Teemu Saali

Modbus TCP -rajapinnan käyttö kaasuvalvontajärjestelmän toteutuk-sessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

8.5.2022

Tiivistelmä

Tekijä:	Teemu Saali
Otsikko:	Modbus TCP -rajapinnan käyttö kaasuvalvontajärjestelmän toteutuksessa
Sivumäärä:	36 sivua + 15 liitettä
Aika:	8.5.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Toimitusjohtaja Jarmo Ekholm Lehtori Kristian Junno

---

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa ohjelma kaasuvalvontajärjestelmään. Työ tehtiin yritykselle KV-Tekniikka Oy, jonka päätoimena on toimittaa kaasuvalvontajärjestelmiä sekä tarjota huoltopalveluja asennetuille järjestelmille.

Työssä perehdyttiin aluksi modbus-kenttäväylään ja sen protokollaan. Selvitettiin, mikä on kaasuvalvonta ja mitä sen suunnittelussa tulisi ottaa huomioon. Työssä käsitellään myös kaasuvalvontaan kuuluvia laitteita. Työssä tarkasteltiin tarkemmin Honeywell Midas-t-004 -kaasunmittausyksikköä, joka toimii Modbus TCP/IP -palvelin laitteena.

Työssä toteutettiin yritykselle kaasuvalvontajärjestelmän ohjelmisto, jota voitaisiin hyödyntää tulevissa projekteissa. Työn ohjelmointi toteutettiin TwinCAT3-ohjelmistolla. Ohjelmointiin kuului PLC, jonka toteutuksessa käytettiin TE1000 TwinCAT3-Engineering kehitysympäristöä. Työssä käytettiin TC1200-komponenttia, joka tukee kaikkia standardissa IEC-61131-3 kuvattuja ohjelmointikieliä sekä TF6250-kirjastoa Modbus TCP/IP -kommunikointiin. Työn ohjelmointikieleksi valittiin Structured Text, koska se tarjosi monipuolisemman ohjelman toteutuksen. HMI:n toteutuksessa käytettiin TwinCAT3:n tarjoamaa TE2000-HMI-Engineering -kehitysympäristöä, jolla voidaan toteuttaa web-pohjaisia käyttöliittymiä.

Avainsanat: kaasuvalvonta, Modbus TCP/IP, TwinCAT 3

Abstract

Author:	Teemu Saali
Title:	Modbus TCP Interface in The Implementation of a Gas Monitoring System
Number of Pages:	36 pages + 15 appendices
Date:	8 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and automation engineering
Professional Major:	Automation engineering
Supervisors:	Jarmo Ekholm, Managing Director Kristian Junno, Serior Lecturer

---

The aim of the thesis work was to design and implement program for a gas monitoring system. The work was carried out for the company KV-Tekniikka Oy, which main activity is to supply gas monitoring systems and provide maintenance services for installed systems.

The work first introduces the Modbus fieldbus and its protocol. It is clarified what gas monitoring is and what equipment should be taken into account in its design. The Honeywell gas metering unit, which act as a Modbus TCP/IP-server device, is examined in more detail.

In this work, the company was provided with program for a gas monitoring system that could be utilized in the company's future projects. The work was programmed with TwinCAT3 software. The programming included PLC program using TE1000, which is the TwinCAT development environment for convenient configuration of control, I/Os and drive control. The TC1200 component which supports all programming languages described in the IEC-61131-3 standard, and the TF6250 library for Modbus TCP/IP communication were used in the work. Structured Text was chosen as the language for programming the work because it offered a more versatile program implementation. The implementation of HMI used TE2000-HMI-Engineering development environment provided by TwinCAT, which can be used to implement web-based user interfaces.

Keywords: gas monitoring, Modbus TCP/IP, TwinCAT 3

Sisällys

Lyhenteet

ADU: Application Data Unit. Modbus-sovellustietoyksikkö.

ADS: Automation Device Specification. TwinCAT -viestintäprotokolla.

BCl: Booritrikloridi.

FB: Function Block. Toimintalohko.

DHCP: Dynamic Host Configuration Protocol. Verkkoprotokolla.

PDU: Protocol Data Unit. Modbus protokolla tietoyksikkö.

LON: Local Operating Network. Ohjaussovellus tarpeisiin luotu protokolla.

PLC: Programmable logic controller. Ohjelmoitava logiikka.

TCP: Transmission Control Protocol. Tiedonsiirtoprotokolla.

HCl: Vetykloridi.

PC: Personal Computer. Tietokone.

HTML: Hypertext Markup Language. Avoimesti standardoitu merkintäkieli.

HTP: Haitalliseksi todettu pitoisuus.

OSI: Open System Interconnection Reference Model. Tiedonsiirtoprotokollien yhdistelmä.

MBAP: Modbus Application Protocol. Modbus -TCP viestiprotokolla.

NO: Normal Open. Normaalisti auki.

NC: Normal Closed. Normaalisti kiinni.

RS: Recommended Standard. Sarjaliikenne standardi.

IP: Internet protocol. Internet-protokolla.

UPS: Uninterruptible power supply. Katkeamaton varavirtalähde.

I/O: Input/Output. sisään- ja ulostulot.

HMI: Human-Machine Interface. Käyttöliittymä.

WYSIWYG: What You See Is What You Get. Editori, joka näyttää muokattavan sisällön niin kuin se näytetään www-sivustolla.

1 Johdanto

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa kaasuvalvontajärjestelmä yritykselle KV-Tekniikka Oy. Tähän kuului PLC-ohjelma (Programmable logic controller) sekä käyttöliittymä. KV-Tekniikka Oy on yritys, jonka päätoimena on toimittaa laadukkaita kaasuvalvonnan laitteita kannettavista kaasuhälyttimistä järjestelmiin. Yrityksen järjestelmiin kuuluvat tällä hetkellä perinteiset analogiset järjestelmät, jotka perustuvat mA-mittaussignaaliin, sekä digitaalinen järjestelmä, joka perustuu LON-arkkitehtuuriin (Local Operating Network).

Tämä työ valittiin opinnäytetyöksi, koska yrityksellä oli tarve vaihtoehtoiselle digitaaliselle järjestelmälle vanhan rinnalle. Yrityksessä oli jo aikaisemmin käytetty Honeywellin valmistamia Midas-sarjan mittayksiköitä sekä analogisessa että LON-järjestelmässä. Haluttiin lähteä kehittämään järjestelmää, jossa hyödynnettäisiin Midas-t-004:n Modbus TCP/IP -rajapintaa. Tätä työtä lähdettiin toteuttamaan Beckhoffin kehittämällä TwinCAT 3 -ohjelmistolla, johon heiltä löytyivät kirjastot Modbus-TCP/IP:tä varten.

Työssä perehdyttiin aluksi modbus-kenttäväylään ja selvitettiin sen protokollaa Modbus TCP/IP:tä varten. Työ toteutettiin TwinCAT 3 -ohjelmistolla, josta lisenssiä vastaan otettiin käyttöön TF6250 PLC -kirjasto. Tehtiin tarvittavat toimenpiteet, jotka vaadittiin ennen kuin yhteys modbus-laitteeseen voitiin muodostaa.

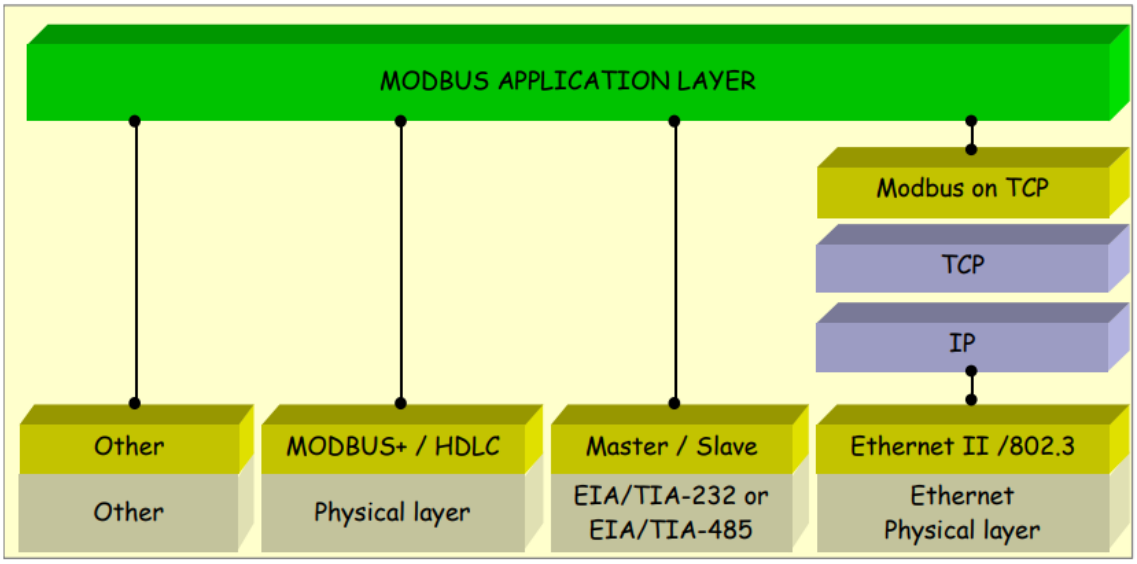
Työn toteutuksessa käytössä oli Midas-t-004 -kaasumittausyksikkö, mutta varsinaista Beckhoffin laitteistoa ei ollut käytössä, joten testaus toteutettiin ainoastaan ohjelmallisesti. Työssä käytettiin TwinCAT3:n tarjoamia kokeilulisenssejä, jotka ladattiin ohjelman tekoa ja testausta varten.

2 Modbus-kenttäväylä

2.1 Yleisesti

Modbus on Modiconin ykyisen Schneider Electricin kehittämä protokolla, joka on tuotu markkinoille vuonna 1979. Laitteita, joissa höydynnetään Modbusia, on maailmassa käytössä miljoonia, ja se on muodostunut alalle “de facto” -standardiksi. [1, s. 2.]

Protokolla toimii OSI-mallin 7. tasolla ja tarjoaa client/server -kommunikaation erityyppisten väylään tai verkkoihin kytkettyjen laitteiden välillä (kuva 1).

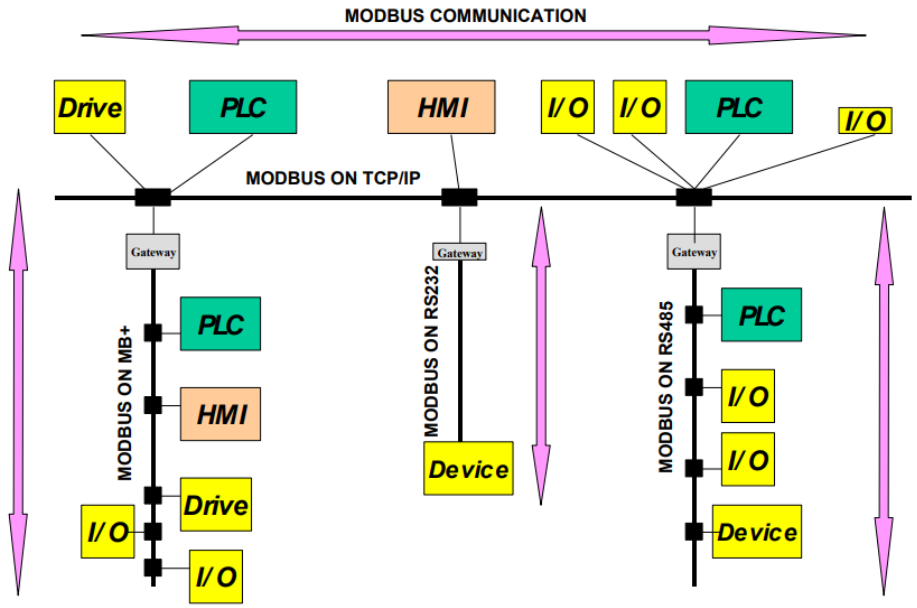


Kuva 1 Modbus-kommunikointipino [1, s. 2.]

Modbus on palvelupyyntö-protokolla (request/reply), joka tarjoaa toimintokoodeilla määriteltyjä palveluja. Kommunikointi voidaan toteuttaa sarjaliikennemuodossa (rs-232, rs-422, rs-485) sekä Ethernet TCP/IP. Sarjaliikenteisessä Modbusissa puhutaan isäntä- (master) ja orjalaitteista (slave), jotka muodostavat verkon, jossa voi olla yksi isäntä ja monta orjaa. Kun taas Modbus TCP:ssä puhutaan asiakas- (client) ja palvelinlaitteista (server). Asiakas vastaa isäntää ja palvelin vastaa orjaa. Modbus-väylässä isäntälaite on pyynnön tekevä laite.

Yleensä tämä on ohjelmoitava logiikka ja näyttöpääte. Orjalaite puolestaan on väylästä löytyvä laite, ohjain tai I/O. [1, s. 2.]

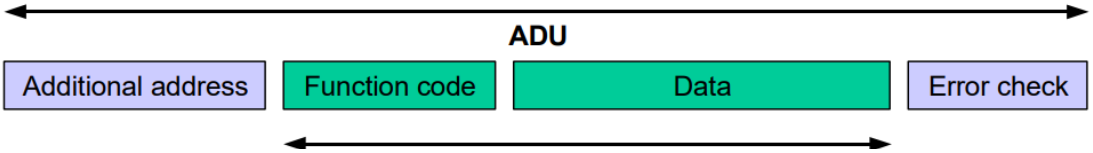
Modbus tarjoaakin helpon kommunikoinnin eri laitteiden välillä kaikenlaisilla verkon rakenteilla. (kuva 2)



Kuva 2 Esimerkki Modbus-verkon rakenne [1, s. 3]

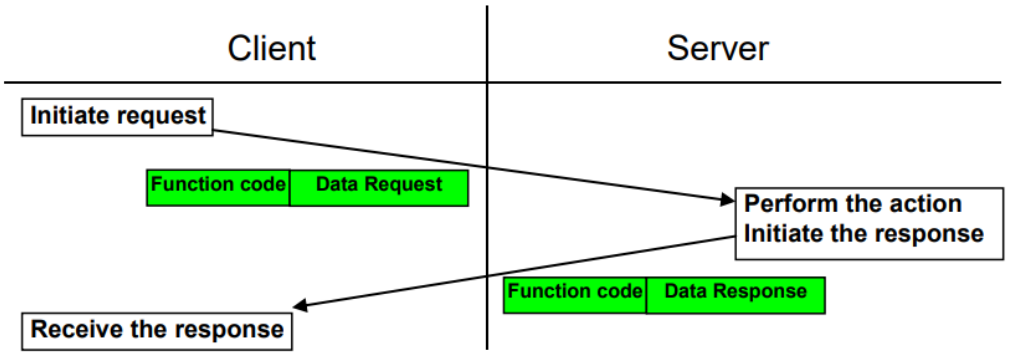
2.2 Protokollan kuvaus

Modbus-protokolla määrittää yksinkertaisen Protocol Data Unitin (PDU), joka on riippumaton taustalla olevista viestintäkerroksista (kuva 3). Modbus-protokollan kartoitus tietyille väylille tai verkko (Ethernet) voi tuoda joitain lisäkenttiä Application Data Unitiin (ADU).



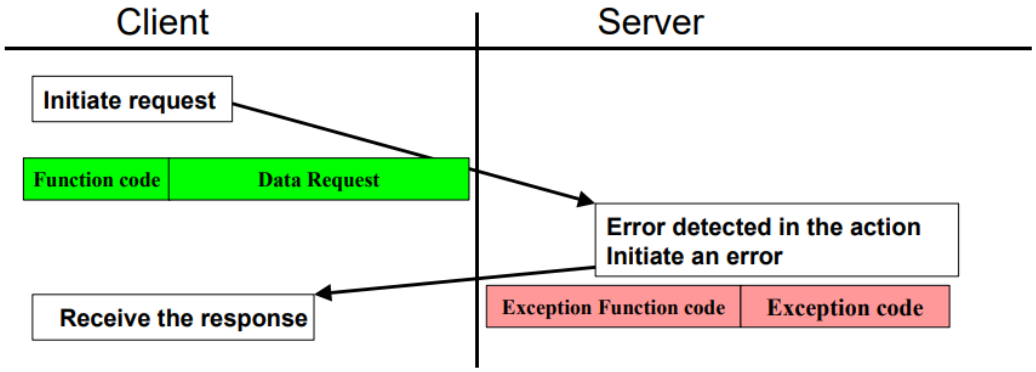
Kuva 3 Yleinen Modbus viestikehys [1, s. 3.]

ADU muodostetaan isäntälaitteessa, joka lähettää pyynnön väylään. Pyyntö sisältää orjalaitteen osoitetiedon ja toimintakoodin, jolla kerrotaan orjalaitteelle



Kuva 4 Onnistunut pyyntö [1, s. 4.]

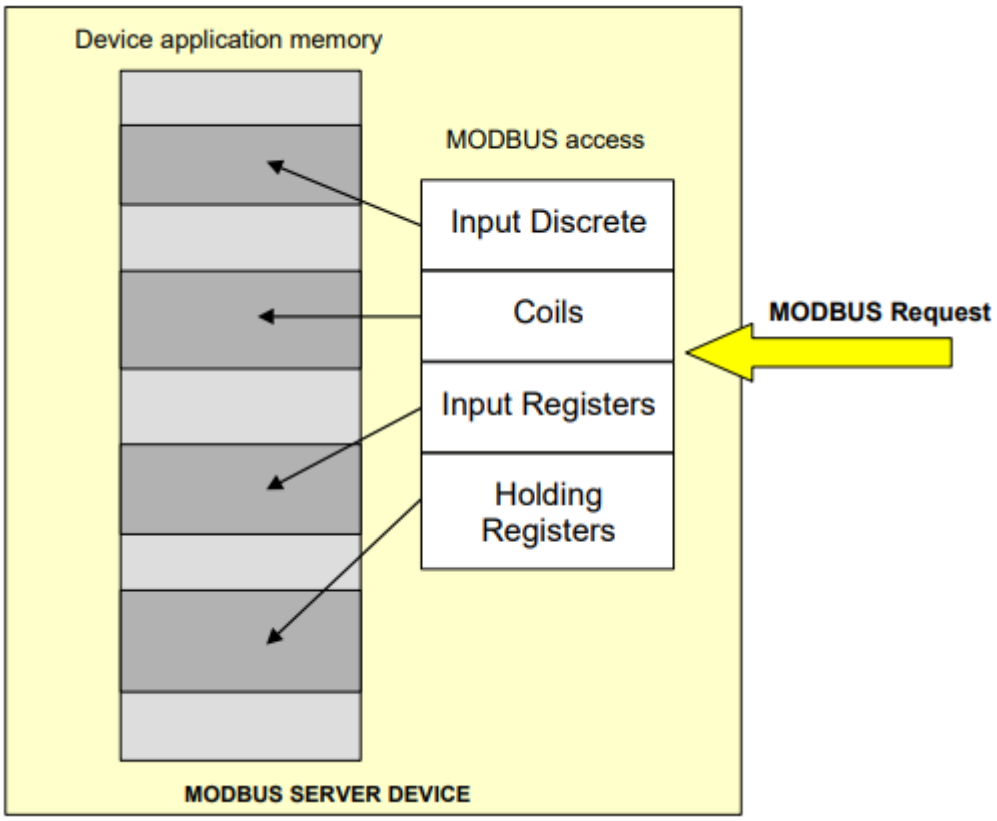
minkälainen, toiminto halutaan suorittaa. Orjalaitteet tarkastavat lähetetystä pyynnöstä osoitetiedon. Jos pyyntö koskee kyseistä laitetta, se lähettää vastauksen. Onnistuneessa pyynnössä vastaus sisältää toimintakoodin ja datan (ks. kuva 4). Virhetilanteessa orja vaihtaa toimintakoodin tilalle poikkeustoimintakoodin (Exception function code) ja lähettää vastauksen isännälle (kuva 5). [2, s. 3.-4.]



