status.sh
```

Listauksessa 5 käytettiin lisäksi kahta muuta ohjelmaa. Listaus 6 esittelee myhash.sh ohjelman, joka laskee pitkän tiivisteen liittäen SHA512 summan ja MD5 summan peräkkäin. Lisäksi se tulostaa dropthis merkkijonon, jonka sitten status.sh tiputtaa. Myös tilanteessa, jossa md5sum komentoa ei löydy ja HASHFUNCTION muuttuja ei ole asetettuna käytetty echo foobarendoffile blargh sisältää tuon tiputettavan osan, blargh. Tästä voimme siis päätellä, että HASHFUNCTION ohjelman tarvitsee tulostaa kaksi sanaa, joista ensimmäistä käytetään. Varoitettakoon myös, että esimerkiksi OpenBSD:n md5 komento tulostaa muodossa MD5 (tiedosto) = tiivistesumma, joka ei tietenkään ole tuollaisena suoraan käytännöllinen. Sen takia tiedostoon on laitettu kommenteiksi hieman eri versio.

HASHFUNCTION sisältö tulee olla suoritettava ohjelma. Toisin kuin esimerkiksi POSTDOTSCRIPT sitä ei kutsuta source komennolla vaan se ajetaan. Tämän takia echo foobarendoffile blargh sisältää ensimmäisenä komennon echo, joka on ohjelma[7].

Postdot.sh listauksessa 7 tekee joka kierroksella kuvan systeemistä. Dot on osa Graphviz-ohjelmistoa on juurikin se työkalu, jolla kuvat on luodaan dot-tiedostosta.

Listaus 6: myhash.sh

```
#bin/bash
FIRSTPART=`sha512sum $1|awk '{print $1}'`
## If we used OpenBSD we might want something like this:
#SECONDPART=`md5 $1 |awk '{print $4}'`
SECONDPART=`md5sum $1|awk '{print $1}'`
echo $FIRSTPART$SECONDPART dropthis
```

Listaus 7: postdot.sh

```
#!/bin/bash
dot -Tjpg system.dot>$OUTPUTDIR/system-$TIMESTAMP.jpg
```

CSV-tiedostoillekin on käyttöä. Niistä voi esimerkiksi tuoda tiedot helposti tietokantaan tai RRDtoolille[8], joka sitten voi kehityksen perusteella piirtää kuvia. Lisäksi tila on annettu, niin erilaisia hälytyksiä voi rakentaa ja CSV-tiedosta on helppo rakentaa tarkkailu web-sivut.

Bashilla ohjelmointi on laaja alue, eikä kuulu tämän käyttöohjeen piiriin. Siitä on jo kirjoitettu useita kirjoja.

[7] Useimmiten Bashin sisällä toteutettuna. Järjestelmästä tulisi myös löytyä tuo ohjelmanakin.

[8] RRDtool, http://oss.oetiker.ch/rrdtool/, Round robin database on tietokanta jota RRDtool käyttää

2.6.4. Käytetyt työkalut

Kerrattakoon vielä työkalut, jotka tarvitaan tilasivun ohjelman suoritukseen. Nämä löytyvät hyvin varustetuista käyttöjärjestelmistä suoraan ja huonompiin saadaan käyttämällä Cygwiniä, `http://cygwin.com/`

1. Bash, komentotulkki, `http://www.gnu.org/software/bash/bash.html`
2. Md5sum, date; apuohjelmia GNU coreutils paketista, `http://www.gnu.org/software/coreutils/`
3. Links, web-selain, `http://links.twibright.com/`
4. Graphviz, grafiikkageneraattori, `http://www.graphviz.org/`

Osa II.
Ongelmat

3. Harjoitustehtävät

3.1. Yleistä harjoitustehtävistä

Harjoitustehtävät ovat tarkoitettu nimenomaan harjoitukseksi. Niihin kuitenkin tarjotaan esimerkkiratkaisut. Nämä ratkaisut syntyivät logiikan toimivuutta testatessa ja ovat juuri niin helppoja. Lisäksi on useampia tapoja ratkaista nämä ongelmat. Tämä laite eroaa muista logiikoista todennäköisesti. Siksi on syytä päästä samalle aaltopituudelle tekijän kanssa, koska ajatuksenjuoksu on kulkenut tikapuuohjelmoinnista kohti oikeaa ohjelmointiympäristöä.

Harjoituksia voi tehdä ilman laitteeseen kytkettyjä Modbus-protokollaa käyttäviä laitteita. Aseta median muuntaminen automaatille tai pois päältä kuten kohdassa 1.4.5. Käytä sisään ja ulostuloina muuttujia ja aseta päivitykset sopivan laiskoiksi.

