Modbus TCP –liitynnän suunnittelu konenäköjärjestelmään

Tero Olli

Opinnäytetyö
Huhtikuu 2013

Automaatiotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) OLLI, Tero	Julkaisun laji Opinnäytetyö	Päivämäärä 05.04.2013
	Sivumäärä 48	Julkaisun kieli suomi
		Verkkojulkaisulupa myönnetty (X)
Työn nimi MODBUS TCP –LIITYNNÄN SUUNNITTELU KONENÄKÖJÄRJESTELMÄÄN		
Koulutusohjelma Automaatiotekniikan koulutusohjelma		
Työn ohjaaja(t) RANTAPUSKA, Seppo		
Toimeksiantaja(t) Vision Systems Oy		

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Modbus TCP -kommunikointifunktiot konenäköjärjestelmään. Työ tehtiin toimeksiantona optisen mittauksen ja konenäkötekniikan asiantuntijayritykselle Vision Systems Oy:lle.

Opinnäytetyössä selvitettiin aluksi konenäköjärjestelmän toimintaperiaatteet, käyttökohteet sekä konenäkölaitteistot. Lisäksi perehdyttiin kenttäväyliin yleisesti sekä perusteellisemmin Modbus TCP -protokollaan ja Ethernettiin.

Selvityksen perusteella ryhdyttiin tekemään käytännön toteutusta. Toteutus muodostui ohjelmakoodin kirjoittamisesta sekä valmiin ohjelman testauksesta. Konenäkökameraan ohjelmoitiin palvelin- ja asiakasohjelmatoiminnallisuudet.

Työn toteutuksen tuloksena saatiin toimiva ohjelma jolla konenäkökamera keskustelee muiden laitteiden kanssa Modbus TCP -protokollan mukaisesti. Ohjelmaa voidaan myöhemmin kehittää luomalla erilliset Modbus-rekisterit palvelimelle ja asiakasohjelmalle sekä lisäämällä mittausarvoille ja parametreille rekisteriin vakiopaikat.

Avainsanat (asiasanat)

teollisuusautomaatio, konenäkö, väylät, Modbus TCP, Ethernet

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Author(s)
OLLI, Tero

Type of publication
Bachelor´s Thesis

Date
05042013

Pages
48

Language
Finnish

Permission for web publication
(X)

Title
DESIGNING MODBUS TCP COMMUNICATION FUNCTIONS FOR THE MACHINE VISION SYSTEM

Degree Programme
Automation Engineering

Tutor(s)
RANTAPUSKA, Seppo

Assigned by
Vision Systems Oy

Abstract

The aim of the Bachelor's thesis was to design and implement the Modbus TCP communication functions for the machine vision system. The thesis project was assigned by Vision Systems Oy.

In the thesis study the operation of the machine vision system and field buses, including Modbus TCP and Ethernet were researched. Implementation included programming and testing. The client and server functionalities were implemented in the machine vision camera.

As a result of the thesis, an application that communicates through Modus TCP -protocol was produced.

Keywords
industrial automation, machine vision, bus, Modbus TCP, Ethernet

Miscellaneous

Sisältö

KUVIOT

TAULUKOT

LYHENTEET JA KÄSITTEET

ADU	Application Data Unit
ASCII	American Standard Code for Information Interchange
BIG-ENDIAN	Merkitsevin bitti kirjoitetaan ensin
MAC	Media Access Control
MDAP	Modbus Application Protocol header
NTSC	National Television System Commitee
OSI	Open Systems Interconnection Reference Model
PAL	Phase AAlternate Line
PDU	Protocol Data Unit
RS232	Sarjaliikennestandardi
RS485	Sarjaliikennestandardi
SECAM	Sequential Color With Memory
SSH	Secure Shell
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
KVANTISOINTI	Analogisen näytteen digitaalisointi

1 JOHDANTO

1.1 Opinnäytetyön taustat ja toimeksiantajan esittely

Opinnäytetyö tehtiin toimeksiantona Vision Systems Oy:lle. Työssä tehtiin VISI 50 -konenäkökameraan Modbus TCP -palvelin (server) sekä asiakassovellus (client) toiminnallisuudet.

Toimeksiantajana toimi Vision Systems Oy, joka on optisen mittauksen ja konenäkötekniikan asiantuntijayritys. Yrityksen asiakaskunta koostuu paperi-, sellu-, metalli-, teräs-, lasi-, elintarvike- ja rengasteollisuuden suuryrityksistä sekä eri alojen laite- ja konevalmistajista. Yritys toimii Jyväskylässä ja toimittaa järjestelmiä ympäri maailmaa. (Vision Systems Oy)

1.2 Opinnäytetyön tavoitteet

Opinnäytetyön tavoitteena oli luoda toimeksiantajan konenäkökameraan Modbus TCP -palvelin sekä asiakassovellus. Nämä toteutettiin kirjoittamalla C-kielinen ohjelmakoodi ja testaamalla sen toimivuus. Testauksen apuna käytettiin tietokoneella toimivia palvelin- ja asiakasohjelmia sekä Siemensin S7-300 sarjan ohjelmoitavaa logiikkaa.

Opinnäytetyössä perehdyttiin konenäköjärjestelmän toimintaan sekä kenttäväyliin. Kenttäväylistä tarkemmin tutustutaan Modbus TCP -protokollaan.

2 KONENÄKÖ

2.1 Konenäkö yleisesti

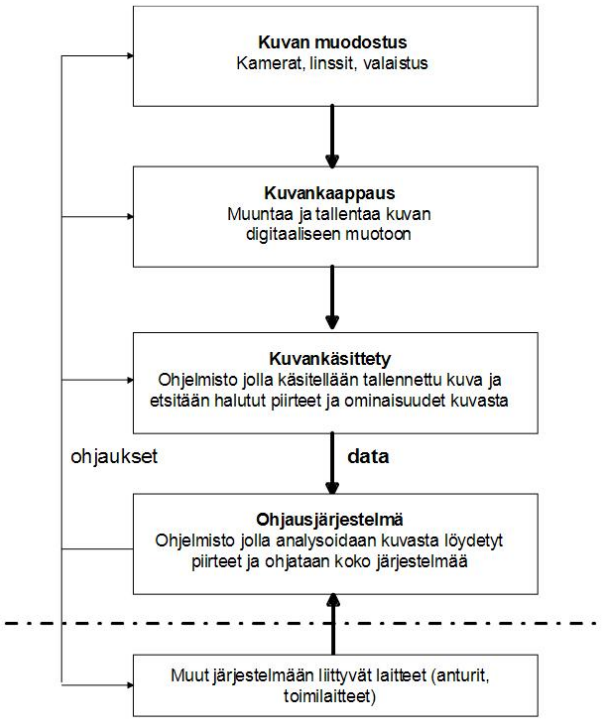
Konenäöllä matkitaan ihmisen näköaistia ja on yleensä osa tuotannon automaatiota. Tuotannossa yleisimmät käyttökohteet ovat laadunvarmistus sekä kohteen paikantaminen. Lisäksi konenäköä käytetään esimerkiksi työvaiheiden dokumentointiin sekä tuotantomäärien seuraamiseen. (Niemi)

2.2 Konenäön käyttökohteet

Konenäköjärjestelmiä on nykyisin käytössä laajasti monilla eri teollisuuden aloilla. Järjestelmien käyttökohteita löytyy erilaisten prosessien automatisoinnista. Lajittelu ja laadunvarmistus ovat yksi suurimmista konenäköjärjestelmien käyttökohteista. Esimerkiksi näitä ovat postin materiaalivirtojen hallinta, tuotteiden pakkausprosessi sekä virheiden etsiminen esimerkiksi paperin valmistuksessa. Lisäksi konenäön avulla voidaan tuotteen paikantaminen ja noukkiminen sekä siirtäminen paikalleen toteuttaa automaattisesti. Tämä tapahtuu kappaleen koordinaatteja hyödyntäen. Kappaleen dimensioita voidaan myös mitata konenäöllä ja esimerkiksi puun sahaus voidaan optimoida. Konenäköä käytetään lisäksi robotin paikantamiseen ympäristön mukaan, jolloin esimerkiksi vihivaunut osaavat liikkua itsenäisesti. (Halinen 2007)

2.3 Konenäön toimintaperiaate

Konenäön toimintaperiaatteena on saada koneellinen järjestelmä tulkitsemaan kameran tai muun sensorin kuvaamaa näkymää ja hyödyntää saatu tieto erilaisissa sovelluksissa (Pietikäinen ja Silven 2008). Järjestelmä rakentuu neljästä eri osasta: kuvanmuodostuksesta eli optiikasta ja kamerasta, kuvankaappauksesta, kuvankäsittelystä ja ohjausjärjestelmästä. (Kuvio 1)



KUVIO 1. Konenäköjärjestelmän toiminta (Halinen 2007)

2.4 Konenäkölaitteisto

2.4.1 Kamera

Konenäköjärjestelmän tärkeimpänä komponenttina pidetään kameraa. Kamera valitaan aina käyttökohteen mukaan. Kohteiden erityisvaatimuksia voivat olla esimerkiksi elintarviketeollisuudessa käytettävien puhdistusaineiden sieto. Käyttökohteen mukaan valitaan myös kameran tarkkuus. Konenäköjärjestelmät käyttävät jopa 2 megapikselin tai 5 megapikselin kameroita, mutta useimmiten riittää 0,3 megapikselin kamera. Tä-

mä vastaa 640 x 480 pikselin kennoa. Liian suuri tarkkuus nostaa kuvankäsittelyltä vaadittavaa prosessointitehoa. Kameran sijoituksella ja optiikalla pyritään saamaan kohde rajattua tarkasti kohteeseen. Monessa käyttökohteessa riittää mustavalkokamera ja värikameran käyttö ei olisi tarkoituksenmukaista, sillä sen kolme värikanavaa moninkertaistavat tarvittavan laskentatehon. (Niemi)

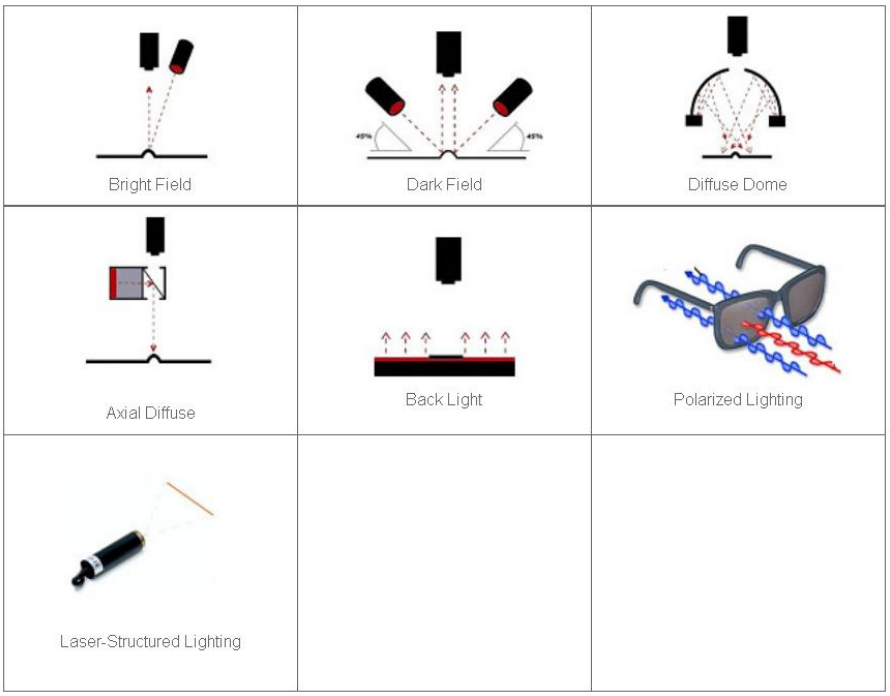
Konenäköjärjestelmissä käytettävät kamerat ovat matriisi- ja viivakameroita. Matriisikamerassa kuvaelementit muodostavat ruudukon ja viivakamerassa kuvaelementit ovat jonossa. Viivakameroita käytetään liikkuviin kohteisiin, esimerkiksi paperiteollisuudessa paperiradan kuvaamiseen. (Halinen 2007, 4)

Kameran videosignaalin siirrossa käytetään analogista tai digitaalista muotoa. Analogisissa signaaleissa käytetään pääosin standardoituja signaaleja, joista yleisimmät ovat PAL, NTSC ja SECAM. Joidenkin valmistajien erikoiskamerat käyttävät omaa videomuotoa. Toisin kuin analogisessa signaalissa, digitaaliselle signaalille ei ole standardeja. (Halinen 2007, 4-5)

Älykamerat edustavat kameroiden uusinta kehityssuuntaa. Niihin on sisäänrakennettu kamera, optiikka, valaistus sekä kuvankäsittelylaitteisto (Soini 2011). Toisin kuin perinteiset konenäköjärjestelmät, älykamerat eivät tarvitse erillistä ohjausyksikköä, vaan voidaan liittää suoraan automaatiojärjestelmään.

2.4.2 Valaistus

Kuvattavasta kohteesta ja kuvauksen tavoitteesta riippuen käytetään erilaisia valaistustekniikoita. Kuvassa 2 ja taulukossa 1 esitellään erilaisia valaistustekniikoita. Esimerkiksi diffuse dome -valonlähdettä käytetään, kun halutaan kuvattavalle pinnalle tasainen valaistus ja back light -valonlähdettä käytetään, kun halutaan kuvata virheitä paperissa tai halutaan kohteen ääriviivat esille.