Kuva 5 Epäonnistunut pyyntö [1, s. 4.]

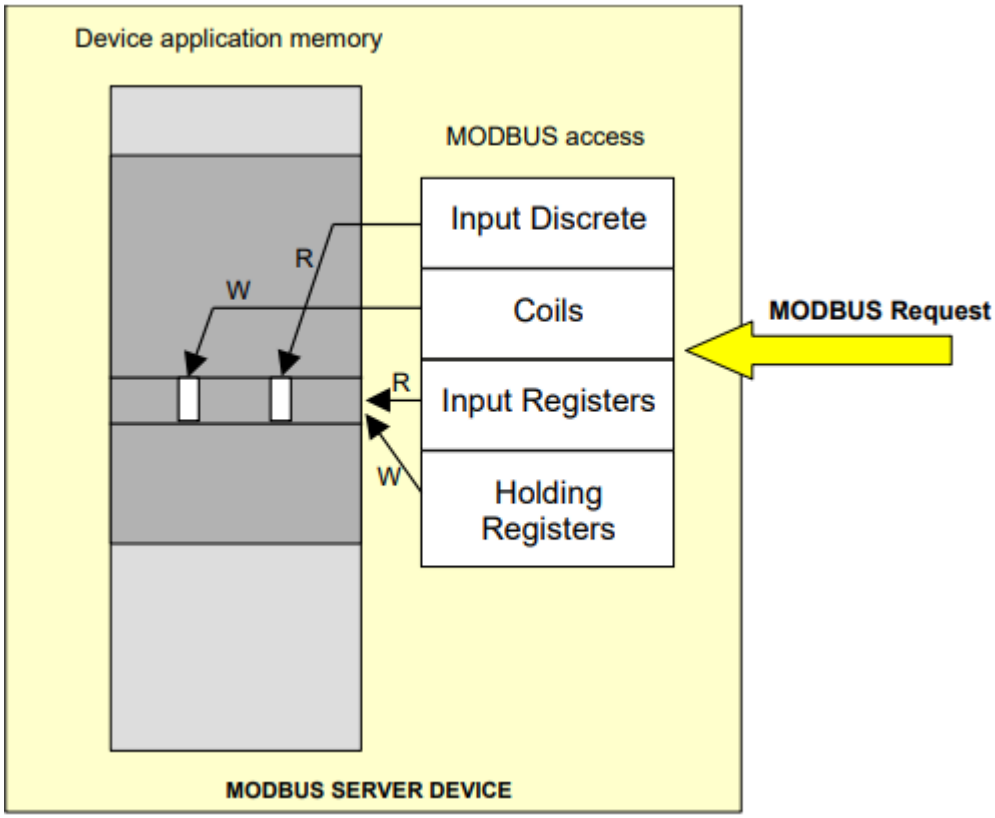
Modbus-protokollassa osoitteet ja data esitetään ‘Big-Endian’-muotoisina. Tämä tarkoittaa sitä, että kun esimerkiksi luetaan 16 bitin rekisteristä tiedot 0x1234 – ensimmäisenä luetaan bitit 0x12 ja sitten 0x34. [1, s. 5.]

Modbusissa on käytössä neljä eri tietotyyppiä, joille jokaiselle löytyvät omat toimintakoodinsa. Nämä tietotyypit ovat discretes input, coils, input registers ja



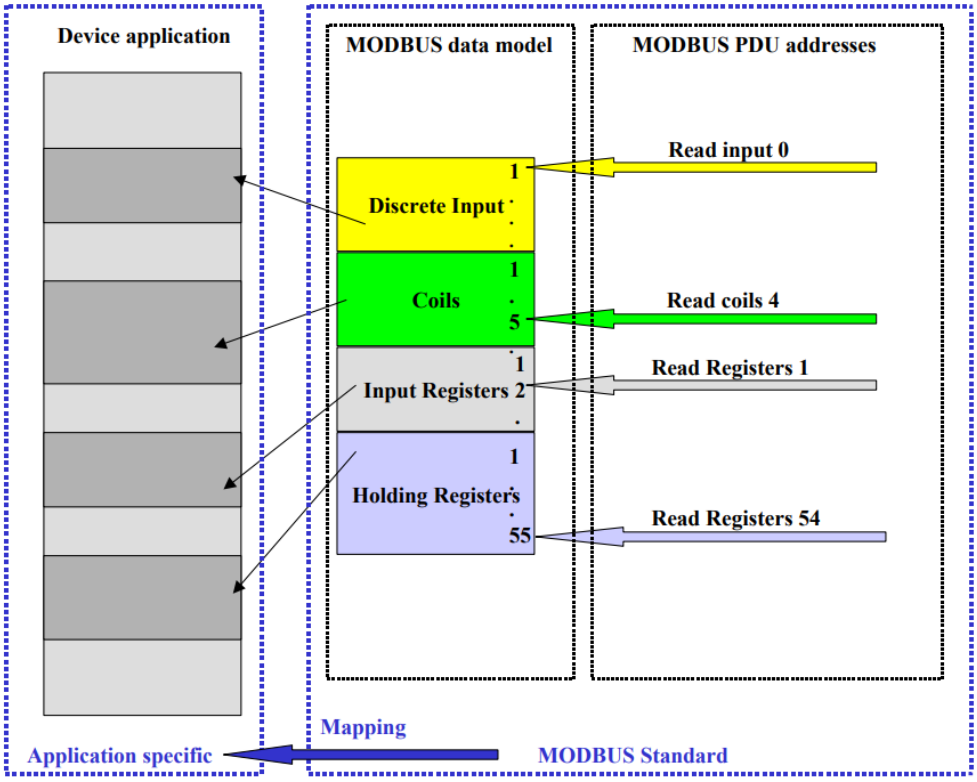
Kuva 6 Modbus-tietomalli omilla lohkoilla [1, s. 6.]

holding registers. Discrete inputs ja coils ovat binäärimuotoisia tietoja, kun taas input registers ja holding registers ovat tavumuotoisia. Standardi ei määrittele laitteiden rekisterien toimintaa, vaan rekisterit voidaan toteuttaa monella eri tavalla. Esimerkkinä otetaan laite, jonka tietotyypit on kaikki jaettu omiin lohkoihinsa (ks. kuva 6). Tällaista ratkaisua joudutaan käyttämään, kun laitteessa on digitaalisia ja analogisia sisään- ja ulostuloja. Jos taas laitteessa ei ole kuin digitaalisia tuloja, voidaan laitteen kaikki data sisällyttää yhteen lohkoon (kuva 7). [1, s. 6.]



Kuva 7 Modbus-tietomalli yhdellä lohkolla [1, s. 7.]

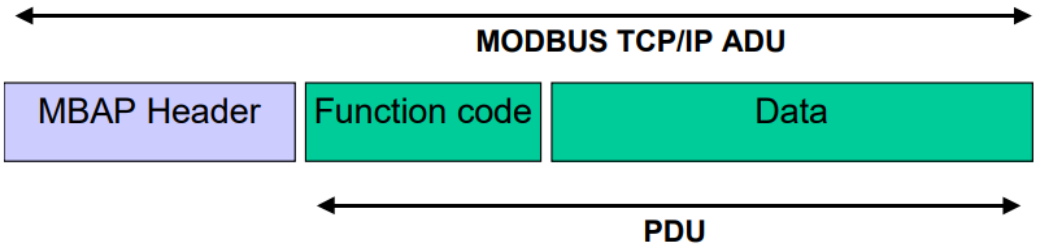
Modbus-protokolla määrittää tarkasti PDU-osoitussäännöt, joissa jokainen data osoitetaan väliltä 0-65535. Se myös määrittelee Modbus-tietomallin, joka koostuu 4 lohkosta. Nämä tietomallit koostuvat monesta elementistä, jotka ovat numeroituna 1-n (kuva 8). [1, s. 7.]



Kuva 8 Modbus-osoitemalli [1, s. 8.]

2.3 Modbus TCP/IP

Modbus TCP/IP:ssä käytetään samaa PDU:ta ja viestin kehys on muutettu (ks. kuva 9). Kehys ei TCP/IP:ssä sisällä enää orjalaitteen osoitetta eikä virheentar-



Kuva 9 Modbus TCP/IP-viestikehys [2, s. 4.]

kastusta. Modbus TCP/IP:n viestiin on lisätty MBAP-otsikko, joka sisältää tapahtumatunnisteen, protokollatunnisteen, viestin pituuden sekä yksikkötunnisteen (ks. kuva 10). [2, s. 3.-4.]

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

Kuva 10 MBAP-otsikkoon sisältyvät tiedot [2, s. 5]

3 Kaasuvalvonta

Kaasuvalvonnan pääsääntöisenä tehtävänä on suojata ihmistä, prosessilaitteita ja ympäristöä. Kaasuvalvonnalla voidaan saavuttaa myös taloudellisia hyötyjä, jos sillä voidaan ilmaista kaasuvuotoja. Tämä tarkoittaa yleisesti tarkempaa mittausta.

Kaasuvalvontaa suunniteltaessa tulee ottaa huomioon mitattavien kaasujen ominaisuudet. On tärkeää, että mittalaite sijoitetaan oikealle paikalle kaasunominaisuuksien mukaan. Tämä määräytyy pääsääntöisesti sen mukaan, onko mitattava kaasu ilmaa kevyempi vai raskaampi. Väärin asennetulla laitteella ei saada mittaustietoa eikä myöskään hälytystietoja. Tärkeää on myös ottaa huomioida kaasujen muita ominaisuuksia, esimerkiksi sen reagointi veden kanssa. Kun kaasu reagoi veden kanssa voimakkaasti, tulee vuototilanteessa ottaa huomioon myös ilmankosteus. Tällöin, jos kaasua vuotaa huonetilaan, on mitattava kaasu eri kuin käytössä oleva kaasu. Esimerkiksi jos yrityksellä on tuotannossa käytössä booritrikloridia (BCl_3), mitataan sen mahdollisesta vuotokohdasta kloorivetyä (HCl). Tämä tehdään siitä syystä, että booritrikloridi reagoi

kiivaasti ilmankosteuden kanssa ja muodostaa kloorivetyä ja boorihappoa. (Liite 3.)

Mittalaitteen sijoitus tulisi kohdistaa mahdollisiin vuotokohtiin ja/tai kaasun poistumisreitille. Todennäköisimmät vuotokohdat ovat venttiilit sekä laitteet, joissa kaasu on käytössä. Poistumisreittiin vaikuttaa kaasun suhteellinen höyryn tiheys, jota verrataan ilmaan. Ilmaa kevyemmät kaasut pyrkivät poistumaan niille helpointa reittiä, joita ovat muun muassa ilmanvaihtokanavat. Tällöin mittaus voidaan sijoittaa joko ilmavaihdon suulle tai kanavaan. Raskaammat kaasut taas puolestaan hakevat reittiä alhaalta, mahdollisesti lattiakaivoista, jolloin mittaus on sijoitettava viemäriputkeen tai sen suulle. Myös osa raskaammistakin kaasuista poistuu ilmanvaihdoin kautta, kun ilmanvaihto sekoittaa tehokkaasti huoneilmaa. Suunnittelussa on otettava huomioon rakenteet, joihin kaasu voisi mahdollisesti kerääntyä vuototilanteessa. Jos kaasua pääsee kertymään tällaiseen rakenteeseen voi seurauksena olla pahimmassa tapauksessa räjähdys.

Tärkeimpiä kaasuvalvonnan teknisiä ominaisuuksia ovat luotettavuus, huollettavuus ja käytettävyys. Näistä kaikkein tärkeimpänä luotettavuus. Tähän voidaan vaikuttaa laitevalinnoilla, suunnittelulla sekä käyttöönotolla, johon kuuluu kaikkien järjestelmään kuuluvien mittalaitteiden testaus. Testauksella varmistutaan siitä, että jokainen yksittäinen mittalaite toimii ja mittaustietoa syntyy. Mittalaitteet testataan siihen tarkoitetulla kaasulla sekä tarvittaessa kalibroida. Nykyisin mittalaitteet voivat sisältää älyn, ja nykyisillä väyläratkaisuilla sitä voidaan helposti hyödyntää. Älyllä varustetut anturit ilmoittavat vanhentuneesta kalibroinnista sekä vioista. Kaikilla edellä mainituilla pyritään estämään mahdollisen kaasuvuodon aiheuttamia henkilö-, ympäristö- sekä materiaalivahinkoja.

Toisena tärkeänä tekijänä on huollettavuus, johon vaikuttavat myös laitevalinnat. Tärkeimpiä ominaisuuksia huollettavuudessa ovat anturien vaihdot, kalibrointi sekä tarvittaessa hälytysrajojen ja viiveiden muutos. Käyttäjälle tärkein tekijä on käytettävyys. Käytettävyydessä tulee ottaa huomioon, että käyttäjät eivät välttämättä ole teknisesti koulutettuja, minkä vuoksi käyttäjille olisikin hyvä tar-

jota käyttökoulutusta. Yleisesti käyttäjä ei käytä järjestelmää muutoin kuin hälytys- tai vikatilanteessa. Tämä on huomioitava järjestelmää suunniteltaessa. Järjestelmän tulisi olla selkeä ja itseään selittävä, jotta pitkänkin käyttämättömyyden jälkeen sitä voitaisiin käyttää ilman epäröintiä.

Kaasuvalvonnassa mittaukset perustuvat kaasuille määritettyjen haitallisiksi tunnettujen pitoisuuksien, eli HTP-arvojen mittaukseen. HTP-arvot ovat ilman epäpuhtauksille määritettyjä arvoja, joiden on todettu olevan ihmiselle haitallisia. HTP-arvoja määritetään niiden vaikutuksesta riippuen eri mittaisille ajanjaksoille. Arvo voidaan antaa kahdeksan tunnin keskiarvolle, joka voidaan ylittää lyhyinä ajanjaksoina, kunhan arvo ei ylity laskennallisesti. Aineille, joilla esiintyy vaikutuksia lyhyilläkin ajanjaksolla, määritetään 15 minuutin arvo. Pitoisuus ei saa nousta tähän arvoon useammin kuin kerran tunnissa, mutta kuitenkin enintään neljä kertaa kahdeksan tunnin työvuoron aikana. [3, s. 10-11]

> HTP-arvot on vahvistettu työturvallisuuslain (738/2002) 38 § 4 momentin nojalla annetulla sosiaali- ja terveysministeriön asetuksella (538/2018). [3, s. 10.]

Osalle kaasuista ei määritetä HTP-arvoja olleenkaan, koska ne voivat suurina pitoisuuksina vaikuttaa tukahduttavasti ilman muita merkittäviä vaikutuksia. Tällaisessa tilanteessa seuraukset voivat olla hengenvaaralliset. Näissä tapauksissa mitataan happikatoa. [3, s. 57.]

4 Kaasuvalvontalaitteet

Kaasuvalvontajärjestelmälaitteet voidaan jakaa pääsääntöisesti kolmeen ryhmään: mittayksiköihin, joilla mitataan kaasunpitoisuuksia, näyttölaitteisiin, joilla voidaan tarkastella järjestelmän tilaa ja siihen kuuluvia mittayksiköitä, sekä logiikkaan, joka koostuu yleisesti PLC:stä tai PC:stä, jolla voidaan ohjata I/O-laitteita. Ohjelmointi tulee olla toteutettuna niin, että voidaan määrittää mittapisteiden asetukset, tarkkailla järjestelmän tilaa, saada mittaustietoa reaaliajassa sekä kerätä mittauksista historiatietoja. Järjestelmään kuuluvat myös hälytys- ja

lukituslaitteet, joilla pyritään estämään ihmiselle ja ympäristölle aiheutuvat vahingot varoittamalla kaasuvuodoista hälytysvalolilla ja äänimerkillä sekä usein myös jatkohälytyksillä rakennusautomaatioon. Lukituslaitteilla voidaan muun muassa katkaista kaasunsyöttö tai sähköt toimilaitteita vuotavasta tilasta.

Mittayksiköt koostuvat laiteosasta ja anturista. Laiteosa käsittelee anturin tuottamaa tietoa, muodostaa hälytystiedot laitteen rekisteriin sekä esittää pitoisuuden näytöllä, jos laitteessa on sellainen asennettuna. Riippuen laitteesta voidaan ulostulona käyttää joko milliamppeeri tai digitaalista viestiä. Jos käytetään digitaalista, voidaan hälytystiedot lukea suoraan laitteelta. Muussa tapauksessa joudutaan hälytystiedot muodostamaan milliampeeriviestistä, jossa mittausalue on yleensä välillä 4-20 mA.

Anturityyppejä löytyy useita erilaisia: muun muassa elektrokemiallinen, pellistori ja infrapuna. Anturit on suunniteltu mittaamaan yleisesti aina yhtä kaasua, mutta osassa niissä esiintyy myös ristiherkkyyttä muihin kaasuihin.

4.1 Honeywell Midas-t-004

Tässä projektissa testaukseen käytettiin Midas-t-004 -mittayksikköä sekä hiilidioksidi- ja otsoni-anturia.

4.1.1 Kuvaus

Midas on Honeywellin valmistama mittayksikkö, jonka toiminto perustuu kalvopumpun avulla kerättävään näytteeseen. Laitetta voidaan käyttää joko paikallisesti tai etäkäyttöisenä. Laite sisältää erillisen anturiosan, joka asennettaan laitteen sisälle. Antureita on laaja valikoima tukahduttavien, myrkyllisten, syttyvien, syövittävien ja hapettavien kaasujen mittaukseen. Antureita on yleisesti käytössä puolijohdeteollisuudessa, muilla teollisuuden aloilla sekä tutkimuksen eri aloilla. [4, s. 1-2.]

Midas on seinään kiinnitettävä mittayksikkö, josta voidaan paikallisesti seurata kaasunpitoisuutta, hälytyksiä, vikatietoja sekä laitteen tilaa sen näytöltä. Laitteeseen asennettujen painonappien avulla voidaan laitetta operoida ja konfiguroida. [4, s. 1-2.]

