Tampereen yliopisto

Nuutti Mikkonen

VENTTIILIN SÄÄDÖN TOTEUTUS MIKROKONTROLLERIPOHJAISEEN OHJAUSKESKUKSEEN

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Tarkastaja: Yliopistonlehtori Erja Sipilä
Huhtikuu 2020

TIIVISTELMÄ

Nuutti Mikkonen: Venttiilin säädön toteutus mikrokontrolleripohjaiseen ohjauskeskukseen
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan TkK-tutkinto-ohjelma
Huhtikuu 2020

Prosessisäädin on olennainen suurien tehtaiden ohjauksesta pieniin elektronisiin säätöihin, kuten moottorien asennon ohjaukseen ja aurinkopaneelien kuormituksen asetukseen. Säädin voidaan toteuttaa usean mallisena, joista kenties yleisin on PID-säädin, jonka suunnittelu ja säätäminen voi olla yksinkertaista, kun ohjattavana on tunnettu järjestelmä.

Tämän työn tarkoitus oli suunnitella ja toteuttaa uusi säädin kiertovoitelussa käytetylle jäähdytysvesiventtiilille. Aiemmin käytetyn venttiilin säädin oli sisäänrakennettu venttiilin elektroniikan logiikkaan, jolloin sen asetusten muuttaminen ja manuaalinen etäkäyttö ei välttämättä ollut mahdollista samassa paikassa kuin missä ohjauskeskus sijaitsi.

Tiedonsiirtoon työssä käytettiin virtaviestiä, joka muodostettiin valitulla lisä-IO -moduulilla. Virtaviesti on yleinen teollisuudessa käytetty viestintätapa, joka sopii hyvin häiriöllisiin ympäristöihin. Moduulille ohjauskeskukselta tiedonsiirto toteutettiin MODBUSilla RS-485:n yli. Nämä osat järjestelmästä oli valittu jo ennen työn aloitusta ja osittain ne vaativat vain muutoksia jo olemassa oleviin menetelmiin.

Säätimestä toteutettiin kaksi eri versiota eri diskretointimenetelmiä käyttäen. Mittaustulosten perusteella näistä valittiin paremmin ideaalista säädintä mukaileva versio, jotta säätöparametrien asetus toimisi mahdollisimman samanlaisella tavalla kuin aiemmalla säätimellä.

Lopullinen säädin ja sen toiminnallisuus vastasivat tavoitteita, jotka oli asetettu työtä aloittaessa. Säätimellä pystyttiin korvaamaan aiemmin käytetyn venttiilin sisäinen säädin muuttamatta toiminnallisuutta merkittävästi. Koska säädin toteutettiin täysin ohjelmistossa, sitä pystytään myös käyttämään muissa sovelluksissa, joissa myös tiedonsiirtomenetelmät voivat muuttua. Toiminnallisuus pysyy kuitenkin oletetunlaisena, hyvin tunnettuna PID-säätimenä.

Avainsanat: PID, MODBUS, RS-485

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ASCII	engl. American Standard Code for Information Interchange, tietokoneiden merkistötyyppi
AD	Analogia-Digitaali
baud	Symbolinopeus tiedonsiirrossa
CRC	engl. Cyclic Redundancy Check, tarkisteavaimen tiivistealgoritmi
DA	Digitaali-Analogia
EIA	engl. Electronic Industries Alliance, entinen elektroniikkastandardeja määritellyt järjestö
IO	engl. Input-Output, sisäänmeno-ulostulo
MSB	engl. Most Significant Bit, merkittävin bitti
OSI	engl. Open Systems Interconnection, avoin järjestelmien liittymä
PID	engl. Proportional-Integral-Derivative, säädintyyppi
PLC	engl. Programmable Logic Controller, ohjelmoitava logiikkaohjain
RTU	engl. Remote Terminal Unit, etäterminaaliyksikkö
U.L.	engl. Unit Load, yksikkökuorma
TCP/IP	engl. Transmission Control Protocol / Internet Protocol, tiedonsiirtoprotokolla
USART	engl. Universal Synchronous-Asynchronous Receiver-Transmitter, universaali synkroninen-asynkroninen lähetin-vastaanotin
XOR	engl. Exclusive OR, poissulkeva looginen TAI
2DOF	engl. 2 Degrees-Of-Freedom, 2 vapausastetta

1. JOHDANTO

Työn tarkoituksena oli toteuttaa kiertovoitelunohjauskeskukseen uusi jäähdytysveden venttiilin säätö. Aiemmassa venttiilissä säädin oli sisäänrakennettu venttiilin ohjauslogiikkaan, joten ohjauskeskus ei pystynyt vaikuttamaan sen toimintaan. Tarkoituksena oli sisällyttää säädin ohjauskeskuksen ohjelmistoon, jotta sen säätö ja etäkäyttö helpottuisi. Säätimeksi valittiin PID-säädin (engl. Proportional-Integral-Derivative), sillä se on yleisesti tunnettu ja suhteellisen helppo virittää stabiiliksi. Lisäksi aiemmassa venttiilissä säätö on toteutettu PID:llä, joten säätöparametrien valinta voidaan toteuttaa samalla tavalla kuin aiemminkin.

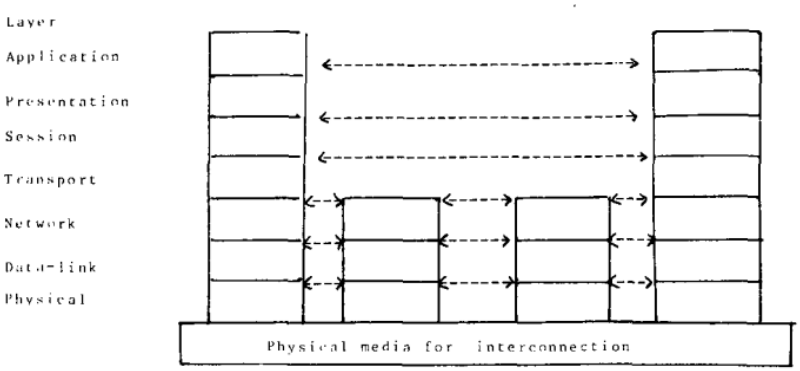
Etuna venttiilin ohjauksen toteutukselle ohjauskeskukseen on säätimen parametrien asetus fyysisesti samassa paikassa kuin missä muukin ohjauksen parametrien säätäminen tapahtuu, sillä itse venttiili ei aina ole lähellä ohjauskeskusta. Lisäksi venttiiliä voidaan näin ohjata poikkeustapauksissa muutenkin kuin PID-säätimen avulla, esimerkiksi kokonaan sulkemalla putkien vuotaessa.

Ohjauskeskuksessa ei ollut vapaita digitaalisia tai analogisia lähtöjä, mutta jo toteutettuun MODBUS-kenttäväylään oli mahdollista liittää lisä-IO -moduuli, minkä ansiosta elektroniikkaa ei tarvinnut muuttaa työtä varten, vaan ohjauksen pystyi toteuttamaan ohjelmistossa.

Tämän työn toisessa ja kolmannessa luvussa tutustutaan työssä tarvittavaan teoriaan tiedonsiirrosta ja säätimestä. Luvussa neljä käydään läpi käytetty laitteisto, toteutus ja luvussa viisi siihen liittyvät mittaukset ja lopuksi tehdään yhteenveto työstä.

2. TIEDONSIIRTO

Tiedonsiirto on olennainen osa monia järjestelmiä. Tärkeitä tiedonsiirtomenetelmää kuvaavia ominaisuuksia ovat muun muassa onko signaali analoginen vai digitaalinen, onko menetelmä langaton vai käyttääkö se väliainetta. Lisäksi on olennaista, onko yhteys yksi- vai kaksisuuntainen, jolloin puhutaan half-duplexista, jos vain toinen laite voi lähettää viestiä samaan aikaan, tai full-duplexista, jos laitteet lähettävät ja vastaanottavat samaan aikaan. Yleinen tapa vertailla menetelmiä on käyttää kuvan 1 mukaista seitsemänkerroksista OSI-mallia (engl. Open Systems Interconnection) [1, s.1334]. Tässä luvussa perehdytään työssä käytettyihin protokolliin, jotka ovat mallin ensimmäisellä, toisella tai seitsemännellä kerroksella.



