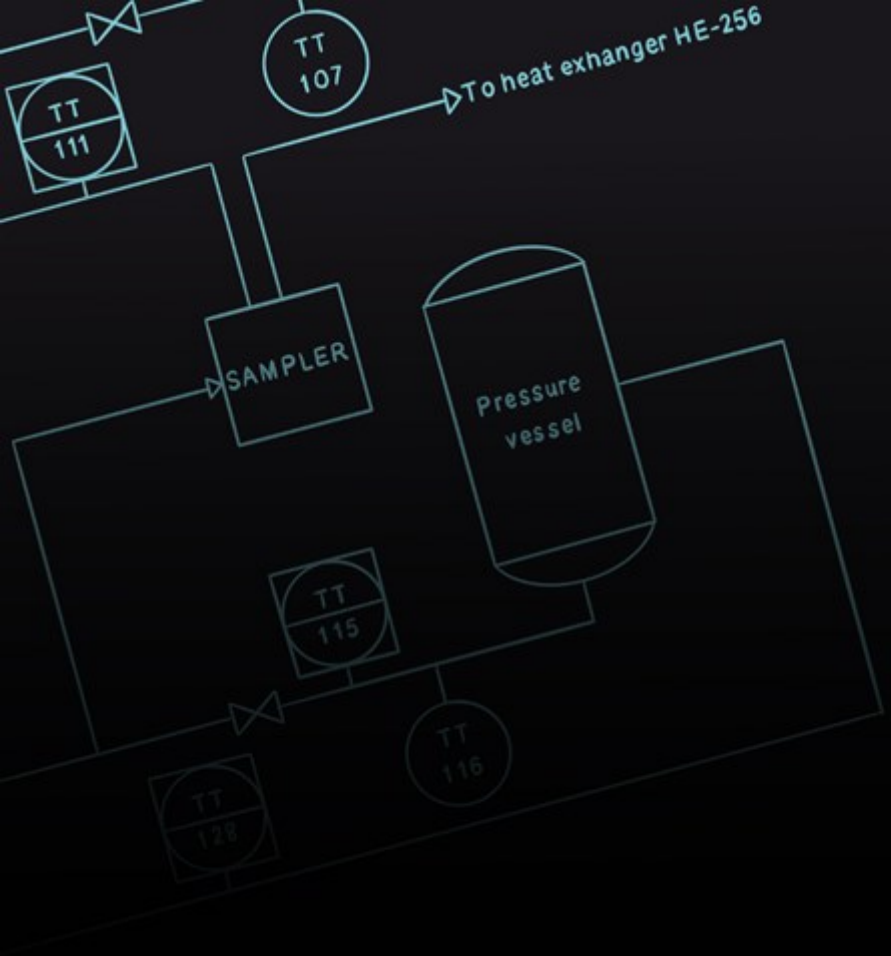
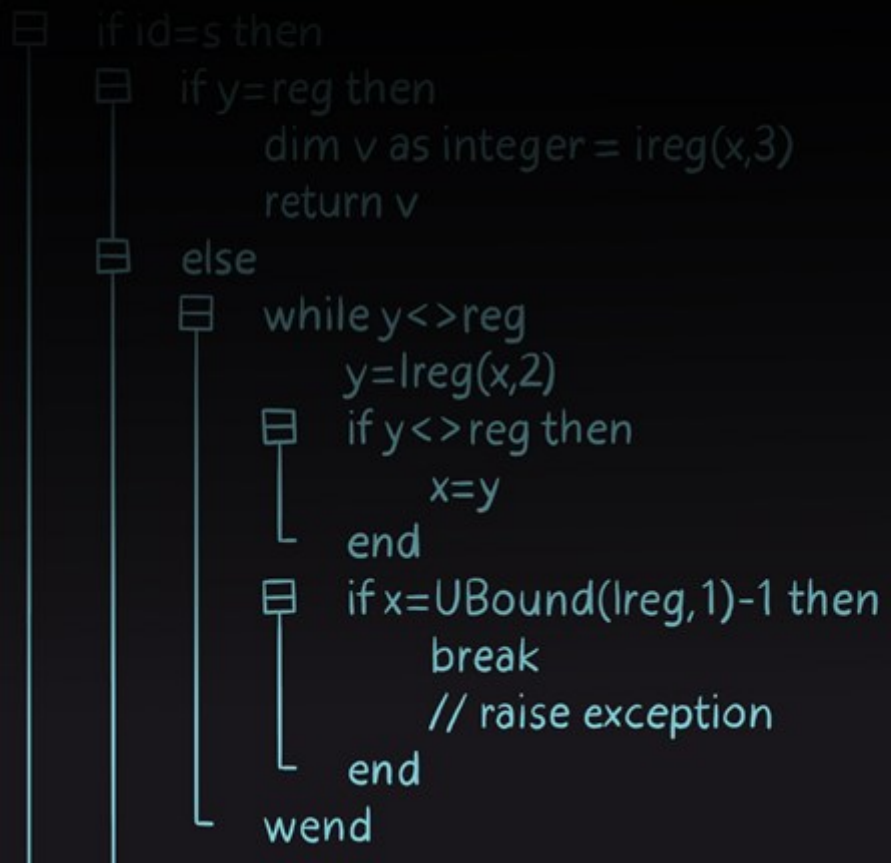


Jari Väisänen

Modbus TCP/IP -palvelimen toteutus



Insinööri (AMK)

Tieto- ja viestintätekniikan koulutus

Kevät 2020

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Väisänen Jari

Työn nimi: Modbus TCP/IP -palvelimen toteutus

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka, monimuoto

Asiasanat: teollisuuden kenttäväylät, tietoliikenne, ohjelmistokehitys, Xojo, Planray Oy, sähkösaatto, palvelin, automaatio, prosessiautomaatio

Tämän opinnäytetyön toimeksiantaja oli Planray Oy, joka valmistaa sähkösaattojen ohjauslaitteita. Opinnäytetyössä kehitettiin teollisuuden kenttäväyliin liittyvä sovellutus Modbus TCP/IP-palvelimesta. Tavoitteena oli saada Planrayn MidiTrace-tuotteelle soveltuva ohjelmisto, joka toimisi pohjana Modbus TCP-palvelimen kehittämisessä paneeli-PC-laitteelle. Paneeli-PC:n kehitys oli jo ennestään ollut käynnissä, mutta liitettävyys teollisuuden automaatiojärjestelmään oli vielä puutteellinen, eikä sitä ollut aloitettu rakentamaan. Työssä tutustuttiin erilaisiin kenttäväyliin sekä niiden vaatimuksiin. Tutustumisen jälkeen valittiin tähän tiettyyn käyttötapaukseen parhaiten soveltuva kenttäväyläratkaisu.

Ohjelmistokehityksessä noudatettiin ketterän kehityksen tapaa, jossa ohjelmakoodia kehitettiin ja testattiin jatkuvasti korkean laadun takaamiseksi. Kehitystyössä käytettiin versiohallintaa ja tutustuttiin git-versionhallinnan käyttöön ohjelmointiympäristössä. Kehitystyö tapahtui Xojo-ohjelmointiympäristössä sekä omalla koneella simuloiden. Näin ohjelmaa testattiin välittömästi työn edetessä ja ongelmien sekä bugien korjaus tapahtui nopealla syklillä.

Lopputuloksena syntyi valmis toimiva palvelinohjelma, jota voidaan kokonaisuudessaan ja jonka osia voidaan hyödyntää myöhemmin Planray Oy:n tuotteissa sekä niiden jatkokehityksessä. Tätä työtä voidaan käyttää katsauksena Modbus-kenttäväyläteknologiaan sekä pikaoppaana Modbus TCP-järjestelmän kehitykseen.

Abstract

Author: Väisänen Jari

Title of the Publication: Modbus TCP/IP -server implementation

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: industrial fieldbus, communication technology, software development, Xojo, Planray Oy, electric trace heating, server, automation, process-automation

This Bachelor's thesis covers the development and testing of industrial field bus implementation, that can be used to connect equipment to a larger process automation control and monitoring system. Different kind of fieldbuses were examined to find the one that fits the use-case of this system best. The fieldbus technology was selected by its robustness and by its large userbase.

Software code was developed and tested continually to guarantee the high quality and fast response to problems. The development was done with a version-control system, to ensure a quick way for keeping track of changes and bugs in code. Code was developed on Xojo IDE, that allowed the built code to be run on almost any machine that has Linux or Windows based operating system. This way, the system implantation was made to be easy for later on application on the end-product.

As a result, the Modbus server software was created and it can be implemented as a whole or partly to be part of Planray's products and their development. This thesis can be used as a quick start guide for implementing Modbus TCP industrial fieldbus.

Alkusanat

Tämä insinöörityö on tehty Planray Oy:lle vuoden 2019 aikana.

Kiitän erityisesti Planray Oy:n tuotepäällikkö Vesa Kovalaista sekä testaaja Seija Kettusta neuvoista sekä tuesta tämän työn tekemisen aikana.

Kiitos myös Caverion Suomi Oy - Teollisuuden Ratkaisut kehityspäällikkö Markku Huhtalalle haastattelusta sähkösaattoihin liittyen.

Lopuksi, haluan myös kiittää Kajaanin Ammattikorkeakoulun työn ohjaajana ja valvojana toiminutta tuntiopettaja Eero Huuskoa.

Kajaanissa 27.1.2020 Jari Väisänen

Sisällys

1 Johdanto 1

2 Planray Oy 2

3 OSI-malli ja TCP/IP-protokollapino 3

3.1 OSI-malli 3

3.2 IP-protokolla 4

3.3 TCP 5

3.4 Modbus TCP/IP ADU 6

4 Teollisuuden kenttäväylät 7

4.1 CAN 8

4.2 ModBus 8

4.3 PROFIBUS 8

4.4 Profinet 9

4.5 EtherCAT 9

4.6 OPC 9

5 Xojo 10

6 GitLab 11

7 MidiTrace 12

8 Sähkösaatto 13

9 Modbus TCP-palvelimen kehitys 14

9.1 Paneeli-PC 14

9.2 Järjestelmän rakenne 16

9.2.1 Modbus RTU:n tärkeitä termejä 17

9.2.2 Modbus TCP:n tärkeitä termejä ja eroja 18

9.3 Järjestelmän vikasietoisuus 18

9.4 Suunnittelu 19

9.5 Pääohjelman luonti ja käyttöliittymän suunnittelu 21

9.6 Modbus-viestin käsittely ja vastauksen luominen 22

9.7 Modbus-funktioiden toteutus 26

10 Yhteenveto 30

Termistö

ADU	Application Data Unit
Aikakatkaisu	Maksimiaika, jonka ohjelma tai prosessi odottaa vastausta tai syötettä.
Broadcast	Yleislähetys
Heijastus	Ilmiö signaalin ”monistuessa” johtimessa kaapelin vapaasta päästä
IP	Internet Protocol
IP-luokitus	Standardisoitu tapa ilmoittaa laitteen pöly- ja vesitiiveyttä
MBAP	Modbus Application Protocol header
Moduuli	Erillinen ohjelmakooditiedosto Basic-kielessä
OSI malli	Open Systems Interconnection Reference Model
Palvelin	Verkossa toimiva laite, joka käsittelee pyyntöjä ja vastaa niihin
PDU	Protocol Data Unit
Säie	”Kevyt” ohjelmaprosessi, jolla ei ole omia resursseja, vaan käyttää isäntänsä resursseja.