Fault Relay Configuration	Relay 1	Relay 2	Relay 3
Instrument Fault Only (1FLt)	Alarm 1	Alarm 2	Instrument Fault
Separate Fault Relays (2FLt)	Any Alarm	Maintenance Fault	Instrument Fault
Combined Fault Relay (CmbF)	Alarm 1	Alarm 2	Any Fault
Network Remote Control (nEtr)	Remote control of relays via Modbus/ TCP or LonWorks®		

Kuva 11 Relekonfiguraatiot [4, s. 4-10]

Laitteesta on joustavat virta- ja viestintäominaisuudet. Näihin kuuluvat laitteeseen asennetut kolme relettä, jotka määritetään ohjelmallisesti toimimaan halutulla tavalla (kuva 11). Laitteesta on 0-20 mA analoginen ulostulo sekä Modbus TCP. Virransyöttö toteutetaan joko 24 VDC tai PoE:llä (Power over Ethernet). [4, s. 1-2.]

4.1.2 Huollettavuus

Midas on täysin huollettava tuote, joka on suunniteltu modulaarisista komponenteista, jotka voidaan helposti vaihtaa koulutetun huoltohenkilön toimesta. Modulaarisuutensa ansiota sen huoltaminen on nopeaa. Tämä minimoi aikaa,

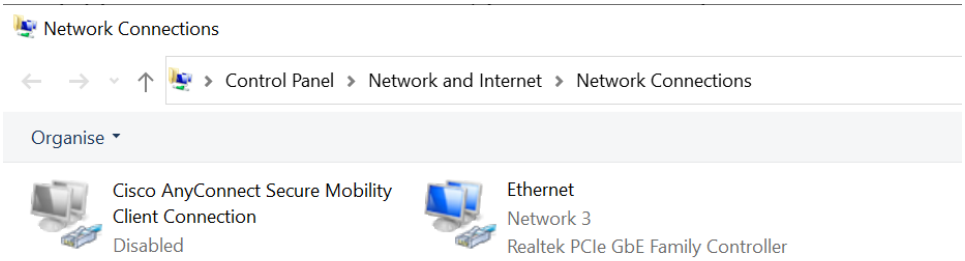
jolloin mittayksikkö ei ole käytössä. Valmistaja ilmoittaa jokaiselle komponentille oman huoltovälin (kuva 12). [4, s. 8-2.]

Recommended Maintenance Schedule	
Component	Frequency
Pump	2 years/as needed
Pyrolyzer (all models)	1 year
Internal Filter	2 years/as needed
External Sample Line Filter	780248 3-6 months 1991-0147 3-6 months 1830-0055 3-6 months 1830-0027 1 month
Leak Check	Leak Check every 6 months or after pump, pyrolyzer or internal filter replacement.
Bump Test	6 months
Flow Calibration	Flow Calibrate after pump, pyrolyzer or internal filter replacement.

Kuva 12 Komponenttien huoltovälit [4, s. 8-2.]

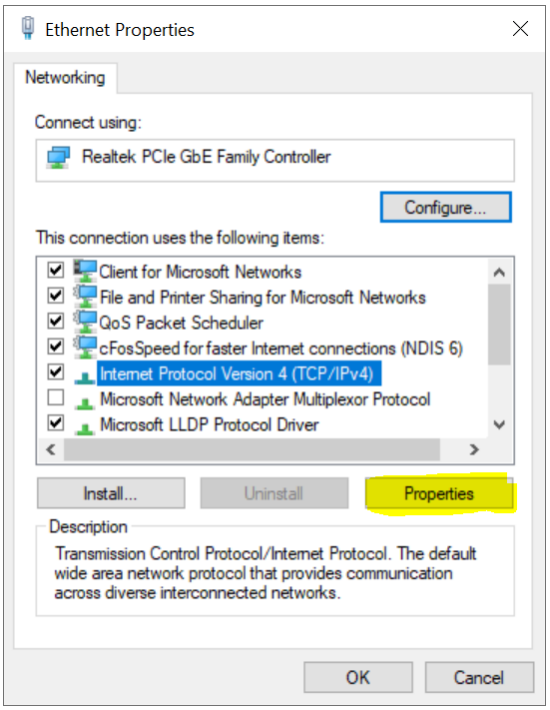
4.1.3 Verkkopalvelin

Midas-mittayksiköstä löytyy HTML-verkkopalvelin, johon voidaan yhdistää tietokoneella selaimen avulla (esim. Google Chrome). Selaimella avautuva sivu toimii laitteen käyttäjän rajapintana, josta päästään käsittelemään kaikkia laitteen asetuksia sekä katselemaan hälytyslokia. Yhdistäminen onnistuu laitteesta löytyvän Ethernet-portin avulla, joka käyttää TCP/IP -protokollaa. Jos Midas ei ole yhteydessä DHCP-palvelimeen, joudutaan osoitteet muuttamaan käsin. Midakseen tehdasasetettu IP-osoite on 169.254.60.47 ja netmask 255.255.255.0. PC:lle määritetään IP-osoitteeksi esimerkiksi 169.254.60.2 ja netmaskiksi sama kuin Midaksessa. [4, s. 13-2.]



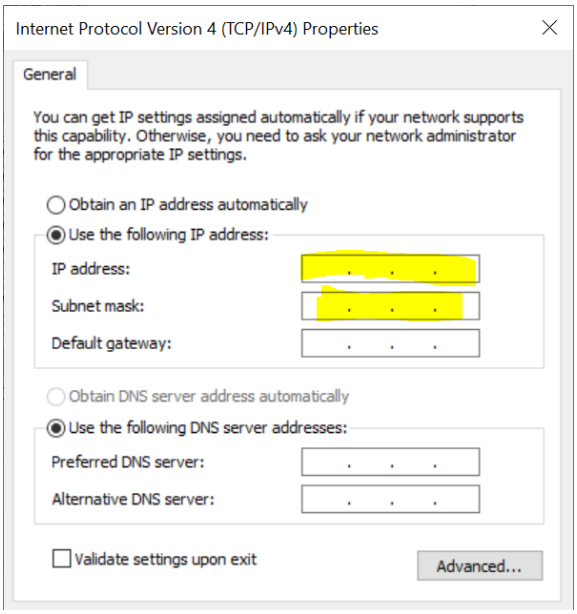
Kuva 13 Verkkoasetukset

PC:llä IP-osoite ja netmaski vaihdetaan Windowsin verkkoasetuksista, joihin päästään verkkoyhteydet sivulta (kuva 13). Valitaan verkkokortti, jota halutaan



Kuva 14 Verkkoadapterin ominaisuudet

käyttää ja mennään sen ominaisuuksiin. Kuvassa 14 nähdään verkkoadapterin ominaisuudet, joista valitaan TCP/IPv4 ja painetaan properties-painiketta. Täältä päästään määrittämään verkkoadapterille IP-osoite ja netmask (kuva 15).



Kuva 15 TCP/IPv4 -ominaisuudet

4.1.4 Modbus/TCP -rajapinta

Midas on Modbus/TCP-palvelin, joka tukee modbus-protokollan komentoa 03(”read holding registers”) rekistereille 40001-40081. Komennoilla 06 (“write single register”) ja 16 (“write multiple registers”) voidaan muun muassa resetoida laitteelta hälytykset, kun kirjoitetaan 350 rekisteriin 40021 ja 13862 rekisteriin 40022. [4, s. A-2.]

5 Twincat 3

5.1 TE1000

TE1000 on TwinCAT -kehitysympäristö, jossa voidaan kätevästi konfiguroida ohjain, taajuusmuuttaja ja I/O. Lisäksi työkalu sisältää Beckhoffin turvallisuusratkaisun TwinSAFEn konfiguroinnin ja ohjelmoinnin.

TwinCAT 3 Engineering on Visual Studioon integroitu kehitystyökalu, jossa voidaan suorittaa PLC-ohjelmointia sen ilmaisella versiolla. C++-ohjelmointi vaatii toimiakseen ohjelmiston täyden version. [5, s. 8.]

5.2 TE2000 HMI Engineering

TwinCAT HMI on työkalu käyttöliittymien luomiseen. Kuten TwinCAT 3 Engineering on se integroitu Visual Studioon. Konfiguroinnissa käytetään WYSIWYG-editoria eli ohjelmointia ei tarvita, vaan komponentit ja niiden toiminnot voidaan valita kehitystyökalulla.

Kehitysympäristön tarjoamat ohjaimet voidaan lisätä käyttöliittymään ja linkittää reaaliaikaisiin muuttujiin esimerkiksi PLC:stä tai C++-moduulista. Käyttäjä voi luoda myös omia ohjaimia, joita voidaan uudelleen käyttää ohjelmassa samanlailla kuin kehitysympäristön tarjoamia. Kehitysympäristössä toimintoja voidaan myös luoda JavaScript-ohjelmointikielellä. Käyttöliittymää voidaan testata koko sen kehityksen ajan sisäänrakennetun web-palvelimen avulla. [6, s. 11.]

5.3 TF6250 Modbus TCP

TwinCAT Modbus toimii yhdyskäytävänä Modbus TCP -laitteiden ja TwinCAT-järjestelmän välillä. Se tarjoaa sekä palvelimen että asiakkaan toiminnallisuuksia. Palvelintilassa useiden järjestelmien muistialueet voidaan yhdistää suoraan modbus-muistialueisiin. PLC-kirjastolla voidaan toteuttaa kommunikointi Modbus-laitteiden kanssa. [7, s. 8.]

6 Ohjelmointi

6.1 PLC

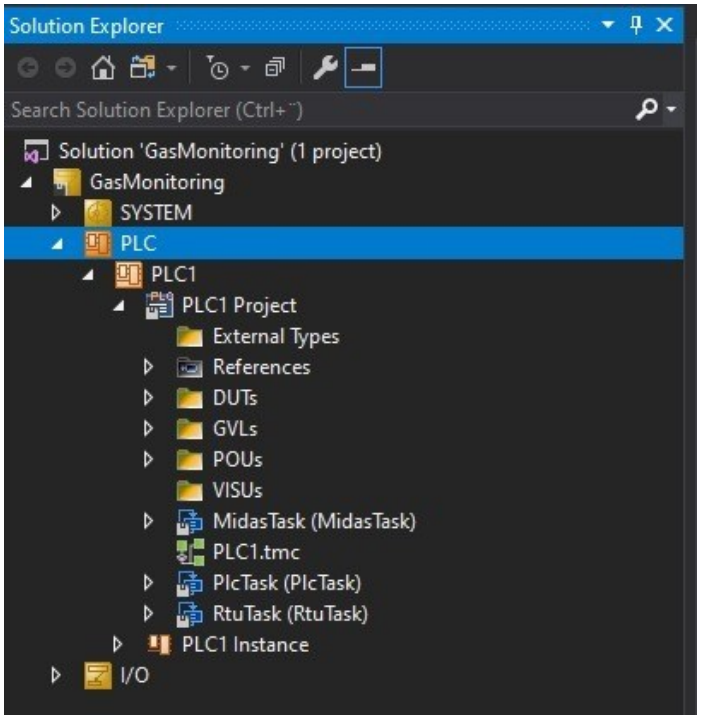
Ohjelma jaettiin tässä projektissa kahteen eri taskiin. Ensimmäisenä taskina oli pääohjelma, jossa hoidettiin lähtöjen ohjaus sekä tallennettiin mittayksiköiden pysyvät tiedot. Toisena taskina oli mittayksiköiden tietojen luku. Pysyvien tietojen tallennukseen käytettiin hyväksi Twincat-ohjelmiston tarjoamia persistent-muuttujia, jotka voidaan kirjoittaa kovalevylle joko manuaalisesti tai kun PLC:stä katkaistaan virta. Tässä projektissa käytettiin manuaalista toimintoa, sillä oli tiedossa, ettei tulevassa paneeli-pc:ssä ollut UPS-yksikköä, jolloin odottamattoman virran katkeamisella olisi vaikutuksena tallennettujen tietojen menetys. Ohjelman eri toiminnot jaettiin function blockeihin (FB).

Pysyviin tietoihin määritetään mittayksikölle nimi, sijainti, laiteryhmä, magneettiventtiiliryhmä, IP-osoite, mitattavan kaasun nimi ja kaava, yksiköt, desimaalien määrä pitoisuudessa, laiteen hiljennys sekä aktivointi. Nämä tiedot tallennetaan kovalevylle ja ne otetaan käyttöön, kun ohjelma käynnistyy.

Ohjelmalla voidaan lukea mittayksikölle määritetyt kaasun nimi ja kaava, yksiköt, desimaalien määrä, laitteen ja anturin sarjanumerot sekä mittayksikön hälytysrajatiedot 1 ja 2. Nämä luetaan laitteen rekistereistä ja voidaan tallentaa kovalevylle. Jokaiselta mittayksiöltä saadaan hälytystiedot sekä useita vikatietoja.

Laiteryhmille pystytään määrittämään nimi sekä sijaintitieto. Laiteryhmät hälyttävät sen mukaan, missä tilassa siihen kuuluvat laitteet ovat.

Työ aloitettiin luomalla uusi projekti TwinCAT 3 -ohjelmistolla, joka sisälsi projektin perusrakenteen sekä PLC-ohjelman (kuva 16). Sen jälkeen lisättiin pro-



Kuva 16 Luotu projekti

jektiin tarvittavat lisenssit. Lisenssien lisääminen tapahtui SYSTEM -> license -> Manage licenses. Manage licenses -välilehdeltä löytyivät kaikki lisenssit, joita TwinCAT:ssa on tarjolla. Projektiin määritettiin lisenssit: TC1200, TF2000 ja

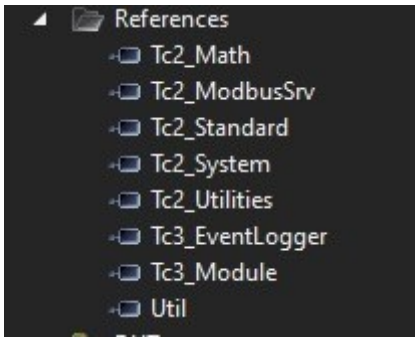
Order No	License	Instances
TC1200	TC3 PLC	cpu license
TF2000	TC3 HMI Server	cpu license
TF6250	TC3 Modbus-TCP	cpu license

Kuva 17 Projektissa käytetyt lisenssit

TF6250 (ks. kuva 17), minkä jälkeen ne aktivoituivat, kun seuraavan kerran painettiin Active Configure-painiketta  . Koska projektissa oli käytössä Modbus-TCP-lisenssi, ladattiin tietokoneelle Modbus-TCP gateway-ohjelma Beckhoffin sivuilta https://www.beckhoff.com/fi-fi/products/automation/twincat/tfxxxx-twincat-3-functions/tf6xxx-tc3-connectivity/tf6250.html. Latauksen jälkeen gateway-

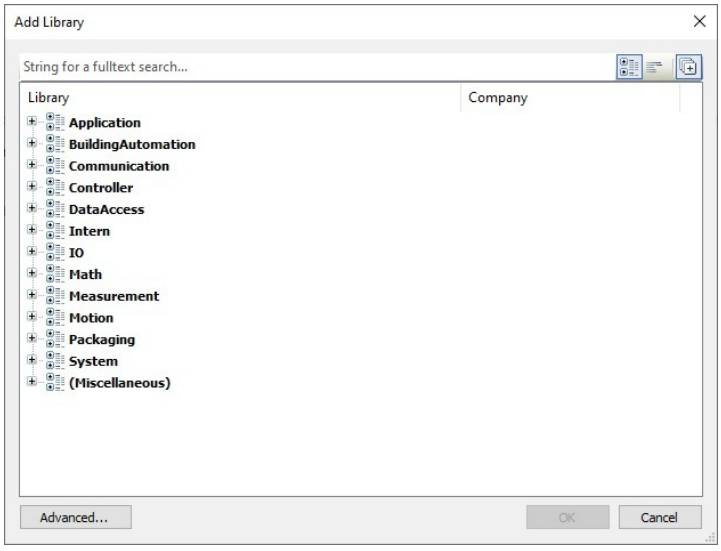
ohjelma asennettiin koneelle, minkä jälkeen TwinCAT 3-ohjelmistolla pystyttiin kommunikoimaan Modbus-TCP-laitteen kanssa.

Tässä vaiheessa lisättiin projektiin muita tarvittavia kirjastoja, jotka näkyvät kuvassa 19. Kirjaston lisäys tapahtui projektipuun references-kohdassa, jota pai-



Kuva 19 Käyttöönotetut kirjastot

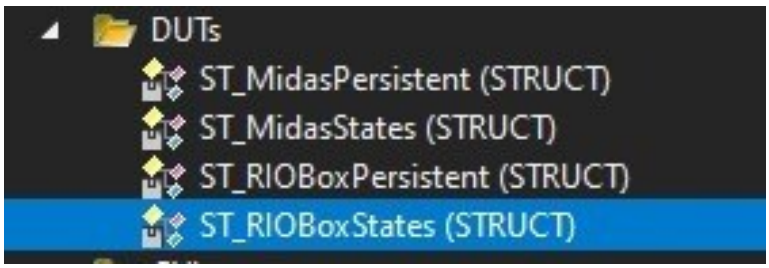
namalla hiiren oikealla-painikkeella valittiin add library. Tätä painamalla aukesi kuvassa 20 näkyvä ikkuna, josta hakukenttään kirjoittamalla haettiin halutut kir-



Kuva 20 Add library -ikkuna

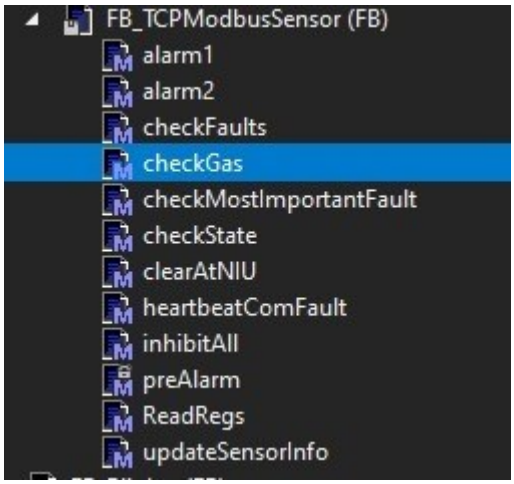
jastot. Kirjastot jouduttiin lisäämään yksitellen. Valittiin haluttu kirjasto hiiren vasemmalla painikkeella ja hyväksyttiin valinta painamalla ok-painiketta.

Tämän jälkeen PLC-projektiin tehtiin tietorakenteita, joita projektissa käytettiin. Nämä tietorakenteet sisälsivät laiteryhmien ja mittayksiköiden pysyviä tietoja sekä tilatietoja (kuva 21).



Kuva 21 Tietorakenteet

Seuraavaksi toteutettiin PLC-ohjelman toimintoja. Luotiin FB_TCPModbusSensor [liite 1], jonka tehtävänä oli kommunikoida Modbus-TCP laitteen kanssa



Kuva 22 FB_TCPModbusSensor ja luodut metodit

sekä palauttaa laitteen tilatiedot. Toiminnot jaettiin metodeihin, joista preAlarm()- ja alarmX()-metodit päivittävät laitteen hälytystiedot FB:n ulostuloille. CheckFaults()-metodilla tarkastetaan laitteelta tulleet vikatiedot. Metodilla checkGas() voidaan tarkastaa laitteen ilmoittama GasID ja SensorCartridgeID, joiden perusteella päivitetään mitattavan kaasun nimi, kaava ja mittausalue.