Kuva 1. *OSI-mallin 7 kerrosta [1, s.1338]*

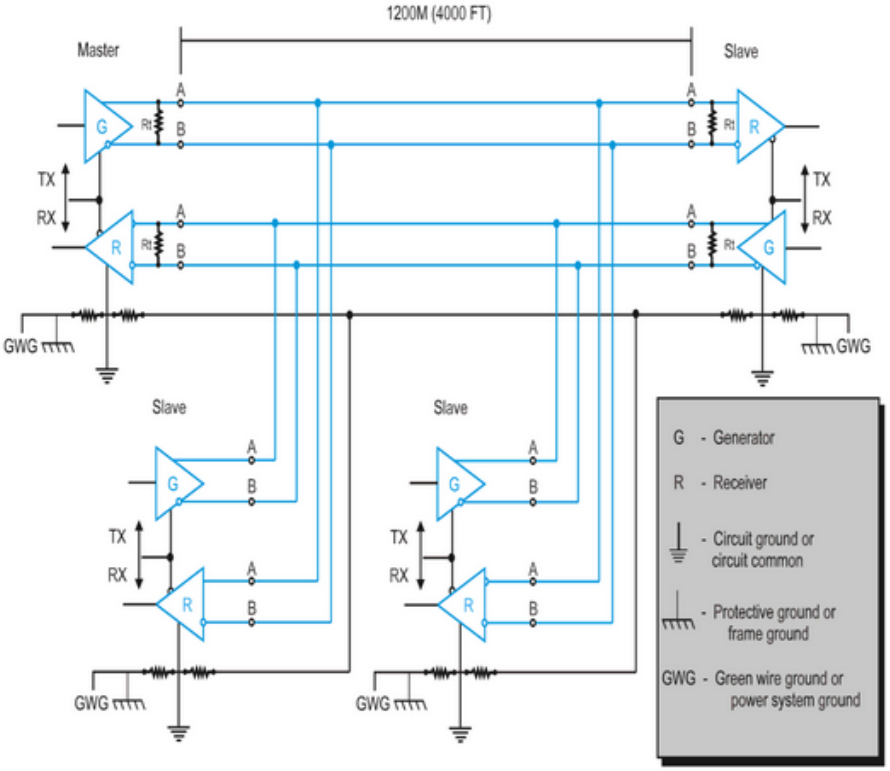
OSI-mallin kerrokset tarjoavat palveluita korkeammalle kerrokselle laitteen sisällä. Jokainen taso pystyy kommunikoimaan ulkoisten laitteiden kanssa vain omalla tasollaan. Ensimmäisellä, fyysisellä kerroksella sijaitsevat tiedonsiirtomenetelmät ovat mekaanisia, sähköisiä ja funktionaalisia standardeja, kuten johtojen ja liittimien määrittelyitä. Toisen kerroksen, siirtokerroksen, protokollat siirtävät dataa verkon laitteiden välillä, joissain tapauksissa havaiten myös virheitä. Fyysisen ja siirtokerroksen välinen raja ei aina ole selvä, ja sama protokolla voi hoitaa molempien tehtävää. [1, s.1339]

Mallin korkein taso, sovellustaso, ei tarjoa palveluita muille kerroksille. Tällöin data on muodossa, jossa sitä tarvitseva sovellus pystyy sitä tulkitsemaan. [1, s.1338]

2.1 RS-485

TIA/EIA-485, yleisesti RS-485, on OSI-mallin fyysisen kerroksen protokolla. Se on EIA:n (engl. Electronic Industries Alliance) kehittämä paranneltu versio 70-luvulla julkaistusta RS-422 standardista, jonka tavoin käytössä on differentiaalinen johtokoodaus enintään

1200 m:n etäisyyksille. RS-485 tukee samassa verkossa enintään 32:ta laitetta, joiden impedanssi on 12 kΩ. Tiedonsiirtonopeudet riippuvat käytetyistä etäisyyksistä. Yleisesti tiedonsiirtonopeuden kerrottuna kaapelin pituudella pitäisi pysyä alle 108 mMbps, ja lyhyillä etäisyyksillä se pystyy jopa 10 Mbps nopeuksiin. RS-485-lähetinvastaanottimia voidaan monesti käyttää myös RS-422:n kanssa. [2, s.79-80][3, s.62] Saman verkon laitteet kytketään toisiinsa kuvan 2 mukaisesti.



Kuva 2. *RS-485-laitteiden verkko [3, s.64]*

RS-485 on aina differentiaalinen, siis symboli tulkitaan kahden johdon jännitteiden erotuksesta, ei jännite-erosta maata vasten. Se voi kuitenkin käyttää yhtä tai kahta differentiaaliparia. Kahdella parilla lähetin ja vastaanotin käyttävät omia parejaan, yksiparisena lähetin ja vastaanotin yhdistetään rinnan. Differentiaalipari kestää paremmin häiriötä kuin maareferenssiä käyttävä johto, sillä yhteismuotoinen häiriö ei näy vastaanottimella, mutta vaatii useampia johtoja ja erilaisia ajuripiirejä. [2, s.80]

Standardi ei määrittele käytettyjä johtoja, mutta yleisesti käytetään kierrettyä paria eromuotoisen häiriön vähentämiseksi. Vaikka differentiaalinen signaali ei tarvitse toimiakseen maareferenssiä, on yleistä tuoda sellainen ylimääräisellä johdolla, sillä pitkillä etäisyyksillä maapotentiaali voi muuttua. Tällöin myös vastaanottimella näkyvä yhteismuotoinen signaali on yhtä suuri kuin lähetetty. RS-485-standardi sallii yhteismuotoisen jännitteen välillä -7...12 V. Käyttämällä samaa referenssiä vastaanotin pystyy lukemaan lähetetyn viestin eikä vahingoitu. Kaikkien laitteiden galvaaninen erottaminen verkko-

virrasta estää virtasilmukan syntymisen suojamaan ja referenssijohdon välille, mutta mikäli se ei ole mahdollista, suosittelee standardi vastusten lisäämistä signaalin ja laitteen maan väliin virran pienentämiseksi. [2, s.80–83]

Käytetyt johdot voidaan nimetä usealla tavalla. Tässä dokumentissa käytetään symboleita A ja B, muita vaihtoehtoja ovat TX+/TX-, RX+/RX-, D+/D- tai B+/A-. Jännite-ero $V_{AB}= V_A – V_B$ tulkitaan loogiseksi nollaksi sen ollessa alle -0,2 V ja ykköseksi yli 0,2 V:lla. [2, s.84] Välillä -0,2 V…0,2 V signaali on määrittelemättömässä tilassa, mitä ei pitäisi tapahtua normaalissa toiminnassa. Kaikkien lähettimien ollessa korkeaimpedanssisessa tilassa näin voi kuitenkin käydä, miltä suojautumiseksi linja voidaan biasoida niin, että jännite-ero on aina yli rajan ±200 mV. [3, s.66] Joillakin vastaanottimilla rajat on muutettu arvoihin -200 mV ja -30 mV, jolloin signaali on ilman biasointia tunnetussa tilassa, mutta tätä ei ole määritelty standardissa. [4, s.6–7]

Lähetettäessä signaalin täytyy olla suurempi kuin minimivaatimus vastaanotolle, sillä jännitehäviö johdoissa pudottaa amplitudia. Lisäksi käytetyt biasointivastukset voivat heikentää signaalia. Standardi määrittelee pienimmän lähetystason olevan ±1,5 V ja suurimman ±6 V. [2, s.84] Tämä tarkoittaa, että lähettimiä pystyy käyttämään 5:n ja 3,3:n voltin käyttöjännitteillä ilman varauspumppua tai jännitekonvertteria, toisin kuin vastaavanlaisissa tapauksissa käytettyä RS232.

RS-422:ta vastoin RS-485:llä voidaan käyttää useampia lähettimiä samalla väylällä. Standardi määrittelee, että käytettävien laitteiden lähettimien täytyy pystyä siirtymään korkeaimpedanssiseen tilaan, jolloin ne eivät kuormita väylää. Nelijohtoisena tämä tarkoittaa, että yksi käytettävistä laitteista on master, jonka lähetin yhdistetään muiden laitteiden vastaanottimiin. Masterin vastaanotin yhdistetään muihin lähettimiin, joista korkeintaan yksi voi olla kerrallaan aktiivinen. [3, s.62] Tämä mahdollistaa full-duplex -toiminnan [2, s.91].