1 Johdanto

Digitaalinen tiedonsiirto oli aikanaan suuri mullistus teollisuuden puolella. Valtavat valvomotilat analogisine mittareineen alkoivat siirtyä historiaan, ja kaikki tieto voitiin kerätä yhteen paikkaan käyttäjän saataville näytölle, yksinkertaisten laitteiden avulla.

Teollisuuden kenttäväylät kehittyvät jatkuvasti, ja osa jää ns. de-facto-standardeiksi. Tässä työssä käsitelty Modbus on yksi vakiintuneimmista ja on erittäin paljon maailmalla käytössä avoimuutensa sekä yksinkertaisen toiminnallisuutensa ansiosta.

Tämä insinöörityö käsittelee tarkemmin Modbus-palvelimen toteuttamista paneeli-PC-laitteeseen, joka tulee olemaan osana suurempaa MidiTrace-ohjausjärjestelmää. Idea tälle työlle löytyi työpaikaltani, koska työn aihe oli kiinnostava ja halusin syventää omaa tuntemustani teollisuuden kenttäväylistä, jotta pystyisin työssäni paremmin ymmärtämään tietoliikenteen merkitystä isoissa automaatiojärjestelmien kokonaisuuksissa. Työn laajuuden todettiin olevan sopiva opinnäytetyön vaatimuksiin ja tästä syystä se valikoitui omaksi opinnäytetyökseni.

2 Planray Oy

Planray Oy on vuonna 1992 perustettu perheyritys, joka kehittää ja myy omia tuotteitaan Kajaanissa. Planrayn juuret juontavat niin teollisuuden sähköasennukseen kuin elektroniikan valmistukseen. Nykyisin yritys keskittyy teollisuuden sähkösaattojen ohjaus- ja valvontajärjestelmien kehittämiseen sekä myyntiin. Planray Oy:n tilat Kajaanissa näkyvät kuvassa 1.

Kuva 1. Planray Oy:n toimitila Kajaanissa

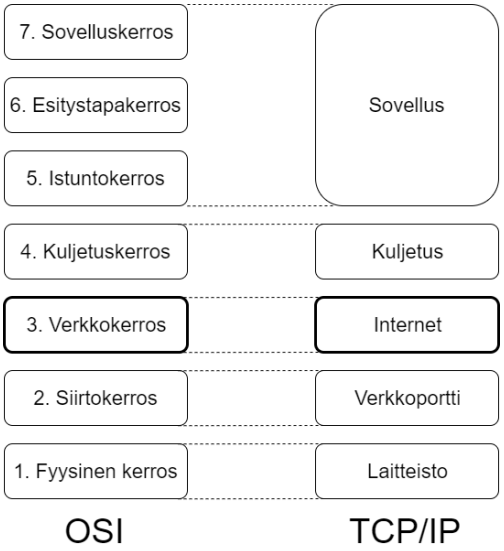
Planrayn tuotepaletti sisältää lähes kaiken sähkösaaton ohjaukseen sekä hallintaan tarvittavan, ja sen tuotteita on myyty jo yli 20:een eri maahan [1].

3 OSI-malli ja TCP/IP-protokollapino

Jotta palvelimen kehitystyö lähtee sujuvasti käyntiin, on tärkeää ymmärtää tietoliikennetekniikan perusteita. Tässä osiossa kerrotaan hieman tietoliikenteen perusasioita, jotka ovat tärkeitä Modbus TCP -palvelimen kehittämisen kannalta. Tuntemus IPv4- sekä TCP/IP-paketin rakenteesta auttaa paremmin ymmärtämään ohjelmiston vaatimaa rakennetta sekä virheen/vianhaku helpottuu tuntemalla myös syvällisemmin verkon digitaalista rakennetta.

3.1 OSI-malli

OSI-malli on määritelty IEC-standardissa IEC 7498-1, ja sen tehtävänä on toimia työkaluna tiedonsiirtoverkon suunnittelussa sekä verkon toiminnan määrittelyssä. Mallin esittämä rakenne on nähtävissä kuvassa 2. Mallin rakenne toimii useiden tiedonsiirtoverkkojen pohjana, mutta OSI-malliin perustuva protokollapino ei koskaan saavuttanut laajaa käyttöä. Mallissa ylemmät kerrokset hyödyntävät alemman kerroksen tarjoamia palveluita, ja alemmat kerrokset tukevat ylempiä. Tietoliikenneverkon rakenteen ja esimerkiksi protokollan sijainti OSI-mallissa helpottaa kokonaisuuden hahmottamista, ja siksi sitä käytetäänkin laajalti sekä opetus- että informatiivisessa tarkoituksessa. [2, s. 82–89.]



Kuva 2. OSI-mallin esittämä kerrosrakenne sekä TCP/IP:n esittämä verkkorakenne [2, mukaillen s. 129, Fig. 8–2].

TCP/IP-protokollapino on OSI-mallin neljännen ja kolmannen kerroksen tapa siirtää tietoa kahden laitteen välillä. TCP/IP-protokollapinon perustana on vahvasti verkkokerroksen IP-protokolla, kuin myös TCP-välitysprotokolla. [2, s. 119–138.]

3.2 IP-protokolla

IP-protokollan tarkoituksena on mahdollistaa tietoliikenne kolmannen kerroksen laitteiden välillä. Käytössä on useita erilaisia protokollia, mutta niistä yleisin ja eniten käytetty on IP eli Internet Protocol. [2, s. 238–240.] IP:stä on tällä hetkellä käytössä kaksi rinnakkaista versiota, IPv4 sekä IPv6. Tässä työssä tarkastellaan tarkemmin vain IPv4:ää, joka on yleisempi.

Alla oleva kuva 3 esittää yleistä IPv4-paketin rakennetta. Rakenteessa on nähtävissä paketin sisältämä upotettu otsake ja data, jonka avulla paketti löytää perille kohdekoneeseen.

IPv4 paketti

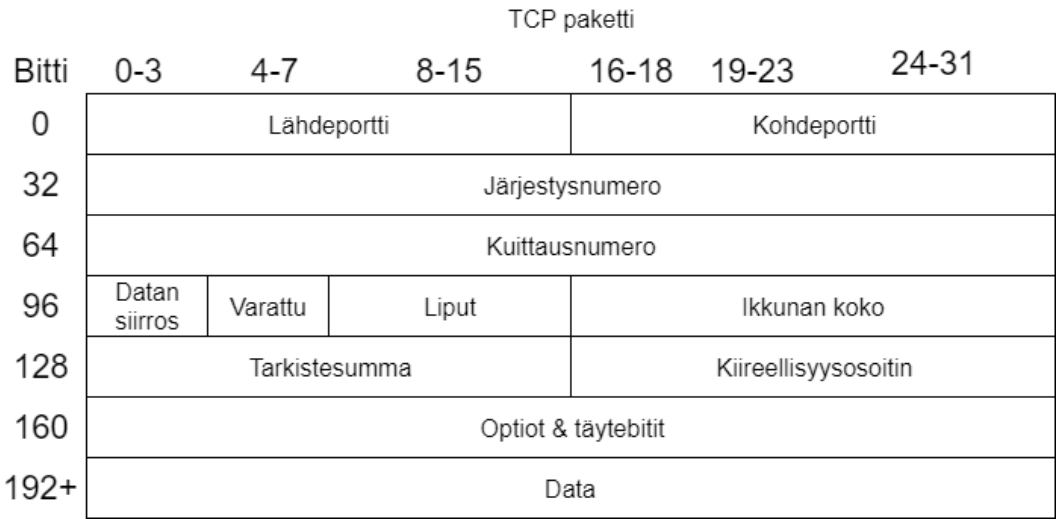
Bitti	0-3	4-7	8-15	16-18	19-23	24-31
0	Versio	Otsakkeen pituus	Palvelun tyyppi	Kokonaispituus		
32	Pilkotun paketin yksilöllinen tunniste			Liput	Pilkotun paketin paikka	
64	Elinaika		Protokolla	Otsakkeen tarkistusumma		
96	Lähdeosoite					
128	Kohdeosoite					
160	Optiot					
192+	Data					

Kuva 3. IPv4-paketin rakenne [3 mukaillen s. 128, fig. 7-9].

Modbus TCP/IP-liikenteessä IPv4-paketin pilkkominen estetään asettamalla lipun arvoksi aina 0x0400, joka kertoo reitittimille sekä muille verkon laitteille, että pakettia ei saa jakaa pienempiin osiin. Paketin protokolla on kuitenkin aina 0x06, eli TCP. [3, s. 121–134.]

3.3 TCP

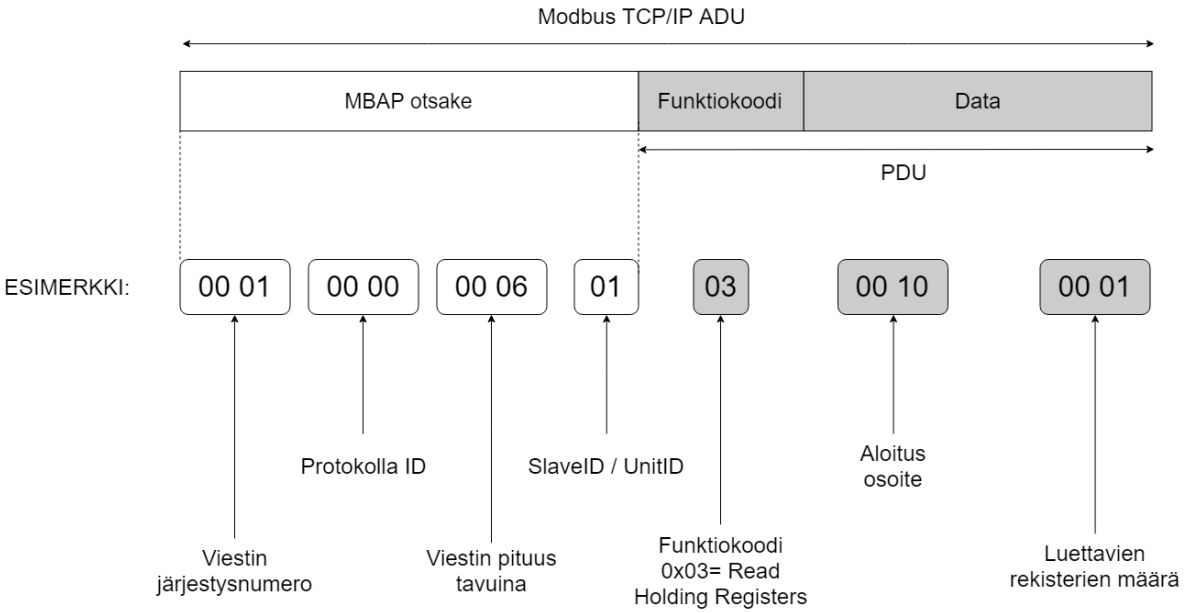
Modbus TCP/IP-viesti upotetaan aina IPv4-paketin DATA-osioon, missä siinä on mukana useita tietoja paketin sisältämään dataan liittyen ja tarkastussummia. Modbus TCP/IP-paketin kohdeportti on useimmiten 502, mutta portti voi periaatteessa olla mitä tahansa muutakin. TCP:n pakettirakennetta kuvaa kuva 4.