Laitteen viimeisin vikatieto voidaan tarkastaa metodilla checkMostImportant-Fault(), joka palauttaa vian numeron sekä valmistajan ilmoittaman kuvauksen vialle. ReadRegs()-metodilla luetaan mittayksikön modbus-rekisterit (holding registers).

```
fbReadRegs(
     sIPAddr        := sIpAddrs,
     nTCPPort       := nTCPPort,
     nUnitID        := nUnitID,
     nQuantity      := nQuantity,
     nMBAddr        := nMAddr,
     cbLength       := SIZEOF(arrData),
     pDestAddr      := ADR(arrData),
     tTimeout       := tTimeout,
     bExecute       := TRUE,
     bBusy          => bBusy,
     bError         => bError,
     nErrId         => nErrorId
);

IF NOT bBusy THEN
     fbReadRegs(bExecute := FALSE);
END_IF

IF fbReadRegs.bError THEN
     FOR i := 0 TO TO_INT(nQuantity) DO
          arrData[i] := 0;
     END_FOR
END_IF
```

Esimerkkikoodi 1. ReadRegs()-metodi

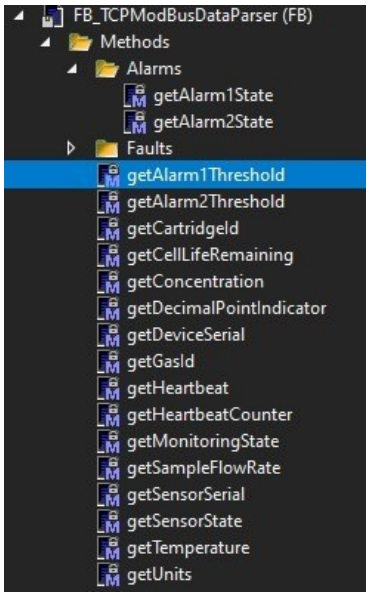
Edellisessä esimerkissä nähdään, kuinka fbReadRegs FB on konfiguroitu. FB:lle annetaan laitteen ip-osoite, TCP-port(502), UnitID, nQuantity, nMBAddr, pDestAddr, cbLength, tTimeout ja bExecute. UnitID:n arvon tulee olla 16#FF, jos laitteen osoitteena käytetään TCP/IP:tä. NQuantity määrittää, montako rekisteriä luetaan, ja arvon on oltava suurempi kuin 0. NMBAddr kertoo aloitusosoitteen, josta rekisterit aloitetaan lukemaan. CbLength sisältää kohdepuskurin maksimikoon tavuina ja pienin koko on oltava nQuantity kertaa kaksi. PDestAddr sisältää kohdepuskurin osoitetiedon, mihin rekisterit luetaan. Tämä voi olla yksittäinen muuttuja, lista tai rakenne, joiden osoite saadaan ADR-operaattorilla. BExecute:n nousevalla reunalla FB aktivoituu. TTimeout ilmoittaa ajan, jota ADS-komennon suorittaminen ei saa ylittää. Mikäli fbReadRegs ilmoittaa virheestä, tyhjennetään kohdepuskurin tiedot, ja jos virhettä ei ole, eikä bBusy

ole TRUE, on lukeminen onnistunut ja voidaan bExecuteen kirjoittaa FALSE. [6, s. 22.]

UpdateSensorInfo()-metodilla päivitetään laitteelta luetut tiedot kohdemuuttujiin ja tallennettiin kovalevylle. Näihin kuuluvat laitteen ja anturin sarjanumerot, hälytysrajat, yksiköt sekä mitattava kaasu.

Modbus-laitteen yhteyden tarkistamiseen voidaan käyttää laitteen heartbeat-bittiä, joka muutti arvoaan kahden sekunnin välein. HeartbeatComFault()-metodilla tarkistetaan edellä mainitun bitin tilaa. Jos tila ei muutu kuuden sekunnin aikana, metodi palauttaa FALSEn, eli yhteys on kunnossa. Kun metodi palauttaa TRUEn, on yhteydessä vika (liite 3).

Luettujen tietojen jäsentelyä varten luotiin FB_ModbusDataParser, joka periytyy FB_TCPModbusSensor FB:ssa. FB_ModbusDataParser ottaa sisäänsä muuttu-



Kuva 23 FB_ModbusDataParser -metodit

jan arrData : ARRAY[0..80] OF WORD. Tämä listamuuttuja sisältää kaikki mittayksiköltä luetut modbus-rekisteritiedot tavumuodossa.FB_ModbusDataParser blockissa toiminnot jaetaan metodeihin (kuva 23).

```
METHOD PRIVATE getAlarm1State : BOOL
VAR
     AlarmData      :WORD := arrData[0];
END_VAR

getAlarm1State := TO_BOOL(AlarmData.6);
```

Esimerkkikoodi 2. FB_TCPModbusDataParser.getAlarm1State() -metodi. Metodi palauttaa hälytystiedon binäärimuotoisena. Metodissa voidaan hyödyntää TwinCAT:in piste-operaatiota, jolla voidaan tavumuotoisesta muuttujasta valita bitti ja tarkastaa sen arvo, eli tässä tapauksessa AlarmData.6.

```
METHOD PRIVATE getCartridgeId : INT
VAR
     nCartridgeId :WORD;
     bitMask : WORD := 16#FF00;
END_VAR

nCartridgeId := arrData[1] AND bitMask;
getCartridgeId := TO_INT(SHR(nCartridgeId, 8));
```

Esimerkkikoodi 3. getCartridgeId:ssa käytetään bittimaskia 16#FF00(2#1111_1111_0000_0000), jonka avulla tarkastetaan AND-operaatiolla, mitkä kohdetavun bitit ovat 1. Tämän jälkeen voidaan siirtää kaikkia bittejä oikealle 8 bitin verran SHR-bittioperaattorilla, joka siirtää bittejä oikealle ja korvaa vasemmalta bitit 0.

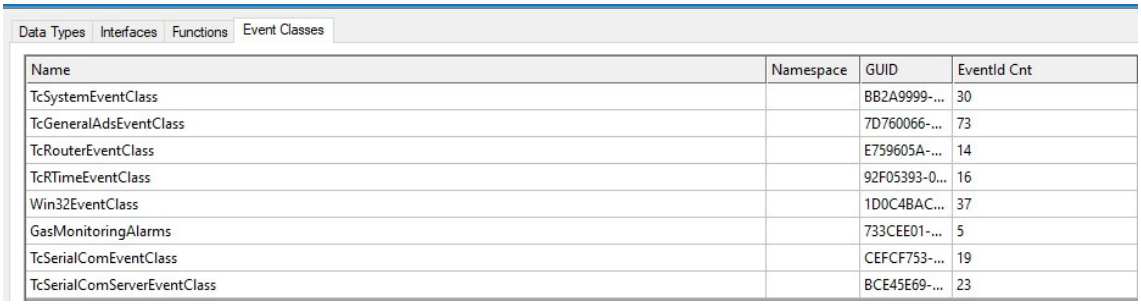
Lähes kaikki metodit tässä projektissa toimivat samalla tavalla paitsi getConcentration(), getAlarmXThreshold() sekä getSensorSerial() ja getDeviceSerial(). Metodeissa getConcentration() ja getAlarmXThreshold() tavumuotoiset rekisterit muutettiin liukuluvuiksi.

GetSensorSerial()- ja getDeviceSerial()-metodit kääntävät tavumuotoiset rekisterit ASCII-muotoon. Esimerkkikoodissa 4 nähdään getSensorSerial() -toiminta. Sarjanumerot koostuvat viidestä modbus-rekisteristä, jotka jokainen sisältävät kaksi kirjainta (liite 1(2)). Bittejä manipuloimalla saadaan jokainen numero eriteltyä, jonka jälkeen tavut muutetaan F_ToCHR:llä ASCII_muotoon. Tämä jälkeen ne voidaan yhdistää toisiinsa CONCAT function blockilla ja lopuksi palauttaa sarjanumero tekstimuotoisena.

```
METHOD PRIVATE getSensorSerial : STRING
VAR
     i              :INT;
     nResult        :WORD;
     bitMask        :WORD     := 16#FF00;
     sSensorSerial  :STRING;
     arrOffSet      :INT      := 75;
END_VAR

FOR i := 0 TO 4 DO
     IF i = 0 THEN
          nResult := arrData[arrOffSet+i] AND bitMask;
          nResult := SHR(nResult, 8);
          sSensorSerial := F_ToCHR(WORD_TO_BYTE(nResult));
          sSensorSerial := CONCAT(STR1 :=sSensorSerial, STR2 := F_To-
CHR(WORD_TO_BYTE(arrData[arrOffSet+i])));
     ELSE
          nResult := arrData[arrOffSet+i] AND bitMask;
          nResult := SHR(nResult, 8);
          sSensorSerial := CONCAT(STR1 := sSensorSerial, STR2 := F_To-
CHR(WORD_TO_BYTE(nResult)));
          sSensorSerial := CONCAT(STR1:= sSensorSerial,  STR2 := F_To-
CHR(WORD_TO_BYTE(arrData[arrOffSet+i])));
     END_IF
END_FOR

getSensorSerial := sSensorSerial;
```

Esimerkkikoodi 4. getSensorSerial().

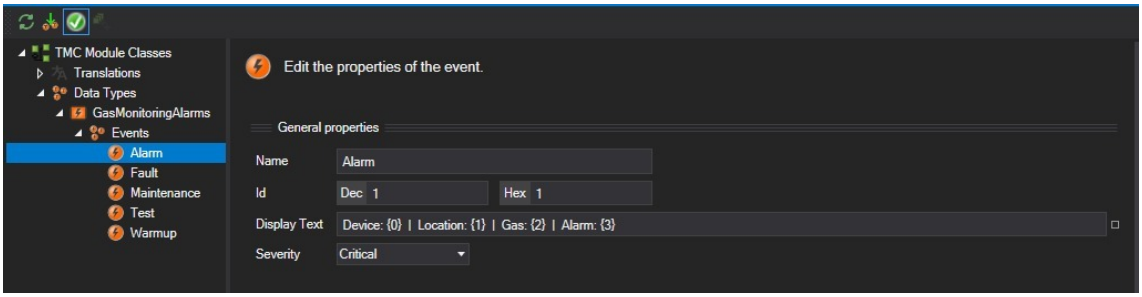
Seuraavana tehtiin FB:tä, jonka tarkoituksena oli luoda tapahtumia (Events) mittayksiköiden tilatiedoista. Tätä ennen luotiin vielä uusi tapahtumaluokka (Event Class) GasMonitoringAlarms, joka on tietotyyppi TwinCAT-tyyppijärjestelmässä (kuva 24). GasMonitoringAlarm -tapahtumaluokka piti sisällään useamman tapahtuman: hälytys (Alarm), vika (Fault), huolto (Maintenance), testi (Test) sekä



Kuva 24 Event Classes lista

lämmitys (Warmup). Tapahtumaa luodessa sille annetiin nimi, id, näytettävä teksti sekä tapahtuman vakavuus. Kuvassa 25 nähdään, kuinka

hälytystapahtuma toteutettiin. Näytettävä teksti voidaan täydentää myöhemmin ohjelmassa (ks. alempaa esimerkkikoodi 5). Käytetään FB_TcAlarm-funktion propertya ipArguments, kun halutaan täydentää tapahtuman viesti addXXX() metodeilla. Käytetään niin montaa addXXX()-metodia kuin viestissä on käytössä ”{} + numero”-yhdistelmää ja viesti täydentyy numerojärjestyksessä pienimmästä suurimpaan. [8, s. 1]

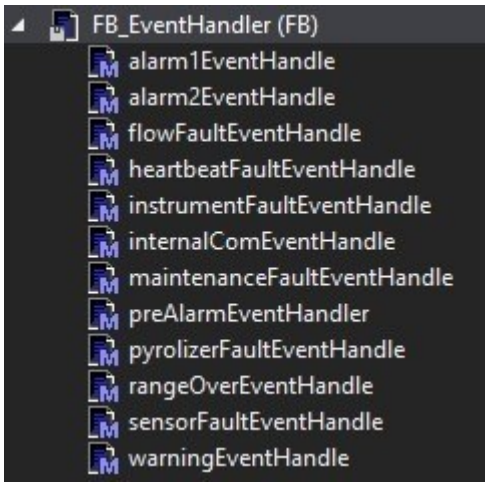


Kuva 25 Hälytystapahtuma

```
METHOD alarm1EventHandle : BOOL
IF bAlarm1 THEN
     Alarm1.ipArguments.Clear().AddString(sTagName).AddString(sLocation
).AddString(sGasFormula).AddInt(1);
     Alarm1.Raise(0);
ELSE
     Alarm1.Clear(0, FALSE);
END_IF
```

Esimerkkikoodi 5. Hälytysrajalle 1 luotu metodi, josta voidaan nähdä kuinka .ipArguments propertyä voidaan käyttää.

Tämän jälkeen luotiin FB_EventHandler, jonka tehtävänä oli tapahtumien akti-



Kuva 26 FB_EventHandler

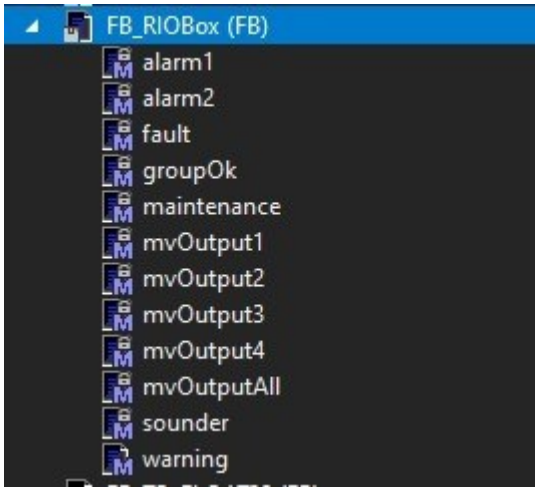
vointi. Sisääntulotietoina laitteen nimi, sijainti, kaasun kaava, raja1, raja2 sekä vikatietoja. Kaikille hälytys- ja vikatapahtumille määritettiin FB_TcAlarm sekä myös FB_TcSourceInfo funktiot. FB_TcAlarm muodostaa TwinCAT 3 EventLoggeriin hälytyksen, kun taas FB_TcSourceInfo:lla voidaan määrittää tapahtuman lähdetiedot [liite 4].

```
IF bInit THEN
     bInit := FALSE;

     PreAlarmSource.ExtendName('LAL Alarm');
     Alarm1Source.ExtendName('Alarm 1');
     Alarm2Source.ExtendName('Alarm 2');
     PyrolizerSource.ExtendName('Pyrolizer');
     InternalComSource.ExtendName('Internal Communication');
     SensorFaultSource.ExtendName('Sensor');
     InstrumentSource.ExtendName('Instrument');
     RangeOverSource.ExtendName('Range over');
     FlowSource.ExtendName('Flow Rate');
     MaintenanceSource.ExtendName('Maintenance');
     HeartbeatSource.ExtendName('Communication');
     WarningSource.ExtendName('Warning');

     preAlarm.CreateEx(stEventEntry        :=
TC_EVENTS.GasMonitoringAlarms.Alarm,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := PreAlarmSource
                       );

     Alarm1.CreateEx(stEventEntry          :=
TC_EVENTS.GasMonitoringAlarms.Alarm,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := Alarm1Source
                       );

     Alarm2.CreateEx(stEventEntry          :=
TC_EVENTS.GasMonitoringAlarms.Alarm,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := Alarm2Source
                       );

     Pyrolizer.CreateEx(stEventEntry  :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := PyrolizerSource
                       );

     InternalCom.CreateEx(stEventEntry     :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := InternalComSource
                       );

     SensorFault.CreateEx(stEventEntry     :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := SensorFaultSource
                       );

     Instrument.CreateEx(stEventEntry :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                       bWithConfirmation  := TRUE,
                       ipSourceInfo       := InstrumentSource
                       );

     RangeOver.CreateEx(stEventEntry  :=
TC_EVENTS.GasMonitoringAlarms.Fault,
```

```
                        bWithConfirmation  := TRUE,
                        ipSourceInfo       := RangeOverSource
                        );

     Flow.CreateEx(stEventEntry            :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation  := TRUE,
                        ipSourceInfo       := FlowSource
                        );

     Maintenance.CreateEx(stEventEntry     :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation  := TRUE,
                        ipSourceInfo       := MaintenanceSource
                        );

     Heartbeat.CreateEx(stEventEntry  :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation  := TRUE,
                        ipSourceInfo       := HeartbeatSource
                        );

     WarningFault.CreateEx(stEventEntry    :=
TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation  := TRUE,
                        ipSourceInfo       := WarningSource
                        );



END_IF
```

Esimerkkikoodi 6. Hälytyksille muodostetaan nimet, jonka jälkeen luodaan hälytykset. Kerrotaan CreateEx-metodille, että käytetään Event Classia GasMonitoringAlarms.XXXX. Hälytyksissä käytetään .Alarm ja vioissa .Fault. BWithConfirmationilla määritetään, vaatiiko hälytys pakollisen vahvistuksen – valitaan TRUE, eli kyllä vaaditaan vahvistus. IpSourceInfo on osoitin lähdetietoihin [8, s. 35;60-61] .

Seuraavaksi luotiin FB_RIOBox (Kuva 27), jonka tehtävänä oli päivittää eri laiteryhmien tilatietoja ja ulostuloina saadaan IO-laitteille tarvittavat tilatiedot. FB



Kuva 27 FB_RIOBox

tarkastaa kaikki ryhmään kuuluvat laitteet. Mikäli yksikin laite ilmoittaa hälytysrajaa 1, aktivoidaan bAlarm1Out, ja jos laitteelle on valittu magneettiventtiiliryhmä aktivoidaan myös sen lähtö (1-4). Jos ryhmässä laite ilmoittaa, hälytysraja 2 aktivoidaan bAlarm2Out sekä bSounderOut, ja jos laite kuulu johonkin magneettiventtiiliryhmään, aktivoidaan ne kaikki. BOkOut = TRUE jos yhdessäkään laitteessa ei ole vikaa aktiivisena. Jos johonkin laitteeseen tulee vika bOkOut = FALSE. Hälytykset kuitataan, kun bResetAlarmIn = TRUE, mutta vain jos hälytykset eivät ole aktiivisia. Magneettiventtiilit kuittaantuu, kun bResetMVIn = TRUE ja hälytykset on kuitattu. Jos bEmergencyIn = TRUE, kaikki magneettiventtiiliryhmät tulevat aktiivisiksi, ja bOkOut vaihtaa tilaa sekunnin välein. Kuittaus onnistuu vasta, kun bEmergencyIn = FALSE (liite 5).