3.2. Kasvihuoneen kastelu

Kasvihuoneessa on kastelujärjestelmä. Kastelujärjestelmä on rakennettu lämmityslaitteistosta putkistosta ja pumpusta. Putkisto on tehty niin, että pumppu pystyy pumppaamaan multaan vettä, vaikka hullunkiertoventtiili olisi auki. Toisaalta putkisto on niin tilava, että sieltä voidaan pumpata hullunkierrolla vettä läpikin kastelematta. Venttiilin ollessa auki kastelujärjestelmässä oleva vesi suodatetaan ja pumpataan takaisin putkistoon. Logiikka saa sisäänsä ulkolämpötilan tiedon, maan kosteuden. Pumppua ohjataan kertomalla sille virtauspyyntö. Lämmityskin on esimerkissämme mitoitettu sopivasti, joten sitäkään ei tarvitse kuin kytkeä päälle. Ratkaistu kohdassa 2.5 sivulla 31.

3.3. Käsisäätö

Kappaleessa 3.2 esitettyä järjestelmää halutaan testata. Lisää käsisäädöksi sisääntuleva lämpötila.

Ratkaisu: Luodaan reaalilukumuuttuja Ulkolämmön testiarvo ja totuusarvomuuttuja Testikäyttö päällä. Lisäksi tarvitaan yksi jos logiikkaoperaattori Lämpötilan valinta. Ehdoksi tulee Testikäyttö päällä, silloin sisääntuloksi Ulkolämmön testiarvo ja muulloin ulkolämpö. Pakkasvahdin ja halutun kosteuden sisääntulo muutetaan Lämpötilan valinnaksi.

3.4. Kolmen sisääntulon keskiarvo

Olkoon olemassa ei-totuusarvo tyyppiset Sisääntulo 1, 2 ja 3. Olkoon myös Summa 1 ja 2 sekä muuttuja kokonaisluku arvolla 3 ja jakolasku. Summa 1 laskee yhteen Sisääntulo 1 ja 2. Summa 2 sisääntulot ovat Summa 1 ja Sisääntulo 3. Jakolaskulla jaettava on Summa 2 ja jakaja muuttuja.

Summien arvo ei voi ylittää 32-bittisen kokonaisluvun suurinta arvoa[9]. Jos näin käy, tulee laskentavirhe. Laskentavirheen estämiseksi on tarkasteltava kaavaa, jolla keskiarvo lasketaan $(a+b+c)/3 = a/3+b/3+c/3$. Vasen puoli yhtälössä kuvaa alkuperäistä ratkaisua. Jos a, b, ja c olisivat olleet suurin mahdollinen luku, olisi keskiarvoksi tullut $\frac{1}{3}$ suurimmasta mahdollisesta luvusta. Oikean puolen yhtälöllä voidaan havaita, että keskiarvo lasketaan oikein. $\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = \frac{3}{3} = 1$Toteutetaan siis parempi ratkaisu. Alkuperäisen ratkaisun jakolaskun voi poistaa. Luodaan kolme jakolaskua tilalle. Jaettava kulkee näissä kesken Sisääntulo 1:stä Sisääntulo 3:een. Kaikilla jakajana on sama muuttuja, jonka arvo on 3. Summista vaihdetaan sisääntulot vastaaviin jakolaskuihin.

Lisäksi tulee muistaa, että summassa ensimmäinen yhteenlaskettava määrää ulostulon tyypin.

4. Vikatilanteet

Tässä laitteessa ei ajeta Windows käyttöjärjestelmää. Käyttämällä laitteen pois päältä, menetät vain asetuksesi.

Aina järjestelmä ei sovi käyttöympäristöönsä helpolla. Tässä kappaleessa käsitellään yleisimpiä mahdollisia vikoja. Mikäli laite on toiminut hyvin ja yhtäkkiä lakkaa toimimasta, ota rauhallisesti.

4.1. En saa yhteyttä laitteeseen

Ovathan johdot kunnolla kiinni ja oikein kytketty? Tiedätkö laitteen IP-osoitteen? Onko se oikea? Saako laite varmasti IP osoitteen? Tarkista DHCP-palvelun logeista. Laite ei tue IPv6:a. Estääkö jokin palomuuri tai muu suojaus yhteyden. Tutustu kohtaan 1.3

4.2. Laite ei saa yhteyttä muihin Modbus-laitteisiin

Onhan yhteys oikea? RS-485 väylälle puhutaan RS-485:ttä? Laitteessa on RS-232 liityntä. Sarjaväylään tarvitaan muunnin jos RS-232 liitynnästä otetaan yhteys RS-485 väylään.

Onko sarjaportin nopeus ja pariteetti oikea? Mikäli laite on muuten asetettu oikeille sarjaportin asetuksille ja laite ei kykene lukemaan muita laitteita, kokeile kahdella stop-bitillä. Lähetettäessä sarjadataa, tavujen väliin tulee kahden bitin tauko yhden sijasta. Tämä voi auttaa tilanteessa, jossa kuuntelevan laitteen UART on epätarkka. Vastaanottoon tuolla asetuksella ei dokumentaation mukaan tulisi olla vaikutusta.

Onko kaapeli rikki? Onko väylässä liian monta laitetta? Onko johto liian pitkä?