Kuva 4. TCP-paketin rakenne [3, mukaillen s. 152 fig. 8-2].

3.4 Modbus TCP/IP ADU

Modbus TCP ADU on TCP-pakettia seuraava dataosa, joka sisältää Modbus-liikenteen kannalta oleelliset tiedot. TCP-yhteyden ollessa kyseessä mukana tulee MBAP-otsake sekä ADU. Kuvassa 5 on esitetty esimerkillä, mitä MBAP-otsake ja PDU voivat sisältää. [4, s. 4–5.]



Kuva 5. Modbus TCP/IP ADU-rakenne [4, mukaillen s. 4 fig. 3].

4 Teollisuuden kenttäväylät

Teollisuudessa käytetään useita digitaalisia väyliä vähentämään kaapelointia sekä tuomaan tietoa tarjolle helposti käsiteltävässä muodossa. Vanhoissa ratkaisuissa, kuten kuvassa 6, oli usein valtavia ohjaamoita, jotka olivat täynnä mittareita sekä ohjauspulpetteja satoine kaapeleineen.

Kuva 6. Vanhanaikainen ohjaamo/valvomotila [5].

Nykyaikaisessa valvomossa käyttäjän edessä on vain muutamia ruutuja ja kaiken hallinnan pystyy toteuttamaan muutamalla hiirenklikkauksella. Tämän ovat mahdollistaneet erilaiset kenttäväylät, joiden eri tyyppejä on lueteltu seuraavilla sivuilla.

4.1 CAN

CAN eli ”Controller Area Network” on Robert Bosch GmbH:n 1980-luvulla kehittämä protokolla, joka kehitettiin helpottamaan ajoneuvoteollisuuden kasvavaa painoa, joka johtui elektroniikan lisääntymisestä ajoneuvoissa. CAN-verkossa jokaisella laitteella on oma viestitunniste, johon perustuen laite päättää, onko lähetetty viesti osoitettu sille. CAN-väylän etuna on monesti yksinkertaisuus sekä nopeus, koska väylässä voidaan siirtää jopa megatavun nopeudella dataa. [6.]

4.2 ModBus

Modbus on Modiconin (nykyisin osa Schneider Electriciä) vuonna 1979 kehittämä protokolla, joka suunniteltiin erityisesti ohjelmoitavia logiikoita silmällä pitäen. Modbus-protokolla ei ole riippuvainen siirtomediasta, vaan sitä voidaan käyttää lähes minkä tahansa langallisen ja langattoman tiedonsiirtoväylän kautta. Yleisimpiä Modbus-tiedonsiirtoväyliä ovat RS232, RS422, RS485-sarjaliikenne sekä TCP/IP. [7.]

Modbus kehitettiin olemaan yksinkertainen, jotta sen käyttäminen olisi mahdollisimman helppoa rajoitetuissa laiteresursseissa, kuten mikro-ohjaimissa. Sitä käytetään nykyään etenkin ohjelmoitavissa logiikoissa, SCADA-automaatiosovellutuksissa, venttiilien ja antureiden kanssa sekä useissa muissa sulautetuissa järjestelmissä. [8.]

4.3 PROFIBUS

Profibus eli ”PROcessFIeldBUS” on saksalaisten teollisuusalan yritysten yhdessä kehittämä token-ring-tyyppinen protokolla, joka suunniteltiin hajautettujen järjestelmien tarpeisiin. Teollisuuden vaatimukset kuitenkin muuttivat väylää, ja nykyään PROFIBUS on keskitettyyn hallintaan perustuva väylätekniikka. [9.]

4.4 Profinet

Profinet eli ”PROcessFIeldNETwork” on PROFIBUSista muokattu TCP/IP-verkkoihin tarkoitettu protokolla, jota käytetään yhä laajemmin teollisuudessa niin mittaus- kuin ohjaustekniikassa. PROFINETin etuna on nopeus, koska toisin kuin PROFIBUSissa, pyyntöjä voidaan tehdä rinnakkain TCP/IP-teknologian ansiosta. [10.]

4.5 EtherCAT

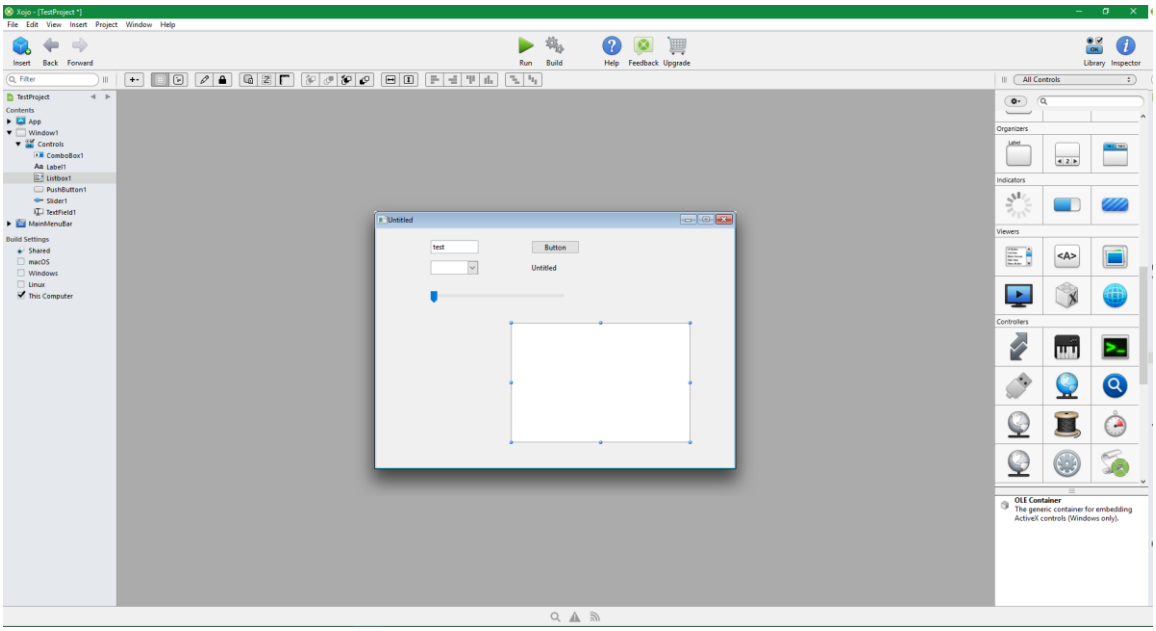
EtherCAT-termi muodostuu englanninkielisistä sanoista “Ethernet for Control Automation Technology”. EtherCAT on suunniteltu suurta nopeutta vaativiin tarkoituksiin, ja sitä käytetään paljon liikeohjaukseen sekä nopeasti vastaaviin sovelluksiin. [11.]

4.6 OPC

OPC on eräs ehkä raskaimmista teollisuuden kenttäväylän toteutuksista, jossa jokaisesta laitteesta, joka liikennöi OPC-verkossa, siirretään paljon tietoa laitteelta toiselle. OPC:tä käytetään usein kenttälaitteiden sekä valvomon SCADA-järjestelmän välillä, koska sitä pidetään turvallisena järjestelmän luonteesta johtuen. OPC:ssa järjestelmässä ei ole merkitystä, kuinka monta isäntää tai orjaa verkossa on, koska jokainen laite voi kommunikoida milloin vain toisilleen. Tämän ansiosta periaatteessa jokainen OPC-laite on sekä palvelin että asiakas verkossa. [12.]

5 Xojo

Xojo on nopeaan ohjelmistokehitykseen kehitetty työkalu, joka pohjautuu vahvasti Microsoftin Visual Basiciin. Aiemmin Xojo tunnettiin nimellä RealBasic, mutta yritys brändäsi itsensä uudelleen vuonna 2013 [13]. Xojo IDE on nähtävillä alla olevassa kuvassa 7.



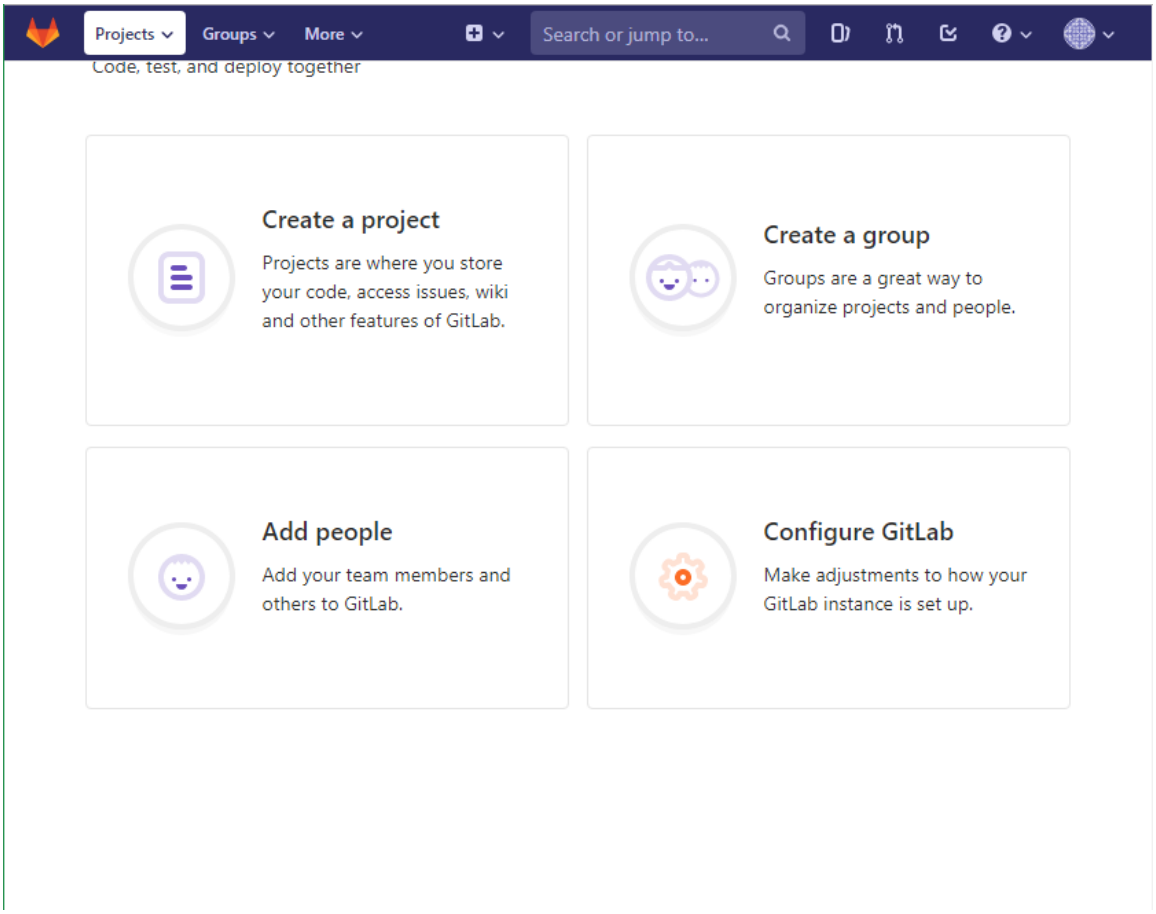
Kuva 7. Xojo IDE

Xojolla ohjelmakehitystä pystytään tekemään Windows-, Mac- sekä Linux-tietokoneilla, ja tästä syystä se soveltuu hyvin järjestelmäriippumattomaan ohjelmistokehitykseen. Ohjelmistolla voidaan kääntää samalla kertaa sekä Windows-, Linux- että MacOS-versiot lopullisesta ohjelmasta.