Tämän jälkeen määritettiin globaalit muuttujat. Tehtiin omat muuttujat PLC:lle, HMI:lle sekä IO:lle.(liite 8)

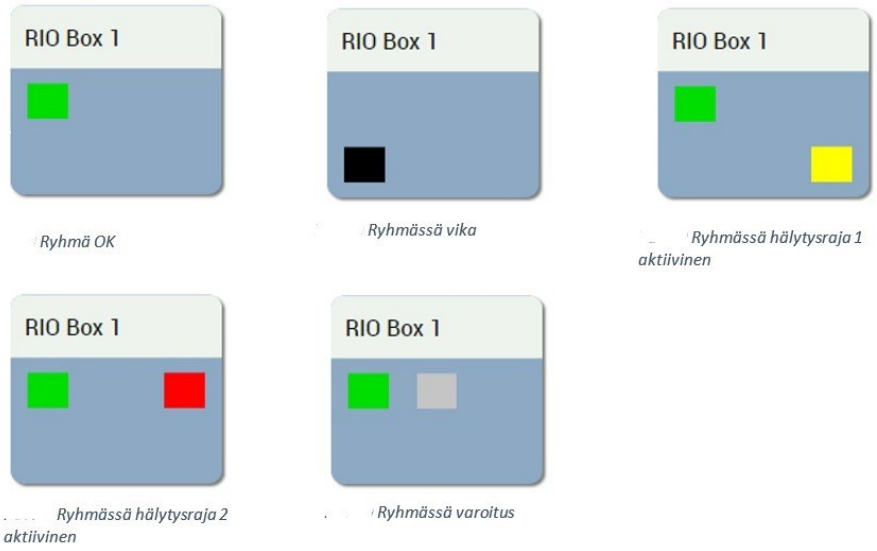
Lopuksi tehtiin ohjelmat PRG_Midas ja PRG_OutputControl. Määritettiin PRG_Midas:lle oma taski ja koodattiin sen toiminta (liite 6). PRG_OutputControl lisättiin MAIN-ohjelmaan, jossa sen toimintaa käytettiin (liite 7).

6.2 Käyttöliittymä

Käyttöliittymän etusivulla voidaan tarkastella eri alueiden aktiiviset hälytykset ja voidaan siirtyä ryhmään painamalla ryhmän kuvaketta tai valitsemalla se navigointipalkista kohdasta laitteet (Kuva 28). Ryhmän hälytys- ja vikatiedot ilmestyvät ryhmän kuvakkeeseen eri värisin merkein (kuva 29). Liitteessä 9 nähdään

Kuva 28 Navigointipalkki

etusivu kokonaisuudessaan. Etusivun taustakuvaksi on tarkoituksena liittää asiakkaan tiloista pohjakuva, jolloin saadaan havainnollisempaa tietoa laiteryhmän



Kuva 29 Ryhmän hälytys- ja vikatilanteet

sijainnista. Kuittaus- ja kirjautumispainikkeet on sijoitettu etusivulle oikeaan yläreunaan, jossa ne pysyvät, kun siirrytään sivulta toiselle.

Ryhmäkohtaisella sivulla voidaan tarkastella mittalaitteiden mittatietoa sekä nähdään myös laitteelle määritetty nimi ja mitattavan kaasun molekyylikaava. Vika- ja hälytystiedot ilmaistaan mittapisteiden kuvakkeissa eri värisillä merkeillä (kuva 30). Jos mittalaitteessa on vika, nähdään se kuvakkeessa mustalla. Jos mittalaite ilmoittaa hälytysraja 1, nähdään se kuvakkeessa keltaisella. Hälytys-

Kuva 4 Mittayksikön hälytys- ja vikatiedot

raja 2 nähdään kuvakkeessa punaisella. Jos mittapiste on hiljennetty, ilmestyy kuvakkeeseen oranssi ja punainen rasti. Jos mittalaite on pois käytöstä, kuvakkeessa lukee NIU. Kun painetaan mittapisteen kuvaketta, siirrytään mittapisteen tiedot sivulle. Liitteessä 10 nähdään ryhmäkohtainen sivu, jossa myös käytetään asiakkaan tiloista pohjakuvaa. Tämä kuva on suurennettu alueesta, jossa mittalaitteet sijaitsevat.

Mittalaitteen tiedot -sivulla voidaan tarkastella kaikkia mittayksikön arvoja (liite 12).

Mittalaitteen konfigurointiin päästään painamalla Config-painiketta, josta voidaan määrittää sille sen tiedot (kuva 31). Jos halutaan muuttaa enemmän lait-

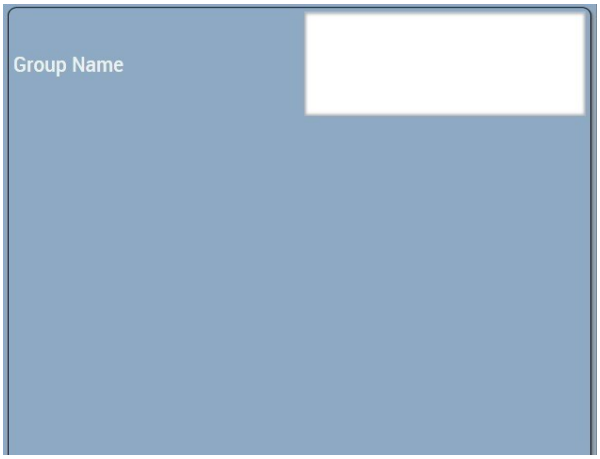
teen asetuksia, voidaan webpage-painikkeella siirtyä laitteen omalle käyttöliittymälle. Täältä löytyvät kaikki laitteen asetukset, kalibrointityökalut sekä laitteen

Kuva 5 Laitteen konfigurointi

tiedot (ks. liite 12).

Käyttöliittymän IO-sivulta voidaan tarkastella IO-laitteiden tilatietoja. Jos tulo tai lähtö on aktiivinen, sen perään ilmestyy punainen merkki. Ryhmän nimitieto päästään muuttamaan painamalla config-painiketta (kuva 32). (liite 11)

Hälytys-sivulta voidaan tarkastella järjestelmän hälytyshistoria sekä aktiiviset hälytykset. Hälytyksestä nähdään sen alkamisaika, sen aiheuttanut laite, laitteen sijainti tieto sekä hälytyksen aiheuttaja. (liite 13)



Kuva 6 Ryhmän nimeäminen. Painamalla close-painiketta nimi tallennetaan.

Järjestelmän trendit -sivulle päästää siirtymään joko navigointipalkista tai mittalaitteen omalta sivulta Trendit-painikkeella. Sivulla voidaan siirtyä laitteesta toiseen nuolipainikkeilla. Järjestelmä tallentaa mittausten historiatietoja neljältä päivältä viiden sekunnin aikavälillä. (liite 14)

Asetukset sivulla voidaan muuttaa järjestelmän kieltä. Valittavina kielinä Suomi ja Englanti. (liite 15)

7 Yhteenveto

Opinnäytetyön tavoitteena oli luoda yritykselle KV-Tekniikka Oy ohjelma digitaaliseen kaasuvalvontajärjestelmään, jossa käytettäisiin Modbus TCP/IP -protokollalla toimivia mittayksiköitä.

Työ aloitettiin perehtymällä käytettävään mittayksikköön Midas-t-004:ään ja sen Modbus TCP/IP -rajapintaan. Samalla myös tutustuttiin Modbus-protokollaan, jotta saatiin ymmärrys sen toiminnasta. Tämän jälkeen alkoi perehtyminen TwinCAT 3 -ohjelmistoon, johon ei ollut aikaisempaa kokemusta. Yrityksen puolesta päästiin Beckhoffin järjestämille PLC:n ja HMI:n peruskursseille.

Projektin ohjelmointia aloittaessa suurin osa ajasta kului aluksi TwinCAT-ohjelmiston testaukseen sekä sen dokumentaatioon perehtymiseen. Tutuksi tulivat Beckhoffin infosys-sivut, joista löytyi tietoa niin kirjastoista kuin myös sen tarjoamista laitteista. Aluksi ongelmia tuotti TwinCAT:n ja Midas-t-004 välinen kommunikointi, joka ei lähtenyt toimimaan. Syyksi kuitenkin selvisi tietokoneelta puuttuva Modbus-gateway-ohjelma, jonka asentamisen jälkeen ongelma ratkesi.

Toiminnan suunnittelussa hyödynnettiin jo olemassa olevia järjestelmiä ja kerättiin niistä parhaita ominaisuuksia. Haastateltiin myös yrityksen edustajaa muista ominaisuuksista, joita järjestelmässä voisi olla. Osa näistä jäi kuitenkin toteuttamatta, koska tietoa TwinCAT 3 -ohjelmiston toiminnoista ei ollut riittävästi, ja

suurin osa suunnittelusta jouduttiin tekemään samaan aikaan toteutuksen kanssa.

Projektin edetessä seuraava ongelma oli Modbus-laitteesta saatava pitoisuuden arvo, joka luettiin tavumuotoisena laitteen rekisteristä. Ratkaisuksi tähän kirjoitettiin oma koodi, joka käänsi rekisterintiedot tavumuodosta liukuluvuksi. Myöhemmin selvisi, että saman asian olisi voinut hoitaa helpomminkin, kun olisi luotu UNION-muuttuja. Tähän muutokseen ei aika enää tässä vaiheessa riittänyt, vaan se hoidettaisiin seuraavassa päivityksessä.

PLC-ohjelmaa testattiin koko sen tekemisen ajan ja seuraavaan vaiheeseen siirryttiin vasta, kun edellinen osio saatiin toimimaan. Näin varmistuttiin siitä, että ohjelman teko pysyi jotenkin kasassa eikä jouduttaisi niin paljoa korjaamaan jälkeenpäin. Tietysti kaikilta virheiltä ei voitu välttyä. Kun ohjelmaa lähdettiin testaamaan kokonaisuudessa, huomattiin, että mittayksiköille luotu ohjelma hidasti tietokonetta. Ongelmana oli FOR-loopin käyttö ohjelmassa, kun yritettiin lukea kaikki laitteet yhdellä ohjelmakierroksella. Tämä korjattiin ohjelmaan niin, että luettiin yksi laite per ohjelmakierros. PLC:n ohjelmointi sujui ripeästi alun opettelun jälkeen, ja ohjelma saatiin lopulta testausten jälkeen toimimaan halutulla tavalla.

Projektissa luodun käyttöliittymän teko oli haastavaa, sillä aikaisempaa kokemusta selainpohjaisen käyttöliittymän teosta ei ollut. Käyttöliittymän teko osoittautuikin suhteellisen helpoksi, sillä komponentteja voitiin vain lisätä haluamalleen sivulle, minkä jälkeen niihin voitiin linkata PLC-muuttujat ja luoda omia tapahtumia. Käyttöliittymän haastavin osuus oli kuitenkin trendien tekeminen. Trendeille piti määrittää omat muuttujansa, minkä jälkeen ne linkattiin trendigrafiikkaohjaimen trendisymboliin, jonka piti olla tekstimuotoinen. Projektissa haluttiin kuitenkin, että voitaisiin selata useampaa laitetta ja valita myös trendi sen mukaan, mikä laite oli valittuna. Nyt ei onnistunutkaan symbolin linkkaus tuohon trendisymboliin, vaan ohjelma ilmoitti virheestä, että symbolin täytyisi olla tekstimuodossa. Ongelmasta oltiin yhteydessä Beckhoffin tukeen, josta saatiin JavaScript-tiedosto, jota muokkaamalla ongelma saatiin ratkaistua. Tähän kului

paljon aikaa myös siksi, ettei ollut aikaisempaa kokemusta JavaScriptistä. Lisäksi käyttöliittymää testattiin koko sen kehityksen ajan

Viimeiseksi oli koko järjestelmän testaus, jossa oli mukana PLC, HMI ja kaksi Midas-t-004:ää. Lähdettiin testaamaan, että laitteisiin saatiin yhteys ja niiden tiedot saatiin päivitettyä ohjelmaan. Yhteys toimi, ja saatiin luettua kaikki rekisterit laitteelta. Sen jälkeen testattiin laitteiden ja ryhmien hälytysten ja vikojen toiminta. Ongelmana oli, että tilatiedot eivät päivittyneet oikein käyttöliittymään. Huomattiin, että hälytystietoja ei ollut linkattuna käyttöliittymän puolelle, koska niiden muuttujien nimiä oli muutettu jälkeenpäin. Muuttujat päivitettiin käyttöliittymään ja linkattiin uudestaan, mikä ratkaisi ongelman.

Opinnäytetyön lopputuotteena syntyi kohdeyritykselle kaasuvalvontajärjestelmän ohjelma, joka hyödyntää Modbus TCP/IP -rajapintaa. Tässä projektissa ei saatu ohjelmaan toteutettu kaikkia haluttuja ominaisuuksia, mutta se voidaan silti ottaa käyttöön asiakaskohteissa, koska sen hälytys- ja lukituspuolen ominaisuudet toimivat niin kuin kuuluukin. Tästä opinnäytetyöstä KV-Tekniikka Oy sai ohjelman, jonka se voi ottaa käyttöönsä, kun se lähtee toteuttamaan uutta digitaalista kaasuvalvontajärjestelmää.

Lähteet

1 Modbus, 2012. MODBUS Application Protocol Specification V1.1b3, Verkkoaineisto. https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. Luettu 5.5.2021.

2 Modbus, 2006. MODBUS Messaging On TCP/IP Implementation Guide V1.0b, Verkkoaineisto. https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. Luettu 26.5.2021.

3 Sosiaali- ja Terveysministeriön julkaisu, 2018. Haitalliseksi tunnetut pitoisuudet 2018, Verkkoaineisto. https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/160967/STM_09_2018_HTParvot_2018_web.pdf?sequence=1&isAllowed=y. Luettu 8.8.2021.

4 Honeywell, 2018. Midas Gas Detector User Manual, Verkkoaineisto. https://prod-edam.honeywell.com/content/dam/honeywell-edam/sps/his/ja/products/gas-and-flame-detection/documents/midasa001_technical_manual_eng_rev22.pdf. Luettu 1.4.2021.

5 Beckhoff, 2022. TwinCAT 3 Manual, Verkkoaineisto. https://download.beckhoff.com/download/document/automation/twincat3/Product_overview_EN.pdf. Luettu 5.3.2021.

6 Beckhoff, 2022. TwinCAT 3 HMI Engineering Manual, Verkkoaineisto. https://download.beckhoff.com/download/document/automation/twincat3/TE2000_TC3_HMI_EN.pdf. Luettu 30.3.2021.

7 Beckhoff, 2022. TF6250 TwinCAT 3 Modbus TCP Manual, Verkkoaineisto. https://download.beckhoff.com/download/Document/automation/twincat3/TF6250_TC3_Modbus_TCP_EN.pdf. Luettu 20.4.2021.

8 Beckhoff, 2021. TE1000 TwinCAT 3 Eventlogger, Verkkoaineisto. https://download.beckhoff.com/download/document/automation/twincat3/TwinCAT_3_PLC_Lib_Tc3_EventLogger_EN.pdf. Luettu 6.6.2021.

Midas Modbus-rekisterit

Ref. Addr Reg. Name	Bits	Function	Value Enumeration
4001 STTS Status			
Nibble 0	0-3	Monitoring state integer	
			0: Warmup
			1: Monitor mode with inhibit state “nonE”
			2: Monitor mode but alarms inhibited, inhibit state “ALm”
			3: Monitor mode but alarms and faults inhibited, inhibit state “AL-Ft”
			4: Monitor mode but fully Inhibited, inhibit state “ALL”
			5: Alarm / Fault Simulation
			6: Bump test mode (largely same as state 2)
			7: 4-20 mA loop Calibration mode
			8: Calibration Mode other than state 7
			9-15: for future expansion
Nibble 1	4-5	Fault status integer	
			0: No Fault
			1: Maintenance fault active
			2: Instrument fault active
	6		Alarm1 active
	7		Alarm2 active
Nibble 2	8		Relay 1 energized
	9		Relay 2 energized
	10		Relay 3 energized
	11		Heartbeat bit - toggles every two seconds to confirm communications
	12		Relays under remote Modbus/TCP control
Nibble 3	13-15	For future expansion	
40002 GASS Gas Selection			
	0-7	Gas ID	
	0-15	Sensor CartridgeID	
40003 FCN1 Gas Concentration in floating point format word 1 of 2			
40004 FCN2 Gas Concentration in floating point format word 2 of 2			
40005 CONC Gas Concentration in Integer Format			
	Conversion equation: ppm value = Registry Value * 1/10 DECP		
40006 NOFT Number of most important active fault			
40007 DPUN Decimal point and units			
	0-2	Decimal point indicator (0-3)	
	3-7	for future expansion	
	8-15	Concentration units	
			1: ppm
			2: ppb
			4: % volume

		8: %LEL
		16: mA
		0, 3, 5, 6, 7, 9-15, 17-255: for future expansion
40008 TEMP Temperature in Celsius		
	Signed 16-bit integer	
40009 CLRH		Cell life remaining in hours
40010 HRTB		Heartbeat Counter, 16 LSB of time in seconds
40011 FLOW		Sample flowrate in cc/minute
40012		Reserved for future expansion, currently zero
40013 A1T1		Alarm 1 threshold in floating point format, Scaling same as FCN1 and FCN2
40014 A1T2		
40015 A2T1		Alarm 2 threshold in floating point format, Scaling same as FCN1 and FCN2
40016 A2T2		
40017 ASI	0-1	Alarm status, equivalent to STTS bits 6-7
	2-15	Zero
40018 FSI	0	Maintenance fault active
	1	Instrument fault active
	2	Flow fault active
	3	Internal communication fault active
	4	Pyrolyzer fault active
	5	Sensor fault active
	6	Range over fault active
	7-15	Reserved for future expansion
40019 HST1		Concentration high scale in floating point format, scaling same as FCN1 and FCN2
40020 HST2		
40071 DSN1		Detector serial number in ASCII code format, maximum 10 characters (2 characters per address)
40072 DSN2		
40073 DSN3		
40074 DSN4		
40075 DSN5		
40076 SSN1		Sensor serial number in ASCII code format, maximum 10 characters (2 characters per address)
40077 SSN2		
40078 SSN3		
40079 SSN4		
40080 SSN5		
40081 MSI	0	Warm-up (STTS nibble0 value = 0)
	1	Monitor mode (STTS nibble0 value = 1, 2, 3, 4)
	2	Test mode (STTS nibble0 value = 5, 6)
	3	Maintenance mode (STTS nibble0 value = 7, 8)
	4-15	Reserved for future expansion

Kemikaalikortti booritrikloridi

BOORITRIKLORIDI	**ICSC: 0616 (Heinäkuu 1997)**
Triklooriboraani	
CAS #: 10294-34-5 **YK #: 1741** **EC Numero: 233-658-4**	

	VÄLITTÖMÄT VAARAT	**TORJUNTA**	**SAMMUTUS**
PALO & RÄJÄHDYS	Ei palava. Vapauttaa ärsyttäviä tai myrkyllisiä huuruja (tai kaasuja) palossa.		EI vettä. Jos palo on lähiympäristössä, käytä sopivaa sammutusainetta. Tulipalotilanteessa: jäähdytä kaasusäiliöitä vesisuihkuilla. EI suoraa kosketusta veden kanssa.