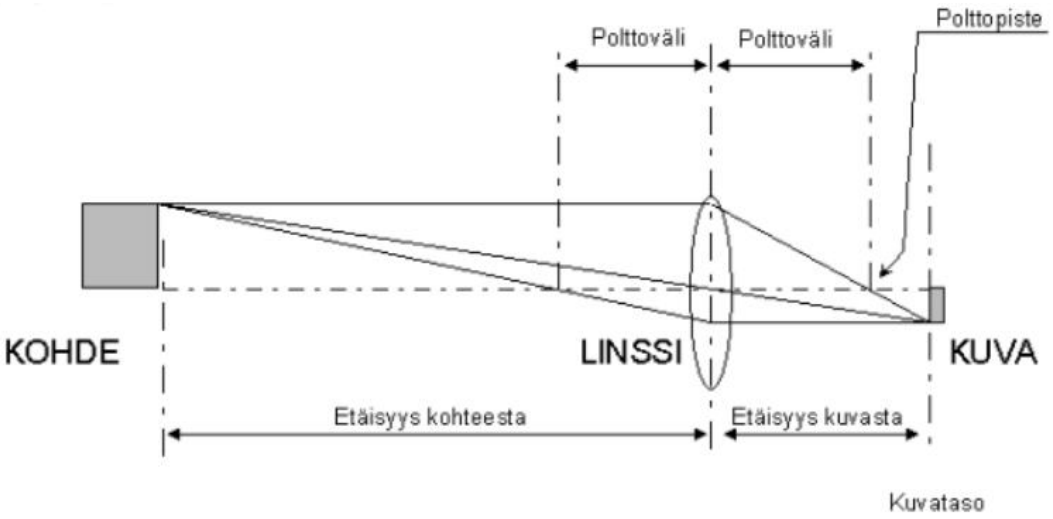
KUVIO 2. Valaistustekniikat (Valaistus)

TAULUKKO 1. Valaistuksen valinta (Valaistus)

Sovelluksen vaatimukset	Tarkasteltava kohde	Valaistussuositus
Heijastusten vähentäminen	Kirkas/kiiltävä	Diffuse Front, Diffuse Axial, Polarizing
Kohteen tasainen valaiseminen	Mikä tahansa	Diffuse Front, Diffuse Axial, Ring Guide
Pinnan muotojen korostaminen	Lähes litteä (kaksiulotteinen)	Single Directional, Structured Light
Kohteen rakenteen korostus varjoilla	Mikä tahansa	Directional, Structured Light
Varjojen vähentäminen	Ulkonemat, kolmiulotteinen	Diffuse Front, Diffuse Axial, Ring Guide
Kohteen vikojen korostus	Läpinäkyvä	Dark Field
Kohteen silhuetti	Mikä tahansa	Backlighting
Kohteen kolmiulotteinen muoto	Ulkonemat, kolmiulotteinen	Structured Light

2.4.3 Optiikka

Optiikkaa tarvitaan, jotta kuva saadaan siirrettyä kameran kennolle. Kuvan ominaisuuksiin vaikutetaan aukolla (f-luku), polttovälillä sekä linsseillä. Aukon suuruus vaikuttaa valon määrään, joka pääsee kennolle, jolloin kuvaaminen heikossa valossa ja lyhyillä suljinajoilla onnistuu. Linssin polttovälillä määritetään otettavan kuvan koko. Linssejä on kahdenlaisia: vakiosuurennus- ja zoomattavat linssit. Zoomattavissa linsseissä esiintyy enemmän virheitä kuin vakiolinsseissä. Näitä virheitä voidaan vähentää erikoislinsseillä tai matemaattisilla menetelmillä kuvanmuokkauksessa. (Halinen 2007, 6-7)



KUVIO 3. Yksinkertaistettu kuva kameran optiikasta (Halinen 2007)

2.4.4 Kuva-analyysi

Kuva-analyysin päävaiheet ovat Pietikäisen ja Silvenin (2008) mukaan:

- kuvan muodostus
- esikäsittely
- segmentointi
- sisällön kuvauksen muodostus
- sovitus

- tunnistus

Kuvan muodostus tarkoittaa kohteen kuvaamista kameralla ja kuvan digitointia. Valaistuksella ja kuvausjärjestelyillä voidaan helpottaa kuvan analysointia. Otetun kuvan kuvapisteiden määrä voi vaihdella sovelluksesta riippuen. Tyypillinen kuvakoko on 512 x 512 kuvapistettä. Harmaasävykuvissa, joita yleensä käytetään, sävyasteikko kvantisoidaan 256 tasoon. Värikuvissa vastaavasti jokainen kolmesta väristä kvantisoidaan 256 tasoon. (Pietikäinen ja Silven 2008)

Esikäsittelyssä kuvaa muokataan digitaalisella kuvankäsittelyllä. Kuvankäsittelyssä voidaan esimerkiksi normalisoida valaistuksessa tapahtuneet vaihtelut, joka vaikuttaa lopputulokseen. Muita toimenpiteitä ovat esimerkiksi kohinan poisto, häiritsevien sävyvaihteluiden poisto ja haluttujen piirteiden korostaminen. (Pietikäinen ja Silven 2008)

Segmentointi tarkoittaa kohteen ja kohteiden osien erottamista toisistaan ja taustastaan. Erotus voidaan toteuttaa esimerkiksi jyrkkien sävynmuutosten eli kappaleen reunojen mukaan. Segmentointi on kuva-analyysin kriittisimpiä vaiheita ja menetelmän valintaan vaikuttaa sovelluksen tyyppi. Segmentoinnin epäonnistuminen saattaa vaikeuttaa analyysiä tai estää sen. (Pietikäinen ja Silven 2008)

Kuvauksen muodostamisessa lasketaan erotettujen alueiden ja reunojen ominaisuuksia, esimerkiksi muotoa, väriä tai pintarakenteen tekstuuria. Usein tarvitaan myös tietoa kappaleen reunojen keskinäisistä suhteista. Kohteen tai poikkeaman tunnistaminen tapahtuu vertaamalla kuvaa konenäköjärjestelmään etukäteen opetettuihin prototyyppikohteiden malleihin. (Pietikäinen ja Silven 2008)

3 KENTTÄVÄYLÄT JA VÄYLÄTEKNIIKAT

3.1 Kenttäväylät yleisesti

Kenttäväyliksi kutsutaan teollisuudessa käytettäviä tiedonsiirtoratkaisuja, jotka yhdistävät laitteet ohjausjärjestelmään. Usein sovellukset ovat sellaisia, että kenttäväyliltä vaaditaan reaaliaikaista vastetta. Kenttäväyliltä vaaditaan myös luotettavuutta sekä riittävää nopeutta. Pitkään suosittu 4 – 20 mA analoginen virtaviesti on tekniikaltaan edullinen sekä yksinkertainen ja varmatoiminen. Toteutukseltaan tekniikka on monimutkainen, koska tekniikkaa voidaan käyttää vain kahden laitteen väliseen tiedonsiirtoon ja se vaatii jokaiselle laitteelle oman kaapeloinnin. (Baumgartner 2009)

Kenttäväylien avulla kaapelointia voidaan vähentää, jolloin pystytään yhdistämään jopa useita satoja laitteita yhteen väylään. Väylän laajennus on myös helpompaa, koska pitkiä kaapelivetoja ei tarvita. Eri kenttäväylillä voidaan käyttää erilaisia verkkotopologioita. Tällaisia ovat mm. daisy-chain-, tähti-, rinki- ja puu-verkkotopologiat. OSI-mallin mukaiseen toteutukseen voidaan fyysinen kerros toteuttaa esimerkiksi optisella tiedonsiirrolla, RS485 parikaapelilla tai koaksiaalikaapelilla. Näistä parikaapeli on käytetyin vaihtoehto optisen tiedonsiirron lisäksi. Taulukossa 2 selvitetään standardin IEC 61558 versiossa 2.0 2007-11 mukaiset protokollaperheet. (Baumgartner 2009)

TAULUKKO 2. Kenttäväyläteknologiat ja protokollat (Baumgartner 2009)

	Teknologia	Profiilin numero	Protokolla
1	FOUNDATION Fieldbus ™	1/1	FF H1
		1/2	FF HSE
		1/3	FF H2
2	CIP ™	2/1	ControlNet™
		2/2	EtherNet/IP™
		2/3	DeviceNet™
3	PROFIBUS, PROFINET	3/1	PROFIBUS DP
		3/2	PROFIBUS PA
		3/3	PROFIBUS CBA
		3/4	PROFINET IO CC-A
		3/5	PROFINET IO CC-B
		3/6	PROFINET IO CC-C
4	P-NET ®	4/1	P-NET RS-485
		4/2	P-NET RS-232
		4/3	P-NET on IP
5	WorldFIP ®	5/1	WorldFIP
		5/2	WorldFIP with subMMS
		5/3	WorldFIP minimal for TCP/IP
6	INTERBUS ®	6/1	INTERBUS
		6/2	INTERBUS TCP/IP
		6/3	INTERBUS minimal subset of CP 6/1
		6/4	
		6/5	
		6/6	
7	-	-	Poistettu
8	CC-Link	8/1	CC-Link/V1
		8/2	CC-Link/V2
		8/3	CC-Link/LT
9	HART	9/1	HART
10	Vnet/IP	10/1	Vnet/IP
11	Tcnet	11/1	Tcnet
		11/2	Tcnet-Loop
12	EtherCAT	12/1	
		12/2	
13	ETHERNET Powerlink	13/1	EPL
14	EPA	14/1	
		14/2	
15	MODBUS ® - RTPS	15/1	MODBUS TCP
		15/2	RTPS
16	SERCOS	16/1	SERCOS I
		16/2	SERCOS II
		16/3	SERCOS III

3.2 Teollisuus-Ethernet

Teollisuus-Ethernet -termillä tarkoitetaan useita eri ratkaisumalleja, joita yhdistää kyky käyttää Ethernet-teknologiaa sekä TCP/IP –protokollaperhettä. TCP/IP-protokollalla on vaikea täyttää reaaliaikaisuuden vaatimuksia. Vaihtoehtoisesti voidaan käyttää myös muita protokollia, jotta vasteajasta saadaan parempi. (Koivisto)

3.3 TCP/IP-protokollaperhe

TCP-protokolla huolehtii luotettavan päästä päähän -yhteyden muodostamisesta. Yhteys luodaan aina kahden kohteen välille. Tätä kutsutaan yhteydelliseksi tiedonsiirtoprotokollaksi. TCP:n rinnalla on myös UDP-protokolla, joka on yhteydetön protokolla. Ethernet-verkko koostuu useista pienemmistä verkoista, jotka kykenevät itsenäiseen toimintaan. Verkot voivat olla fyysisiltä olemuksiltaan erilaisia ja IP-protokolla hoitaa reitityksen näiden verkkojen välillä. Tämä mahdollistaa sen, että viestintä on mahdollista verkkojen välillä ilman monimutkaisia järjestelmänmuutoksia. Viestien välitykseen käytetään loogisia osoitteita, jotka ovat numeerisesti ilmaistuja osoitteita tai aakkosellisia laitenimiä. Fyysisesti liikennöinti tapahtuu laitetason osoitteilla, joita kutsutaan MAC-osoitteiksi. (Vainio ja Hakala 2005)

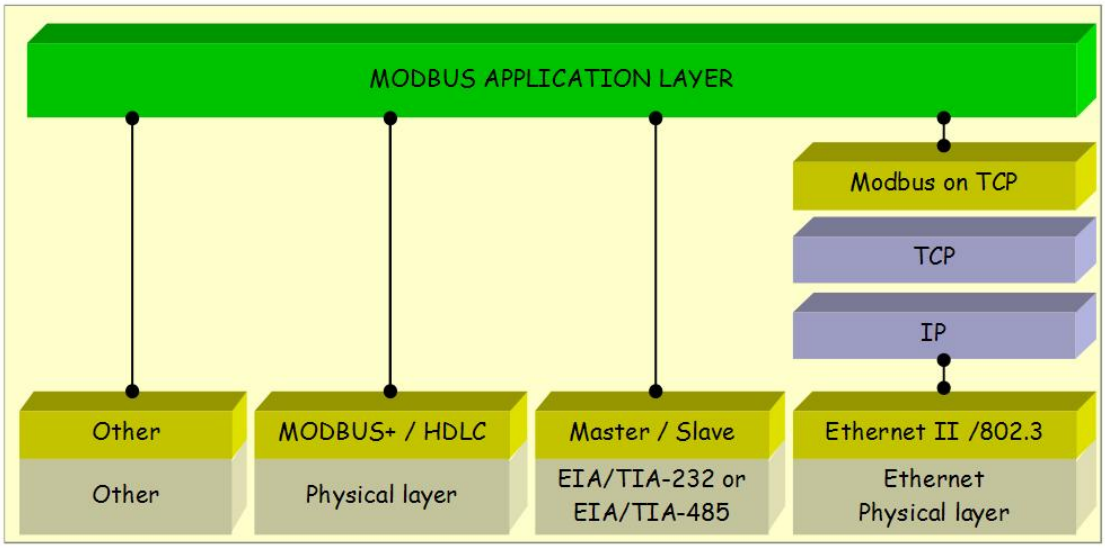
4 MODBUS-KENTTÄVÄYLÄ

4.1 Modbus yleisesti

Modbus on protokolla, joka toimii OSI-mallin 7. tasolla. Protokolla hoitaa palvelimen ja asiakassovelluksen välisen tiedonsiirron erilaisten väyläratkaisujen välillä. Modbusista on muodostunut alalle ’de facto’-standardi. Standardi on kehitetty vuonna 1979 ja sitä käyttävät miljoonat laitteet ympäri maailman. TCP/IP-pinossa on varattu portti 502 Modbus TCP:tä varten. Modbus on palvelupyyntö (request/reply) protokolla ja se toimii eri toimintoja kuvaavilla funktiokoodeilla (function codes). (Modbus 2012, 2)