Xojoa käytetään paljon vanhojen Visual Basic -ohjelmien korvaamiseen sekä niiden päivittämiseen nykytasolle. Xojon ohjelmointikieli vastaa pitkälti Visual Basicia ja on siitä syystä aiemmin sillä aiemmin kehittäneelle nopeasti opittava. Tunnetuimpia Xojon käyttäjiä ovat mm. seuraavat yritykset: Cisco, Intel, AT&T, Xerox, Google ja NASA. [14.]

6 GitLab

Gitlab on avoimeen Git-versiohallintaan perustuva järjestelmä, jossa voidaan seurata lähdekoodin muutoksia, pitää ohjelmistoprojektikohtaisia wiki-tietosivuja, raportoida ohjelmavirheitä, kehitystoiveita sekä hallita tehtäviä. GitLabia voidaan hyödyntää myös lähdekoodin jatkuvaan testaamiseen, jolloin järjestelmä kääntää, ajaa, testaa ja raportoi jatkuvasti koodin toimintaa. GitLab-aloitusikkuna on esitetty kuvassa 8. [15.]



Kuva 8. GitLab -aloitusikkuna

7 MidiTrace

MidiTrace on Planray Oy:n kehittämä saattolämmityksen yksikkösäädin, joka on suunniteltu asennettavaksi sähkökeskukseen. MidiTrace-laitteissa ei ole BlueTrace-laitteen tavoin omaa näyttöä, vaan kaikki ohjaus ja hallinta tehdään ulkoisien väylien kautta. MidiTrace-yksikkösäädin näkyy kuvassa 9. [16.]

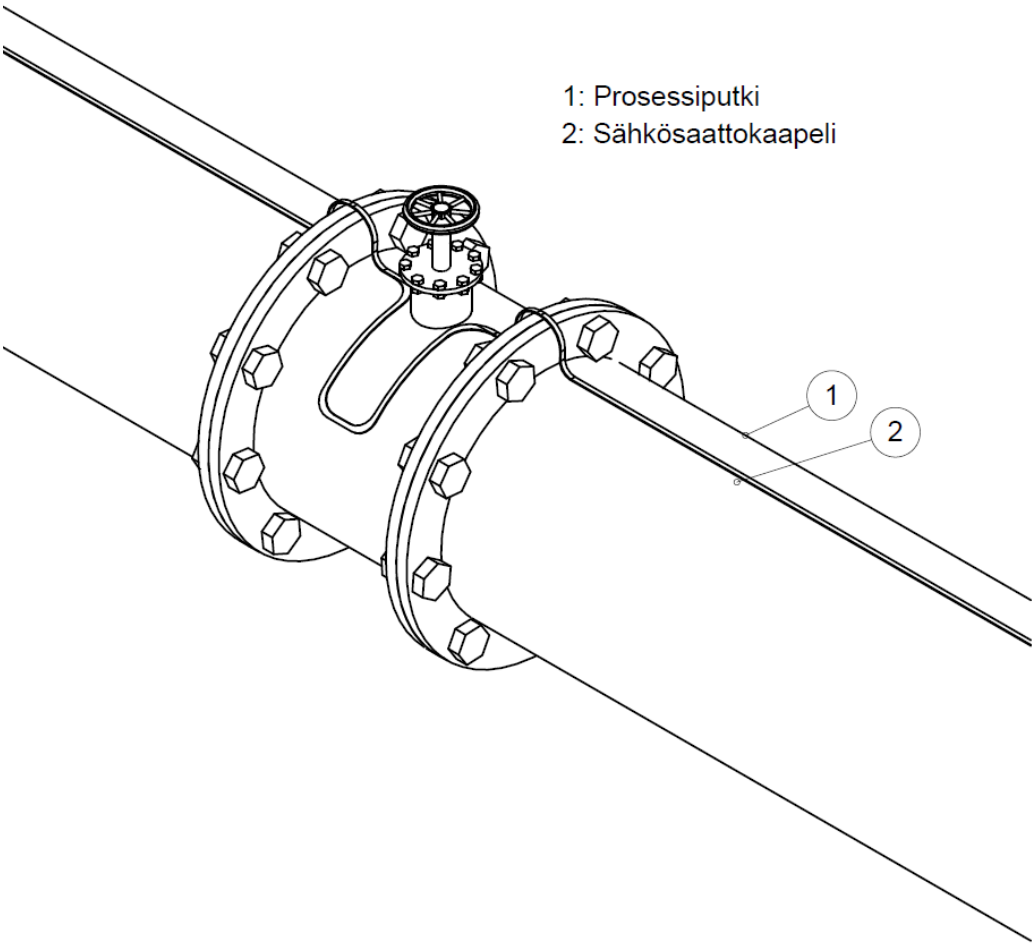
Kuva 9. MidiTrace -yksikkösäädin. [17.]

MidiTrace-laitteen virrankesto on 32 A, ja laitteessa on kaksi optisesti erotettua tuloa (24 VDC/230 VAC) sekä kaksi relelähtöä (277 VAC, 10 A max). Laitteessa on lämmityksen ohjausta varten tyypistä riippuen kaksi tai kolme tyristoria, joilla saavutetaan tarkka lämmityksen ohjaus halutussa kohteessa. Laitteessa on vakiona kaksi lämpötilatuloa, mutta lisäkortin avulla lämpötilatulojen määrä voidaan nostaa jopa neljään. Lisäksi lämpötila voidaan tuoda laitteelle myös Modbus-väylän kautta jakamalla, jolloin säästetään anturointikustannuksissa. [16.]

8 Sähkösaatto

Sähkösaatoksi kutsutaan sähkölämmitystä, jossa sulatetaan, ylläpidetään tai kuumennetaan aineen lämpötilaa. Sähkösaattolämmityksiä käytetään paljon prosessiteollisuudessa huolehtimaan prosessin nopeista muutoksista, esimerkiksi siirryttäessä tuotannosta putkistojen huuhteluun. Sen etuna on lämpötilan tarkkuus, huollettavuus ja energiatehokkuus. [17.]

Saattolämmityksiä asennetaan myös liukkaudentorjuntaan laattoihin sekä autolastauspaikoille, ja lämmitettäviä kohteita voi olla letkuista savupiippuihin. Suomen olosuhteissa saattolämmitykset ovat välttämättömiä monissa teollisuuden ratkaisuissa ja niitä hyödynnetään esimerkiksi palovesiputkien jäätymisen estämisessä. [17.] Esimerkki saattolämmityksestä esitetään kuvassa 10.



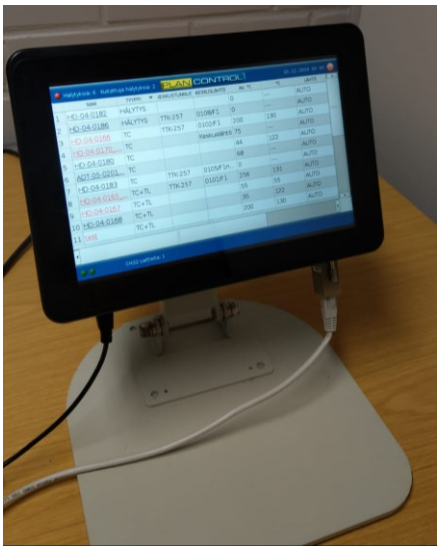
Kuva 10. Esimerkkipiirros sähkösaatosta

9 Modbus TCP-palvelimen kehitys

Modbus TCP-palvelimen kehitystä varten on tärkeää tuntea ohjattavan järjestelmän rakenne sekä käytettävä tietoliikenne. Näiden tietojen avulla järjestelmän eri osat nitoutuvat yhteen ja saavutettavissa on luotettava, suorituskykyinen ohjausjärjestelmä.

9.1 Paneeli-PC

Kuvassa 11 nähdään Paneeli-PC, joka on 10” kosketusnäytöllä varustettu teollisuuskäyttöön suunniteltu PC-tietokone. Ulkoiset liitännät laitteesta ovat RS485, RS232, Ethernet sekä 2xUSB. Paneeli-PCllä on sähkökeskuksen oveen asennettuna ulkopuolelta IP65-luokitus, joka tarkoittaa, että laite on pölytiivis sekä suojattu suoralta vesisuihkulta. [18].



Kuva 11. Paneeli-PC

Ohjelmisto on kehitetty *Windows 10 IoT Enterprise* -version päälle, joka on *Windows 7 Embedded* -versiota vastaava käyttöjärjestelmä. Kuvassa 12 nähdään esimerkki Windows 10 IoT:ssä ajettavasta ohjelmasta. Laitteen päänäkymänä toimii taulukkonäkymä, jossa on mahdollista nopeasti todeta lämmitysten toiminta sekä senhetkiset lämpötilat.

Kuva 12. MidiTrace-ohjelmiston käynnistyminen Win10 IoT-järjestelmässä

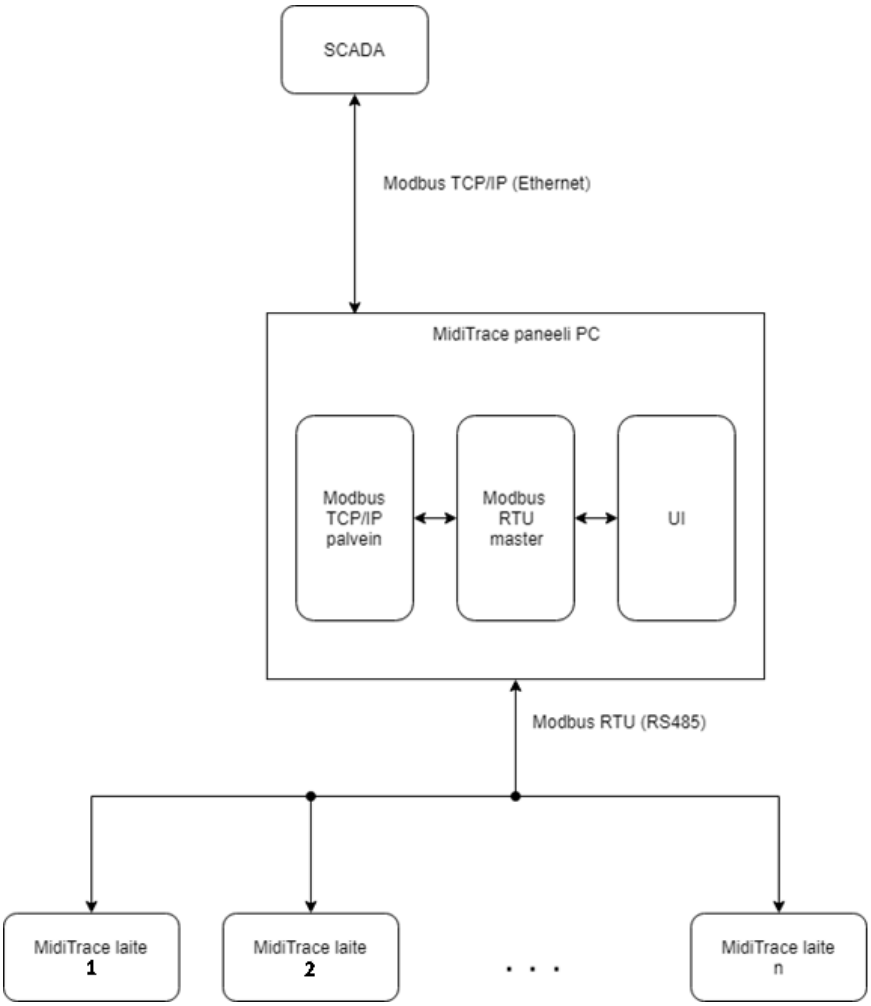
Ohjelmistoa voidaan käyttää keskuksen toiminnan valvontaan sekä järjestelmän päivittäiseen käyttöön (ks. kuva 13). Useimmiten kuitenkin päätettä käyttävät sähköasentajat ja muut sähköalan ammattihenkilöt johtuen laitteiston sijoituspaikasta.