Kaksijohtoisella järjestelmällä kaikki lähettimet ja vastaanottimet kytketään toisiinsa, jolloin ei ole välttämätöntä käyttää master/slave-asetelmaa, sillä kaikki vastaanottimet voivat lukea kaikki lähetetyt viestit. Tällöin vain half-duplex on mahdollista. [2, s.95]

RS-485:n laitteiden kuormitus määritellään yksiköllä U.L. (engl. Unit Load), joka mitataan laitteen ollessa korkeaimpedanssisessa tilassa (vastaanottimena) niin, että 1 U.L. vastaa 12 kΩ:n impedanssia. Tavallista 32:n laitteen enimmäisrajoitusta voi siis nostaa käyttämällä vastaanottimia, joiden kuormitus on alle 1 U.L. [2, s.86–87]

Mahdollisia verkkotopologioita on useampia. Yksinkertaisin kahden laitteen yhdistäminen molemmissa päissä käytettävin terminointivastuksin on häiriönkestoisin vaihtoehto

ja sallii suuret tiedonsiirtonopeudet. Tällaiseen verkkoon lisälaitteiden liittäminen onnistuu joko ohjaamalla koko väylä uusien laitteiden kautta tai liittymällä väylärunkoon uusilla johdoilla. Ensimmäisessä tapauksessa väylä vaikuttaa edelleen yhdeltä siirtolinjalta, eivätkä heijastukset lisäänny. Jälkimmäisessä vaihtoehdossa pitkät terminoimattomat johtojen päät saattavat lisätä heijastumista ja niin ollen signaalin heikkenemistä. [2]

Signaalin eheyden parantamiseksi käytetään RS-485:ssä terminointia signaalitien päissä vähentämään heijastuksia, joskin lyhyillä matkoilla ja matalilla tiedonsiirtonopeuksilla on mahdollista selvitä ilman niitä. Nelijohtoisella systeemillä terminointia ei tarvitse tehdä muualla kuin vastaanottimen päässä, kaksijohtoisena molemmat päät toimivat vastaanottimina, joten myös terminointi tarvitaan molempiin päihin. Yleisesti terminointi kannattaa laittaa fyysisesti kauimpana olevien laitteiden lähelle, sillä pisimmän matkan kulkeneet heijastukset ovat todennäköisemmin eri vaiheessa ja aiheuttavat siten häiriötä. [4, s.5]

Terminointi voidaan toteuttaa vastuksilla ja/tai kondensaattoreilla. Optimaaliseen tulokseen päästään käyttämällä johdon karakteristisen impedanssin suuruista terminointia, RS-485:llä noin 90...120 Ω. Käyttämällä vastuksen kanssa sarjassa kondensaattoria pienennetään biasoinnista johtuvaa tehohäviötä samalla rajoittaen signaalin taajuutta aikavakion mukaan. [2, s.86–87][4, s.5]

Useamman lähettimen ollessa päällä virrat pystyisivät kasvamaan suuriksi. Tätä varten standardin mukaan kaikkien lähettimien täytyy rajoittaa virta 250 mA:iin toiminta-alueella. Lisäksi jotkut piirit pystyvät sammuttamaan itsensä suuren virran aiheuttaessa lämpenemistä. Ylemmän tason protokollien pitäisi kuitenkin estää tällaiset tapaukset. [2, s.88–89]

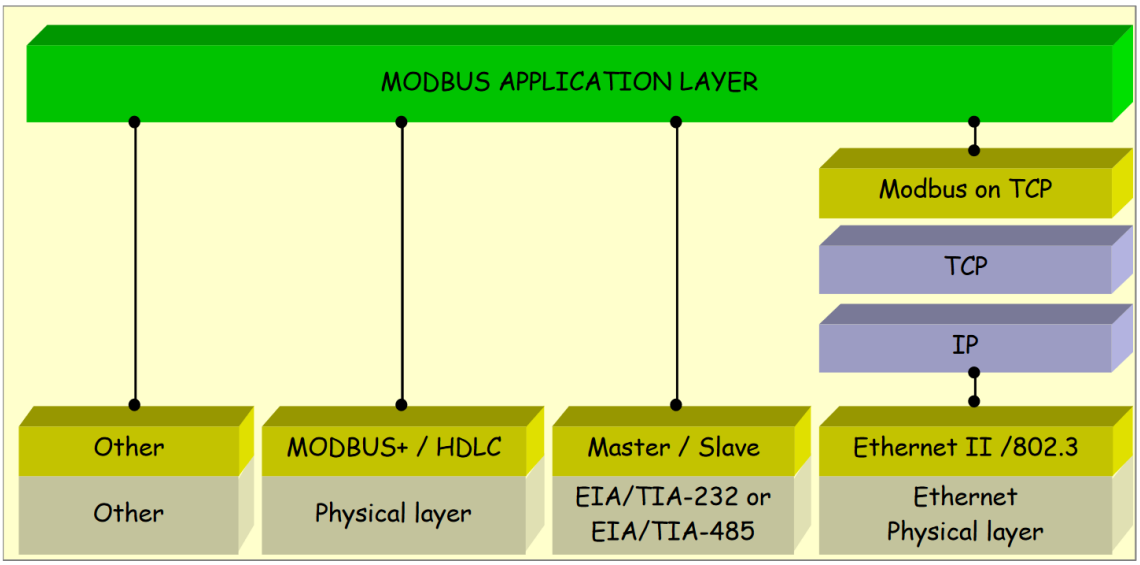
RS-485 on hyvän häiriönsietoisuuden ja mahdollisten pitkien etäisyyksien takia yleisesti käytetty teollisuuden kenttäväylissä, kuten Modbusissa [5], jotka käyttävät useampaa laitetta väylällä.

2.2 MODBUS

Modbus on OSI-mallin sovellus- ja siirtokerroksen protokolla. Modiconin vuonna 1979 kehittämä malli suunniteltiin alun perin Modiconin omien PLC:iden (engl. Programmable Logic Controller) väliseksi liitännäksi, mutta on sittemmin kasvanut tärkeäksi automaatiossa käytettäväksi työkaluksi. [5, s.1]

Siirtokerroksella protokollalla on kuvan 3 mukaisesti useampia toteutustapoja. Käsitellään tässä vain master/slave-tyyppiä, jolta vaadittuja tiedonsiirtonopeuksia ovat 9600 ja

19200 baudia. Väylällä voi olla vain yksi master-laite, jonka lisäksi slavejä voi olla standardin mukaan enintään 247. Ainoastaan master voi aloittaa kommunikaation, jonka jälkeen slave vastaa kyselyyn. Jokaiselle slavelle on asetettu oma osoite välillä 1...127, mikä näkyy lähetetyssä viestissä. Oman numeron lukiessaan slave kuuntelee viestin ja vastaa siihen. Master-laitteella ei ole osoitetta, sillä kaikki viestit jotka master vastaanottaa luetaan. Osoite 0 ei vastaa mitään laitetta, eikä siihen siis vastata, mutta kaikki laitteet kuuntelevat kyseistä osoitetta. [5, s.1–2][6, s.7–8]



Kuva 3. *Modbus-protokollan rakenne OSI-mallissa [7]*

Modbus ei määrittele käytettävää fyysisen tason protokollaa. Tämä tekee siitä joustavamman kuin esimerkiksi Ethernet, sillä toteutus voi olla mitä tahansa. Alun perin Modbus käytti RS-232:ta, mikä ei tarjonnut useamman laitteen väylää. Nykyään käytössä on kuitenkin myös mm. RS-422, RS-485, valokuitu, radiotaajuudet ja TCP/IP (engl. Transmission Control Protocol / Internet Protocol) varatussa portissa 502. [7, s.2]

Modbusilla on kaksi eri tapaa lähettää dataa, muuttaen käytettyä tavun pituutta. Modbus RTU:lla (Remote Terminal Unit) tavun pituus on 8 bittiä ja Modbud ASCII:lla (engl. American Standard Code for Information Interchange) 7, vastaten yhtä ASCII-merkkiä. Käsitellään tästä eteenpäin ainoastaan Modbus RTU:ta. Kehys lähetetään tavuittain niin, että kehyksien väli on vähintään 3,5 kertaa tavun lähettämiseen kulunut aika ja kehyksen sisäisten tavujen väli on korkeintaan 1,5 kertaa tavuun kulunut aika. Tavu alkaa startbitillä, joka on looginen 0. Tämän jälkeen tavu lähetetään MSB (engl. Most Significant Bit) ensin. Standardi vaatii laitteiden pystyvän käyttämään vähintään parillista pariteettia, mutta on myös mahdollista käyttää muita pariteetteja tai stop-bittiä sen sijaan. Riippumatta pariteetin tyypistä viimeinen bitti on stop-bitti, joka on looginen 1. [6, s.8,13]

Uusi kehys alkaa osoitteen lähettämisellä yhdessä tavussa. Slave-laitteet päättävät tämän perusteella, kuuntelevatko ne kyseistä viestiä. Seuraavaksi lähetetään yhden tavun funktiokoodi, joka määrittelee halutun toimenpiteen. Datatavuja on funktiokoodin määrittelemä määrä, joka voi olla myös 0. Kehys loppuu CRC-tarkistukseen (engl. Cyclic Redundancy Check), josta voidaan päätellä, onko viesti korruptoitunut matkalla. Data saattaa sisältää 16-bittisiä sanoja, jolloin korkeampi tavu lähetetään ensin. CRC:ssä matalampi tavu lähetetään aina ensin. [7, s.3–4] Kuvassa 4 on esitetty viestin rakenne sarjaväylällä, missä datatavujen määrä riippuu käytetystä funktiokoodista.