VÄLTÄ KAIKKEA KOSKETUSTA! OTA KAIKISSA TAPAUKSISSA YHTEYS LÄÄKÄRIIN!			
	OIREET	**TORJUNTA**	**ENSIAPU**
Hengitystiet	Kurkkukipu. Yskä. Polttava tunne. Vaikeutunut hengitys. Vaikeutunut hengitys. Oireet voivat viivästyä. Ks. Huomautukset.	Käytä ilmanvaihtoa, paikallispoistoa tai hengityksensuojainta.	Raitis ilma, lepo. Puoli-istuva asento. Tekohengitystä voidaan tarvita. Toimita lääkärin hoitoon.
Iho	Kipu. Punoitus. Rakkulat. Ihon palovammat. NESTEKOSKETUS: PALELTUMA.	Kylmänsuojakäsineet. Suojavaatetus.	PALELTUMA: huuhtele runsaalla vedellä, ÄLÄ riisu vaatteita. Toimita lääkärin hoitoon .
Silmät	Kipu. Punoitus. Vakavia syviä palovammoja. Näönmenetys.	Käytä kasvojensuojainta tai silmiensuojainta yhdessä hengityksensuojaimen kanssa.	Huuhtele ensin runsaalla vedellä usean minuutin ajan (poista piilolinssit mikäli mahdollista), toimita sitten lääkäriin.
Nieleminen			

TOIMINTA VUODON SATTUESSA	**LUOKITUS & MERKINNÄT**
Evakuoi vaara-alue! Kysy asiantuntijalta neuvoja! Tuuletus. Henkilönsuojaimet: kaasutiivis kemikaalisuojapuku, paineilmahengityslaite.	**GHS-järjestelmän mukainen luokitus ja merkinnät**
VARASTOINTI	
Paloturvallinen tila, jos sisällä rakennuksessa. Erillään elintarvikkeista ja eläinravinnosta. Ks. Kemialliset vaarat.	**Kuljetus** **YK-luokitus** YK-vaaraluokka: 2.3; YK-lisävaara: 8
PAKKAUS	
Älä kuljeta elintarvikkeiden ja eläinravinnon kanssa.	

International Labour Organization
World Health Organization

Kansainvälinen asiantuntijaryhmä on laatinut kemikaalikortit englanniksi ILO:n ja WHO:n toimesta EU:n taloudellisen tuen avustuksella.

European Commission

BOORITRIKLORIDI **ICSC: 0616**

FYSIKAALISET & KEMIALLISET TIEDOT

Fysikaalinen olomuoto; esiintyminen
KAASU TAI VÄRITÖN SAVUAVA NESTE, JOLLA PISTÄVÄ HAJU.

Fysikaaliset vaarat
Kaasu on ilmaa raskaampaa.

Kemialliset vaarat
Hajoaa kuumentuessaan. Tämä muodostaa myrkyllisiä ja syövyttäviä huuruja sisältäen kloorivetyä. Reagoi kiivaasti veden ja kostean ilman kanssa. Tämä muodostaa kloorivetyä ja boorihappoa. Reagoi kiivaasti aniliinin, fosfiinin, alkoholien, hapen ja orgaanisen aineen, kuten rasvojen kanssa. Syövyttää monia metalleja veden läsnä ollessa.

Molekyylikaava: BCl_3
Molekyylimassa: 117.19
Kiehumislämpötila: 12.5°C
Sulamislämpötila: -107 °C
Suhteellinen tiheys (vesi = 1): 1.35
Liukoisuus veteen: reaktio
Höyrynpaine, kPa 20 °C:ssa: 150
Suhteellinen höyryn tiheys (ilma = 1): 4.03

ALTISTUMINEN & TERVEYSVAIKUTUKSET

Altistumisreitit
Aine voi imeytyä elimistöön hengitysteitse.

Lyhytaikaisen altistumisen vaikutukset
Aine on syövyttävää silmille, iholle ja hengityselimille. Tämän kaasun hengittäminen voi aiheuttaa keuhkopöhön. Ks. Huomautukset. Nesteen nopea haihtuminen voi aiheuttaa paleltuman. Altistuminen korkeille pitoisuuksille voi aiheuttaa kuoleman. Vaikutukset voivat ilmetä viivästyneinä. Lääkärin tarkkailu on tarpeen.

Hengitystieriski
Tästä kaasusta syntyy hyvin nopeasti haitallinen pitoisuus ilmaan säiliön rikkoutuessa.

Pitkäaikaisen tai toistuvan altistumisen vaikutukset

TYÖHYGIEENISET RAJA-ARVOT

TLV: (kattoarvo): 0.7 ppm STEL

YMPÄRISTÖ

HUOMAUTUKSET

Reagoi kiivaasti sammutusaineiden, kuten veden kanssa.
Keuhkopöhön oireet ilmaantuvat usein vasta tuntien kuluttua, ja fyysinen rasitus pahentaa niitä.
Lepo ja lääkärin tarkkailu ovat siten tärkeitä.
Lääkärin tai muun lääkintähenkilöstön välittömästi aloittaman hengitystä tukevan hoidon tarpeellisuutta tulee harkita.

LISÄTIETOJA

LC50-arvo hengittämällä rotalla 1 tunnin kokeessa: 2541 ppm. **EU-luokitus**
Varoitusmerkki: T+; R: 14-26/28-34; S: (1/2)-9-26-28-36/37/39-45

Työterveyslaitos | Arbetshälsoinstitutet Finnish Institute of Occupational Health

FB_TCPModbusSensor

```
FUNCTION_BLOCK FB_TCPModbusSensor
VAR CONSTANT
    nTCPPort                    :UINT := 502;
    nUnitID                     :BYTE := 16#FF;
    tTimeout                    :TIME := T#2000MS;
END_VAR
VAR_INPUT
    sIpAddrs                    :STRING(15);
    sTagName                    :STRING(20);
    sLocation                   :STRING(20);
    sUnitIn                     :STRING(80);
    sGasNameIn                  :STRING(50);
    sGasFormulaIn               :STRING(50);
    sSensorSerialIn             :STRING(80);
    sDeviceSerialIn             :STRING(80);
    sMeasurementMinIn           :REAL;
    sMeasurementMaxIn           :REAL;
    nAlarm1ThresholdIn          :LREAL;
    nAlarm2ThresholdIn          :LREAL;

    bSelectLessThan             :BOOL;

    bResetAlarm                 :BOOL;
    bWriteRegs                  :BOOL;
    bInhibitIn                  :BOOL;
    bUpdate                     :BOOL;
    bNIU                        :BOOL;
END_VAR
VAR
    bInit                       :BOOL := FALSE;
    fbTCPmodbusDataParser       : FB_TCPModBusDataParser;
    eReadStates                 :(READ1, READ2);
    eModeState                  :(WARMUP, MONITORING, MAINTENANCE, TEST, MODE_ERROR);
    fbReadRegs                  :FB_MBReadRegs;
    nQuantity                   :WORD                   := 81;
    nMAddr                      :WORD                   := 0;
    bReadRegs                   :BOOL;
    arrData                     :ARRAY[0..80] OF WORD; //Midas Data
    fbMBWriteRegs               :FB_MBWriteRegs;
    alarm1SR                    :SR;
    alarm2SR                    :SR;
    inhibitSR                   :SR;
    inhTrig                     :R_TRIG;
    inhibitTimer                :TON;
    nInhibitTime                :TIME;
    bInhibitTimer               :BOOL;
    resetAlarmTrig              :R_TRIG;
    nGasId                      :UINT;
    nCartridgeId                :INT;
    nDecimalPointIndicator      :INT;
    nHeartbeatCounter           :UINT;
    nFaultStatus                :INT;
    bBusy                       :BOOL;
    bError                      :BOOL;
    nErrorId                    :UDINT;
    bHeartbeat                  :BOOL;
    nHeartbeatRef               :UINT;
    bHeartbeatInit              :BOOL;
    heartbeatTimer              :TON;
    heartbeatTrigger            :R_TRIG;
    fbEventHandler              :FB_EventHandler;
    bCheck: BOOL;
    tick: ULINT;
    ModbusCycle: UINT;
    lasttick: ULINT;
    heartbeatSR: SR;
    bNIUFlag: BOOL;
    heartbeatSetTrig: R_TRIG;
END_VAR
```

```
VAR_OUTPUT
    bPreAlarm                   :BOOL;
    bAlarm1                     :BOOL;
    bAlarm2                     :BOOL;
    nAlarm1ThresholdOut         :LREAL;
    nAlarm2ThresholdOut         :LREAL;
    bMaintenanceFault           :BOOL;
    bInstrumentFault            :BOOL;
    bFlowFault                  :BOOL;
    bInternalCommunicationFault :BOOL;
    bPyrolizerFault             :BOOL;
    bSensorFault                :BOOL;
    bRangeOverFault             :BOOL;
    bFault                      :BOOL;
    nMostImportantFault         :WORD;
    sMostImportantFault         :STRING;

    nCellLifeRemaining          :UINT;
    bHeartbeatCom               :BOOL;

    nTemperature                :WORD;
    sDeviceSerialOut            :STRING;
    sMonitoringState            :STRING;
    nConcentrationFloat         :STRING;
    nConcentrationInt           :INT;
    sSensorSerialOut            :STRING;
    nSensorState                :INT;
    sUnitOut                    :STRING;
    sGasNameOut                 :STRING;
    sGasFormulaOut              :STRING;
    sMeasurementMinOut          :REAL;
    sMeasurementMaxOut          :REAL;
    sStateMode                  :STRING(80);

    bOk                         :BOOL;

END_VAR
```

```
// Alarm Handling
fbEventHandler(
    sTagName            := sTagName,
    sLocation           := sLocation,
    sGasFormula         := sGasFormulaOut,
    bPreAlarm           := bPreAlarm,
    bAlarm1             := bAlarm1,
    bAlarm2             := bAlarm2,
    bWarning            := FALSE,
    bPyrolizer          := bPyrolizerFault,
    bInternalCom        := bInternalCommunicationFault,
    bSensorFault        := bSensorFault,
    bInstrument         := bInstrumentFault,
    bRangeOver          := bRangeOverFault,
    bFlow               := bFlowFault,
    bMaintenance        := bMaintenanceFault,
    bHeartbeat          := bHeartbeatCom
);
```

```
IF bNIU THEN
    clearAtNIU();
END_IF
```

```
IF NOT bNIU THEN
    resetAlarmTrig(CLK:=bResetAlarm);
    ReadRegs();
    fbTCPmodbusDataParser.ArrayData := arrData;

    // Check device state
    checkState();

    checkMostImportantFault();
    checkFaults();

    // Check heartbeat communication
    heartbeatTrigger(CLK:=fbTCPmodbusDataParser.Heartbeat);
    heartbeatSetTrig(CLK:=heartbeatComFault(nHeartbeat:=fbTCPmodbusDataParser.HeartBeatCounter));
    heartbeatSR(SET1:=heartbeatSetTrig.Q,RESET:=heartbeatTrigger.Q, Q1 => bHeartbeatCom);

    CASE eModeState OF

        WARMUP:
            //idle
        MONITORING:
            IF NOT bInhibitIn THEN
                nConcentrationFloat             := LREAL_TO_FMTSTR(fbTCPmodbusDataParser.ConcentrationFloat, nDecimalPointIndicator, TRUE);
                nConcentrationInt               := fbTCPmodbusDataParser.ConcentarionInteger;

                // Get Alarms and Faults
                preAlarm(bDirection := bSelectLessThan);
                bAlarm1 := alarm1(bDirection := bSelectLessThan);
                alarm2SR(SET1:= alarm2(bDirection:=bSelectLessThan), RESET:=resetAlarmTrig.Q, Q1 => bAlarm2);
            END_IF
            IF bInhibitIn THEN
                nConcentrationFloat := '*****';
                bMaintenanceFault := TRUE;
            END_IF

        TEST:
            //code
            bAlarm1 := alarm1(bDirection := bSelectLessThan);
            alarm2SR(SET1:= alarm2(bDirection:=bSelectLessThan), RESET:=resetAlarmTrig.Q, Q1 => bAlarm2);

        MAINTENANCE:
            //code

        MODE_ERROR:
            //code
            bOk := FALSE;
    END_CASE

    updateSensorInfo();
END_IF
```

FB_EventHandler

```
FUNCTION_BLOCK FB_EventHandler
VAR_INPUT
    sTagName        :STRING;
    sLocation       :STRING;
    sGasFormula     :STRING;
    bAlarm1         :BOOL;
    bAlarm2         :BOOL;
    bPyrolizer      :BOOL;
    bInternalCom    :BOOL;
    bSensorFault    :BOOL;
    bInstrument     :BOOL;
    bRangeOver      :BOOL;
    bFlow           :BOOL;
    bMaintenance    :BOOL;
    bPreAlarm       :BOOL;
    bHeartbeat      :BOOL;
    bWarning        :BOOL;
END_VAR
VAR
    PreAlarm                :FB_TcAlarm;
    Alarm1                  :FB_TcAlarm;
    Alarm2                  :FB_TcAlarm;
    Pyrolizer               :FB_TcAlarm;
    InternalCom             :FB_TcAlarm;
    SensorFault             :FB_TcAlarm;
    Instrument              :FB_TcAlarm;
    RangeOver               :FB_TcAlarm;
    Flow                    :FB_TcAlarm;
    Maintenance             :FB_TcAlarm;
    Heartbeat               :FB_TcAlarm;
    WarningFault            :FB_TcAlarm;

    PreAlarmSource          :FB_TcSourceInfo;
    Alarm1Source            :FB_TcSourceInfo;
    Alarm2Source            :FB_TcSourceInfo;
    PyrolizerSource         :FB_TcSourceInfo;
    InternalComSource       :FB_TcSourceInfo;
    SensorFaultSource       :FB_TcSourceInfo;
    InstrumentSource        :FB_TcSourceInfo;
    RangeOverSource         :FB_TcSourceInfo;
    FlowSource              :FB_TcSourceInfo;
    MaintenanceSource       :FB_TcSourceInfo;
    HeartbeatSource         :FB_TcSourceInfo;
    WarningSource           :FB_TcSourceInfo;

    rAlarm1                 :R_TRIG;
    rAlarm2                 :R_TRIG;
    bInit                   :BOOL                           := TRUE;
END_VAR
```

```
IF bInit THEN
    bInit := FALSE;

    PreAlarmSource.ExtendName('LAL Alarm');
    Alarm1Source.ExtendName('Alarm 1');
    Alarm2Source.ExtendName('Alarm 2');
    PyrolizerSource.ExtendName('Pyrolizer');
    InternalComSource.ExtendName('Internal Communication');
    SensorFaultSource.ExtendName('Sensor');
    InstrumentSource.ExtendName('Instrument');
    RangeOverSource.ExtendName('Range over');
    FlowSource.ExtendName('Flow Rate');
    MaintenanceSource.ExtendName('Maintenance');
    HeartbeatSource.ExtendName('Communication');
    WarningSource.ExtendName('Warning');

    preAlarm.CreateEx(stEventEntry      := TC_EVENTS.GasMonitoringAlarms.Alarm,
                      bWithConfirmation := TRUE,
                      ipSourceInfo      := PreAlarmSource
                      );

    Alarm1.CreateEx(stEventEntry        := TC_EVENTS.GasMonitoringAlarms.Alarm,
                    bWithConfirmation   := TRUE,
                    ipSourceInfo        := Alarm1Source
                    );

    Alarm2.CreateEx(stEventEntry        := TC_EVENTS.GasMonitoringAlarms.Alarm,
                    bWithConfirmation   := TRUE,
                    ipSourceInfo        := Alarm2Source
                    );

    Pyrolizer.CreateEx(stEventEntry     := TC_EVENTS.GasMonitoringAlarms.Fault,
                       bWithConfirmation := TRUE,
                       ipSourceInfo      := PyrolizerSource
                       );

    InternalCom.CreateEx(stEventEntry   := TC_EVENTS.GasMonitoringAlarms.Fault,
                         bWithConfirmation := TRUE,
                         ipSourceInfo      := InternalComSource
                         );

    SensorFault.CreateEx(stEventEntry   := TC_EVENTS.GasMonitoringAlarms.Fault,
                         bWithConfirmation := TRUE,
                         ipSourceInfo      := SensorFaultSource
                         );
```

```
    Instrument.CreateEx(stEventEntry    := TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation   := TRUE,
                        ipSourceInfo        := InstrumentSource
                        );

    RangeOver.CreateEx(stEventEntry     := TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation   := TRUE,
                        ipSourceInfo        := RangeOverSource
                        );

    Flow.CreateEx(stEventEntry          := TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation   := TRUE,
                        ipSourceInfo        := FlowSource
                        );

    Maintenance.CreateEx(stEventEntry   := TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation   := TRUE,
                        ipSourceInfo        := MaintenanceSource
                        );

    Heartbeat.CreateEx(stEventEntry     := TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation   := TRUE,
                        ipSourceInfo        := HeartbeatSource
                        );

    WarningFault.CreateEx(stEventEntry  := TC_EVENTS.GasMonitoringAlarms.Fault,
                        bWithConfirmation   := TRUE,
                        ipSourceInfo        := WarningSource

END_IF

preAlarmEventHandler();
alarm1EventHandle();
alarm2EventHandle();
flowFaultEventHandle();
instrumentFaultEventHandle();
internalComEventHandle();
maintenanceFaultEventHandle();
pyrolizerFaultEventHandle();
rangeOverEventHandle();
sensorFaultEventHandle();
heartbeatFaultEventHandle();
warningEventHandle();
```