Kuva 13. MidiTrace-laitteen hallintanäkymiä

Ohjaus ja valvonta toteutetaan useimmiten automaatiojärjestelmään, jota varten tässä työssä toteutettua Modbus TCP/IP-palvelinta tarvitaan. On myös mahdollista, että järjestelmää käytetään omana osanaan, ilman ylempää automaatiojärjestelmää.

9.2 Järjestelmän rakenne

Työssä kehitettävä Modbus-palvelin tulee olemaan osana järjestelmää, jossa laitteet kommunikoivat toisilleen kenttäväylien avulla. Järjestelmän rakennetta kuvaa alla oleva kuva 14.



Kuva 14. MidiTrace-järjestelmän rakenne

Rakenteessa tärkeää on havaita, että käytössä on kaksi erillistä Modbus-väylää:

- Modbus RTU – laiteväylään päin, orjana toimivien MidiTrace-laitteiden ohjauksessa. RTU on tähän tarkoitukseen valittu, koska RS485-väylä on vikasietoinen ja kustannustehokas toteuttaa suurissakin lämmitysten ohjauskeskuksissa.

- Modbus TCP/IP – automaatiota ja SCADA-tiedonkeruu- sekä ohjausjärjestelmiä varten, MidiTrace-laitteiden portti ylempään automaatio- tai ohjausjärjestelmään.

Näiden kahden väylän kommunikaatioprotokollana Modbus on yhteistä, mutta tärkeää on huomata, että Modbus RTU:ssa väylä on sarjaliikennepohjainen. Tästä syystä yhdessä Modbus RTU -väylässä voi olla vain yksi ns. Modbus Master -laite, joka aloittaa kommunikoinnin ja käskee allaan olevia Modbus RTU -orjalaitteita.

9.2.1 Modbus RTU:n tärkeitä termejä

Modbus RTU on RS485/422/232 -väylälle tarkoitettu toteutus Modbus -kommunikaatiosta. RTU:ssa kaikki data liikkuu bittimuodossa, eikä ole täten ihmisen luettavissa. Modbus ASCII taas käyttää ASCII-merkkejä ja on selkokielisessä, ihmisen luettavassa muodossa. Molempien toteutusten viestin lopussa on aina CRC- tarkastus-summa, jonka perusteella laitteistot tarkistavat, oliko saapunut viesti muuttunut matkalla.

Master eli isäntä

Modbus RTU:ssa Master toimii keskitetyn hallinnan kulmakivenä. Master on RTU -väylän pääasiallinen kommunikoija. Näitä voi olla vain yksi, ja sille ei määritetä SlaveID:tä, johtuen sarjaliikenneverkon luonteesta.

Slave eli orja

Väylässä oleva osoitteellinen orjalaite. Näitä voi olla useitakin, mutta jokaisella laitteella pitää olla yksilöllinen osoite välillä 1—255. Osoite 0 on varattu ns. broadcast -osoitteeksi ja sitä käytetään, jos halutaan, että kaikki väylän laitteet ottavat kirjoituskäskyn vastaan. Yksikään laite **EI** saa vastata tähän osoitteeseen lähetettyyn pyyntöön.

9.2.2 Modbus TCP:n tärkeitä termejä ja eroja

Server eli palvelin

TCP/IP-verkossa toimiva palvelin, jonka tehtävänä on ottaa vastaan pyyntöjä ja käsitellä niitä. Modbus TCP/IP:ssä server voi toimia pelkkänä välittäjänä eli ”gateway”-laitteena, jolloin toiminnassa noudatetaan tiettyjä standardissa määritettyjä ehtoja.

Client eli asiakas

Client eli asiakas on toiminnaltaan vastaava kuin Modbus RTU:n isäntä, mutta pienellä poikkeuksella: Modbus TCP:ssä näitä ”isäntiä” voi olla useita, eli useampi verkon kone voi käydä kysymässä Modbus-palvelimelta dataa ja näin verkon rakenteen ei tarvitse olla niin monimutkainen kuin RS485-verkon vastaavassa.

Laiteväylän kannalta tässä käyttötapauksessa tärkeää on, että väylän fyysinen kunto ja valittu liikennenopeus voidaan saavuttaa. Modbus RTU-väylässä liikenne joutuu pyörimään syklisesti, eli jokaisen laitteen kanssa liikennöidään vuorotellen ja Paneeli-PC:n ohjelmisto huolehtii liikenteestä sekä laiteväylän tutkimisesta.

9.3 Järjestelmän vikasietoisuus

MidiTrace-järjestelmän vikasietoisuus on korkea. Jokaisella MidiTrace-laitteella on oma ”äly”, jonka ansiosta ne voivat jatkaa toimintaansa normaalisti, vaikka paneeli-PC tai yläjärjestelmä putoaisi pois pelistä. Mikäli yksittäinen MidiTrace-laite rikkoutuu, väylässä vain yksittäinen laite jää vastaamattomaksi.

Mikäli yläjärjestelmä rikkoutuu, laitteisto jatkaa normaalia toimintaansa. MidiTrace-laitteisto turvaa saattojen ohjauksen ja toiminnan myös tietoliikenteen vikatilanteissa, joissa Ethernet-verkko tulee alas jostakin syystä.

Jatkokehityksellisenä ideana paikallinen paneeli voisi tarjota myös ohjeita käyttäjälle, miten selvittää mahdollista RS485-väylävikaa tai muita ohjeita esimerkiksi saattolämmityspiirin käyttöönottoon.

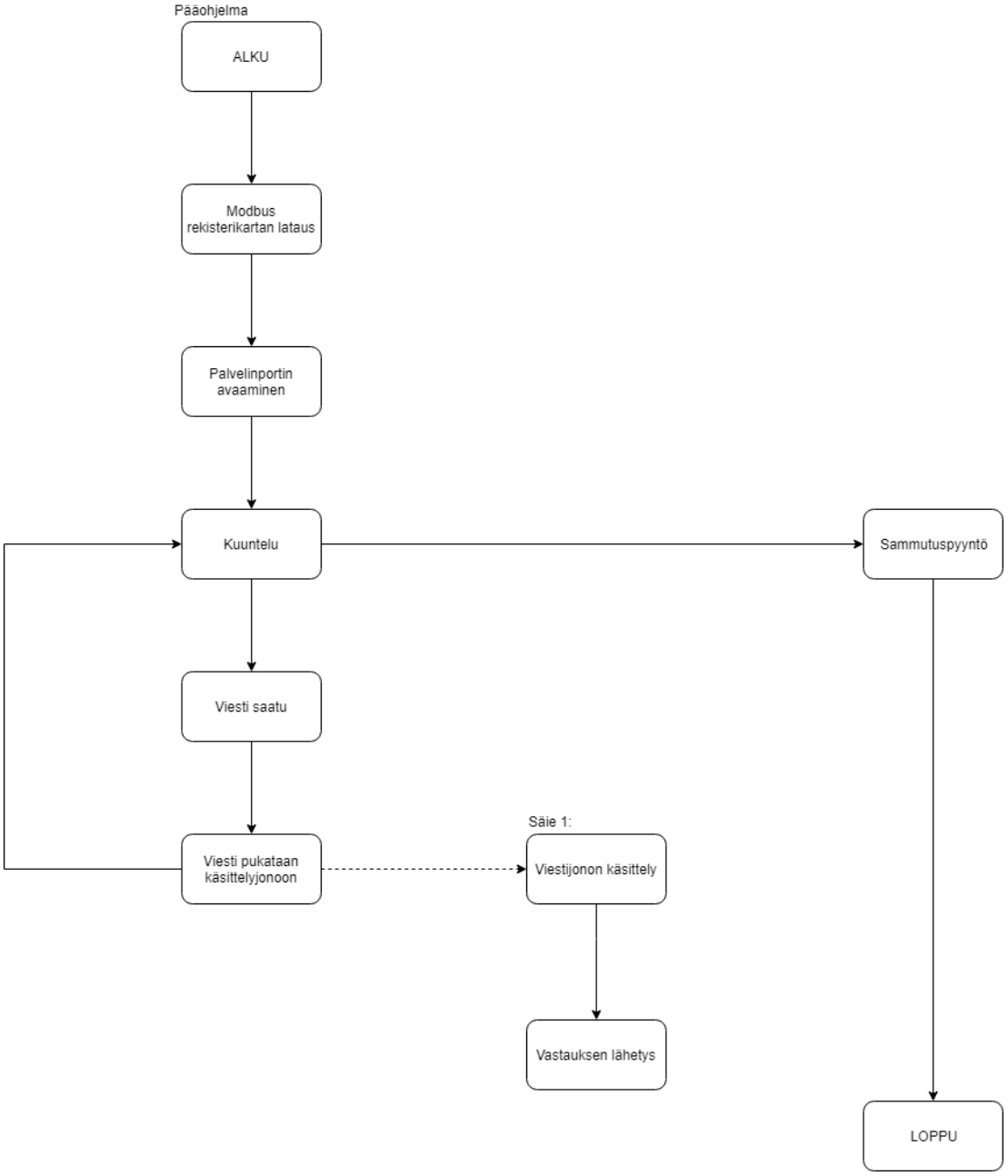
9.4 Suunnittelu

Palvelinohjelmiston suunnittelu aloitettiin mallintamalla ohjelmiston kulkua askel askeleelta kaavioina. Ohjelmiston suunnittelussa käytettiin apuna Modbus IDA:n julkaisemaa “Modbus application protocol”-kirjasta, jossa käydään läpi ohjelmiston toiminta eri tilanteissa seikkaperäisesti kaavioina.

Palvelinohjelmiston kehitys aloitetaan siten, että asetetaan tavoitteet ohjelmalle ja sen toteutukselle:

- Palvelin kykenee käsittelemään Modbus TCP-viestin ja vastaamaan siihen sovellusprotokollan mukaisesti.
- Palvelin kykenee käsittelemään virheet sekä toipumaan niistä itsenäisesti.
- Palvelin pystyy näyttämään rekisterien sisällöt kehitysversiossa.
- Palvelin toimii mahdollisimman jouhevasti vähintään kahden laitteen yhdyskäytävänä, mutta tässä toteutuksessa sitä simuloidaan dataa muuttamalla.

Ohjelmistolle luotiin pohjaksi kuva 14, josta käy ilmi sen toiminta eri tilanteissa, kuitenkaan sen syvemmin ohjelmakoodiin menemättä.