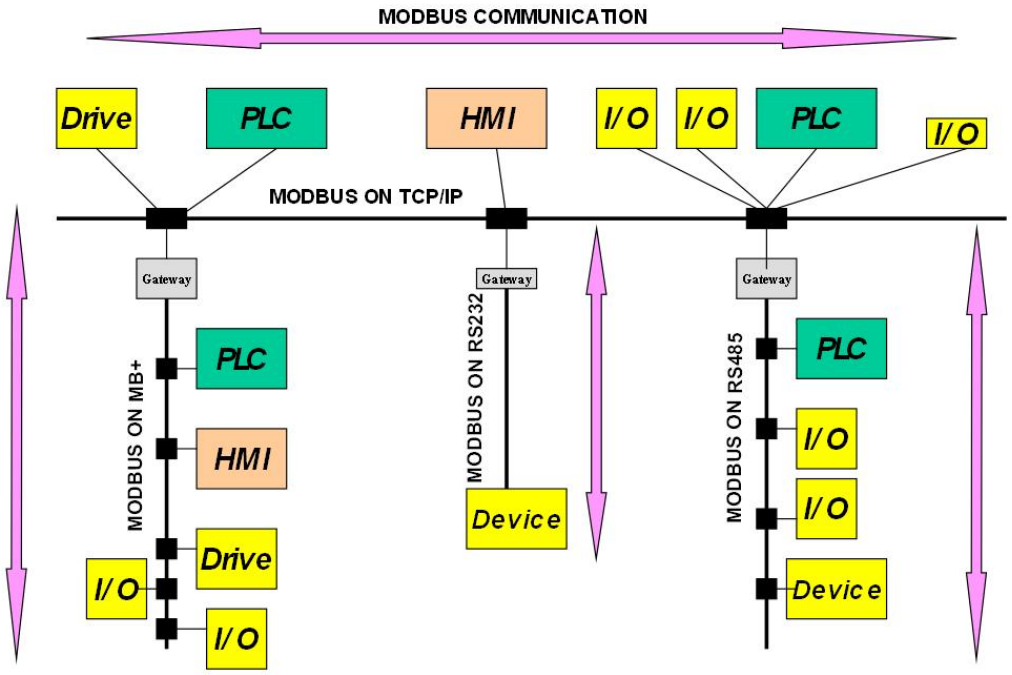
Modbusin yleisimmät fyysisen kerroksen liikennöintitavat ovat RS232, RS485 sekä Ethernet TCP/IP. Ethernet on nousemassa suosituimmaksi vaihtoehdoksi. Näiden lisäksi Modbus-protokollaa voidaan käyttää esimerkiksi valokuitu- tai langattomilla yhteyksillä. Protokolla ei määritä tiettyä fyysistä yhteyttä, vaan sitä voidaan käyttää kaikenlaisilla yhteyksillä. Sarjaliikennekäytössä Modbus on yhden isännän ja monen orjan verkko, mutta on olemassa myös Modbus+-laajennus, joka toimii multimaster-

verkkona. Modbus+-standardissa on määritelty myös fyysinen kerros. Kuviossa 4 esitellään fyysisiä kerroksia. Modbus-sarjaliikenteen yhteydessä käytetään nimityksiä isäntä (master) ja orjalaite (slave) ja Modbus TCP:tä käytettäessä käytetään nimityksiä asiakas (client) sekä palvelin (server). Master ja client ovat pyynnön tekeviä laitteita, yleensä ohjelmoitava logiikka tai näyttöpaneeli. Slave ja server ovat pyyntöön vastaavia laitteita, yleensä mitta- tai ohjainlaitteita. (Modbus 2012, 2)



KUVIO 4. Modbus-protokolla (Modbus 2012, 2)

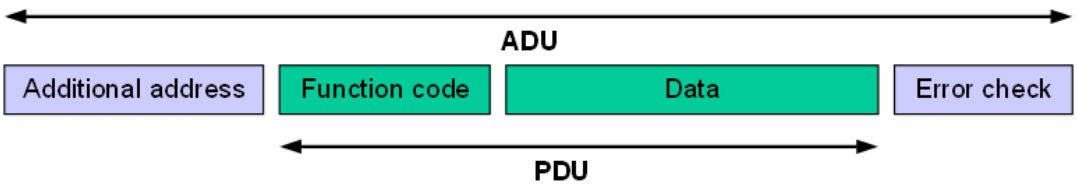
Erilaisten verkkojen toisiinsa liittäminen on helppoa. Protokollan viestikehys ei muutu, joten muuntimet hoitavat ainoastaan fyysisen kerroksen muuntamisen. (Kuvio 5)



KUVIO 5. Modbus-kommunikointi erilaisissa väylissä (Modbus 2012, 3)

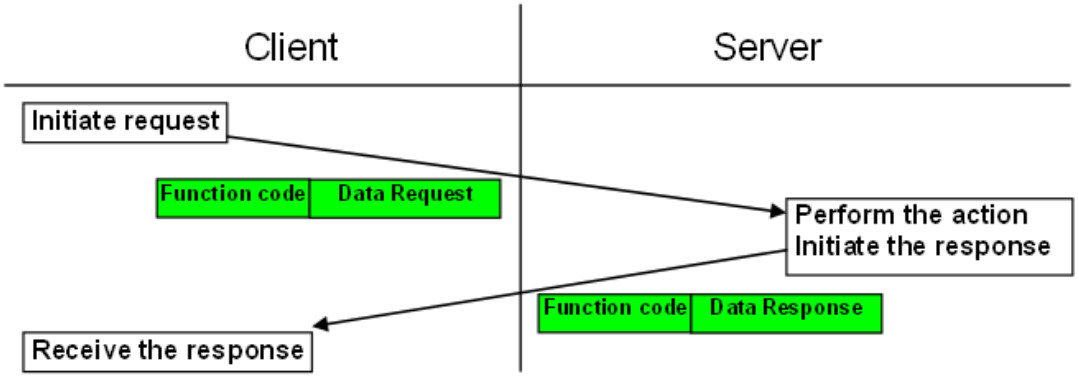
4.2 Protokollan kuvaus

Modbus-protokolla määrittelee Protocol data unitin (PDU) sekä Application data unitin (ADU) (kuvio 6). PDU on itsenäinen yhteyskerroksesta erotettu paketti, joka pitää sisällään function coden sekä datapaketin. ADU muodostuu PDU:sta sekä yhteyteen liittyvistä tiedoista. PDU:n erottaminen omaksi paketiksi mahdollistaa yksinkertaisten muuntimien käyttämisen eri verkkojen välillä. (Modbus 2012, 3)

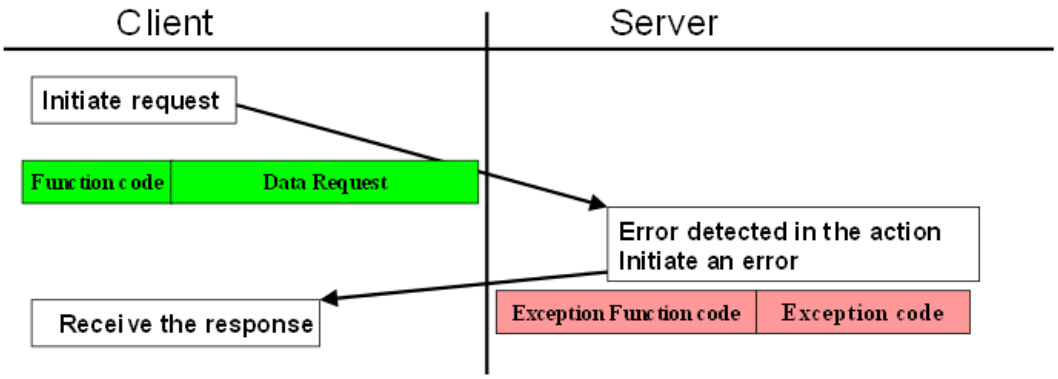


KUVIO 6. Modbus-paketti (Modbus 2012, 3)

Modbus-pyyntö tapahtuu siten, että client muodostaa ADU-paketin ja lähettää sen kenttäväylään. Server tarkistaa pyynnön ja lähettää vastauksen, mikä sisältää function coden sekä datan (Kuvio 7). Virhetilanteessa server vaihtaa function coden tilalle exeption function coden sekä lähettää vastauksen clientille (Kuvio 8). (Modbus 2012, 4)



KUVIO 7. Onnistunut Modbus-kysely (Modbus 2012, 4)



KUVIO 8. Virheellinen Modbus kysely (Modbus 2012, 4)

Modbus käyttää ’Big-Endian’ tavujärjestystä, jossa merkitsevin tavu lähetetään ensimmäisenä. Esimerkiksi:

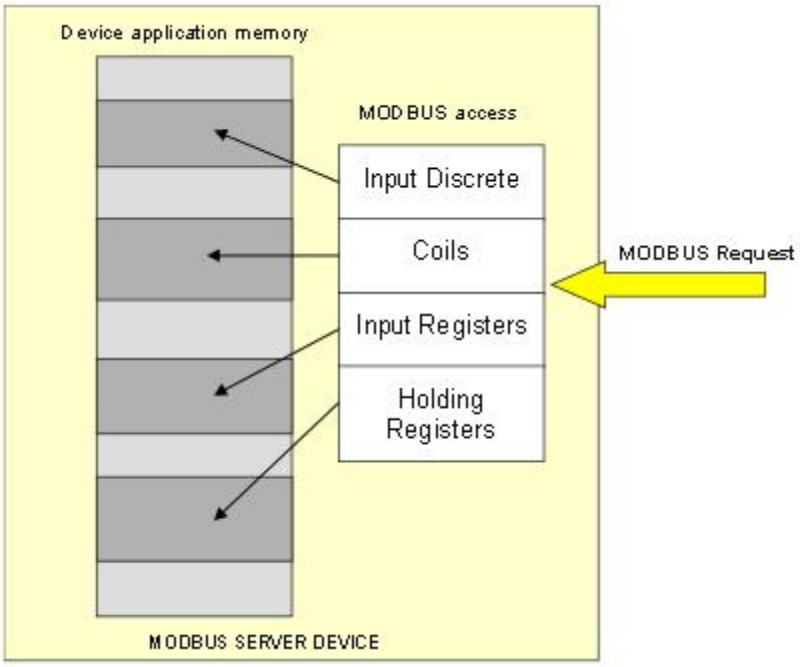
Rekisterin koko	arvo	
16-bittiä	0x1234	ensin lähetetään 0x12 toisena 0x34

Modbusissa on käytössä neljä eri tietotyyppiä. Nämä ovat discrete input, coils, input registers sekä holding registers. Kaksi ensimmäistä ovat binäärityyppisiä ja jälkimmäiset 16-bittisiä lukuja (Taulukko 3). (Modbus 2012, 5)

TAULUKKO 3. Modbus-tietotyypit (Modbus 2012, 6)

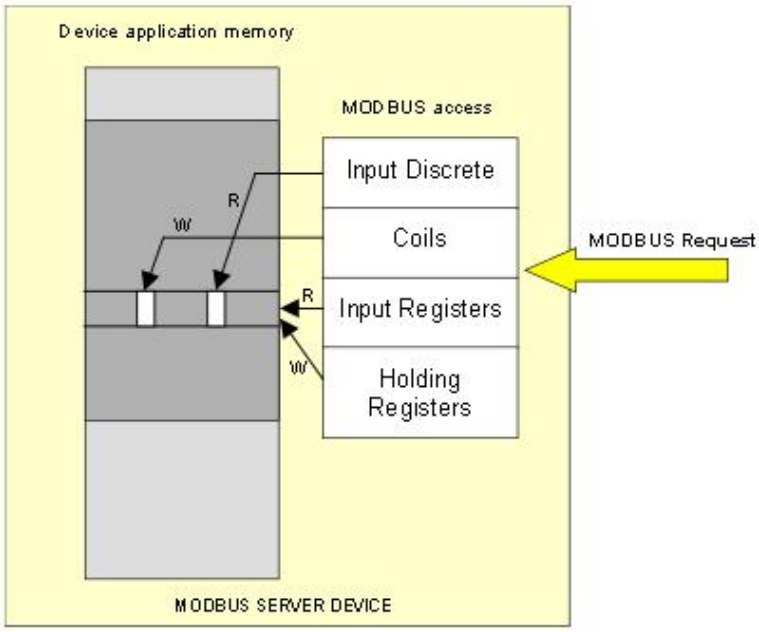
Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Jokaiselle tietotyypille löytyy omat function codensa. Modbus-standardi ei ole määritellyt laitteiden rekisterin toimintaa. Rekisteri voidaan toteuttaa siten, että jokaisella tietotyypillä on oma alueensa rekisterissä (Kuvio 9.). (Modbus 2012, 6)



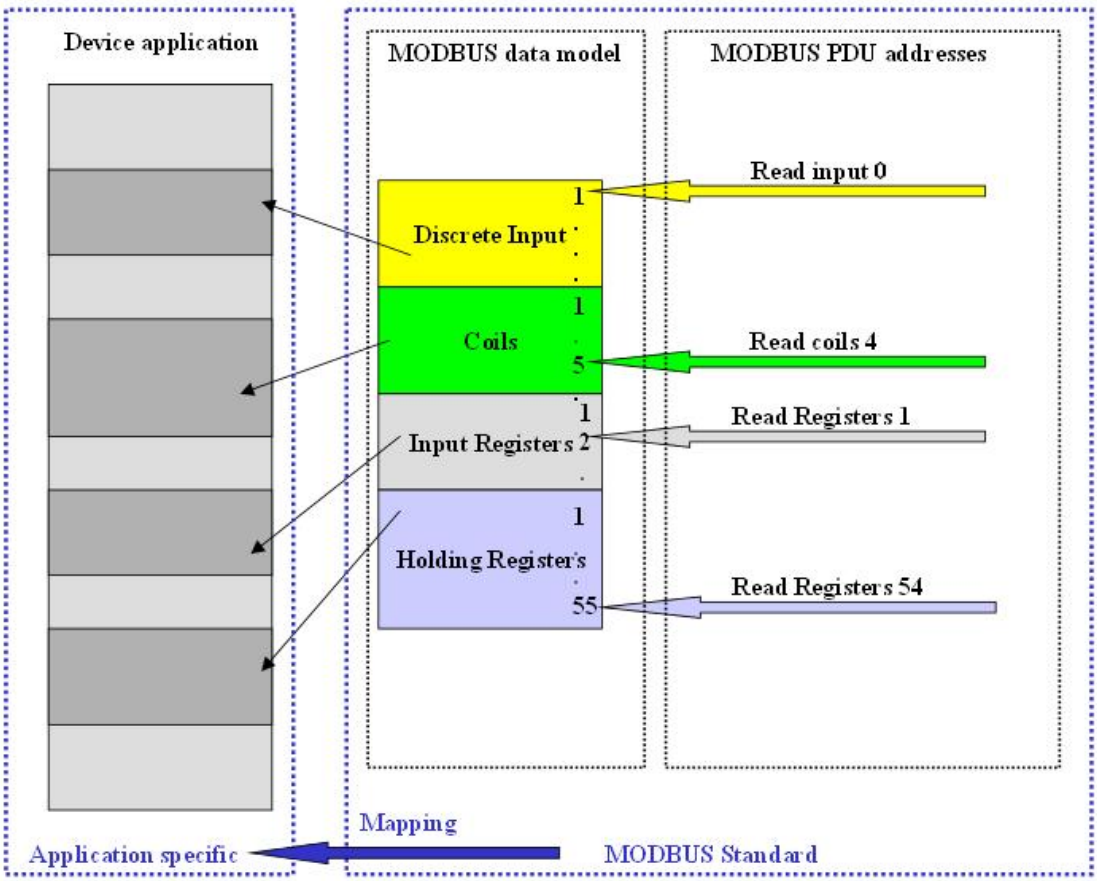
KUVIO 9. Modbus-rekisteri erillisillä lohkoilla (Modbus 2012, 6)

Vaihtoehtoisesti rekisteri voidaan myös toteuttaa yhteisellä rekisterillä (Kuvio 10.). Tämä mahdollistaa sen, että esimerkiksi yksitäisiä binääritietoja voidaan lukea 16 bitin erissä. (Modbus 2012, 7)