FB_RIOBox

```
FUNCTION_BLOCK FB_RIOBox
VAR_INPUT
    nId                 :INT; //Give Group ID number 0..
    nMVGroupId          :INT; //Give Sensors MV ID number 0..2
    nLightGroup         :INT;
    nSensorCountIn      :INT;
    sGroupNameIn        :STRING(80);
    sGroupLocationIn    :STRING(80);
    arrSensorStates     :ARRAY[0..HMI.MAX_SENSOR_COUNT] OF ST_MidasStates;
    arrSensorPersistent :ARRAY[0..HMI.MAX_SENSOR_COUNT] OF ST_MidasPersistent;

    bEmergencyIn        :BOOL;
    bResetAlarmIn       :BOOL;
    bResetMVIn          :BOOL;

END_VAR
VAR_OUTPUT
    bAlarm1Out          :BOOL;
    bAlarm2Out          :BOOL;
    bSounderOut         :BOOL;
    bOkOut              :BOOL;
    bFaultOut           :BOOL;
    bWarningOut         :BOOL;
    bOutputGroup1       :BOOL;
    bOutputGroup2       :BOOL;
    bOutputGroup3       :BOOL;
    bOutputGroup4       :BOOL;
    bEmergencyOut       :BOOL;
    bMaintenance: BOOL;
END_VAR
VAR
    groupOkSR           :SR;
    alarm2SR            :SR;
    sounderSR           :SR;
    mv1SR               :SR;
    mv2SR               :SR;
    mv3SR               :SR;
    mv4SR               :SR;
    mvAllSR             :SR;
    bSounderMuted       :BOOL := TRUE;

    EmergencyTrigger    :R_TRIG;
    EmergencySR         :SR;

    fbBlink             : FB_Blinker;

END_VAR
```

```
EmergencyTrigger(CLK:=bEmergencyIn);
//IF EmergencySR.Q1 THEN
//  fbBlink(bEnable:=bEmergencyIn, bReset:= bResetMVIn, IntervalTime:=T#1000MS, bOut => bOkOut);
//END_IF

EmergencySR(SET1:=bEmergencyIn, RESET:= bResetMVIn AND NOT bEmergencyIn);
fbBlink(bEnable:=EmergencySR.Q1, bReset:= bResetMVIn, IntervalTime:=T#1000MS, bOut => bOkOut);

//EmergencySR(SET1:=EmergencyTrigger.Q, RESET:=bResetMVIn AND NOT bEmergencyIn);

bAlarm1Out := alarm1();
alarm2SR(SET1 := alarm2(), RESET:= bResetAlarmIn, Q1 => bAlarm2Out);
sounderSR(SET1 := sounder(bAlarm2 := bAlarm2Out) OR EmergencyTrigger.Q, RESET := bResetAlarmIn, Q1 => bSounderOut);

mv1SR(SET1:=mvOutput1(bAlarm1 := bAlarm1Out) OR mvOutputAll() OR EmergencySR.Q1, RESET:= bResetMVIn, Q1 => bOutputGroup1);
mv2SR(SET1:=mvOutput2(bAlarm1 := bAlarm1Out) OR mvOutputAll() OR EmergencySR.Q1, RESET:= bResetMVIn, Q1 => bOutputGroup2);
mv3SR(SET1:=mvOutput3(bAlarm1 := bAlarm1Out) OR mvOutputAll() OR EmergencySR.Q1, RESET:= bResetMVIn, Q1 => bOutputGroup3);
mv4SR(SET1:=mvOutput4(bAlarm1 := bAlarm1Out) OR mvOutputAll() OR EmergencySR.Q1, RESET:= bResetMVIn, Q1 => bOutputGroup4);

bFaultOut       := fault();
bWarningOut     := warning();
bMaintenance    := maintenance();

IF NOT EmergencySR.Q1 THEN
    groupOkSR(SET1:= groupOk(), RESET:= fault() OR bEmergencyIn, Q1=> bOkOut);
END_IF
```

PRG_Midas

```
PROGRAM PRG_Midas
VAR
    i: INT := 0;
    bSimulate: BOOL;
    nMidasCount : INT := 21;
END_VAR

IF NOT bSimulate THEN
    IF i > nMidasCount THEN
        i := 0;
    END_IF

        GVL.fbMidas[i](
            sIpAddrs                    := HMI.stMidasPersistent[i].sIpAddr,
            sLocation                   := HMI.stMidasPersistent[i].sLocation,
            sTagName                    := HMI.stMidasPersistent[i].sTagName,
            sUnitIn                     := HMI.stMidasPersistent[i].sUnit,
            sGasNameIn                  := HMI.stMidasPersistent[i].sGasName,
            sGasFormulaIn               := HMI.stMidasPersistent[i].sGasFormula,
            sSensorSerialIn             := HMI.stMidasPersistent[i].sSensorSerial,
            sDeviceSerialIn             := HMI.stMidasPersistent[i].sDeviceSerial,
            sMeasurementMinIn           := HMI.stMidasPersistent[i].sMeasurementMin,
            sMeasurementMaxIn           := HMI.stMidasPersistent[i].sMeasurementMax,
            nAlarm1ThresholdIn          := HMI.stMidasPersistent[i].nAlarm1Threshold,
            nAlarm2ThresholdIn          := HMI.stMidasPersistent[i].nAlarm2Threshold,
            bInhibitIn                  := HMI.stMidasPersistent[i].bInhibit,
            bNIU                        := HMI.stMidasPersistent[i].bNIU,
            bSelectLessThan             := HMI.stMidasPersistent[i].bSelectLessThan ,
            bUpdate                     := HMI.stMidasStates[i].bUpdate,
            bResetAlarm                 := HMI.bReset,
            bPreAlarm                   => HMI.stMidasStates[i].bPreAlarm,
            bAlarm1                     => HMI.stMidasStates[i].bAlarm1,
            bAlarm2                     => HMI.stMidasStates[i].bAlarm2,
            bFault                      => HMI.stMidasStates[i].bFault,
            bMaintenanceFault           => HMI.stMidasStates[i].bMaintenanceFault,
            bInstrumentFault            => HMI.stMidasStates[i].bInstrumentFault,
            bFlowFault                  => HMI.stMidasStates[i].bFlowFault,
            bInternalCommunicationFault => HMI.stMidasStates[i].bInternalCommunicationFault,
            bPyrolizerFault             => HMI.stMidasStates[i].bPyrolizerFault,
            bSensorFault                => HMI.stMidasStates[i].bSensorFault,
            bRangeOverFault             => HMI.stMidasStates[i].bRangeOverFault,
            nMostImportantFault         => HMI.stMidasStates[i].nMostImportantFault,
            sMostImportantFault         => HMI.stMidasStates[i].sMostImportantFault,
            nCellLifeRemaining          => HMI.stMidasStates[i].nCellLifeRemaining,
            bHeartbeatCom               => HMI.stMidasStates[i].bHeartbeatCom,
```

```
            nTemperature                    => HMI.stMidasStates[i].nTemperature,
            sDeviceSerialOut                => HMI.stMidasPersistent[i].sDeviceSerial,
            sMonitoringState                => HMI.stMidasStates[i].sMonitoringState,
            nConcentrationFloat             => HMI.stMidasStates[i].nConcentrationFloat,
            nConcentrationInt               => HMI.stMidasStates[i].nConcentrationInt,
            sSensorSerialOut                => HMI.stMidasPersistent[i].sSensorSerial,
            nSensorState                    => HMI.stMidasStates[i].nSensorState,
            sUnitOut                        => HMI.stMidasPersistent[i].sUnit,
            sGasNameOut                     => HMI.stMidasPersistent[i].sGasName,
            sGasFormulaOut                  => HMI.stMidasPersistent[i].sGasFormula,
            sMeasurementMinOut              => HMI.stMidasPersistent[i].sMeasurementMin,
            sMeasurementMaxOut              => HMI.stMidasPersistent[i].sMeasurementMax,
            nAlarm1ThresholdOut             => HMI.stMidasPersistent[i].nAlarm1Threshold,
            nAlarm2ThresholdOut             => HMI.stMidasPersistent[i].nAlarm2Threshold,
            bOk                             => HMI.stMidasStates[i].bOk,
            //bInhibitOut                       => HMI.stMidasPersistent[i].bInhibit,
            sStateMode                      => HMI.stMidasStates[i].sSensorMode
        );

        IF GVL.fbMidas[i].bOk THEN
            HMI.HistorizedMidas[i] := TO_LREAL(GVL.fbMidas[i].nConcentrationFloat);
        END_IF

        i := i + 1;
END_IF
```

PRG_OutputControl

```
PROGRAM PRG_OutputControl
VAR
    i : INT := 0;
    bResetAlarm : BOOL;
    bResetMV    : BOOL;
    bEmergency  : BOOL;

END_VAR
IO.bRIOBox2InputArr[2] := TRUE; // COMMENT THIS IF GROUP 2 HAVE EMERGENCY BUTTON
IO.bRIOBox3InputArr[2] := TRUE; // COMMENT THIS IF GROUP 3 HAVE EMERGENCY BUTTON
bResetAlarm := HMI.bReset OR IO.bRIOBox1InputArr[0]; //OR IO.bRIOBox2InputArr[0] OR IO.bRIOBox3InputArr[0];
bResetMV    := HMI.bResetMV;//  OR IO.bRIOBox1InputArr[1]; //OR IO.bRIOBox2InputArr[1] OR IO.bRIOBox3InputArr[1];
bEmergency  := NOT IO.bRIOBox1InputArr[2];// OR NOT IO.bRIOBox3InputArr[2]; // IF Group 2 have emergency button append this -> OR NOT IO.bRIOBox2InputArr[2]

FOR i := HMI.MIN_ARRAY_LENGHT TO HMI.nNumberOfGroups - 1 DO
    HMI.stRIOBoxPersistent[i].nID := i;
    GVL.fbRIOBox[i](
        nId                         := HMI.stRIOBoxPersistent[i].nID,
        nSensorCountIn              := HMI.MAX_SENSOR_COUNT,
        sGroupNameIn                := HMI.stRIOBoxPersistent[i].sGroupName,
        nLightGroup                 := HMI.stRIOBoxPersistent[i].nLightId,
        sGroupLocationIn            := HMI.stRIOBoxPersistent[i].sGroupLocation,
        arrSensorPersistent         := HMI.stMidasPersistent,
        arrSensorStates             := HMI.stMidasStates,
        bEmergencyIn                := bEmergency,
        bResetAlarmIn               := bResetAlarm,
        bResetMVIn                  := bResetMV,
        bAlarm1Out                  => HMI.stRIOBoxStates[i].yellowLight,
        bAlarm2Out                  => HMI.stRIOBoxStates[i].redLight,
        bSounderOut                 => HMI.stRIOBoxStates[i].sounder,
        bOkOut                      => HMI.stRIOBoxStates[i].greenLight,
        bFaultOut                   => HMI.stRIOBoxStates[i].fault,
        bWarningOut                 => HMI.stRIOBoxStates[i].warning,
        bOutputGroup1               => HMI.stRIOBoxStates[i].mvGroup1,
        bOutputGroup2               => HMI.stRIOBoxStates[i].mvGroup2,
        bOutputGroup3               => HMI.stRIOBoxStates[i].mvGroup3,
        bOutputGroup4               => HMI.stRIOBoxStates[i].mvGroup4,
        bEmergencyOut               => HMI.stRIOBoxStates[i].emergencyStop,
        bMaintenance                => HMI.stRIOBoxStates[i].maintenance
    );
END_FOR

IO.bRIOBox1OutputArr[0] := GVL.fbRIOBox[0].bokOut AND GVL.fbRIOBox[1].bokOut AND GVL.fbRIOBox[2].bokOut;
IO.bRIOBox2OutputArr[0] := GVL.fbRIOBox[0].bokOut AND GVL.fbRIOBox[1].bokOut AND GVL.fbRIOBox[2].bokOut;
IO.bRIOBox3OutputArr[0] := GVL.fbRIOBox[0].bokOut AND GVL.fbRIOBox[1].bokOut AND GVL.fbRIOBox[2].bokOut;

IO.bRIOBox1OutputArr[1] := GVL.fbRIOBox[0].bAlarm1Out OR GVL.fbRIOBox[1].bAlarm1Out OR GVL.fbRIOBox[2].bAlarm1Out;
IO.bRIOBox2OutputArr[1] := GVL.fbRIOBox[0].bAlarm1Out OR GVL.fbRIOBox[1].bAlarm1Out OR GVL.fbRIOBox[2].bAlarm1Out;
IO.bRIOBox3OutputArr[1] := GVL.fbRIOBox[0].bAlarm1Out OR GVL.fbRIOBox[1].bAlarm1Out OR GVL.fbRIOBox[2].bAlarm1Out;

IO.bRIOBox1OutputArr[2] := GVL.fbRIOBox[0].bAlarm2Out OR GVL.fbRIOBox[1].bAlarm2Out OR GVL.fbRIOBox[2].bAlarm2Out;
IO.bRIOBox2OutputArr[2] := GVL.fbRIOBox[0].bAlarm2Out OR GVL.fbRIOBox[1].bAlarm2Out OR GVL.fbRIOBox[2].bAlarm2Out;
IO.bRIOBox3OutputArr[2] := GVL.fbRIOBox[0].bAlarm2Out OR GVL.fbRIOBox[1].bAlarm2Out OR GVL.fbRIOBox[2].bAlarm2Out;

IO.bRIOBox1OutputArr[3] := GVL.fbRIOBox[0].bSounderOut OR GVL.fbRIOBox[1].bSounderOut OR GVL.fbRIOBox[2].bSounderOut;
IO.bRIOBox2OutputArr[3] := GVL.fbRIOBox[0].bSounderOut OR GVL.fbRIOBox[1].bSounderOut OR GVL.fbRIOBox[2].bSounderOut;
IO.bRIOBox3OutputArr[3] := GVL.fbRIOBox[0].bSounderOut OR GVL.fbRIOBox[1].bSounderOut OR GVL.fbRIOBox[2].bSounderOut;

IO.bRIOBox1OutputArr[4] := GVL.fbRIOBox[0].bOutputGroup1 OR GVL.fbRIOBox[1].bOutputGroup1 OR GVL.fbRIOBox[2].bOutputGroup1;
IO.bRIOBox2OutputArr[4] := GVL.fbRIOBox[0].bOutputGroup1 OR GVL.fbRIOBox[1].bOutputGroup1 OR GVL.fbRIOBox[2].bOutputGroup1;
IO.bRIOBox3OutputArr[4] := GVL.fbRIOBox[0].bOutputGroup1 OR GVL.fbRIOBox[1].bOutputGroup1 OR GVL.fbRIOBox[2].bOutputGroup1;

IO.bRIOBox1OutputArr[5] := GVL.fbRIOBox[0].bOutputGroup2 OR GVL.fbRIOBox[1].bOutputGroup2 OR GVL.fbRIOBox[2].bOutputGroup2;
IO.bRIOBox2OutputArr[5] := GVL.fbRIOBox[0].bOutputGroup2 OR GVL.fbRIOBox[1].bOutputGroup2 OR GVL.fbRIOBox[2].bOutputGroup2;
IO.bRIOBox3OutputArr[5] := GVL.fbRIOBox[0].bOutputGroup2 OR GVL.fbRIOBox[1].bOutputGroup2 OR GVL.fbRIOBox[2].bOutputGroup2;

IO.bRIOBox1OutputArr[6] := GVL.fbRIOBox[0].bOutputGroup3 OR GVL.fbRIOBox[1].bOutputGroup3 OR GVL.fbRIOBox[2].bOutputGroup3;
IO.bRIOBox2OutputArr[6] := GVL.fbRIOBox[0].bOutputGroup3 OR GVL.fbRIOBox[1].bOutputGroup3 OR GVL.fbRIOBox[2].bOutputGroup3;
IO.bRIOBox3OutputArr[6] := GVL.fbRIOBox[0].bOutputGroup3 OR GVL.fbRIOBox[1].bOutputGroup3 OR GVL.fbRIOBox[2].bOutputGroup3;

IO.bRIOBox1OutputArr[7] := GVL.fbRIOBox[0].bOutputGroup4 OR GVL.fbRIOBox[1].bOutputGroup4 OR GVL.fbRIOBox[2].bOutputGroup4;
IO.bRIOBox2OutputArr[7] := GVL.fbRIOBox[0].bOutputGroup4 OR GVL.fbRIOBox[1].bOutputGroup4 OR GVL.fbRIOBox[2].bOutputGroup4;
IO.bRIOBox3OutputArr[7] := GVL.fbRIOBox[0].bOutputGroup4 OR GVL.fbRIOBox[1].bOutputGroup4 OR GVL.fbRIOBox[2].bOutputGroup4;
```

Globaalit muuttujat

GVL

```
{attribute 'qualified_only'}
VAR_GLOBAL
    fbRIOBox        : ARRAY[0..2] OF FB_RIOBox;
    fbMidas         : ARRAY[HMI.MIN_ARRAY_LENGHT..HMI.MAX_SENSOR_COUNT - 3] OF FB_TCPModbusSensor;
END_VAR
```

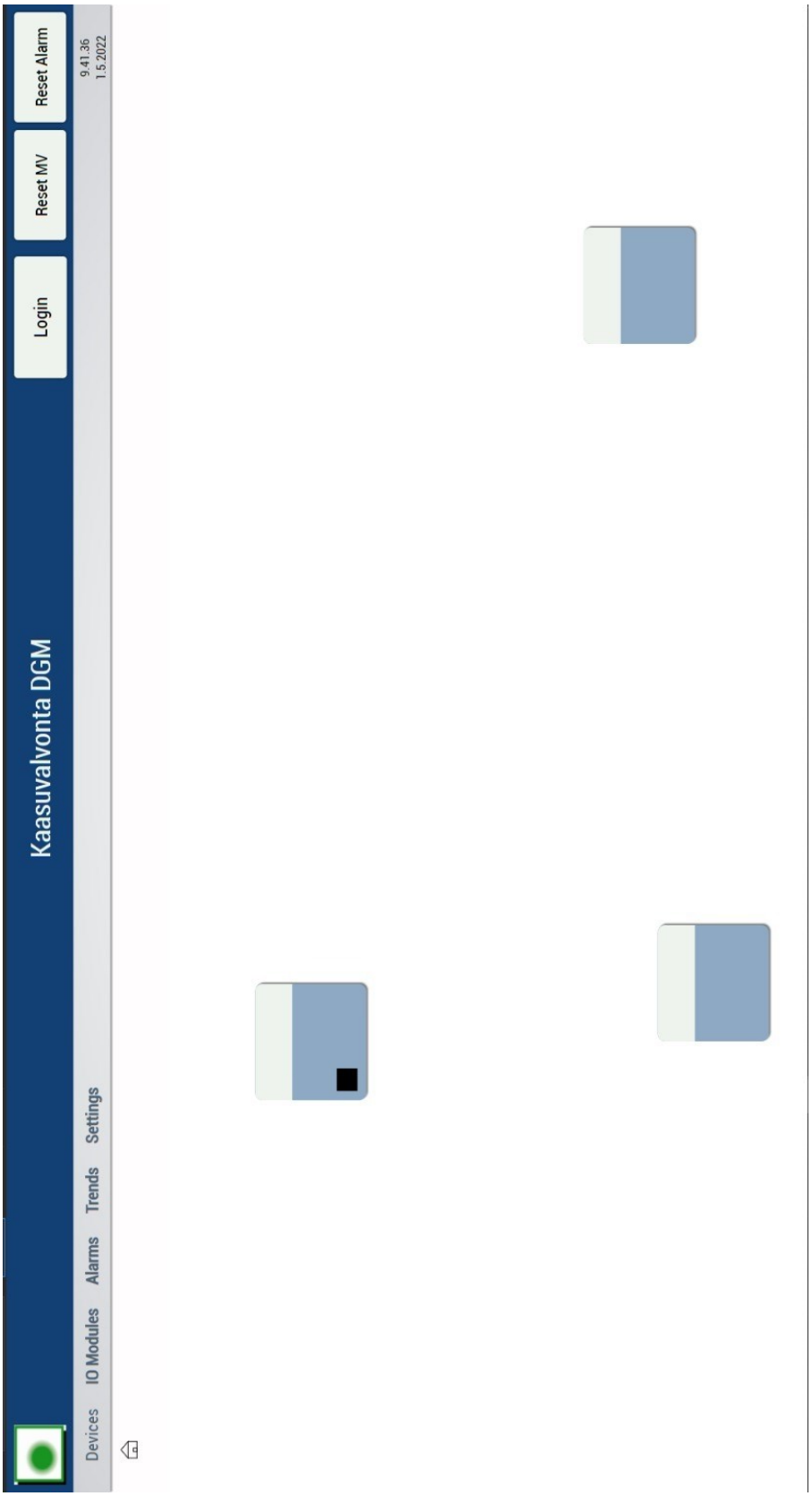
HMI

```
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
    MAX_SENSOR_COUNT    : INT := 24;
    MIN_ARRAY_LENGHT    : INT := 0;
    nNumberOfGroups     : INT := 3;
END_VAR
VAR_GLOBAL
    //stMidasParams     :ARRAY[MIN_ARRAY_LENGHT..MAX_SENSOR_COUNT] OF ST_MidasParams;
    HistorizedMidas     :ARRAY[MIN_ARRAY_LENGHT..MAX_SENSOR_COUNT] OF LREAL;
    stMidasStates       :ARRAY[MIN_ARRAY_LENGHT..MAX_SENSOR_COUNT] OF ST_MidasStates;
    stRIOBoxStates      :ARRAY[MIN_ARRAY_LENGHT..2] OF ST_RIOBoxStates;
    bReset              :BOOL;
    bResetMV            :BOOL;
    bWritePersistent    :BOOL;
END_VAR
VAR_GLOBAL PERSISTENT
    stRIOBoxPersistent  :ARRAY[MIN_ARRAY_LENGHT..2] OF ST_RIOBoxPersistent;
    stMidasPersistent   :ARRAY[MIN_ARRAY_LENGHT..MAX_SENSOR_COUNT] OF ST_MidasPersistent;
    nNumberOfSensors    :INT;
END_VAR
```