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Kuva 4. *Modbus RTU kehys [6]*

Kaikkiin kehyksiin odotetaan vastausta, mikäli osoite ei ole 0. Vastauksen rakenne on sama kuin kyselyn; Alussa sama osoite ja funktiokoodi, lopussa CRC. Datan rakenne ja pituus riippuvat jälleen vastaanotetusta funktiokoodista. Mikäli viesti ei ollut käsiteltävissä vastauksen funktiokoodi sisältää poikkeusbitin ja data poikkeuksen tyypin. [6, s.47-48]

Funktiokoodit ovat alueella 1...127 eli seitsemänbittiset luvut pois lukien 0. Ne määrittelevät, minkä tyyppiseen rekisteriin toteutettava operaatio kohdistuu ja mitä sille tehdään. Kehyksen datatavujen tulkinta riippuu funktiokoodista. Alueet 1...63, 73...99 ja 111...127 ovat julkisia funktioita, joiden toiminta on määritelty samalla tavalla kaikille Modbus-yhteensopiville laitteille. Vain noin 20 julkisista funktiokoodeista on määritelty, niiden lisäämisestä vastaa Modbus-organisaatio. Alueiden 64...72 ja 100...110 koodit voidaan määritellä käytettävässä sovelluksessa, eivätkä ne siis ole kaikille laitteille toimivia. [6, s.10] Kuvassa 5 esimerkki funktiokoodin 3 rakenteesta.

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**

Kuva 5. *Esimerkki funktiokoodin 3 rakenteesta [7]*

Taulukossa 1 kuvataan Modbus-protokollaan kuuluvat neljä eri rekisterityyppiä, joista kaikkia ei ole välttämätöntä sisällyttää laitteisiin. Kaikki rekisterit ovat 16-bittisiä sanoja

riippumatta niiden tarkoituksesta tai sisällöstä. 0- ja 1-alkuiset rekisterit ovat luettavia ja kirjoitettavia binääreita. Näiden tapauksessa yksittäinen rekisteri sisältää vain yhden bitin, mutta funktiokoodilla, joka käsittelee useampia rekistereitä, bitit pakataan järjestyksessä yhteen tai useampaan tavuun. 3- ja 4- alkuiset rekisterit voivat sisältää korkeintaan 16-bittisiä lukuja. [7, s.6–8]

Fyysisessä toteutuksessa rekisterit voi olla toteutettu eri tavalla. Binäärisiä rekistereitä ei välttämättä ole tai ne on pakattu pienempään tilaan. Input rekisterit voi olla siirretty kokonaan holding rekistereihin, jotta niille voisi käyttää samoja funktiokoodeja. Monet funktiokoodit, kuten 16, määrittelevät käytetyn alueen. Tällöin datassa annettu osoite on suhteellinen alueen ensimmäiseen rekisteriin, koodin 16 tapauksessa kirjoitettu rekisteri 100 viittaa todelliseen rekisteriin 400001+100=400101. [7, s.6–8] Joskus käytetään vain viiden numeron osoitteita, jolloin toinen numero jätetään pois ja rekisterit rajoitetaan suhteelliselle välille 0...9999. [5, s.3]

Sanoman virheiden tarkistukseen käytetään CRC-16/MODBUS-algoritmia. CRC alkaa arvosta 0xFFFF. Dataa käsitellään jokaista tavua kohden bitti kerrallaan. Kun tavu on käsitelty siirrytään seuraavaan tavuun ja XOR:ataan uusi tavu CRC:n kanssa. Siirryttäessä seuraavaan bittiin tarkastellaan nykyistä CRC:n viimeistä bittiä; sen ollessa asetettu XOR:ataan CRC valitun luvun kanssa, Modbusin tapauksessa luvun 0xA001. CRC:tä siirretään yhden bitin verran oikealle ja tarkastellaan uutta viimeistä bittiä. Lopulta, kun CRC:tä on siirretty koko sanoman verran oikealle eli kaikki tavut on tarkastettu, on jäljelle jäänyt CRC lähetettävissä. Modbusin tapauksessa se lisätään sanoman perään matalampi tavu ensin. [6, s.14–15] Laskennallisesti kevyempää on tehdä lista mahdollisista CRC:n arvoista, jolloin vältetään suuri määrä XOR-operaatioita (9 jokaista tavua kohden). Tämän lähestymistavan haittapuoli on suuri ohjelmamuistin kulutus, 16 bitin CRC vaatii 512 tavua muistia.

Mikäli tiedonsiirrossa on tapahtunut virhe, vastaava laite ilmoittaa sen asettamalla funktiokoodin ensimmäisen bitin. Virheen tyyppi ilmoitetaan yhdellä datatavulla. [7, s.4]

2.3 Virtaviesti

Virtaviesti on yksinkertainen analoginen tapa siirtää dataa. Yleisesti käytetty alue on 4–20 mA, jota käsitellään tässä dokumentissa. Virran käyttö tuo muutamia tärkeitä etuja verrattuna jänniteviestiin, minkä takia sen käyttö on yleistä teollisuusympäristöissä. [8, s.3]

Virtaviestin tulkinta riippuu käytettävästä laitteesta. Yleisesti 4 mA vastaa suureen pienintä arvoa ja 20 mA suurinta. Koska pienin arvo on nollasta poikkeava, on avoin virtapiiri helppo tunnistaa. Lisäksi pienitehoiset laitteet voivat käyttää signaalin virtaa tehonlähteenä. [8, s.1]

Pitkillä matkoilla johdotuksen resistanssi alkaa olla merkittävä, mikä aiheuttaa jännite-viestien epävarmuutta. Mittauksen aiheuttama kuormaresistanssi muodostaa jännitejaon johtojen kanssa, jolloin mitattu signaali ei kaikki päädykään vastaanottimelle. Tätä vaikutusta voidaan pienentää, mikäli mahdollista, kuormaresistanssia kasvattamalla. Tämä kuitenkin aiheuttaa suurempia indusoituneita häiriöitä vastaanottimelle, joten jännitesignaaleilla täytyy löytää kompromissi haittojen minimoimiseksi. [8, s.1]

Mahdolliset pitkät etäisyydet ja parempi häiriönsieto ovat haluttavia ominaisuuksia teollisuusympäristöissä, mutta kaikkiin sovelluksiin virtaviesti ei ole paras vaihtoehto. Analogisen signaalin tulkitsemiseen käytettävät AD-muuntimet (Analogia-Digitaali) mittaavat yleensä jännitettä, joten tarvittava virta-jännitemuutos vaatii ylimääräisiä komponentteja. Yksinkertaisin tapa tähän on lisätä vastus signaalitielle ja mitata sen yli jännitehäviötä. Vastuksessa syntyy tehohäviöitä, jotka voivat olla liikaa pienikulutuksisissa sovelluksissa. Tehohäviötä pystyy laskemaan pienentämällä vastuksen kokoa, mutta tällöin jännitettä pitää vahvistaa tai menetetään mittauksen resoluutiota. [8, s.1-2]

Lisäksi signaalin kuljettaminen useammalle mittauspisteelle on haastavampaa. Joissain tapauksissa mitattavia laitteita pystyy lisäämään sarjaan, jolloin sama virta kulkee kaikkien mittavastusten läpi. Jos järjestelmä mittaa virtaa maata vasten, tarvitaan virtaviestille toistaja, joka tuottaa kaksi eri virtasilmukkaa. Muussa tapauksessa laitteita voi liittää sarjaan ilman maareferenssiä, jonka kautta signaali pääsisi ohittamaan toisen laitteen. [8, s.1]

Käytettävää kuormaresistanssia ei ole määritelty, vaan kaikilla laitteilla voi olla oma kuormituksensa, jotka ovat yleisesti alueella 100...750 Ω. Tämän takia lähettimen täytyy pystyä tukemaan tarpeeksi suuria kuormia, toisin kuin jänniteviestillä, jonka kuormitus on yleisesti melko pientä. [8, s.2]

3. PID

PID-säädin on yleinen säätöalgoritmi. Yksinkertaisimmassa muodossa se koostuu kolmesta komponentista; proportionaalisesta, integraalisesta ja derivoivasta osasta rinnan kytkettynä. Sisäänmenona toimii säädettävän suureen virhe asetusarvosta ja ulostulona prosessia ohjaava suure. [9, s.59]

Säätimen toimintaa voidaan kuvata säätöparametreillä K_p, K_i ja K_d, jotka toimivat eri osien vahvistuksina. Aikariippuvaisten integroivan ja derivoivan termin vahvistus voidaan korvata integrointi- ja derivointiajoilla T_i ja T_d, jotka kuvaavat paremmin säätimen toimintaa ajan suhteen.