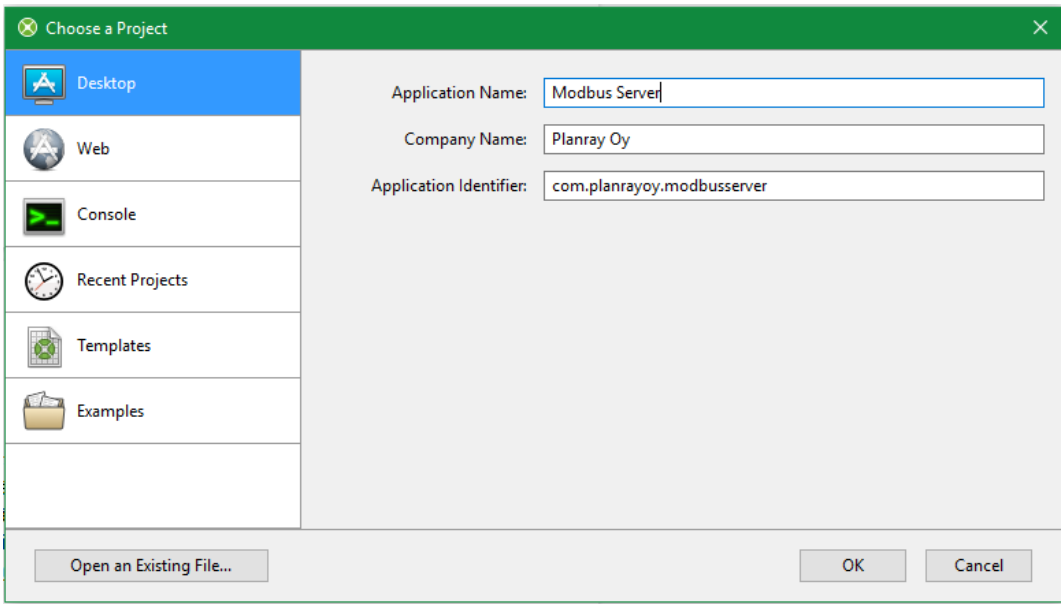


Kuva 14. Modbus-pääohjelman kaavio

9.5 Pääohjelman luonti ja käyttöliittymän suunnittelu

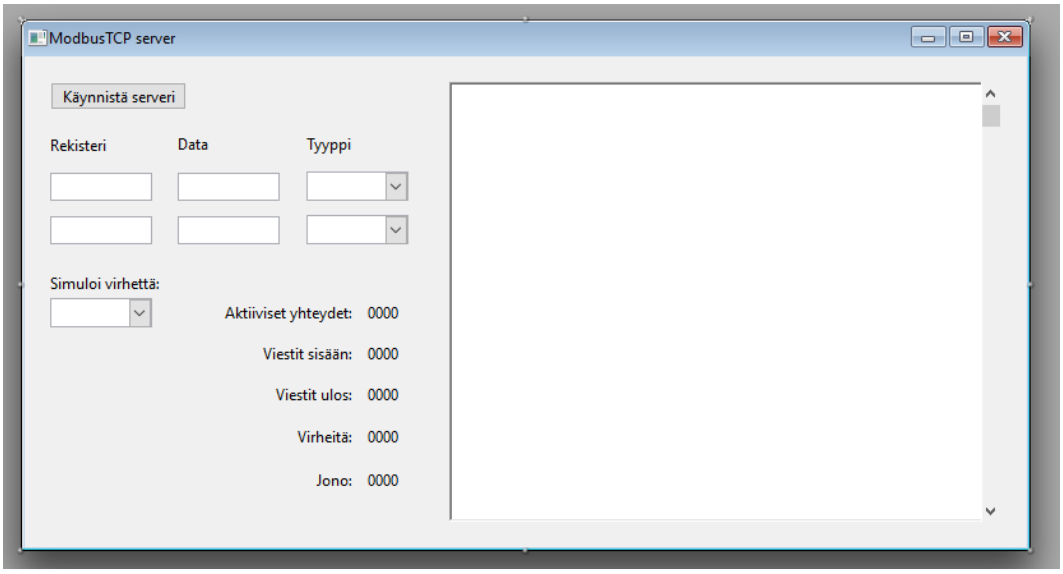
Ohjelman kehitys aloitettiin Xojo-ympäristössä luomalla normaali työpöydälle tarkoitettu ohjelma, koska ohjelmassa on ikkuna ja ohjelmointiympäristönä ikkunassa on huomattavasti käyttäjäystävällisempää käsitellä säikeitä sekä dataa.

Ohjelmalle määritettiin aloitusikkunassa alla nimi sekä yritys, joka ohjelmaa kehittää (ks. kuva 15). Työ tallennetaan ”Xojo-project”-muotoon ihmisen luettavassa muodossa, jotta se pystytään ottamaan mukaan lähdekoodin hallintaan, kuten GitLabiin.



Kuva 15. Xojon aloitusikkuna

Ohjelmisto näyttää tarvittavat tiedot tilasta sekä pienen loki-ikkunan, johon kerätään ajonaikaista tietoa. Pääikkunan asettelu (kuvassa 16) tehtiin siten, että sillä pystytään alussa testaamaan sekä tutkimaan palvelinohjelmiston toimintaa eri tilanteissa.



Kuva 162. Ohjelmiston käyttöliittymän suunnittelu

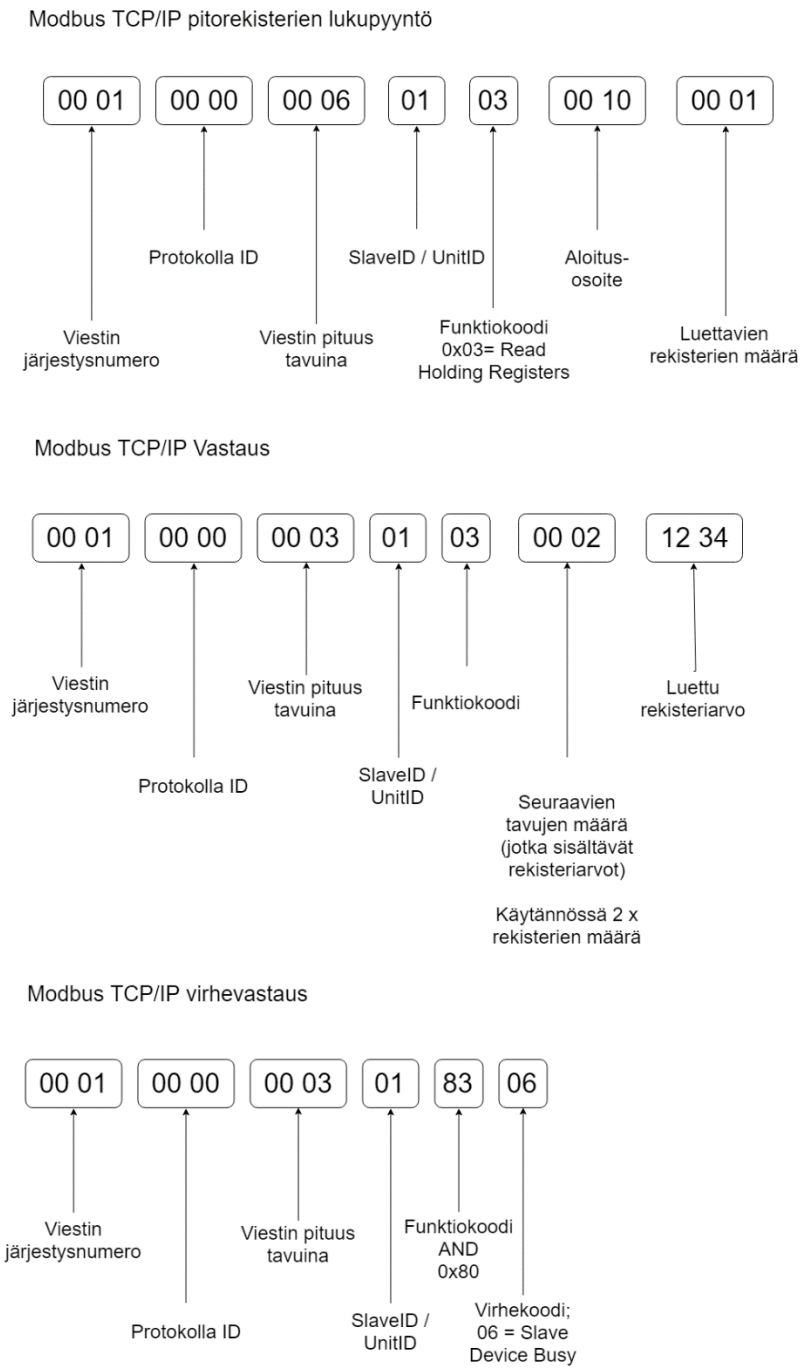
9.6 Modbus-viestin käsittely ja vastauksen luominen

Ohjelmaan lisättiin heti aluksi nimetty putki, “soketti”, jonka kautta Modbus-yhteys palvelimelle tapahtuu. Kun tähän putkeen avataan TCP-yhteys ennalta määritettyyn porttiin 502, se luo TCP/IP-yhteyttä varten oman säikeen, jossa “ModbusSocket” -moduuli hoitaa liikenteen käsittelyn ja viesteihin vastaamisen.

”ModbusSocket” -moduulissa on määritelty, mitä ohjelma tekee eri tilanteissa. Jos ohjelma havaitsee yhteydessä dataa, se luetaan muistiin ja otetaan heti käsittelyyn aikakatkaisun välttämiseksi. Aluksi testattaessa määriteltiin ohjelma siten, että se lukee viestin ja vastaa siihen “Slave Busy” -virhekoodilla.

Modbus-viestiin vastattaessa virhekoodilla sen funktiokoodin ylin bitti käännetään ykköseksi, jotta toinen järjestelmä tietää, että mukana seuraa virhekoodin sisältävä vastaus. Kuva 17 esittää Modbusin viestirakennetta.

Modbusin erilaisia virhekoodeja ovat mm. ”Illegal Data Address”, ”Invalid Value” sekä ”Slave Device Busy”. Kyseisiä koodeja käytetään järjestelmän tilan ilmaisemiseen, koska ModBus-väylä ei voi jatkuvasti ilmaista tilaansa, vaan vain vastattaessa saatuun Modbus-pyyntöön. Tietyissä tapauksissa ohjatun laitteen tilaa voidaan ilmaista erillisessä Modbus-rekisterissä ja täten saada tietoa järjestelmän tilasta ilman, että laite on Modbusin kannalta virheellisessä tilassa.