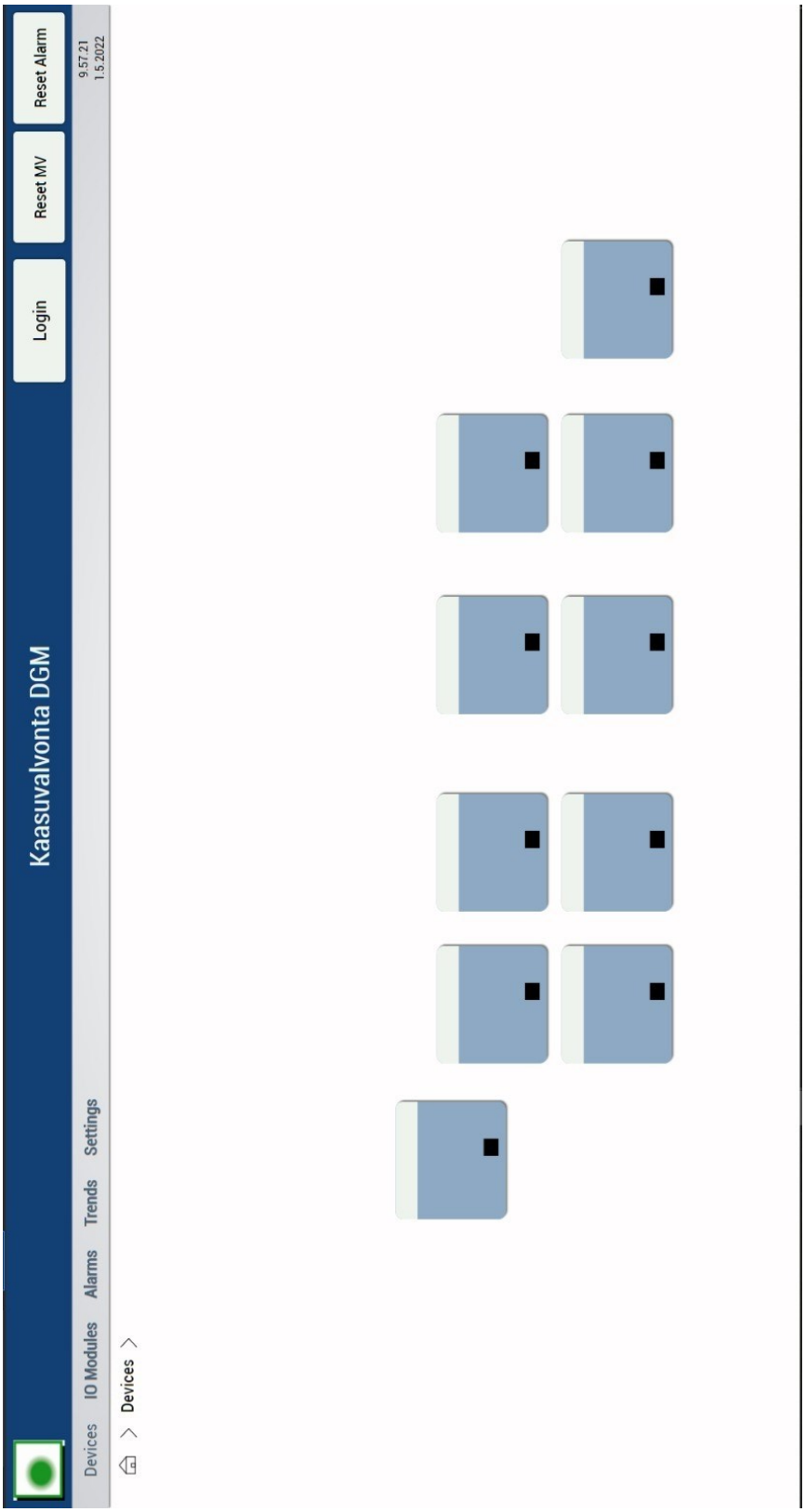
IO

```
{attribute 'qualified_only'}
VAR_GLOBAL
    // DIGITAL INPUTS
    bRIOBox1InputArr AT %I* : ARRAY[0..7] OF BOOL;  // 0. Emergency Stop 1. Reset 2-7. Optio
    bRIOBox2InputArr AT %I* : ARRAY[0..7] OF BOOL;  // 0. Emergency Stop 1. Reset 2-7. Optio
    bRIOBox3InputArr AT %I* : ARRAY[0..7] OF BOOL;  // 0. Emergency Stop 1. Reset 2-7. Optio

    // DIGITAL OUTPUTS
    bRIOBox1OutputArr AT %Q* : ARRAY[0..7] OF BOOL; // 0. Green 1. Yellow 2. Red 3. Sounder
    bRIOBox2OutputArr AT %Q* : ARRAY[0..7] OF BOOL; // 0. Green 1. Yellow 2. Red 3. Sounder
    bRIOBox3OutputArr AT %Q* : ARRAY[0..7] OF BOOL; // 0. Green 1. Yellow 2. Red 3. Sounder
END_VAR
```

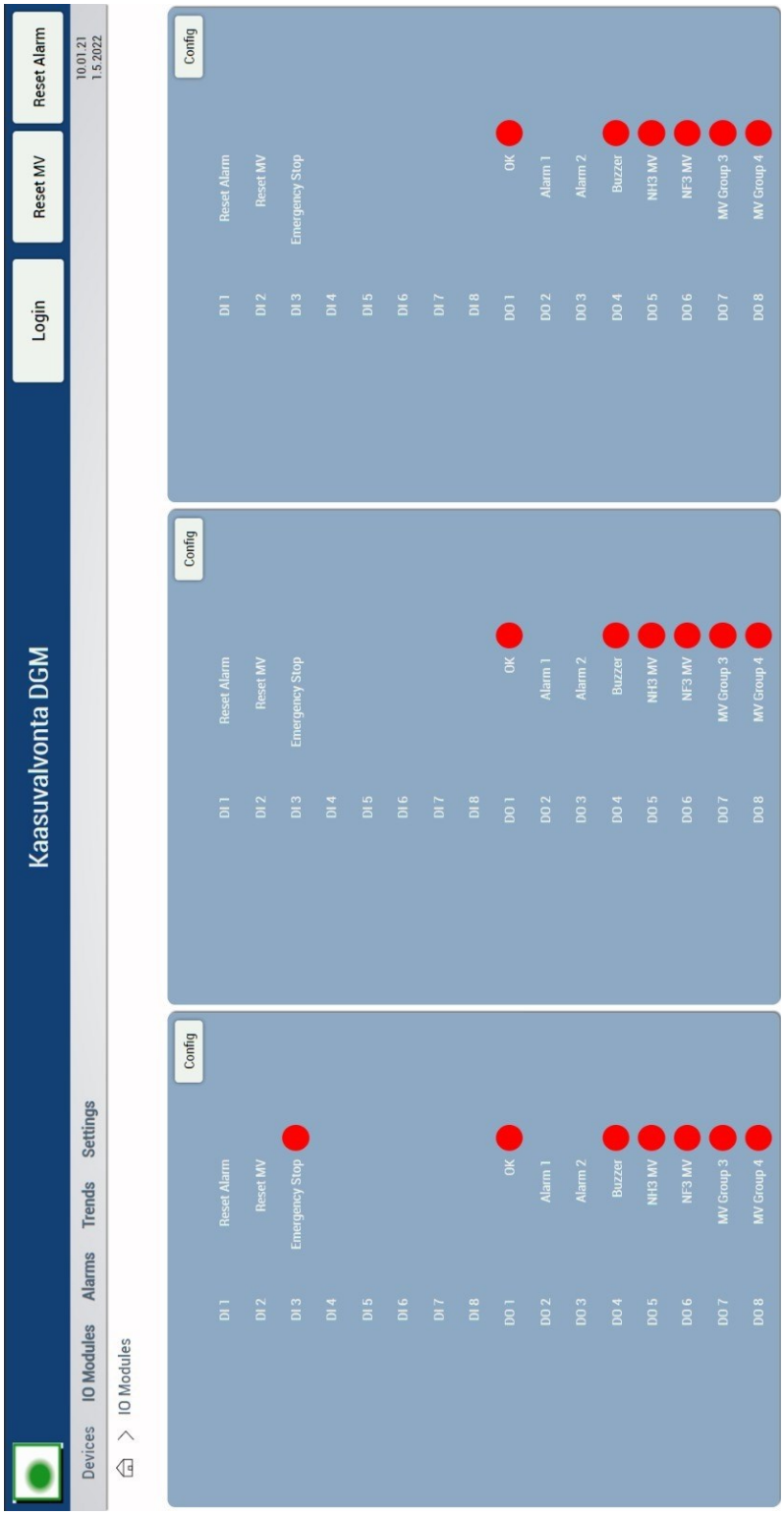
Etusivu


Kaasuvalvonta DGM
Login
Reset MV
Reset Alarm
9.41.36
1.5.2022
Devices
IO Modules
Alarms
Trends
Settings

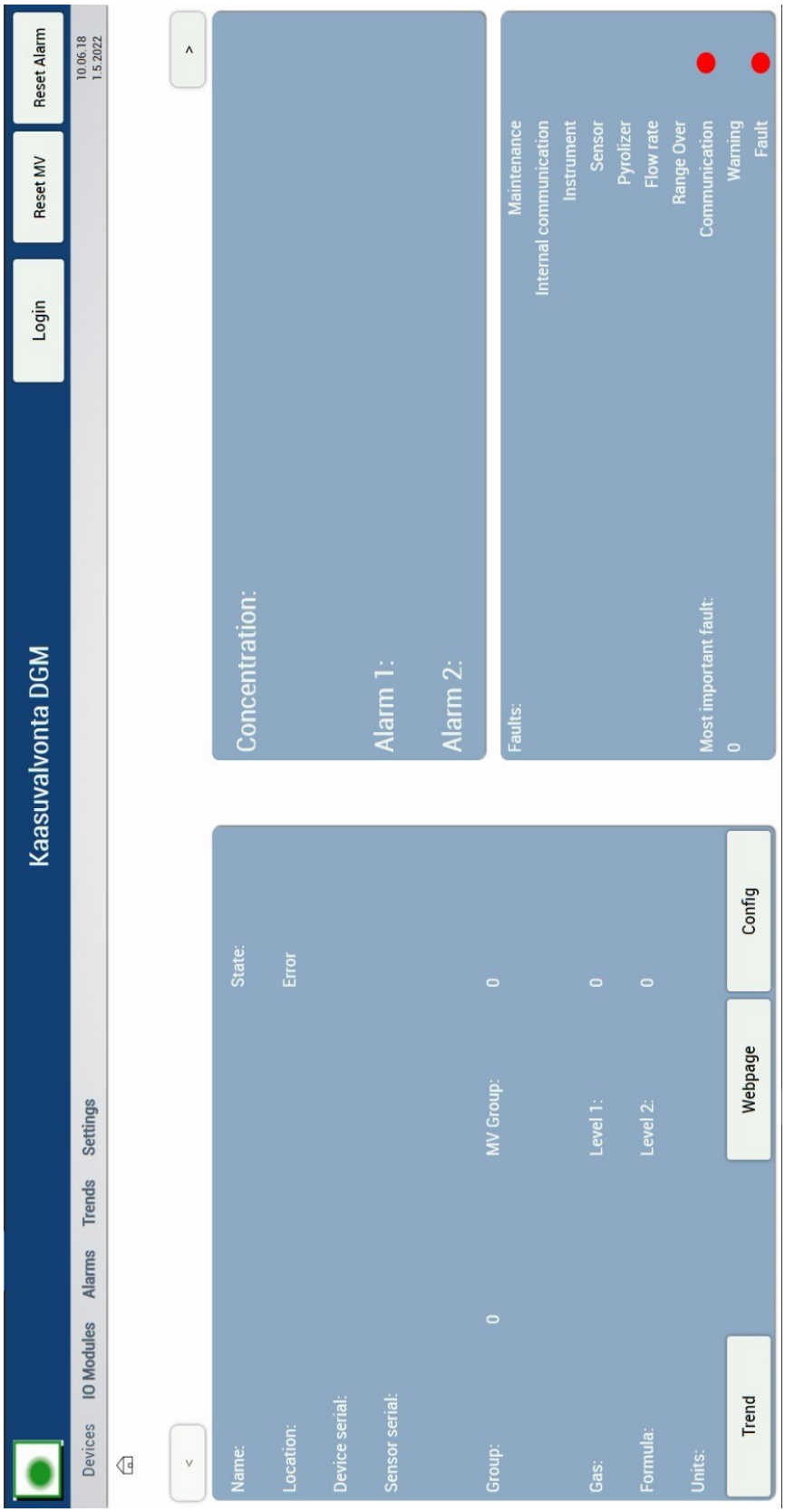
Laiteryhmäsivu



IO-sivu

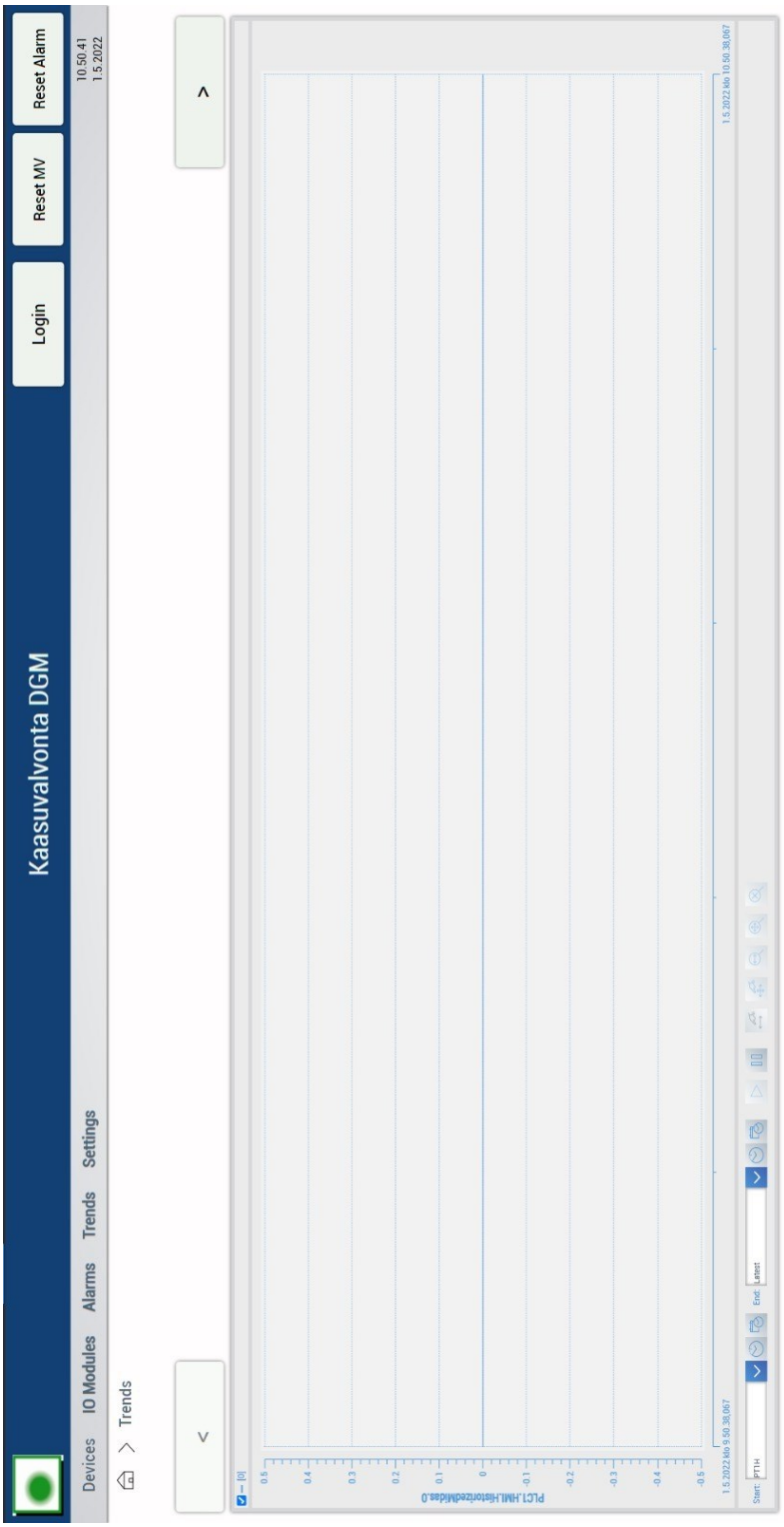

Kaasuvalvonta DGM
Login
Reset MV
Reset Alarm
Devices
IO Modules
Alarms
Trends
Settings
10.01.21
1.5.2022
IO Modules
Config
DI 1
Reset Alarm
DI 2
Reset MV
DI 3
Emergency Stop
DI 4
DI 5
DI 6
DI 7
DI 8
DO 1
OK
DO 2
Alarm 1
DO 3
Alarm 2
DO 4
Buzzer
DO 5
NH3 MV
DO 6
NF3 MV
DO 7
MV Group 3
DO 8
MV Group 4

Mittalaitteen tiedot sivu



Hälytykset-sivu

Trendit-sivu



Asetukset-sivu