3.1 Jatkuva-aikainen

Jatkuva-aikaisena PID-säädin voidaan kuvata funktiolla

$$\mathrm{u(t)} = K_p e(t) + K_i \int e(t)\mathrm{d}t + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{1}$$

tai korvaamalla integrointi- ja derivointivahvistukset vastaavilla ajoilla

$$\mathrm{u(t)} = K_p \left(e(t) + \frac{1}{T_i} \int e(t)\mathrm{d}t + T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right), \tag{2}$$

joissa *u* on säätimen ulostulo ja *e* virhe ajanhetkellä *t*. Helpommin säätimen käyttäytymistä pystytään havainnoimaan Laplace-muunnoksen

$$G(s) = K_p \left(1 + \frac{1}{sT_i} + sT_d \right) \tag{3}$$

avulla, missä *G* on säätimen siirtofunktio ja *s* on Laplace-muunnoksen muuttuja. Tästä muodosta pystytään helposti määrittämään säätimen käyttäytyminen taajuustasossa sekä napa-nolla-kuvio. [9, s.64, 70–71] Nämä auttavat säätöparametrien asettamisessa, mikä ei ole tämän työn kannalta olennaista.

3.2 Diskreettiaikainen

Digitaaliselle alustalle siirtyessä jatkuva-aikaisia aika- tai s-tason esityksiä täytyy approksimoida. PID-säätimen tapauksessa on siis käsiteltävä integroivaa ja derivoivaa osaa. Näiden arviointia voidaan tehdä numeeristen määritelmien tai z-muunnoksen avulla. Laplace-muunnosta vastaavan diskreettiaikaisen z-muunnoksen muuttujaa *z* voidaan myös approksimoida useilla eri tavoilla. [10, s.1]

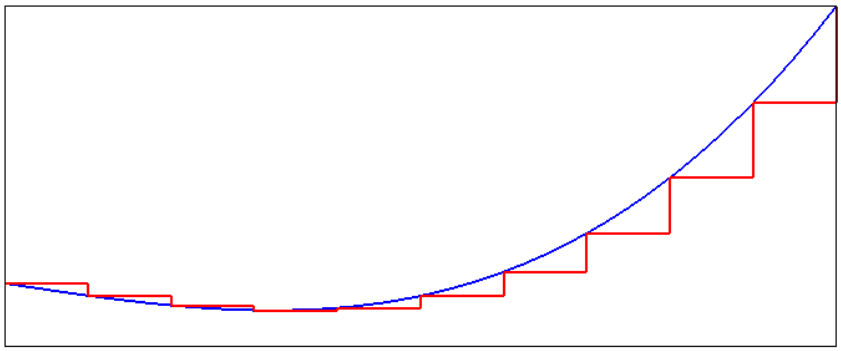
Laskennallisesti todennäköisesti helpoin ratkaisu on käyttää zero-order hold -menetelmää, jolloin z muunnosta ei tarvita. Integraali lasketaan nyt mittauspisteistä syntyneen pinta-alan kautta kuvan 6 mukaan. Kaavalla

$$I = \sum e_k T_s \tag{4}$$

jossa e_k on virhe ja T_s on mittausaika, saadaan helposti laskettua integroivan osan aiheuttama ulostulo, kun mittausaika tiedetään. Derivoiva osa lasketaan yhden tai useamman aiemman erotermin ja nykyisen erotermin erotuksesta kaavalla

$$D = \frac{e_{k-1} - e_k}{T_s} \tag{5}$$

jossa e_k on nykyinen virhe, e_{k-1} on edellinen virhe ja T_s mittausaika. Näillä kaavoilla mittausajan lähestyessä nollaa tulokset vastaavat jatkuva-aikaista integraalia ja derivaattaa. Pitkällä mittausajalla approksimoinnin tarkkuutta voidaan parantaa käyttämällä z-muunnosta. [11, s.489]



Kuva 6 *Zero-order hold integraali*

Yleisiä tapoja muuttaa Laplace z-muunnokseksi on Tustinin sekä Eulerin taaksepäin ja eteenpäin menetelmät. Käsitellään tässä Eulerin taaksepäin sijoitusta. Derivaatan määritelmästä

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x) - f(x-h)}{h} \tag{6}$$

saadaan helposti diskreettiaikainen antamalla h:n kasvaa nollasta poikkeavaksi. Nyt ottamalla oikealta puolelta z-muunnos ja vasemmalta puolelta Laplace-muunnos saadaan muunnosten välille approksimointi. Derivaatan määritelmän mukaan mittausajan lähestyessä nollaa yhtälö pitää paikkansa, joten approksimoidaan niin myös nollaa suuremmilla arvoilla. Tällöin funktiot voidaan poistaa, jolloin jäljelle jää

$$s = \frac{z - 1}{T_s z}, \tag{7}$$

missä määritelmän derivaatan h on korvattu mittausajalla T_s. Tätä voidaan käyttää harkiten muuttamaan kaikki Laplace-muotoiset siirtofunktiot z-muunnoksiksi. Tustinin ja Eulerin eteenpäin menetelmät tuottavat samankaltaisen tuloksen. [12][13]

Nyt saatu approksimointi voidaan sijoittaa PID-säätimen Laplace-muunnokseen. Säädin saa muodon

$$C(z) = K_p + \frac{K_i T_s z}{z-1} + \frac{K_d (z-1)}{T_s z} \tag{8}$$

jossa K_p, K_i ja K_d ovat säätöparametrit, T_s mittausaika ja z z-muunnoksen muuttuja.

Koska z-muunnos voidaan muuttaa nykyisiksi ja edellisiksi sisäänmenoiksi ja ulostuloiksi, tämän muotoinen säädin on mahdollista toteuttaa digitaalisena. Lisäksi voidaan lisätä myös kaikki jatkuva-aikaisen säätimen lisäominaisuudet, kuten derivointisuodin ja 2DOF-säätö (engl. 2-Degrees-Of-Freedom).

4. TOTEUTUS

Keskuksen piirikaavio ja piirilevy oli jo suunniteltu ennen työn aloittamista. Samoin myös Modbus-protokolla oli toteutettu ohjelmaan. Tämän ansiosta toteuttaa täytyi vain lisämoduulin vaatimat Modbus-rutiinit ja PID-säädin.

4.1 Käytetty laitteisto

Työ toteutettiin ST-2240-CIRC-kiertovoitelunohjauskeskukseen. Keskus käyttää Microchipin PIC18F6621-mikrokontrolleria, joka toimii 4 MHz:n kellotaajudella. Ohjelma on kirjoitettu C-kielellä Microchipin omassa MPLAB X -ympäristössä ja käännetty yhteensopivalla XC8-kääntäjällä. Työssä kontrolleri ohjelmoitiin ICD3-debuggaustyökalulla, mutta lopullinen tuote sisältää bootloaderin, jonka avulla keskus pystytään ohjelmoimaan sarjaliitännän kautta.

Keskuksen näyttönä toimi Schneiderin HMISTU65, joka on yhteydessä keskukseen RS232:n läpi Modbusilla. Näytön ohjelma luotiin Vijeo Designer -ohjelmalla ja vietiin näytölle Ethernetin kautta.

Lisä-IO:na käytettiin Schneiderin OTB1S0DM9LP-modulia, joka toimii RS485:n läpi Modbusilla. Tämän lisäksi TM2AMM6HT-lisämodulia käytettiin analogisten tulojen ja lähtöjen lisäämiseksi.

Ohjattavana venttiilinä toimi Bürkertin 3280-säätöventtiili analogiohjauksella. Säädintä testattiin myös Bürkertin paineilmatoimisella 8693-venttiilillä.

4.2 Tiedonsiirto

Käytetyllä mikrokontrollerilla PIC18F6621 on kaksi USART-väylää (engl. Universal Synchronous/Asynchronous Receiver-Transmitter), joista toinen muunnetaan RS485:ksi. Kenttäväylä erotetaan galvaanisesti optoerottimilla ja muutetaan differentiaaliseksi SN75176D-piirillä. Piirin ulostulo on RS485-4W eli nelijohtoinen väylä. Käytetyt laitteet toimivat kuitenkin kaksijohtoisena, joten lähetyksen ja vastaanoton johdot yhdistetään lähes aina.