KUVIO 10. Modbus-rekisteri yhteisellä data-alueella (Modbus 2012, 7)

Modbus-protokollan mukaan PDU:ssa käytettävät rekisterit ovat 0 – 65535. Modbus-laittessa olevat rekisterit alkavat kuitenkin numerosta 1, joten käytössä on yhden numeron poikkeama. (Kuvio 11.) (Modbus 2012, 7)



KUVIO 11. Modbus-rekisterin poikkeama (Modbus 2012, 8)

Modbus-protokolla määrittelee kolmenlaisia function codeja. Näitä ovat yleiset koodit (public function codes) (katso Taulukko 4), käyttäjän määrittelemät koodit (user-defined function codes) sekä varatut koodit (reserved function codes). Yleiset koodit ovat hyvin määriteltyjä, dokumentoituja, Modbus.org:in hyväksymiä sekä varmuudella ainutlaatuisia. Käyttäjän määrittelemät koodit ovat laitevalmistajan omia ratkaisuja, joten niistä ei voi varmistua, että koodi olisi yksilöllinen. Varatut koodit ovat laitevalmistajien käytössä, mutta niitä ei voi käyttää julkisesti. (Modbus 2012, 10)

TAULUKKO 4. Julkiset koodit (Modbus 2012, 11)

				Function Codes		
				code	*Sub code*	*(hex)*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**
		Internal Bits Or Physical coils	Read Coils	**01**		**01**
			Write Single Coil	**05**		**05**
			Write Multiple Coils	**15**		**0F**
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**
			Write Single Register	**06**		**06**
			Write Multiple Registers	**16**		**10**
			Read/Write Multiple Registers	**23**		**17**
			Mask Write Register	**22**		**16**
			Read FIFO queue	**24**		**18**
	File record access		Read File record	**20**		**14**
			Write File record	**21**		**15**
Diagnostics			Read Exception status	**07**		**07**
			Diagnostic	**08**	**00-18,20**	**08**
			Get Com event counter	**11**		**0B**
			Get Com Event Log	**12**		**0C**
			Report Server ID	**17**		**11**
			Read device Identification	**43**	**14**	**2B**
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**
			CANopen General Reference	**43**	**13**	**2B**

Esimerkki Modbus-pyynnöstä ja -vastauksesta:

Client lähettää palvelimelle pyynnön, jolla luetaan rekisterit 108 – 110. Viesti pitää sisällään funktiokoodin, aloitusosoitteen sekä rekisterien lukumäärän. (Taulukko 5.)

TAULUKKO 5. Modbus-pyyntö (Modbus 2012, 15)

Request

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Palvelin lähettää vastauksen, joka koostuu funktiokoodista, rekisterien määrästä sekä rekistereiden arvosta. (Taulukko 6.)

TAULUKKO 6. Modbus-vastaus (Modbus 2012, 15)

Response

Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	

*N = Quantity of Registers

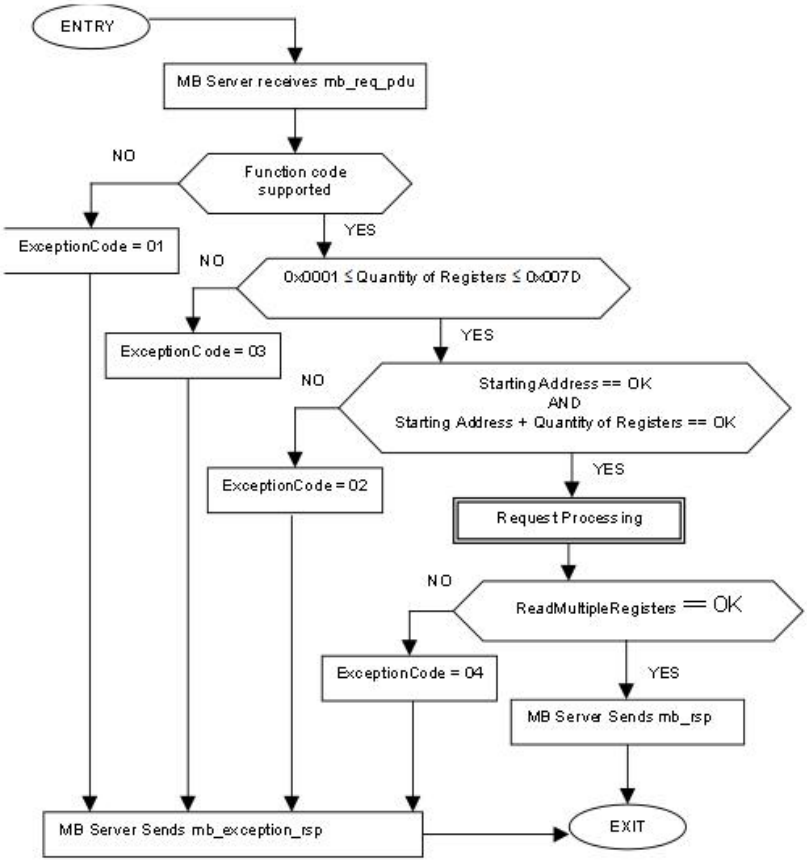
Virheen sattuessa vastaus muodostuu virhekoodista (taulukko 7.), joka on alkuperäinen funktiokoodi lisättynä 0x80 heksadesimaalia. Lisäksi viestiin tulee exception code, joka ilmaisee millainen virhe on kyseessä.

TAULUKKO 7. Modbus-virhevastaus (Modbus 2012, 15)

Error

Error code	1 Byte	**0x83**
Exception code	1 Byte	01 or 02 or 03 or 04

Virhekoodit määräytyvät seuraavan Kuvion 12. mukaan.



KUVIO 12. Modbus-virhekoodin määräytyminen (Modbus 2012, 16)

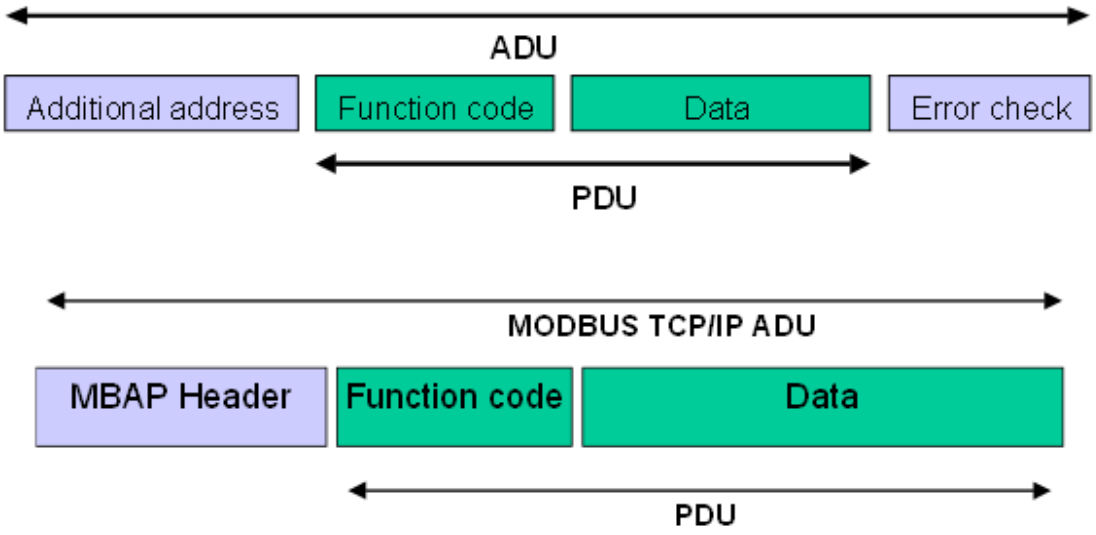
Taulukossa 8. esitellään esimerkit Modbus-pyynnöstä ja –vastauksesta, jolla luetaan rekisterien 108 – 110 arvot.

TAULUKKO 8. Modbus-kysely sekä -vastaus (Modbus 2012, 15)

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**03**	Function	**03**
Starting Address Hi	**00**	Byte Count	**06**
Starting Address Lo	**6B**	Register value Hi (108)	**02**
No. of Registers Hi	**00**	Register value Lo (108)	**2B**
No. of Registers Lo	**03**	Register value Hi (109)	**00**
		Register value Lo (109)	**00**
		Register value Hi (110)	**00**
		Register value Lo (110)	**64**

4.3 Modbus/TCP

Modbus TCP-protokollassa käytetään samaa PDU-viestiä, joka on määritelty Modbus-protokollassa. Verrattuna sarjaliikenne-Modbusiin, TCP-laajennuksessa ei käytetä virheentarkistusta eikä osoitetietoa. (Kuvio 13.) Sen sijaan Modbus TCP/IP ADU -pakettiin lisätään MDAP-otsikkolohko. MDAP pitää sisällään viestin järjestysnumeron, protokollatunnuksen, pituuden sekä laitenumeron. (Taulukko 9.) (Modbus 2006, 4)



KUVIO 13. Modbus ADU sekä Modbus TCP/IP ADU (Modbus 2006, 4)

TAULUKKO 9. Modbus TCP/IP MDAP header (Modbus 2006, 5)

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the client	Recopied by the server from the received request
Length	2 Bytes	Number of following bytes	Initialized by the client (request)	Initialized by the server (Response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the client	Recopied by the server from the received request

5 VISI-KAMERAJÄRJESTELMÄ

5.1 Visi-järjestelmä

VISI-tuoteperhe

Vision Systems Oy:n VISI-tuoteperheeseen kuuluvat konenäkökamerat, optiset mittalaitteet sekä LED-valaisimet. Kameroita ovat VISI 50 SMART -älykamera sekä VISI HT -analogikamera. Optisiin mittalaitteisiin kuuluvat VISI LT100 –katkonvalvonta-anturi sekä VISI LT200 –reunanseuranta-anturi. Lisäksi tuoteperheestä löytyvät VISI EYE SMART -integroitu konenäköjärjestelmä, joka perustuu VISI 50 SMART kameraan, VISI 3D -profiilinmittausjärjestelmä sekä VISI DMS –optinen halkaisijanmittausjärjestelmä. (Vision Systems Oy)

VISI 50 SMART

VISI 50 SMART on integroitu konenäköanturi, jota käytetään teollisissa mittaus- ja laadunvalvontasovelluksissa. Anturin vakiomittaustoimintoja ovat reunan paikan mittaus, keskeisyyden mittaus, leveyden mittaus, kappaleen keskipisteen mittaus sekä kappalemäärän laskenta. Vakiotoimintojen lisäksi kameraan voidaan vapaasti ohjelmoida asiakaskohtaisia sovelluksia. Tällaisia voivat olla esimerkiksi keskitys, värin tunnistus, muodon mittaus tai reunanohjaus. Kameran perusversiossa on mustavalkokuvakenno, jonka vaihtoehtona kameran on mahdollista valita värisuodattimella varustettu kenno. Anturissa käytetään tehokasta DSP-kuvankäsittelytekniikkaa, joka käsittelee 5 megapikselin matriisikennolla tuotettuja kuvia. Kameran tekniset tiedot esitellään taulukossa 10. (Vision Systems Oy 2012)

TAULUKKO 10. VISI 50 SMART tiedot (Vision Systems Oy 2012)

Tekniset tiedot:	
Kuvaelementti	CMOS-matriisikenno 2592 x 1944 kuvapistettä, optiona Gr R B Gb Bayer -värisuodatin
Kuvankäsittely	Sisäinen DSP TI TMS320C64
Mittausalue	Vapaasti valittavissa
Asennusetäisyys	Vapaasti valittavissa
Erottelutarkkuus	2592 x 1944 kuvapistettä
Tarkkuus	Mittausalueesta ja sovelluksesta riippuen
Suorituskyky	Yleensä 50…1000 mittausta / s (jopa yli 1000 / s)
I/O-liitännät	Ethernet 2 digitaalituloa 24 VDC 100 mA 4 digitaalilähtöa 24 VDC 400 mA Optiot: 1 analogitulo 0-10 VDC tai 4…20 mA 1 analogilähtö 0-10 VDC tai 4…20 mA Optio: ProfiBus
Syöttöjännite	24 VDC 1 A
Optiikkasovitus	C-kierre
Mitat	80 x 150 x 225 mm (ilman suojakoteloa)
Paino	1 kg (ilman suojakoteloa)
Kotelointi	IP55
Liittimet	12 pin Ø 16 mm, 4 pin Ø 8 mm, 10/8 mm paineilma
Visi50 UI -käyttöliittymäohjelma PC:lle	Windows-ohjelma käyttöönotto- ja ylläpitotoimintoihin. PC-vaatimukset: Win9x, Win2k, WinXP, Ethernet (10/100 RJ45).

Vision Systems Oy:n ratkaisut eri teollisuuden aloille.

Paperiteollisuudessa Vision Systems Oy:n konenäköratkaisuja voidaan käyttää radan laadunvalvonnassa, ratakatkojen ilmaisussa ja –valvonnassa sekä kudosten ohjauksessa. Lisäksi ratkaisuja voidaan hyödyntää myös radan reunanmittauksessa ja –ohjauksessa, rullien tunnistuksessa sekä laboratorion laatumittauksissa ja analysoinneissa. Selluteollisuuteen yritys on tehnyt AVM –kuitupuun tilavuudenmittausjärjestelmät sekä AQS – kuitupuun mittaus- ja laadutusasemat (Rahkola 2008). Rahkolan (2010) mukaan paperiteollisuudessa reunamerkin tunnistus, radan reunanseuranta sekä radan ja kudosten reunanohjaus voidaan toteuttaa VISIEDGE ja LINE-TEC-200 -ratkaisuilla sekä ratakatkoilmaisu VISIEYE ja LINE-TEC-100 -ratkaisuilla.

Terästeollisuudessa yrityksen konenäköratkaisuja voidaan käyttää aihion mittaukseen ja tunnistukseen, pään leikkauksen optimointiin, pinnanlaadun tarkastukseen sekä rullan muodon mittaukseen. Lisäksi ratkaisuja voidaan käyttää myös nauhan kolmiulotteiseen muodon mittaukseen, nauhan leveyden mittaukseen, nauhan reunanohjaukseen sekä nauhan keskitykseen (Rahkola 2008). Rahkolan (2010) mukaan metalliteollisuudessa leveyden mittaus, nauhan reunanohjaus ja keskitys sekä radan ja kudostenohjaus voidaan toteuttaa VISILINE-ratkaisuilla sekä kolmiulotteiset mittausjärjestelmät ja xy-muodonmittaus voidaan toteuttaa asiakaskohtaisilla ratkaisuilla.