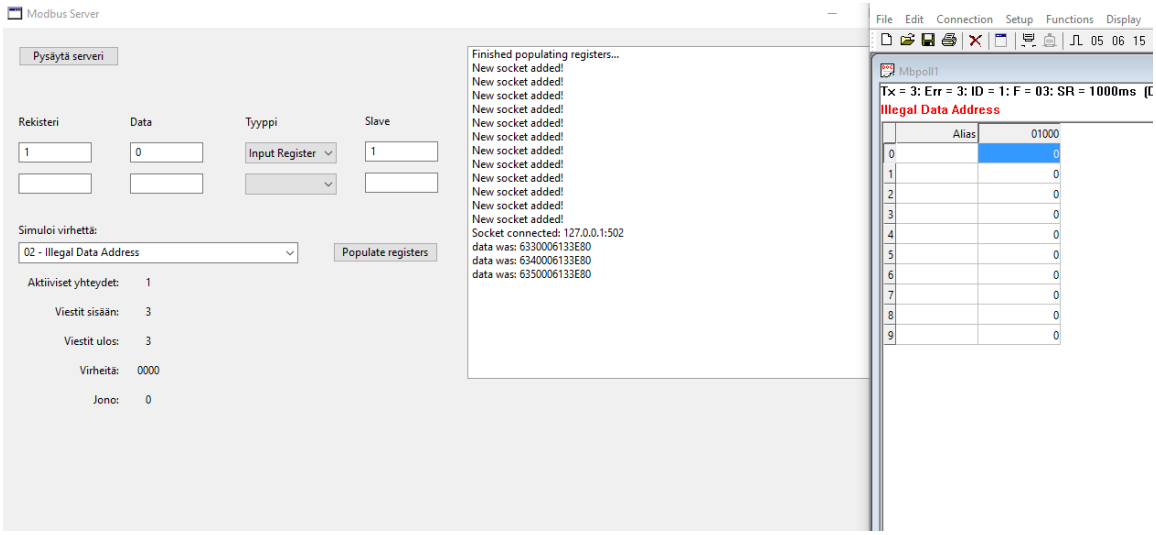


Kuva 17. Modbus TCP/IP -viestin rakenne.

Testaamisessa käytettiin Witte Softwaren kehittämää ModbusPoll-ohjelmaa. Ohjelma tuntee suurimman osan eri Modbus-virhekoodeista, sekä sillä nähdään myös muut virheet, mitä Modbus-viesteissä voi olla, kuten esimerkiksi puuttuvia tavuja viestin pituudessa. [19.]

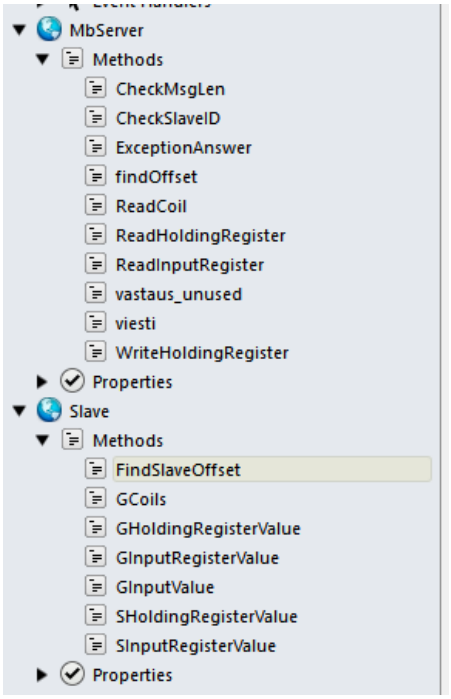
Ohjelmaan luotiin funktio, joka toimii siten, että se tarkistaa Modbus-funktion koodin, onko pyydetty rekisteri pitorekisterialueella, sekä palauttaa rekisteriarvot, mikäli aiemmat ehdot täyttyvät.

Tämä testattiin myös samaisella ModbusPoll-ohjelmistolla ja tarkkailtiin, että muuttunut arvo palautuu oikein ohjelmasta. Esimerkki testauksesta nähtävissä kuvassa 18.



Kuva 183. Modbus virhekoodin testaus

Tässä vaiheessa ohjelmaan syntyneet Modbus-funktiot otettiin omaan paikkaansa, ”moduuliin” (ks. kuva 19), jotta ne ovat ohjelman käytettävissä moduulin nimen takaa ja jotta ohjelmistokoodin uudelleenkäyttö myös muissa projekteissa olisi mahdollisimman helppoa.



Kuva 19. Moduulien rakenteet ja alifunktiot

Taulukossa 1 nähdään esimerkkinä Modbus-kartta, joka kertoo ohjelmoijalle, miten rekisterit eri orjayksiköiden välillä jakautuvat esimerkkiohjelmassa.

Tyyppi	Rekisteri	Selite
Pitorekisteri	1000	Lämpötila (x 10)
Pitorekisteri	1001	Asetusarvo (x 10)
Pitorekisteri	1002	Virranmittaus
Pitorekisteri	1003	Vikavirran mittaus
Digitaalinen tulo	200	Kytkin
Analoginen tulo	100	Potentiometri

Taulukko 1. Modbus-kartta

Ohjelmaan luotiin aluksi vain kaksi orjalaitetta, jotta testaus olisi helpompaa ja muistin käyttöä voisi tarkkailla paremmin.

Pitorekisterien alustus yhdelle orjalaitteelle tehdään koodilla, joka näkyy kuvassa 20.

```
dim apu, slaveToHave, oldOffset as integer=0
dim max as integer=55

slaveToHave=1

oldOffset=ubound(slave.HRegister,1) // haetaan taulukon entinen yläraja
if oldOffset=-1 then // jos taulukko oli ennestään tyhjä…
oldOffset=0 // offset=0
end
redim slave.HRegister(ubound(slave.HRegister,1)+max,3) // slavekenttä pituus+ rekisterimaara
// kasvattaa rekisterimäärää max:in arvolla

while apu<>max
slave.HRegister(oldOffset+apu,1)=SlaveToHave // slave jolle rekisterit varataan
slave.HRegister(oldOffset+apu,2)=1000+apu // register address
slave.HRegister(oldOffset+apu,3)=100+apu // data value
apu=apu+1
wend

ListBox1.AddRow("Finished populating registers...")
```

Kuva 204. Pitorekisterin alustuskoodi

Ylläolevan ohjelmakoodin avulla saadaan muistia varattua dynaamisesti, tarvittavien laitteiden mukaan. Jos varaukset tehtäisiin aina kaikille orjalaitteille, vaikka ne eivät ole verkossa, ohjelmiston muistirasitus olisi hetkessä valtava.

Ohjelmakoodin suoritusaikaa pystyy seuraamaan Xojon sisäänrakennetun koodinprofilointijärjestelmän avulla, joka mittaa eri funktioiden viemää aikaa ja niiden toimintaa. Esimerkin Xojon koodiprofiloinnin tuottamasta tiedosta on nähtävissä kuvassa 21.

Name	Called	Total (milliseconds)	Average (milliseconds)
Main Thread ModbusSocket.Connected	1	0,0000	0,0000
Main Thread ModbusSocket.DataAvailable	21	0,0000	0,0000
MbServer.viesti	21	20,0000	0,9524
MbServer.CheckMsgLen	21	0,0000	0,0000
MbServer.CheckSlaveID	21	0,0000	0,0000
MbServer.ReadHoldingRegister	21	5,0000	0,2381
MbServer.ExceptionAnswer	11	0,0000	0,0000
MbServer.findOffset	21	0,0000	0,0000
Slave.GHoldingRegisterValue	511	9,0000	0,0176
Main Thread Window1.Open	1	0,0000	0,0000
Main Thread Window1.PushButton1.Action	1	622,0000	622,0000
Main Thread Window1.PushButton2.Action	1	0,0000	0,0000
Main Thread Window1.ServerSocket1.AddSocket	12	1,0000	0,0833
Main Thread Window1.Timer1.Action	13	14,0000	1,0769

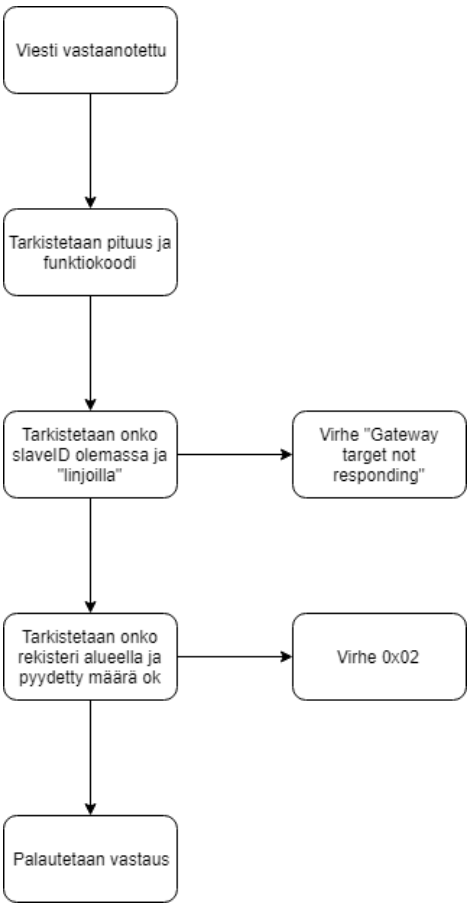
Kuva 21. Xojon profiloinnin tuottama taulukko

Kuten edellä olevasta kuvasta 21 voi päätellä, ohjelman Modbus-funktioiden suoritusaika oli noin 1 ms luokkaa keskimäärin, mikä on loistava verrattuna normaalisti Modbus-laitteissa käytettyyn 500 ms aikakatkaisuun.

Huomioitava on kuitenkin se, että tämä testijärjestelmä ajaa sekä Modbus-palvelinta että asiakasta samalla koneella, toisin kuin todellisessa ympäristössä, jossa palvelin ja asiakas voivat olla jopa satojen kilometrien päässä.

9.7 Modbus-funktioiden toteutus

Modbus-funktiot toteutettiin ohjelmaan omaan moduuliinsa, joka etsii taulukosta vastaavat arvot ja palauttaa datan käyttäjän tarpeen mukaan. Alkujaan toteutuksessa käytettiin taulukkoja, joihin saapuvat viestit pukattiin jonoon yhden säikeen käsiteltäväksi. Jonojen pitkä purkuaika aiheutti sen, että monet Modbus-viestit jäivät ilman vastauksia, eli viestit aikakatkaisivat itsensä. Tästä syystä ohjelman toimintaa muutettiin siten, että jokainen viesti otetaan välittömästi käsittelyyn omaan säikeeseensä, ja tällä tavoin ohjelmiston toiminta oli sujuvampaa ja saavutettiin nopeammat vasteajat kuin vanhalla jonoon perustuneen käsittelyn mallilla. Kuva 22 kuvastaa Modbus-virheiden palautusta eri ohjelman vaiheissa.



Kuva 22. Modbus viestin virhekäsittely [6. mukaillen fig. 13.]

Aiemmin luotu käsittelijä tutkii funktiokoodia ja antaa viestin käsiteltäväksi sen mukaan eri funktioille.