RS485 vaatii toimiakseen terminointivastukset [7, s.85]. Ohjauskeskuksella ja taajuusmuuntajilla nämä voidaan kytkeä liittämällä siihen tarkoitetut jumperit. Ohjaus-

keskuksella ja taajuusmuuntajilla väylään voi kiinnittyä riviliittimillä, joten kaapelina käytetään yleisesti 0,5 mm^2 johtoa, josta yksi suojattu pari kulkee rinnankytketyille taajuusmuuntajille. Lisämoduuli tarvitsee RJ45-liittimen kaapelin päähän.

Ohjauskeskus toimii kenttäväylän masterina, joten väylää ei tarvitse kuunnella ennen datan lähettämistä. Lähetys on ajastettu niin, että uusi käsky lähetetään vähintään 0,2 sekunnin välein tai 0,4 sekunnin jälkeen, jos vastausta ei saada. USARTin `TXREG1`-rekisteriin kirjoitetaan seuraava tavu `RC1IF`-lipun keskeytyksellä [13]. Koko viestin lähetyksen jälkeen suljetaan lähetin ja kuunnellaan vastaanotettua viestiä. 9600 baudilla kaksi ensimmäistä vastaanotettua tavua ovat piirin suunnittelusta johtuen lähetettyä dataa, joten näitä ei lueta vastaanottopuskuriin.

Lisämoduuli vaatii datalta 8 databittiä, parillisen pariteetin ja yhden stop-bitin. Nämä saadaan ohjelmoitua mikrokontrollerille asettamalla `RX91` ja `TX92` -liput ja käyttämällä pariteetin tarkistukseen tehtyä hakutaulukkoa [14]. Myös taajuusmuuntajat pystyvät käsittelemään tämän muotoista dataa.

Virtaviesti luetaan keskukselle ohjaamalla virta 249 Ω vastuksen yli, jolloin muodostuu mitattava jännite, jota lisäksi suodatetaan ennen mittausta. Mittaamiseen käytetään mikrokontrollerin sisäistä AD-muunninta. Säädettäviä virtalähtöjä ei keskuksessa ole, joten niiden lisäämiseksi käytetään lisämodulia.

4.3 Säädin

Säätimen ensimmäinen versio toteutettiin zero-order hold -muotoisena PID-säätimenä parametreilla *Kp*, *Ki* ja *Kd* sekä yksinkertaisella derivointisuotimella. Diskretointina käytettiin integroivalle termille summaa ja derivoivalle termille edellisen ja nykyisen virheen erotusta. Säätimelle luotiin struct-tyyppi `PID_t`, jotta tulevaisuudessa mahdollisesti tarvittavia ylimääräisiä säätimiä voidaan luoda helposti. `PID_t` sisältää paitsi säätöparametrit, myös muut tarvittavat jokaisen säätimen omat arvot, kuten säätöalueen rajat, mittaukseen kuluneen ajan ja konfigurointiasetukset.

Joissain tapauksissa säätimellä on hyödyllistä olla vaihtoehtoiset säätöparametrien arvot, joita pystyy käyttämään asettamalla `paramChange`-lipun. Nykyisessä säätimessä ei ole hyödynnetty tätä ominaisuutta. Säätimen aluetta on rajoitettu kolmessa kohdassa; Integroivaa ja derivoivaa termiä, sekä ulostulon arvoa. Nämä rajat on sisällytetty tyyppiin.

Lisäksi tyyppiin kuuluvat pointerit mittaukseen, asetusarvoon ja ulostuloon. Alkuperäisessä versiossa kaikki muuttujat paitsi liput ovat tyyppiä int. Käytetyssä PIC18F-alustassa int on 16 bittinen, joten kaikki muuttujat mahtuvat siihen [14]. Liput ovat `pidflags`-structissa bitfieldeinä.

Säätimen uusi ulostulo lasketaan `PIDcalculate(PID_t*)`-funktiota kutsuessa. Ensimmäisenä lasketaan kulunut aika sekunnin tarkkuudella. Mittausten välinen aika on sekunneista kymmeniin sekunteihin, joten tämä tarkkuus riittää. Erosuure riippuu siitä, käytetäänkö säädintä normaalisti vai käänteisenä ohjauksena, joka on toimintatapa venttiilin tapauksessa. Tällöin erosuure siis lasketaan mittauksen ja asetusarvon erotuksena.

Vahvistusparametrit on skaalattu, jotta niille saataisiin suurempi alue. Pitkästä mittausajasta johtuen I-termi kannattaa jakaa suuremmalla luvulla ja D-termi pienemmällä, jotta niiden säätöalue on samankaltainen.

Ulostuloa laskettaessa I-termi rajoitetaan välille 0...4095, jolloin ohjauksen ollessa ääriasennossa I-termi ei kasva enää suuremmaksi. D-termille tehdään yksinkertainen alipäästösuodin, joka rajoittaa ohjauksen valitun taajuuden suurimmaksi muutosnopeudeksi kaavalla

$$D_{limit=} f \Delta t \tag{9}$$

jossa *f* on ohjelmassa asetettu rajataajuus ja *Δt* mittausaika. Lopulta rajoitetaan vielä ulostulo välille 0...4095, joka on käytetyn DA-muuntimen alue.

Ensimmäisten testien jälkeen säätimen parametrit vaihdettiin yleisemmin käytettyihin ja paremmin järjestelmää kuvaaviin muotoihin *Kp*, *Ti* ja *Td*. Nyt ulostulo sai muodon

$$u = \frac{K_p e}{100} + \frac{K_p}{100 T_i} \sum e \Delta t + \frac{K_p T_d}{100} \frac{\Delta e}{\Delta t} \tag{10}$$

jossa *e* on virhe, *Δt* on mittausaika ja K_p, T_i ja T_d ovat säätöparametrit. T_i ja T_d eivät enää tarvinneet erillistä skaalausta, sillä ne vastaavat integrointi- ja derivointiaikoja sekunteina. Termien rajat pysyvät samana.

Mittaustulosten perusteella säätimessä oli vielä parannettavaa. Integrointi- ja derivointitermien diskretointi muutettiin z-muunnokseksi, millä tavoiteltiin paremmin jatkuvaaikaista säädintä vastaavia tuloksia. Derivoivaan termiin lisättiin N-kertoiminen suodin aiemman rajan sijaan. Laplacesta z-muunnokseen pääsemiseksi käytettiin Eulerin taaksepäin muunnosta ja säädin saatiin kaavasta (8) muotoon

$$u_k = -\frac{a_1}{a_0} u_{k-1} - \frac{a_2}{a_0} u_{k-2} + \frac{b_0}{a_0} e_k + \frac{b_1}{a_0} e_{k-1} + \frac{b_2}{a_0} e_{k-2} \tag{11}$$

jossa $a_0...a_2$ ja $b_0...b_2$ ovat säätöparametreista, suodinkertoimesta ja mittausajasta riippuvia termejä ja e_i erosuureita ja u_i ulostuloja ajanhetkellä i.

Kertoimien laskut ovat melko pitkiä ja vaativat liukulukuja, joten ei ole mielekästä laskea niitä joka kerta PID:n ulostuloa laskettaessa. Tehtiin siis oma funktio kertoimien laskemiseksi, jota kutsutaan parametrejä muutettaessa tai mittausajan muuttuessa yli 20 %. Lisäksi kulunut aika mitattiin 0,1 sekunnin tarkkuudella tarkkuuden parantamiseksi.

4.4 Lisämoduuli

Lisä-IO -moduuli toimii 0-pohjaisessa liikenteessä, siis ensimmäinen Modbus-rekisteri vastaa rekisteriosoitetta 0. Kaikki rekisterit kirjoitetaan ja luetaan Holding-rekistereistä, joten kirjoittamiseen voidaan käyttää funktiokoodia 6 tai 16 ja lukemiseen koodia 3 [15][7, s.11]. Koska ohjauskeskuksessa on jo valmiiksi käytetty koodeja 16 ja 3, toteutetaan myös moduulin kommunikaatio näillä.

Moduuli ei käytä normaalisti käynnistyessään analogiaportteja, vaan ne täytyy valita käynnistyksessä. Analogiaportit sijaitsevat moduulin lisäosassa, joten niiden rekisterit löytyvät osoitteista 214...241. Ennen näiden kirjoittamista varmistetaan lisäosan olevan oikea kysymällä moduulin rekisteristä 1006 lisäosan tunnistetta, jonka pitäisi tässä tapauksessa olla 4008h [14]. Oikean tunnisteen tapauksessa voidaan asetusten kirjoittamista jatkaa.