Rengasteollisuudessa radan leveysmittaus, radan keskityspinnanmuodon mittaus sekä renkaan tunnistus voidaan toteuttaa yrityksen ratkaisuilla. Lasiteollisuuteen yrityksen ratkaisuja ovat tasolasin laaduntarkistus, reunavikojen valvonta, virhemerkkien tunnistaminen sekä lasilevyjen muodon mittaaminen. Muita teollisuudenaloja, joille konenäköä voidaan soveltaa, ovat mekaaninen metsäteollisuus, elektroniikkateollisuus, elintarviketeollisuus, kappaletavarateollisuus sekä lääketeollisuus (Rahkola 2010). Rahkolan (2002) mukaan muille teollisuudenaloille sopivia ratkaisuja ovat tunnistusjärjestelmät, nauhamateriaalin leveysmittaus, halkaisijan mittaus, nauhan mittaus, kappaleen käsittelyn ohjaus, 3D-mittausjärjestelmät, XY-muodonmittaukset sekä uunikamerat.

6 KÄYTÄNTÖ

6.1 Käytännön työ

Työn tekeminen aloitettiin selvittämällä Modbus TCP -protokollan toimintaa ja tutustumalla ohjelmointiympäristön sekä kameran ohjelmointiin yksinkertaisen ”Hello-world” -ohjelman avulla. Protokollasta selvisi se, että siitä on olemassa kolme eri conformance class -luokkaa. Luokan 0 laitteet tukevat vain muutamaa funktiota ja luokan 2 laitteet tukevat kaikkia protokollan funktioita. Työssä päädyttiin käyttämään luokkaa 0, koska muille funktiokutsuille ei koettu tarvetta. Lisäksi protokollan toiminnasta selvisi viestin muoto, joka selvennettiin opinnäytetyön luvussa 4.3.

Esiselvityksen jälkeen aloitettiin funktioiden ohjelmoiminen, joka tapahtui Teksas Instrumentsin CodeComposer-ohjelmointiympäristöä käyttäen. Palvelimen ohjelmoiminen aloitettiin luomalla Ethernet-socketti, joka kuuntelee porttia 502. Seuraavaksi luotiin funktio, joka kuuntelee luotua porttia ja muodostaa uuden socketin, kun asiakas haluaa luoda yhteyden. Luotua toiminnallisuutta muokattiin tukemaan kymmentä yhteyttä samanaikaisesti. Tämän jälkeen luotiin ohjelma, joka käy läpi luodut yhteydet ja sulkee toimimattomat socketit. Ohjelma tehtiin sellaiseksi, että se tarkistaa toimivista yhteyksistä onko niihin tullut viestiä. Seuraavaksi luotiin ohjelmat kummallekin funktiokutsulle, mitkä Modbus conformance class 0 on määritellyt. Näitä olivat ”lue monta rekisteriä” sekä ”kirjoita monta rekisteriä”. Viimeisenä palvelimen toimintaan lisättiin sammutuksen yhteydessä tapahtuva yhteyksien katkaisu.

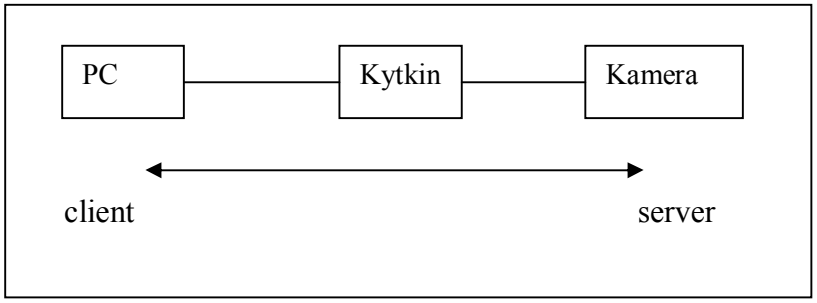
Asiakasohjelman luominen tapahtui samalla tavalla kuin palvelimen luominen. Ensiksi luotiin ohjelma, joka ottaa yhteyttä annettuun osoitteeseen ja luo yhteyden. Tämän jälkeen luotiin ohjelma, joka lähettää pyyntöjä annetuin väliajoin ja huolehtii siitä, että jos vastauksen saapuminen kestää liian kauan, ohjelma lähettää uuden pyynnön. Asia-

kasohjelmaan luotiin myös yhteyksien katkaisutoiminnot. Liitteessä 1 löytyy ohjelmien tarkemmat kuvaukset ja aliohjelmat.

Ohjelmien kääntäminen onnistui CodeComposer Studio 3.1:llä. Ohjelmasta saatiin myös ASCII-koodattu konekielinen ohjelma, joka ladattiin kameralle. Lataaminen tapahtui SSH-yhteyttä käyttämällä, jolla saatiin myös konsoliyhteys kameraan. Tuotantokäytössä olevassa kamerassa on ohjelmien automaattinen käynnistys päällä, mutta testausvaiheessa tämä toiminto otettiin pois käytöstä ja ohjelmat käynnistettiin käsin SSH-yhteyden yli. SSH-yhteysohjelmaksi olisi kelvannut mikä tahansa ohjelma, mutta työssä päädyttiin käyttämään TeraTerm -ohjelmaa, koska myös toimeksiantaja käyttää sitä.

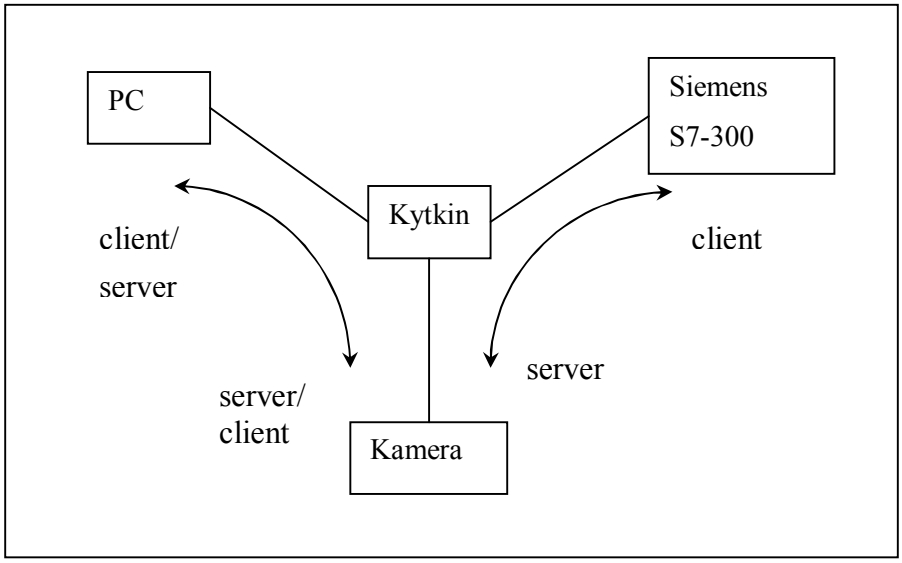
Ohjelmien testaus tapahtui kolmessa osassa. Ohjelmoinnin aikaiseen testaukseen käytettiin tietokonetta. Valmiit ohjelmat testattiin käyttäen Siemensin ohjelmoitavaa logiikkaa sekä servo-ohjainta. Testauksen eteneminen on kuvattu tarkemmin liitteessä 2

Ohjelmoinnin aikainen testaus tapahtui ensin käyttäen SocketSniff-ohjelmaa, jolla voitiin todentaa että socketti on saatu luotua. Tämän jälkeen samalla ohjelmalla varmistettiin, että viestit ovat Modbus-protokollan mukaisia. Seuraavaksi Modpoll-ohjelmalla lähetettiin kyselyjä, jolla testattiin palvelinohjelmiston toimivuutta. Ensiksi varmistuttiin funktioiden toiminnasta ja sen jälkeen suoritettiin virheellisten sanomien oikea toiminta. Diagslave-ohjelmalla testattiin asiakasohjelman toiminta. Ohjelma luo tietokoneelle Modbus-palvelimen ja näyttää konsolissa kyselyjen tiedot. Ohjelmalla voitiin varmistua asiakasohjelman toimivuudesta. Lisäksi palvelinohjelmisto testattiin SimplyModbus TCP -ohjelmalla sekä Ananas-ohjelmalla, jotta saatiin varmistus toiminnallisuuksien toiminnasta. Ananas-ohjelmalla testattiin myös asiakasohjelman toiminta. Kuviossa 14 on esitetty järjestelmäkaavio ohjelmoinnin aikaisesta järjestelmästä.



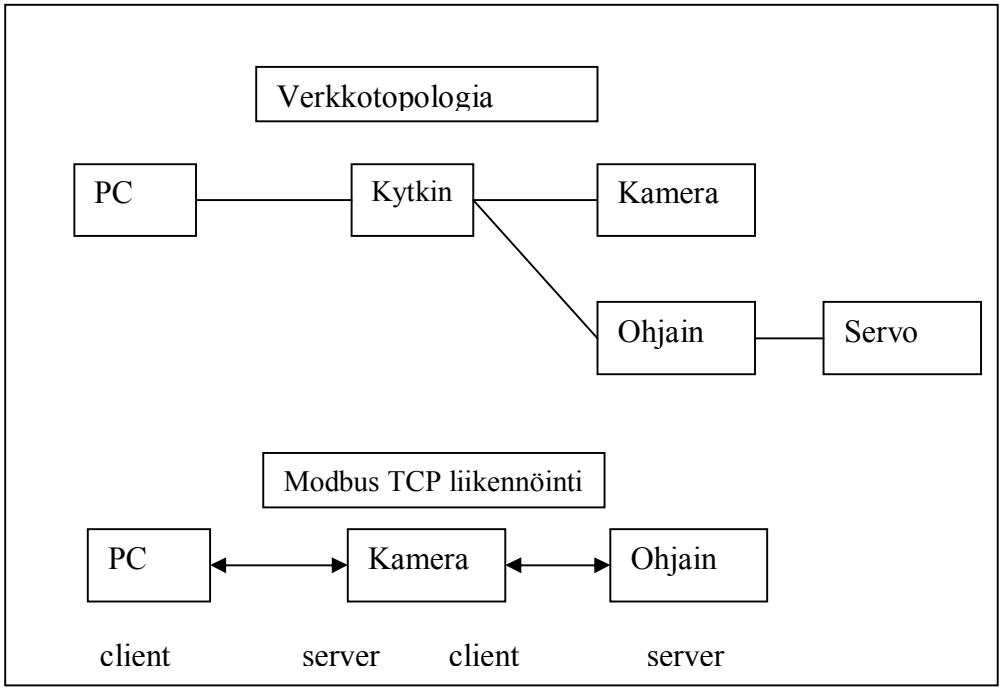
KUVIO 14. Järjestelmäkaavio ohjelmoinnin aikaisesta testauksesta

Seuraavaksi suoritettiin testaus Jyväskylän ammattikorkeakoulun laboratoriotiloissa. Siellä testauksessa käytettiin Siemens S7-300 -sarjan logiikkaa, johon oli liitetty tietoliikennepalikka. Logiikkaan tehtiin kaksi ohjelmaa, joista toinen oli palvelin ja toinen asiakasohjelma. Näitä ohjelmia ei voitu käyttää samaan aikaan, joten niitä käytettiin erikseen. Kummankin ohjelman toimivuus varmistettiin tietokoneella käyttäen edellisessä kappaleessa mainittuja ohjelmia. Kun oli saatu varmuus että ohjelmat toimivat, kokeiltiin niitä kameran avulla. Laboratorioon luotiin koejärjestely, jossa Siemensin logiikka, tietokone ja kamera oli kytketty toisiinsa kytkimen avulla. Koejärjestelyssä logiikka toimi asiakasohjelmana ja teki kirjoitus- ja lukupyyntöjä kameralle. Tämän lisäksi tietokoneella oli käynnissä kaksi eri asiakasohjelmaa, jotka suorittivat lukupyyntöjä sekä näiden lisäksi yksi ohjelma, joka kyseli virheellisiä lukupyyntöjä kameralta. Kamerassa oli asiakasohjelma samaan aikaan käytössä ja se lähetti tietokoneelle lukupyyntöjä. Koe oli käytössä muutaman tunnin ja sen jälkeen voitiin todeta, että kameran ohjelmisto toimi moitteettomasti. Kuviossa 15. on esitetty järjestelmäkaavio ohjelmoitavalla logiikalla tapahtuneesta testauksesta.



KUVIO 15. Järjestelmäkaavio PLC-testauksesta

Viimeisenä kameraa testattiin toimeksiantajan tiloissa käyttäen servo-ohjainta. Kamerassa oli käytössä palvelin sekä asiakasohjelma. Kamera suoritti luku- ja kirjoituspyyntöjä servo-ohjaimelle. Tietokoneella oli asiakasohjelma käytössä, jolla suoritettiin kamerasta kirjoitus- ja lukupyyntöjä. Tietokoneella kirjoitettiin kameran rekisteriin lukuarvo, jonka kamera lähetti servo-ohjaimelle. Tämän jälkeen ohjaimen näytöltä käytiin varmistamassa, että lukuarvo meni perille asti. Lisäksi ohjaimen näytöltä annettiin lukuja, jotka luettiin kameralle ja sieltä edelleen tietokoneelle. Kuviossa 16 esitetään järjestelmäkaavio sekä tiedon liikkuminen tietokoneen kameran sekä servon välillä.