4.3. Laite on jumissa

Olisiko käynyt niin, että laite yrittää lukea jotain olematonta laitetta? Ja näitä laitteita on niin paljon, että aikakatkaisujen summa on suurempi kuin päivitysväli. Ratkaisu: Hanki ne laitteet näkyville tai käynnistä uudelleen.

[9] 2 147 483 647 ja etumerkittömänä 4 294 967 295. Reipas kaksi miljardia ja neljä miljardia

4.4. Median muuntaminen ei toimi

Median muunnosta ei tieto kulje lävitse. Jotkut Modbus-protokollan toteutukset ovat huonompia, kuin toiset. Toisinaan käy niin, että laite sulkee yhteyden tietyn ajan kuluttua käyttämättömänä. Jos toisen pään laite ei tästä ymmärrä yrittää uutta yhteyttä, on tuloksena juuri tuollainen tilanne.

Toinen ongelma voi olla siinä miten kyselyt tehdään. Modbus-TCP-protokollassa käytetään laitenumeroa 255 ilmaisemaan, että käytämme nyt tätä verkkolaitetta. Sen ei välttämättä tarvitse olla 255 vaan voi olla myös jotain muuta. Tämän laitteen kanssa sen tarvitsee olla jotain muuta. Arvot 1-247 ovat hyviä. Lue uudelleen kohta 1.4.5 sivulla 8.

5. Kummalliset tilanteet

5.1. Lisäsin Modbus-laitteen ja...

Laitteen lisäyksessä tapahtuu seuraavaa. Ensin syötetään tiedot web-lomakkeelle ja tiedot lähetetään laitteelle. Mikäli laite toteaa tiedot oikeiksi, pysähtyy logiikan suoritus ja median muunto hetkeksi. Tämän hetken aikana tiedot uudesta laitteesta lisätään laitteen muistiin. Web-käyttöliittymän tulisi vastata tässä vaiheessa. Seuraavaksi laitteessa tapahtuu lukutapahtumien uudelleen luonti. Eli lisätty Modbus-laite menee lukulistoihin. Tämän jälkeen kaikki laitteet luetaan uudelleen. Sen jälkeen laitteen toiminta jatkuu normaalisti.

5.2. Modbus-laitteiden lukeminen on hidasta.

Mikäli laitetta ei ole luotu, sitä ei lueta. Jos olet luonut esimerkiksi rekisterit 2, 4, 6... 250, mutta et parittomia rekisterejä väliin, niin niitä ei tosiaankaan lueta. Luku komentojen lähetys nopeudella 9600 tarkoittaa silloin 125 komentoa, jossa jokaisessa on 8 tavua ja yhteensä 1000 tavua tulee siirtää. Aikaa kuluu reilu sekunti. Lisäksi vastaanottajan tarvitsee vastata ja se on toinen sekunti vähintään.

Tilanteessa, jossa on rekisterit 2... 250 käytössä, lähetykseen menee kaksi kertaa 8 tavua ja vastaukseen kaksi 255 tavua. Tämä tarkoittaa neljäsosa ajassa siirtyi tuplaten dataa.

Ensin kuvatun kaltaisen tilanteen saa aikaiseksi mm. laittamalla joka toiselle eri päivitysvälin.

5.3. Modbus-laitteiden kirjoitus on hidasta.

Sitä se on. Vahinkojen estämiseksi, laite ei kirjoita muualle kuin haluttuihin paikkoihin. Eli niihin paikkoihin joissa on tapahtunut muutos, jonka sisäinen logiikka on tehnyt. Lisäksi kirjoitus tapahtuu lukuryhmittäin prosessointi ajan ja muistin kulutuksen rajoittamiseksi. Tämä ominaisuus tarjoaa mahdollisuuden käyttää useampaa Modbus-laitetta ja useampaa logiikkayksikköä.

Lukuryhmissä on perättäiset kelat, sisääntulot ja rekisterit kukin eri ryhmässä. Lisäksi ryhmän voi katkaista eroava päivitysväli.

5.4. Kohdassa 1.1 luki, ettei lueta ja kirjoiteta kuin omia. Eikö voisi...

Eikö voisi lukea lukea aina koko alueen ja kirjoittaa sitten sinne takaisin? Ei. Helpoin tilanne: Kahdessa rekisterissä on aikaleima ja se antaa kuluneiden sekuntien määrän 32-bittisenä lukuna hetkestä 1.1.1970 0:00:00 eli perinteinen UNIX aika. Lukemalla nuo rekisterit ensin sisään ja sitten kirjoittamalla, kello hidastuisi.

Toinen syy. Jossain laitteessa on anturi, joka ottaa näytteen joka kerta kysyttäessä ja näytteen otto kestää 3 sekuntia. Asettamalla sarjaportin aikakatkaisun tätä suuremmaksi järjestelmä toimii. Jos se kuitenkin luettaisiin aina, koko järjestelmä hidastuisi ja aina tuolta laitteelta jotain kysyttäessä, kuluisi tuo kolme sekuntia turhaan. Ja tietoa voidaan tarvita vaikka kerran päivässä.