Moduulin lisäväylä (engl. expansion bus) täytyy nollata, jotta moduuli hyväksyy asetusten muuttamisen. Tämän jälkeen ohjelman lähetyspuskurin maksimikoon takia lähetetään asetukset kymmenen kerrallaan käyttämällä funktiokoodia 16. Lopulta lisäväylän nollaus voidaan poistaa ja valitut analogiaportit toimivat halutulla tavalla. Moduuli on kuitenkin hidas käynnistymään, joten lähetetään asetukset vasta 3 ja 10 sekuntia sähköjen kytkemisen jälkeen.

Uusi ulostulo lasketaan, kun Modbus-väylän pollauskierros on suoritettu, noin 3 sekunnin välein. Säätimen sisäänmenon, siis lähtevän lämpötilan mittaus, on jo toteutettu keskuksen ohjelmassa. Lämpötilatieto saadaan 4–20 mA virtaviestinä, joka muutetaan 10 bittisellä AD-muuntimella ja skaalataan 0.0…100.0, joka vastaa lämpötilaa celsiusasteina. Säädin toimii 12 bittisillä kokonaisluvuilla, joten AD-muunnosrutiiniin lisättiin toinenkin skaalaus [15].

12 bittinen arvo kirjoitetaan moduulin rekisteriin 101, jolloin analogialähdön AO0 DA-muunnin (Digitaali-Analogia) muuttaa ohjauksen virran arvoa [15]. Ohjauksen arvoilla 0...0FFFh virtalähtö saa arvot 4,10...19,91 mA. Tärkeää on, että ilman ohjausta venttiili

sulkeutuu kokonaan, joten 4,10 mA ohjaus ei riitä varmistamaan jäähdytysveden virtauksen täydellistä pysähtymistä. Valitulla venttiilillä pystytään valitsemaan minimiohjaus, jolla se aukeaa, mutta tätä ominaisuutta ei ole kaikissa säätöventtiileissä. Tällöin voidaan käyttää erillistä sulkuventtiiliä, jota ohjataan lisämoduulin yhdellä relelähdöllä. Tämä voidaan toteuttaa sulkemalla moduulin digitaalilähtö DO3, kun ohjaus saavuttaa arvon 0.

4.5 Poikkeukset

Jos keskusta käynnistäessä moduuli on valittu käytettäväksi ja sen tunniste ei ole oikea, ei alustuksia tehdä ja keskus antaa Modbus-tiedonsiirtohälytyksen. Sama hälytys tehdään myös, jos 12 peräkkäistä vastausta ovat toiminnan aikana virheellisiä tai aikakatkaistiin.

Säätöventtiilit eivät ole aivan varmatoimisia, joten ohjauksen lisäksi tarkkaillaan myös venttiilin oikeaa arvoa, mikäli venttiili ilmoittaa sen 4–20 mA virtaviestillä. Käytetään lisämoduulin analogiatuloa AI0 sen mittaamiseen, ja käynnistetään hälytys sen ollessa kolmen mittauksen ajan liian kaukana asetusarvosta. Hälytys ei muuta ohjausta tai keskeytä järjestelmän toimintaa, ainoastaan näkyy näytöllä ja keskuksen hälytysvalona. Venttiilin palatessa sijainnin toleranssien rajojen sisään lopetetaan hälytys.

Pumppauksen sammuessa ja kylmäkäynnistyksen aikana venttiilin ohjaus keskeytetään ja se asetetaan kokonaan kiinni. Uudelleen käynnistäessä ohjaus ei palaa aiempaan arvoon, vaan alkaa nollasta.

4.6 Mittaukset

Modbusia testattiin tiedonsiirtonopeuksilla 4800 ja 9600 baud pumppauksen ollessa päällä ja pois päältä. Virtausmittaria ei kytketty, joten väylän liikenne oli jatkuvaa IO-moduulin (engl. Input-Output) ja kahden taajuusmuuntajan välillä. Virheet laskettiin aiheutuneista aikakatkaisuista ja virhekoodin sisältävistä vastauksista. Mittauksissa lähetettiin noin 2000 kehystä 200 ms:nnin välein.

Taulukko 1. *Modbus-virheet mittauksissa*

	4800 baud	*9600 baud*
Taajuusmuuntajat päällä	1/2000 = 0,05%	40/2002 = 2,00%
Taajuusmuuntajat pois päältä	1/2001 = 0,05%	39/2000 = 1,95%

Taulukosta 1 nähdään, että taajuusmuuntajien käyttö ei vaikuttanut tietoliikenteeseen. Tiedonsiirtonopeuden kasvattaminen kuitenkin kasvatti virheiden määrää merkittävästi. Suurin osa näistä virheistä oli taajuusmuuntajilta aiheutuvia aikakatkaisuja. Lähetetty sanoma oli näissäkin tapauksissa oikeanlainen, joten oletettavasti ajastus tavujen tai kehysten välissä ei riittänyt taajuusmuuntajien vaatimuksiin. Keskus voi lähettää dataa molemmille USARTeille yhtä aikaa, jolloin päällekkäin lähetettyjen viestien ajoitukset hidastuvat. Suuri virheprosentti on kuitenkin tässä tapauksessa hyväksyttävissä, sillä kaikkia käskyjä lähetetään jatkuvasti.

Säätimen testaaminen oikealla järjestelmällä ei ollut mahdollista, mutta sen toimintaa pystyi vertaamaan ideaaliseen säätimeen. Tämä onnistui korvaamalla lämmönvaihdin tunnetulla siirtofunktiolla ja simuloimalla saman siirtofunktion toimintaa jatkuva-aikaisella PID-säätimellä.

Valittiin yksinkertainen alipäästösuodin testikytkennäksi sen helpon toteutuksen ja hyvin tunnetun käyttäytymisen mukaan. Ulostuloa ja mittausta käytettiin jännitealueella 0...10 V. Siirtofunktioksi saadaan

$$H(s) = \frac{R_L}{R_f R_L C_f s + R_L + R_f} \tag{12}$$

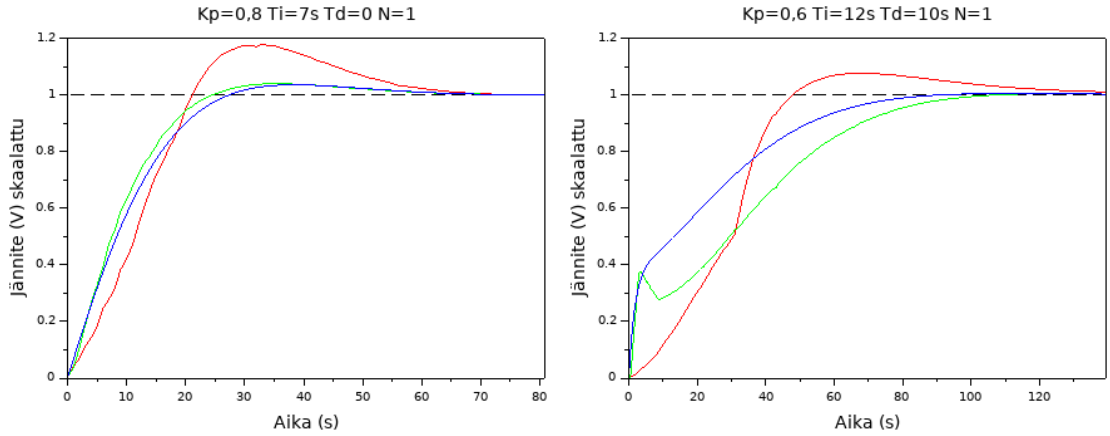
jossa R_L on mittauksen aiheuttama kuormaresistanssi, R*f* on suotimen vastuksen resistanssi ja C_f suotimen kondensaattorin kapasitanssi. Valitaan R_f=1 kΩ ja C_f=12 mF. Mittausten perusteella R_L on merkittävästi suotimen resistanssia suurempi, joten siirtofunktio voidaan yksinkertaistaa muotoon

$$H(s) = \frac{1}{R_f C_f s + 1} \tag{13}$$

josta järjestelmän aikavakioksi saadaan 12 s. Tämä on samaa luokkaa kuin oikean järjestelmän dynamiikka. Mittausaika on melko pitkä tähän verrattuna, joten järjestelmä saattaa toimia odottamattomalla tavalla.