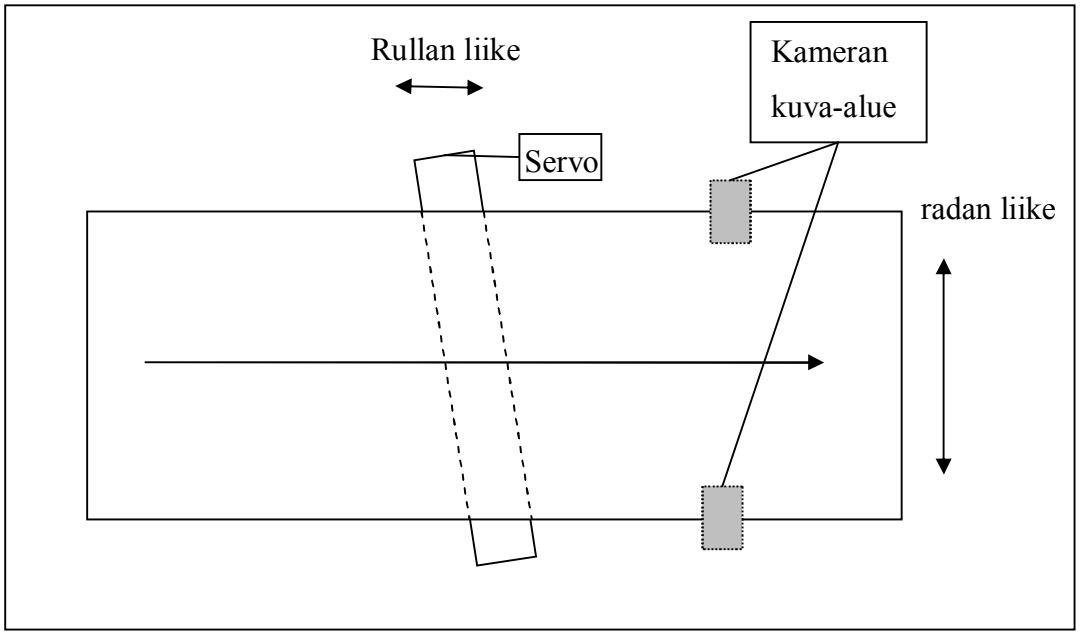


KUVIO 16. Järjestelmäkaavio servon testauksessa

6.2 Modbusin tuomat lisähyödyt ja järjestelmän vertailu väylän näkökulmasta

Opinnäytetyössä tehty toiminnallisuus mahdollistaa uusia vaihtoehtoja ohjaus- ja mittausjärjestelmän toteutukseen. Suurin eroavaisuus aikaisempaan on kamerassa oleva Modbus TCP -asiakassovellus. Tämä mahdollistaa suoran liittymisen toiseen kameraan tai laitteeseen. Tällöin ei tarvita muuta ohjausjärjestelmää, vaan esimerkiksi olemassa oleva automaatiojärjestelmä antaa vain ohjearvot joiden mukaan kamera toimii.

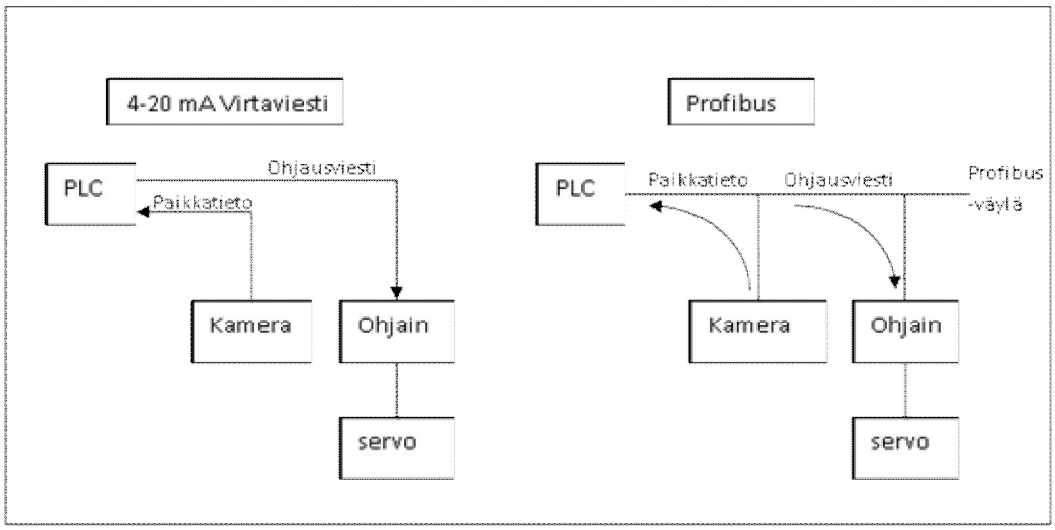
Esimerkiksi paperikoneessa paperin reunanohjaus tapahtuu servolla, joka liikuttaa paperiradan mukaan olevaa telaa. Kuvassa 17. esitetään periaatekuva radan ohjauksesta.



KUVIO 17. Periaatekuva reunan ohjauksesta

Järjestelmä on voitu toteuttaa esimerkiksi virtaviestillä tai Profibus-väylällä sekä ohjelmoitavalla logiikalla. Virtaviestillä toteutetun ratkaisun periaatekuva esitetään kuviossa 18. Toteutus vaatii kameralle sekä servo-ohjaimelle omat kaapeloinnit logiikalta. lisäksi se vaatii logiikalle tehtävää sovelluskohtaista ohjelmointia. Lisäksi ratkaisu ei tarjoa minkäänlaista vikatietoa tai tilatietoa. Kameralta pystytään vain lukemaan reunan paikkatieto ja servolle voidaan kirjoittaa vain ohjaustieto, minkä mukaan servo hake paikkansa. Kaikki ohjaus tapahtuu ohjelmoitavalla logiikalla.

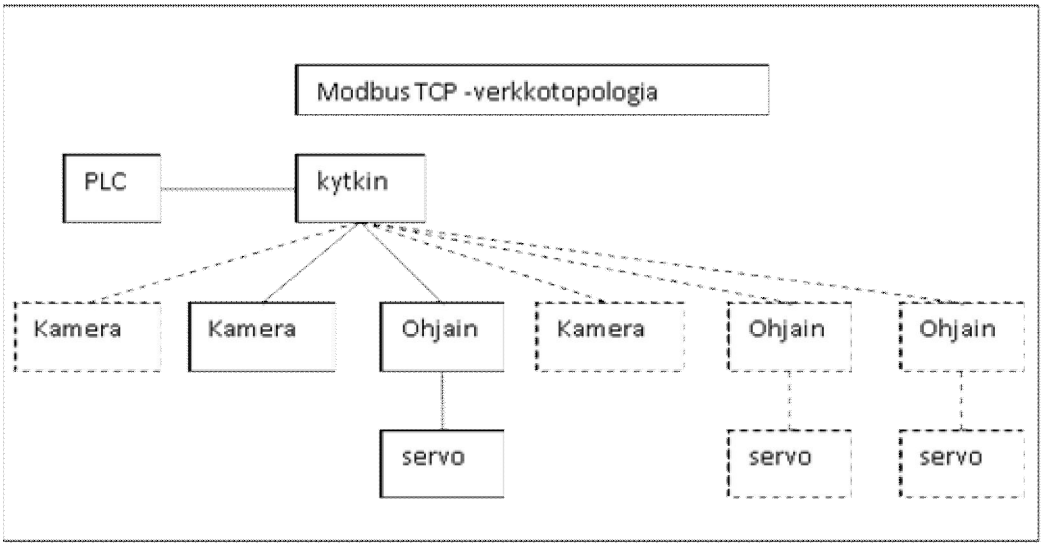
Toteutus voidaan tehdä myös jotain väylää käyttäen, esimerkiksi Profibus, jolloin saadaan kameralta ja servo-ohjaimelta paikka ja ohjaustiedon lisäksi myös vika- ja tilatietoja. Lisäksi kamerasta ja servo-ohjaimesta voidaan saada useita mittaustietoja sekä kirjoittaa useita ohjaustietoja tai asetustietoja. Ohjaus tapahtuu kuitenkin ohjelmoitavalla logiikalla.



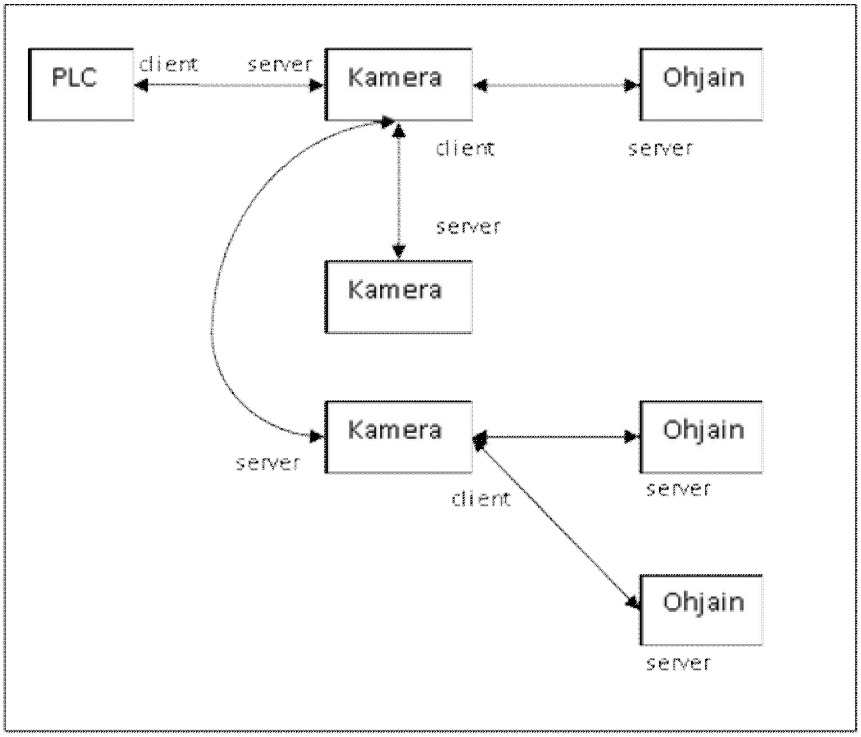
KUVIO 18. Järjestelmäkaavio, Profibus ja virtaviesti

Perinteisellä Modbus TCP -toteutuksella järjestelmä toimii samalla tavalla kuin Profibusin tapauksessa. Kameraan tehty Modbus-asiakastoiminnallisuus mahdollistaa kuitenkin logiikan poisjättämisen, koska kamera osaa ohjata servoa suoraan. Järjestelmä käyttää Ethernettiä, jonka rakenne selvitetään kuviossa 19.

Järjestelmä mahdollistaa usean kameran ja servo-ohjaimen käyttämisen helposti. Kameran Modbus-rekisteriin vakioidaan mittaustiedoille omat rekisterit, jotka mahdollistavat kameroiden välisen yhteyden tekemisen vaivattomasti. Yksi kamera voi saada olemassa olevalta logiikalta paikkatiedon, johon kamera alkaa ohjaamaan paperin reunaa. Tämä kamera pyytää muilta kameroilta mittaustiedot ja lähettää servo-ohjaimelle ohjaustiedon. Lisäksi logiikka voi lukea yhdestä kamerasta usean kameran mittaus- ja tilatiedon. Modbus TCP -asiakasohjelma mahdollistaa myös näytön sekä io-palikoiden liittämisen kameraan, joten järjestelmän vodaan toteuttaa ilman ohjelmoitavaa logiikkaa. Kuviossa 20. selvitetään tiedon kulkeminen laitteiden välillä.



KUVIO 19. Järjestelmäkaavio Modbus TCP



KUVIO 20. Tiedon liikkuminen Modbusin tapauksessa

7 POHDINTA

Työn tuloksen pohdinta

Opinnäytetyön tavoitteena oli luoda kommunikointifunktiot toimeksiantajan konenäön kamerajärjestelmään. Työn alussa oli mielenkiintoista palautella mieleen ohjelmoinnin perusteita sekä tutustua kameran ohjelmointiin. Aluksi ohjelmointi oli haastavaa ja hidasta, koska edellisestä laiteläheisestä ohjelmointityöstä oli kulunut melko paljon aikaa. Lisäksi aikaa kului paljon ohjelman toimimattomuuden selvittämiseen, jonka syyksi paljastui toimeksiantajan koodissa ollut ominaisuus. Tämän löydyttyä ohjelmointi alkoi edistyä nopeasti. Opinnäytetyön tekeminen palautti mieleen ohjelmoinnin perusteet sekä opetti paljon konenäön ja kenttäväylien toiminnasta.

Työn aloituksessa ensimmäinen iso haaste oli päästä yhteisymmärrykseen toimeksiantajan kanssa, koska toimeksiantajalla ei ollut aiempaa kokemusta Modbusista. Aluksi toimeksiantajalle täytyi selvittää Modbusin perusasiat ja kun nämä olivat selvillä, pystyttiin suunnittelemaan yhdessä mitä työltä tarkemmin vaaditaan. Opinnäytetyön tekijälle jäi päätettäväksi suurin osa pienistä yksityiskohdista.

Kun toimeksianto oli määritelty, aloitettiin ohjelmointi. Ohjelmoinnissa ilmeni jo aikaisessa vaiheessa ongelma, kun koodissa ilmeni ikuinen looppi. Kun kameraan laitettiin virta, kamera käynnisti ohjelman heti käynnistyksen jälkeen. Koska ohjelma oli jumissa, uutta ohjelmaa ei saatu ladattua. Toimeksiantajan ohjeilla koetettiin ensiksi saada automaattista käynnistystä pois päältä ottamalla yhteys mahdollisimman nopeasti kameran käynnistyttyä, jolloin kamera ohittaisi automaattisen käynnistyksen. Jonkin aikaa yritettyä ja pohdittua, menetelmä onnistui, mutta se vaati sen, että käyttöön otettiin uudempi tietokone, joka pystyi tekemään yhteyden tarpeeksi nopeasti. Tietokone, jolla kameraa ohjelmoitiin, oli tähän tarkoitukseen liian hidas. Kun ikuinen looppi ei enää mennyt päälle, voitiin ottaa automaattinen käynnistys pois päältä ja korjata ohjelma.

Seuraavana haasteena oli saada TCP-yhteys kameran ja tietokoneen välille. Kameran ohjekirjassa oli tähän ohjeet, mutta ne niiden avulla tehtynä kamera meni aina jumiin. Tehtyä koodia tarpeeksi kauan tutkittua, todettiin että jumiutumisen syy on pakko olla jossain muualla. Lisäksi huomattiin, että jos kameraan ottaa ensin toimeksiantajan konfigurointiohjelmalla yhteyden, niin jumiutumista ei tapahdu. Tästä keksittiin lisätä koodin sekaan kirjoituskäskyjä, jotka tulostavat konsoliin tekstiä. Näiden avulla paikallistettiin ongelma toimeksiantajan koodiin. Löydöistä ilmoitettiin toimeksiantajalle, joka korjasi ongelman.

Ohjelmoinnin aikaisen testauksen suurin haaste oli löytää kattava valikoima ohjelmia, joilla toiminnallisuutta pystyttäisiin testaamaan. Kaupallisia sovelluksia löytyi useita, mutta aikaa jouduttiin käyttämään vähän enemmän, jotta löydettäisiin vastaavia ilmaisia sovelluksia. Näitä löytyi tarpeeksi, jotta testaus saatiin suoritettua. Ohjelmointi sujui ilman suurempia haasteita ja ongelmia.