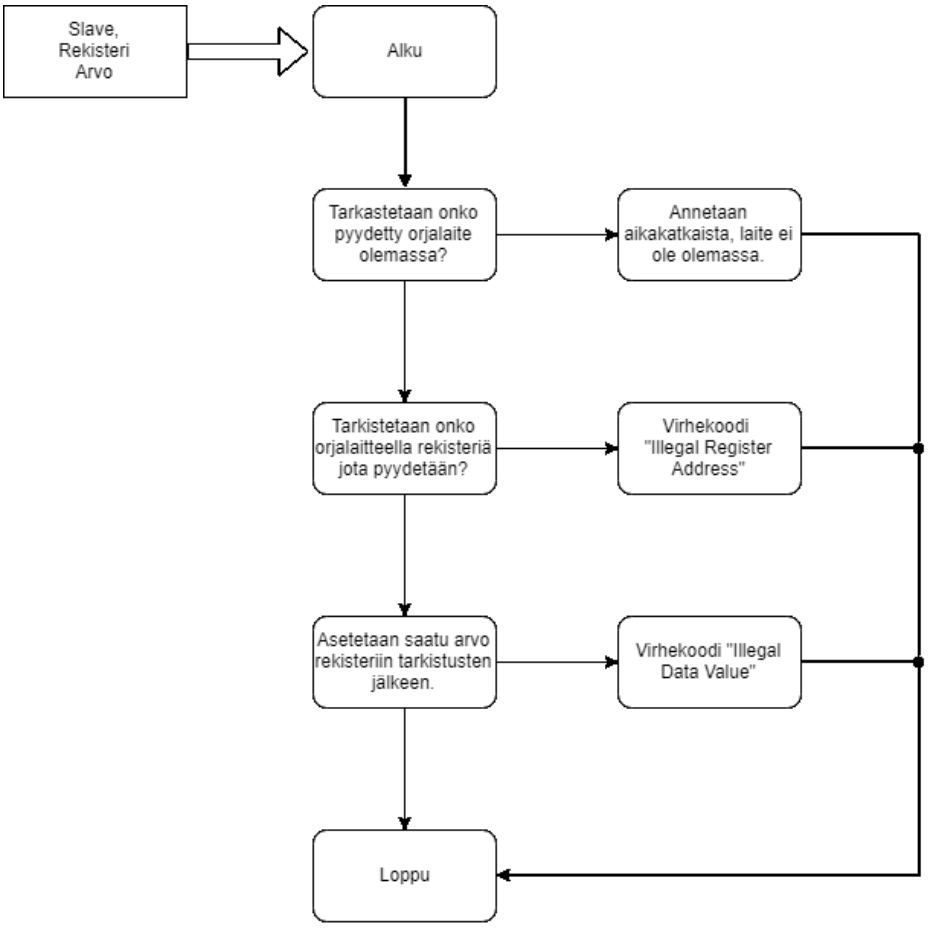
Eräs tärkein funktio, jota Modbus-liikenteessä käytetään, on lähes aina pitorekisterien luku. Tätä varten ohjelmaan luotiin yksinomaan rekisterien lukemiselle omistettu funktio. Kehityksen edetessä kaikki luku-, kirjoitus- sekä virheviestinnät on pyritty erottamaan toisistaan, jotta ohjelmisto olisi mukautuva ja koostuu helposti käsiteltävistä pienistä kokonaisuuksista.

Pitorekisterin lukemiseen liittyvän kuvan 17 perustella on nähtävissä, että viesti sisältää aina rekisterien määrän sekä aloitusrekisterin. Tätä tietoa hyödyntäen funktio pystyy etsimään oikean rekisterin, jonka tietoja palautetaan, ja määrittelemään paluuviestin pituus etukäteen pyydettyjen rekisterien määrän perusteella. Kuten alla oleva kuva 23 osoittaa, Modbus -viestin rakenteena käytetään aiemmin saatua viestiä muistialueelta ”data”, josta poimittuja tietoja hyödynnetään suoraan vastausviestin luomisessa.

```
modbusFrame.byte(0)=data.byte(0) // Järjestysnro saadusta modbus framesta (ylemmät bitit)
modbusFrame.byte(1)=data.byte(1)  //   Järjestysnro saadusta modbus framesta (alemmat bitit)
modbusFrame.byte(2)=data.byte(2) // protokolla (aina nolla, mutta kopioidaan vanhasta suoraan)
modbusFrame.byte(3)=data.byte(3) // protokolla (aina nolla, mutta kopioidaan vanhasta suoraan)
modbusFrame.byte(4)=Bitwise.ShiftRight(pituus-6,8,16) //tämän viestin pituus (pituus - 6)
modbusFrame.byte(5)=pituus-6 and &h00FF // viestin pituus, alemmat bitit
modbusFrame.byte(6)=data.byte(6) // slaveID
modbusFrame.byte(7)=data.byte(7) // funktio
modbusFrame.byte(8)= vastaus.size()  // vaihtoehtoisesti data10,data11 yhdistetään 16bit luvuksi ja 16bit arvo * 2
```

Kuva 23. Modbus-muistialueen käyttö vastausviestissä06 – Write Holding Register eli pitorekisterin kirjoitus

Rekisterien lukemisen lisäksi on tärkeää myös saada kirjoitettua dataa laitteelle. Kun rekisterien lukeminen onnistui, piti luoda myös rekisterien kirjoittamiselle omistettu funktio, joka hoitaa datan ottamisen serverin muistiin ja käsittelee sen sitten eteenpäin. Alla oleva kuva 24 esittää ohjelmiston rekisteriin saapuvan arvon käsittelyn eri vaiheet.



Kuva 24. Modbus-pitorekisterin kirjoitus [6 mukaillen fig. 16].

Rekisterin kirjoittaminen on hyvin pitkälti sama asia kuin rekisterin luku, mutta nyt viestin mukana seuraa arvo, jota rekisteriin halutaan kirjoittaa. Tämä arvo otetaan käyttötapauksen mukaan joko

suoraan palvelimen muistiin tai se lähetetään eteenpäin ohjattavalle laitteelle käsiteltäväksi, jotta muutos ei heijastu ennen kuin se on otettu oikeasti laitteessa käyttöön. Työssä toteutetussa versiossa tuota arvoa ei nyt lähetetä erikseen laitteelle, vaan kirjoituksen onnistumiseen vastataan automaattisesti ja arvo kopioidaan luettavaksi pitorekisteriin muistialueelle.

Modbus-liikenteessä rekisterissä olevaa arvoa voidaan myös käsitellä enemmänkin, ja sille voidaan määrittää sallittuja arvoja sekä tarkistuksia, joiden perusteella luettu arvo voidaan jättää huomioimatta tai siihen voidaan vastata virhekoodilla. Tätä menetelmää käyttämällä laitevalmistaja voi etukäteen varmistaa, että Modbus-kartan määrittämättä jättämiä arvoja kirjoitettaessa tai luettaessa laite ei toimi väärällä tavalla eikä väärä arvo aiheuta ohjelman väärää toimintaa tai jopa kalliita vahinkoja tuotannonmenetyksinä tai laiterikkoina.

10 Yhteenveto

Modbus -palvelimen suunnittelu ja toteutus itsessään on yllättävän haastava projekti, ja varsinkin projektin kuluessa joutuu tutustumaan monipuolisesti erilaisiin kenttäväyliin ja niiden käyttötarkoituksiin. Mielenkiintoisina vaihtoehtoina Modbus-väylälle ovat mm. avoimeen standardiin perustuvat EtherCAT sekä OPC, ja näiden toteuttaminen esimerkiksi Arduinoon tai muuhun vastaavaan järjestelmään on kenen tahansa ulottuvilla.

Aikataulullisesti tämän opinnäytetyön ajoitus oli hieman huono, koska työnantajan puolella sattui olemaan erittäin kiireinen kausi menossa, joten aikaa työn tekemiselle ei meinannut löytyä työajalta. Muistinhallinnan tärkeys korostui palvelimen toimintoja toteutettaessa, jottei resursseja käytetä turhaan. Palvelimen toiminnallinen toteutus jäi huomattavasti suunniteltua pienemmäksi rajallisen ajan ja resurssien puuttuessa. Toteutuksessa parannettavaa jäi etenkin ohjelmiston alustuksessa, jotta ohjelman voisi helposti sovittaa eri käyttötarkoituksiin ilman lähdekoodin muokkausta.

Kuitenkin nykyisessä muodossaan ohjelmisto pystyi muuttamaan periaatteessa minkä tahansa Windows- tai Linux-koneen Modbus TCP/IP -palvelimeksi. Pienellä lisäohjelmoinnilla siitä saisi tehtyä osan isompaa ohjelmistoa, tai vaikka liitettyä sen sarjaväylällä ulkoisiin laitteisiin. Xojolle löytyy Raspberry PI -korttitietokonetuki, joten funktioiden toteutus sille olisi vain kääntäjän asetusten muokkaamisesta kiinni.

Lähteet

1. Planray Oy. Tietoja yrityksestä. [Internet] Saatavilla 3.6.2019. www.planray.com

2. Kozierok, C M. *TCP/IP Guide.* USA. No Starch Press; 2005;

3. Sanders, C. *Practical packet analysis.* 3. p. USA. No Starch Press; 2017.

4. Modbus.org. Modbus messaging on TCP/IP implementation guide V1.0b. [Internet] Modbus.org; 2006. Saatavissa 19.5.2019. http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

4. "Bergadder". Image of power plant control room. [Internet] Pixabay; 2014. Saatavissa 3.6.2019. https://pixabay.com/photos/power-plant-conducting-electric-old-344231/

5. R. Bosch GmbH.*CAN Bus Specification 2.0*. NXP; 1998. Saatavissa 19.5.2019. https://www.nxp.com/docs/en/reference-manual/BCANPSV2.pdf

6. Modbus.org. MODBUS Application Protocol Specification V1.1b3. [Internet] Modbus.org; 2012. Saatavissa 19.5.2019. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

7. Rinaldi, J S. *The Everyman's Guide To Modbus.* USA. Rinaldi, John S; 2015.

8. Felser M, Mitchell R. PROFIBUS. Teoksessa Bogdan, M., Wilamowski, J. & Irwin, D. (toim.) *Industrial Communication Systems* (32-1). 2. p. UK. CRC Press; 2011.

9. Felser M, Paolo F, Alessandra F. PROFINET. Teoksessa Bogdan, M., Wilamowski, J. & Irwin, D. (toim.) *Industrial Communication Systems (40-1 – 40-2).* 2. p. UK. CRC Press.

10. Cena, G., Valenzano, A. & Zuniono, C. EtherCAT. Teoksessa Bogdan M., Wilamowski, J. & Irwin, D. (toim.) *Industrial Communication Systems* (38-1). 2. p. UK. CRC Press; 2011.

11. Dang, T & Renaud, A. OPC-UA. Teoksessa Bogdan, M., Wilamowski, J. & Irwin, D. (toim.) *Industrial Communication Systems* (57-1). 2. p. UK. CRC Press; 2011.

12. RealSoftware. RealBasic is now known as Xojo. [Internet] RealSoftware; 2013. Saatavilla 6.11.2019. http://www.realsoftwareblog.com/2013/06/real-studio-is-now-xojo.html.

13. Xojo organization. Xojo: Cross-platform App Development Tool. Xojo; 2019. Saatavilla 17.3.2019. www.xojo.com

14. GitLab Inc. About GitLab. [Internet] Gitlab Inc; 2019. Saatavilla 17.3.2019. https://about.gitlab.com/product/.

15. Planray Oy. Tuotteet. [Internet] Planray Oy; 2019. Saatavilla 17.12.2019. http://www.planray.com/fi/tuotteet/miditrace-sahkolammityssaadin+keskusasennuk-siin/

16. Planray Oy. Extranet. [Internet] Planray Oy; 2019. Saatavilla 17.12.2019. http://www.planray.com/fi/extranet

17. Huhtala, M. (2019). Kehityspäällikkö/ Caverion Suomi Oy Teollisuuden ratkaisut. Haastattelu 12.11.2019.

18. WinMate. W10IB3S-EHH2. WinMate;2019. Saatavilla 17.12.2019. https://winmate-rugged.com/HMI/W10IB3S-EHH2.asp?Prod=03_1383

19. Witte Software. Modbuspoll [Internet]. Witte Software.com; 2019. Saatavilla 17.3.2019. www.modbustools.com