Säätimiä verrattiin askelvasteen avulla. Mittauksissa askeleena toimi asetusarvon muutos 0 V -> 5 V. Ensimmäisissä simuloinneissa käytettiin vakiomuotoista PID-säädintä ilman derivointisuodinta, uudistettuun säätimeen verratessa käytettiin suodatusvakiota *N*=1. Testi toteutettiin molempia säätimiä kohden kahdesti arvoilla *Kp* = 0,8, *Ti* = 7s, *Td* = 0s ja *Kp* = 0,5, *Ti* = 12s, *Td* = 10s. Kuvasta 7 nähdään, että ensimmäisellä säätimellä toteutetun järjestelmän vaste on huomattavasti aggressiivisempi kuin simuloinneissa. Lopullisessa toteutuksessa mittaus ilman derivoivaa osaa on lähellä ideaalista. Käytettyjen komponenttien arvoja ei ollut mahdollista mitata, minkä takia simulointi ei täysin

kuvaa oikeaa järjestelmää. Lisäksi pitkä mittausaika suhteessa järjestelmän dynamiikkaan aiheuttaa hieman muutoksia.



Kuva 7. *Säätimen testit (sininen=simulointi, punainen=1. säädin, vihreä=2. säädin)*

Suurempi ongelma on derivoivan termin aiheuttama hyppiminen asetusarvoa muutettaessa. Mittauksissa derivointiaika oli asetettu huomattavasti suuremmaksi kuin on järkevää, jolloin ongelma korostuu, mutta se saattaa aiheuttaa toivomatonta muutosta pienemmiläkin arvoilla. Lyhyempi mittausaika vähentäisi vaikutusta, mutta koska tämä ei ole mahdollista, täytyy parametreja asettaessa käyttää omaa harkintakykyä.

4.7 Jatkokehitys

Tietoliikenne toimii nykyisessä versiossa riittävän hyvin, mutta kaksi selkeästi jatkokehitystä vaativaa ongelmaa nousivat esiin. Ensimmäinen on keskusten omien tavujen lukeminen Modbus-kehyksen lähetyksen jälkeen. Todennäköisesti tämä johtuu väylän toteutuksesta piirilevyllä, sillä sama ongelma esiintyy ilman muita laitteita. Riviliittimien ja mikrokontrollerin välissä on neljä piiriä, joissa häiriö saattaa syntyä. Tähän on kuitenkin olemassa toimiva ratkaisu, joten suurin aiheutunut ongelma on ohjelman lisääntynyt monimutkaisuus.

Toinen kehityskohde on vastaamatta jätetyt kyselyt taajuusmuuntajille. Tämäkin ongelma on todennäköisesti fyysisellä tasolla; tietokoneella luettuna lähetetty kehys on oikeaa muotoa, mutta taajuusmuuntaja ei reagoi siihen. Tämä on suurempi ongelma, sillä mahdollisten uusien lisälaitteiden tapauksessa liikenne lisääntyy, jolloin sanomia ei pystytä toistamaan yhtä usein. Ratkaisu alkaisi signaalin tutkimisesta oskilloskoopilla tai logiikka-analysaattorilla.

Itse säädin toimii lähes odotetulla tavalla. Tarkoitukseen suunnitellulla säätimellä olisi mahdollista päästä parempiin tuloksiin, mutta helpon käytettävyyden ja yleisyyden ansiosta PID on parempi ratkaisu yleiskäyttöiseksi säätimeksi. Derivoivan osan säätäminen vaatii harkintaa, mutta hitaassa prosessissa ja hyvin mitoitettuna myös sitä on mahdollista käyttää nykyisellä mittausajalla.

5. YHTEENVETO

Työssä toteutettiin säätö öljyvoitelukeskuksen jäähdytysvesiventtiilille. Säädin toteutettiin PID-säätimellä, sillä aiemmin venttiiliin sisäänrakennettu säädin oli saman tyyppinen, jolloin säätimen parametrien asettaminen vastaa aiempaa.

Uutta venttiiliä ohjattiin 4–20 mA virtaviestillä, joka on yleinen teollisuudessa käytetty tiedonsiirtomuoto sen häiriönsietoisuuden ansiosta. Keskuksessa ei ollut vapaita analogialähtöjä, joita varten ohjauskeskukseen lisättiin lisä-IO -moduuli.

Moduulille tiedonsiirto toimi MODBUSilla RS485-yhteyden yli. RS485 on yleinen differentiaalinen sarjaviestintäprotokolla, joka tukee useaa laitetta samassa väylässä. Lisäksi sen vaatimat jännitteet ovat helppo toteuttaa ja differentiaalisuus lisää häiriönkestoa verrattuna maahan referoituihin viesteihin, mikä tekee siitä hyvän valinnan teollisuudessa.

MODBUS on korkeamman tason protokolla, jonka avulla samalle väylälle voidaan helposti jakaa yhden master-laitteen välittämiä viestejä. MODBUS tarjoaa viestien oikeellisuuden tarkistuksen ja siihen on yksinkertaista lisätä uusia laitteita pienellä laskentateholla, joten se on hyvä ratkaisu sulautetuille järjestelmille.

PID on yksinkertainen ja yleisesti käytetty säädintyyppi. Se on pohjimmiltaan jatkuva-aikainen, mutta sen muuttaminen diskreettiaikaiseksi on yksinkertaista, jolloin toteutus digitaalisilla alustoilla on mahdollista. Zero-order-hold -menetelmällä diskretoitu säädin on helppo toteuttaa ja z-muunnetulla säätimellä päästään vastaavasti paremmin ideaalista vastaavaan tulokseen.

Säädin toteutettiin molemmilla diskretointimenetelmillä, joista mittausten perusteella päädyttiin z-muunoksella toteutettuun. Säädin toimi lähes simuloidun ideaalisen säätimen tavoin, poislukien lyhyellä derivointiajalla testattuna. Tämä ei kuitenkaan ole ongelma, sillä derivoivaa osaa käytetään erittäin harvoin.

Työssä päästiin asetettuihin tavoitteisiin ja säädin toimi halutulla tavalla testilaitteistolla. Myös jatkokehitettäviä osia projektiin jäi, mutta näiden toteuttaminen vaatisi mahdollisesti muutoksia piirilevyyn.

LÄHTEET

[1] Day, J.D., Zimmermann, H., The OSI reference model. Proceedings of the IEEE, vol. 71, Iss. 12, 1983, pp.1334–1340.

[2] Axelson, J., Serial port complete, Lakeview Research, 2007, 380 p.

[3] Park, J., Mackay, S., Wright, E., Practical data communications for instrumentation and control, Newnes, 2003, 400 p.

[4] RS-485/RS-422 Circuit Implementation Guide, Analog Devices, Literature Number AN-960. Saatavissa (viitattu 16.8.2019): https://www.analog.com/media/en/technical-documentation/application-notes/AN-960.pdf

[5] George Thomas, Introduction to the Modbus Protocol, The Extension, Vol. 9, Iss 4, 2008, pp. 1–4

[6] MODBUS over serial line specification and implementation guide V1.0, MODBUS.ORG, 2002. Saatavissa (viitattu 17.8.2019): http://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf

[7] MODBUS Application Protocol Specification V1.1b3, MODBUS.ORG, 2012. Saatavissa (viitattu 17.8.2019): http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[8] The Science of 4-20 mA Current Loops, Building Automation Products, Inc. Saatavissa (viitattu 24.3.2020): https://www.bapihvac.com/application-note/the-science-of-4-to-20-ma-current-loops-application-note/

[9] Åström, K. J., Hägglund T., PID Controllers; Theory, Design and Tuning, Instrument Society of Automation, 1995, 343 p.

[10] Jury, E. J., Theory and Application of the z-Transform Method, Robert E. Krieger Publishing Co, 1973, 337 p.

[11] Helmicki A.J., Jacobson C.A., Nett C.N., On zero-order hold equivalents of distributed parameter systems, IEEE,Transactions of Automatic Control, Vol. 37. Iss 4, April 1992, pp. 488–491.

[12] El-Sharif, I.A., Hareb, F.O., Zerek, A.R., Design of Discrete time PID controller. Proceedings of International Conference on Control, Engineering and Information Technology, Sousse, Tunisia, March 22-25, 2014, pp. 110–115.

[13] Tong, J.L., Bobis, J.P., A model for designing digital PID controllers. Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation. November 9-13, 1992, pp. 1157–1162.

[14] PIC18F6621, Microchip. Saatavissa (viitattu 17.8.2019): http://ww1.microchip.com/downloads/en/DeviceDoc/39612C.pdf

[15] Modbus Advantys OTB, 2008, Schneider Electric. Saatavilla (viitattu 17.8.2019): https://www.schneider-electric.com/