Toimeksiantaja halusi, että toiminnallisuus testattaisiin myös mahdollisimman monella laitteella. Suureksi haasteeksi muodostui se, että toimeksiantajalla ei ollut protokollaa tukevia laitteita. Ammattikorkeakoulun laboratoriotiloista ei ensiksi meinannut löytyä testaukseen sopivia laitteita, mutta tarpeeksi asiaa selvitettyä, löytyi laitteita joihin ohjelmallisesti oli saatavilla kyseinen toiminta. Siemens S7-300 ohjelmoitavien logiikoiden tuoteperheeseen on saatavilla Ethernet-kommunikointimoduli, johon on mahdollista ladata valmistajalta Modbus TCP -ohjelmalohko. Ohjelman sai tietyin edellytyksin ladattua ilmaiseksi, joten tästäkään ei muodostunut kustannuksia.

Työn edetessä toimeksiantajalle tuli toisen projektin kautta mahdollisuus testata ohjelmaa servo-ohjaimessa, joka tukee Modbus TCP -protokollaa.

Testauksessa suurimpana haasteena oli saada ohjelmoitava logiikka toimimaan, joka lopulta osoittautuikin kohtuullisen helpoksi työksi. Koulun laboratoriotiloissa suoritettu testaus onnistui niin hyvin, että koodiin ei tarvinnut tehdä enää minkäänlaisia muutoksia.

Viimeisenä testaus suoritettiin toimeksiantajan tiloissa olevalla servo-ohjaimella, joka pystyi kommunikoimaan Modbus TCP –väylällä. Tietokone, servo-ohjain sekä kamera kytkettiin samaan verkkoon. Testaus suoritettiin siten, että tietokoneella kirjoitettiin kameran rekistereihin ja kamera kirjoitti tiedon servo-ohjaimelle. Tämä varmistettiin lukemalla se ohjaimen näytöltä. Testauksen aikana ei ilmennyt mitään ongelmia jotka olisivat vaatineet toimenpiteitä.

Liitteessä 2. on kuvattu testauksen eteneminen sekä tarkempi kuvaus testausmenetelmistä sekä ongelmista ja niiden ratkaisuista.

Kehitysehdotukset

Opinnäytetyön tavoitteeksi otettiin Modbus TCP -kommunikoinnin toteuttaminen, johon päädyttiin sekä palvelin- että asiakasohjelman puolella. Näiden toiminta jäi kuitenkin melko suppeaksi ja seuraavia asioita voisi vielä kehittää.

Palvelinohjelmistoa tulisi kehittää siten, että siihen lisätään lippurekisteri, joka ilmaisee onko rekistereihin tullut uutta dataa. Tämän lisäksi sieltä tulisi selvitä palvelimen tila. Näitä voisivat olla esimerkiksi varataan porttia, odotetaan saapuvia yhteyksiä sekä erilaiset virhetilat. Lisäksi rekisterissä voisi olla avoimien yhteyksien lukumäärä sekä virheellisten luku- ja kirjoituspyyntöjen määrä.

Asiakasohjelmistoa tulisi kehittää siten, että yhteyksien konfigurointitiedot sekä ajonaikaiset tiedot laitetaan eri taulukoihin. Tällä hetkellä kaikki ovat samassa taulukossa ja se hankaloittaa konfigurointien tekemistä sekä mahdollistaa tietojen laittamisen vääriin paikkoihin. Tässä on sellainen vaara, että ohjelma ei toimi halutulla tavalla. Lisäksi nykyinen tilanne ei mahdollista yhdelle yhteydelle kuin yhden luku- ja kirjoitusoperaation. Jotkin laitteet eivät tue montaa samanaikaista yhteyttä, joten tiedonsiirto hidastuu huomattavasti, jos tarvitaan useita yhteyksiä. Taulukossa 11. on esitetty yksi malli, jolla voidaan useita operaatioita tehdä samalle yhteydelle. Lisäksi taulukossa on esitetty malli usean taulukon käytölle. Asiakasohjelmistolle tulisi luoda li-

säksi samankaltainen lippurekisteri kuin palvelinohjelmiston kehitysehdotuksissa on esitetty.

TAULUKKO 11. Vanha ja uusi taulukointi

Nykyinen lista

#	selitys
0	**yhteyksien määrä**
1	**ip -osoite**
2	socketin numero
3	seuraava vapaa transaction numero
4	**luettavien määrä**
5	**luettavien oma aloituspaikka**
6	**luettavien etä aloituspaikka**
7	lukupyynnon trasaction numero
8	aikaa lähetyksestä
9	**kirjoitettavien määrä**
10	**kirjoitettavien oma aloituspaikka**
11	**kirjoitettavien etä aloituspaikka**
12	kirjoittuspyynnön transaction numero
13	aikaa lähetyksestä
14	ip2.....

Parannusehdotus

Taulukko1
ip
lukuja
kirjoituksia
luku1 määrä
luku1 etä aloitusrekisteri
luku1 oma aloitusrekisteri
luku2...
kirjoitus1 määrä
kirjoitus1 etä aloitusrekisteri
kirjoitus1 oma aloitusrekisteri
kirjoitus2...

taulukko0
socket
transaction1
timer1
transaction2...

Lisäksi palvelin- ja asiakasohjelmisto käyttää samaa rekisteriä. Tässä tulisi selvittää onko järkevämpää käyttää omia rekistereitä. Tällöin voitaisiin vakioida palvelimen käyttämiin rekistereihin vakiotiedot. Esimerkki rekistereistä on esitetty taulukossa 12. Tämä mahdollistaisi tuotteen nopeamman liittämisen olemassa oleviin automaatiojärjestelmiin. Tämä mahdollistaisi myös usean kameran liittämisen toisiinsa siten, että yksi kamera lukee tarvittavat tiedot muilta kameroilta ja tekee tarvittavat säätötoimenpiteet, esimerkiksi ohjaamalla yhtä tai useampaa servo-ohjainta.

TAULUKKO 12. Esimerkkirekisteri

Rekisteri	rekisterin sisältö	selitys
3001	Mode	Tilavalinta, Esim Pakko-ohjaus/automaattiohjaus
3002	enable/disable	Ohjaus käytössä/pois käytöstä
3003	pakkoohjaus	Pakkoohjauksen ohjaustieto
3004	ohjearvo1	Ohjauksen ohjearvot
3005	ohjearvo2	
3006	DO1	ulostulojen ohjearvot
3007	DO2	
3008	AO1	
3009	AO2	
4001	mittausarvo1	Kameran mittausarvot
4002	mittausarvo2	
4003	mittausarvo3	
4004	mittausarvo4	
4005	mittausarvo5	
4006	AI1	Sisääntulojen mittausarvot
4007	AI2	
4008	DI1	
4009	DI2	
4010	tila	Tilatieto, disable/enable/error/…
4011	vikakoodi	vian tyyppi

Modbus TCP -asiakasohjelman käyttö mahdollistaa myös väylään liitettävien io-palikoiden sekä näyttöpaneelien käytön. Tämä mahdollistaa järjestelmän tekemisen ilman ohjelmoitavaa logiikkaa. Viimeisenä parannusehdotuksena voisi lisätä toimeksiantajan nykyiseen konfigurointiohjelmistoon Modbus TCP -palvelin ja asiakasohjelmiston tarvittavat tiedot. Tämä mahdollistaisi väylän nopean konfiguroinnin.

LÄHTEET

Baumgartner, H. 2009. Profibus ja sercos II –kenttäväylien nopeusvertailu. Kandidaatintyö. Lappeenrannan teknillinen yliopisto, Teknillinen tiedekunta. Viitattu 11.3.2013. http://www.doria.fi/bitstream/handle/10024/59485/nbnfi-fe201003221535.pdf

Halinen, M. 2007. Konenäkö robotin ohjauksessa. www dokumentti. Viitattu 6.3.2013. http://automation.tkk.fi/attach/AS-0-2230/lab3c_teoria.pdf

Koivisto, H. n.d. Automaation digitaaliset kenttäväylät: Industrial Ethernet. Tampereen teknillinen yliopisto, automaatio ja säätötekniikan laitos. www dokumentti. Viitattu 11.3.2013.
http://www.ac.tut.fi/aci/courses/7601000/pdf/Averkot_2_industrial_3p.pdf

Modbus. 2012. MODBUS Application ProtocolSpecification V1.1b3. www-dokumentti. Viitattu 12.3.2013.
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Modbus. 2006. MODBUS Messaging on TCP/IP implementationguide V1.0b. www-dokumentti. Viitattu 12.3.2013.
http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

Niemi, P. n.d. Koneäkö – lyhyt oppimäärä. www dokumentti. Viitattu 6.3.2013.
http://www.orbis.fi/konenako-lyhyt-oppimaara

Pietikäinen, M. Silven, O. 2008. Oulun Yliopisto. www dokumentti. Viitattu 7.3.2013
http://www.cse.oulu.fi/CMV/AboutCMV?action=AttachFile&do=view&target=konenako.pdf

Rahkola, K. 2008. Konenäkö prosessiteollisuudessa. Luentomateriaali. SKS Vision Systems Oy.

Rahkola, K.2010. Konenäköratkaisut metalli- ja paperiteollisuudessa. Luentomateriaali. SKS Vision Systems Oy.

Soini, A. 2011. Konenäkö. Suomen automaatioseura ry. www dokumentti. Viitattu 7.3.2013. http://www.automaatioseura.fi/index/tiedostot/Konenako.pdf

Valaistus. n.d. Orbis Oy. www dokumentti. Viitattu 7.3.2013.
http://www.orbis.fi/valaistus

Vainio, M., Hakala, M. 2005. Tietoverkon rakentaminen. 2.p., Docendo. E-kirja.

Vision Systems Oy. www-dokumentti. Viitattu 11.3.2013.
http://www.visionsystems.fi/

Vision Systems Oy. 2012. VISI50 SMART integroitu konenäköanturi. www-dokumentti. Viitattu 25.3.2013.
http://www.visionsystems.fi/uploads/esitteet/fin/Vision_visi50_smart_suomi.pdf

LIITTEET

Liite 1. Aliohjelmat ja niiden lyhyt kuvaus

modbus_server()

Modbus-palvelimen aliohjelma, jota kutsutaan jokaisella ohjelmakierroksella. Ohjelma hoitaa socketin varauksen, uuden yhteyden muodostamisen sekä toimimattomien yhteyksien katkaisun. Lisäksi protokollan vaatima toiminnallisuus on toteutettu tässä ohjelmassa. Ohjelmalle annetaan kaksi taulukkoa, rekisterinä toimiva taulukko sekä sockettien varastoimisena toimiva taulukko.

Ohjelma toimii siten, että kun palvelinportti on avattu, odotetaan saapuvia yhteyksiä. Kun yhteys saapuu, luodaan siitä socketti ja lisätään taulukkoon.

Jokaisella ohjelmakierroksella tarkistetaan taulukossa olevien yhteyksien tila. Jos yhteys on katkennut, poistetaan se taulukosta ja jos pyyntöjä on tullut, lähetetään vastaus.

new_modbus_server()

Aliohjelma, joka luo socketin, jota kutsutaan modbus_server aliohjelmasta. Ohjelma hoitaa socketin luomisen sekä portin aukaisun palvelimelle.

modbus_client()

Ohjelma hoitaa asiakasyhteyksien hallinnan. Ohjelmaan määritellään Modbus-rekisteri taulukkomuodossa sekä asetukset, jotka ovat taulukossa. Ohjelma tarkistaa taulukossa olevat luku- ja kirjoituspyynnöt ja luo new_client aliohjelmalla yhteyden oikeaan ip-osoitteeseen. Lisäksi se hoitaa yhteyden katkaisun, mikäli siinä on jotain ongelmia. Ohjelma myös hallinnoi lähetettyjä pyyntöjä ja lähettää pyynnöt uudelleen mikäli vastausta ei ole tullut määriteltynä aikana.

new_client()

Aliohjelmaa kutsutaan modbus_client -funktiosta. Sille määritellään osoite ja socketin muistipaikka. Se muodostaa yhteyden annettuun osoitteeseen ja yhteyden muodostuttua tallentaa socketin annettuun paikkaan.

w_f_mb()

Aliohjelma, joka muuttaa 32 bittisen liukuluvun kahdeksi 16 bittiseksi lukujonoksi ja kirjoittaa sen kahteen peräkkäiseen Modbus-rekisteriin.

w_i_mb()

Aliohjelma, joka muuttaa 32 bittisen integerin kahdeksi 16 bittiseksi lukujonoksi ja kirjoittaa sen kahteen peräkkäiseen Modbus-rekisteriin.

r_f_mb()

Aliohjelma, joka lukee kaksi Modbus-rekisteriä ja palauttaa reaalilukuarvon.

r_i_mb()

Aliohjelma, joka lukee kaksi Modbus-rekisteriä ja palauttaa integerin.

clean_list()

Aliohjelma, joka lukee annetun listan ja järjestää sen siten, että siinä ei ole nollarivejä keskellä.

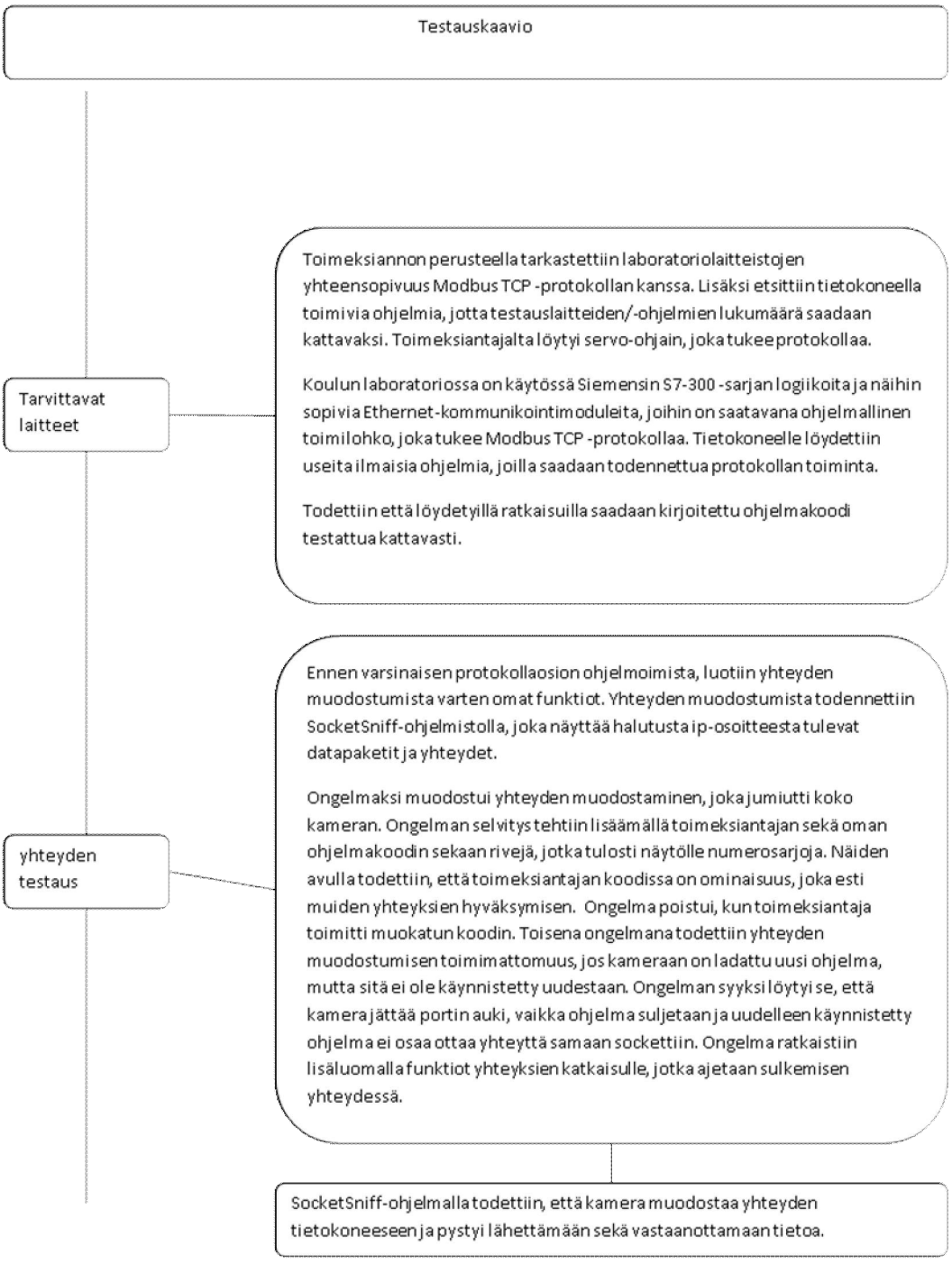
modbus_server_close()

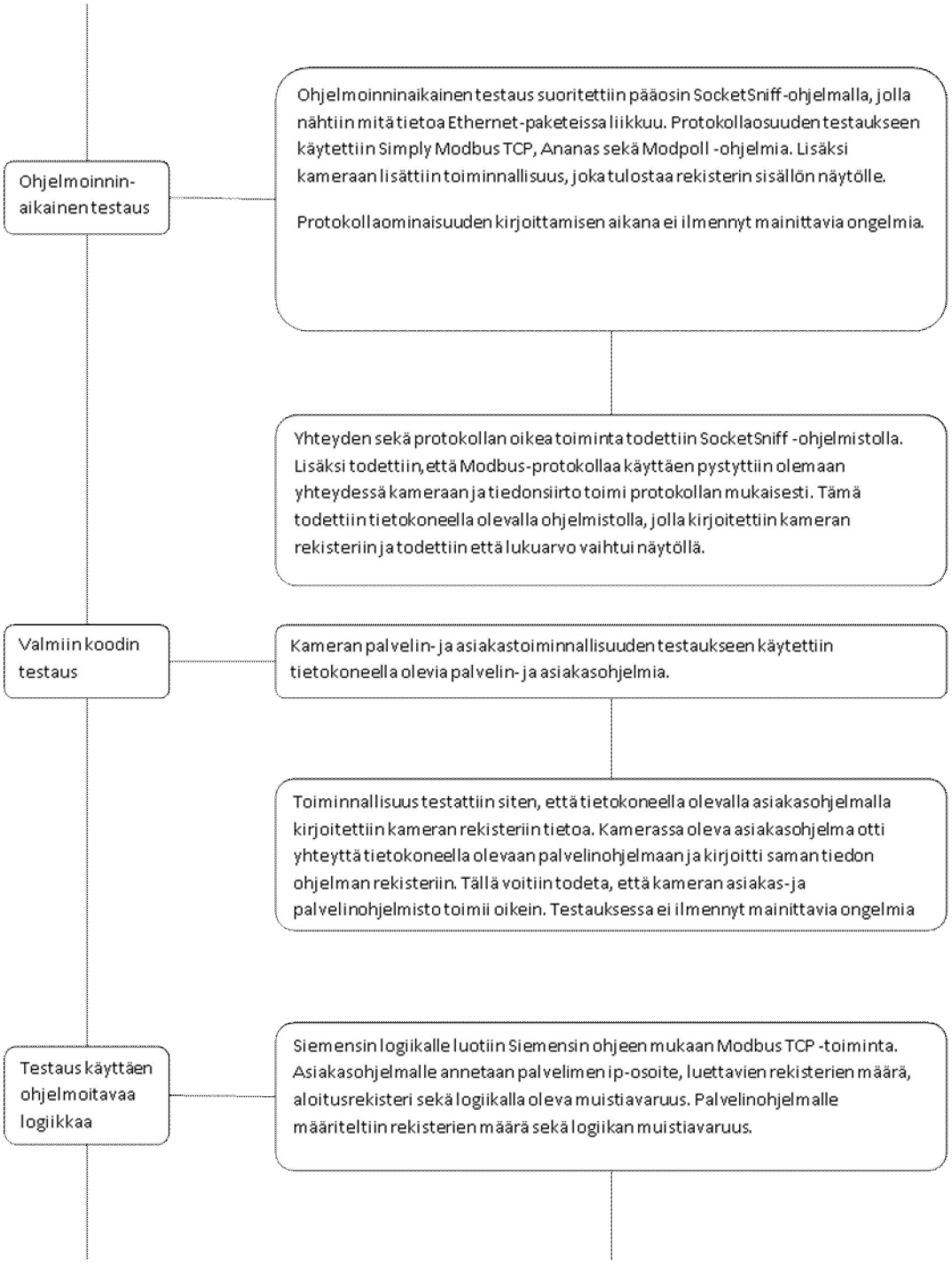
Aliohjelma, joka ajetaan ohjelman lopetusrutiineissa. Sitä kutsutaan, kun halutaan sulkea kommunikointiohjelma, mutta ei käynnistetä kameraa uudelleen. Mikäli yhteyksiä ei suljeta, palvelinaliohjelma ei pysty aukaisemaan tarvittavaa porttia.

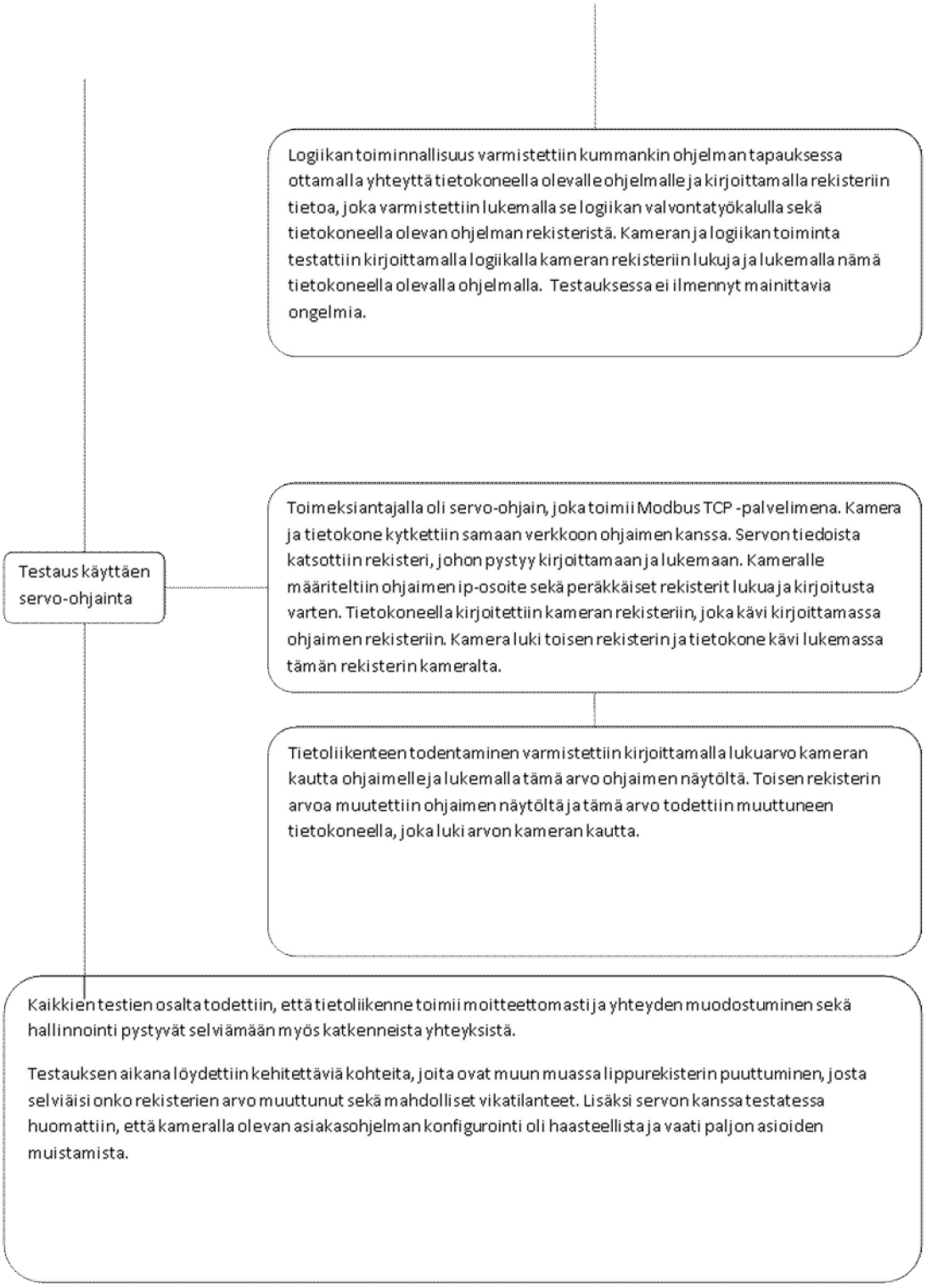
modbus_client_close()

Aliohjelma, joka ajetaan ohjelman lopetusrutiineissa. Sitä kutsutaan, kun halutaan sulkea kommunikointiohjelma, mutta ei käynnistetä kameraa uudelleen.

Liite 2. Testauskaavio




Ohjelmoinnin-aikainen testaus
Ohjelmoinninaikainen testaus suoritettiin pääosin SocketSniff-ohjelmalla, jolla nähtiin mitä tietoa Ethernet-paketeissa liikkuu. Protokollaosuuden testaukseen käytettiin Simply Modbus TCP, Ananas sekä Modpoll -ohjelmia. Lisäksi kameraan lisättiin toiminnallisuus, joka tulostaa rekisterin sisällön näytölle.
Protokollaominaisuuden kirjoittamisen aikana ei ilmennyt mainittavia ongelmia.
Yhteyden sekä protokollan oikea toiminta todettiin SocketSniff -ohjelmistolla. Lisäksi todettiin, että Modbus-protokollaa käyttäen pystyttiin olemaan yhteydessä kameraan ja tiedonsiirto toimi protokollan mukaisesti. Tämä todettiin tietokoneella olevalla ohjelmistolla, jolla kirjoitettiin kameran rekisteriin ja todettiin että lukuarvo vaihtui näytöllä.
Valmiin koodin testaus
Kameran palvelin- ja asiakastoiminnallisuuden testaukseen käytettiin tietokoneella olevia palvelin- ja asiakasohjelmia.
Toiminnallisuus testattiin siten, että tietokoneella olevalla asiakasohjelmalla kirjoitettiin kameran rekisteriin tietoa. Kamerassa oleva asiakasohjelma otti yhteyttä tietokoneella olevaan palvelinohjelmaan ja kirjoitti saman tiedon ohjelman rekisteriin. Tällä voitiin todeta, että kameran asiakas-ja palvelinohjelmisto toimii oikein. Testauksessa ei ilmennyt mainittavia ongelmia
Testaus käyttäen ohjelmoitavaa logiikkaa
Siemensin logiikalle luotiin Siemensin ohjeen mukaan Modbus TCP -toiminta. Asiakasohjelmalle annetaan palvelimen ip-osoite, luettavien rekisterien määrä, aloitusrekisteri sekä logiikalla oleva muistiavaruus. Palvelinohjelmalle määriteltiin rekisterien määrä sekä logiikan muistiavaruus.


Logiikan toiminnallisuus varmistettiin kummankin ohjelman tapauksessa ottamalla yhteyttä tietokoneella olevalle ohjelmalle ja kirjoittamalla rekisteriin tietoa, joka varmistettiin lukemalla se logiikan valvontatyökalulla sekä tietokoneella olevan ohjelman rekisteristä. Kameran ja logiikan toiminta testattiin kirjoittamalla logiikalla kameran rekisteriin lukuja ja lukemalla nämä tietokoneella olevalla ohjelmalla. Testauksessa ei ilmennyt mainittavia ongelmia.
Testaus käyttäen servo-ohjainta
Toimeksiantajalla oli servo-ohjain, joka toimii Modbus TCP -palvelimena. Kamera ja tietokone kytkettiin samaan verkkoon ohjaimen kanssa. Servon tiedoista katsottiin rekisteri, johon pystyy kirjoittamaan ja lukemaan. Kameralle määriteltiin ohjaimen ip-osoite sekä peräkkäiset rekisterit lukua ja kirjoitusta varten. Tietokoneella kirjoitettiin kameran rekisteriin, joka kävi kirjoittamassa ohjaimen rekisteriin. Kamera luki toisen rekisterin ja tietokone kävi lukemassa tämän rekisterin kameralta.
Tietoliikenteen todentaminen varmistettiin kirjoittamalla lukuarvo kameran kautta ohjaimelle ja lukemalla tämä arvo ohjaimen näytöltä. Toisen rekisterin arvoa muutettiin ohjaimen näytöltä ja tämä arvo todettiin muuttuneen tietokoneella, joka luki arvon kameran kautta.
Kaikkien testien osalta todettiin, että tietoliikenne toimii moitteettomasti ja yhteyden muodostuminen sekä hallinnointi pystyvät selviämään myös katkenneista yhteyksistä.
Testauksen aikana löydettiin kehitettäviä kohteita, joita ovat muun muassa lippurekisterin puuttuminen, josta selviäisi onko rekisterien arvo muuttunut sekä mahdolliset vikatilanteet. Lisäksi servon kanssa testatessa huomattiin, että kameralla olevan asiakasohjelman konfigurointi oli haasteellista ja vaati paljon asioiden muistamista.