Tampereen ammattikorkeakoulu

Robotisoidun tuotantoympäristön tiedonsiirron kehittäminen OPC UA –protokollan avulla

Aliina Jääskeläinen

OPINNÄYTETYÖ
Joulukuu 2023

Konetekniikan tutkinto-ohjelma

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Konetekniikan tutkinto-ohjelma
Koneautomaatio

JÄÄSKELÄINEN, ALIINA:
Robotisoidun tuotantoympäristön tiedonsiirron kehittäminen OPC UA –protokollan avulla
Opinnäytetyö 37 sivua, joista liitteitä 13 sivua
Joulukuu 2023

Opinnäytetyön tarkoituksena oli kehittää kahden eri valmistajan robottien välistä kommunikaatiota hyödyntäen alustariippumatonta OPC UA -arkkitehtuuria. Opinnäytetyö toteutettiin osana Tampereen ammattikorkeakoulun FieldLab-ympäristön kehittämistä. Opinnäytetyö toteutettiin toiminnallisena työnä, jossa suunniteltiin OPC UA -ympäristön toteutus ja demonstroitiin tiedonsiirron toiminta laitteiden välillä.

OPC UA -palvelin toteutettiin Yaskawan yhteistyörobottiin ja palvelimen asiakkuus Beckhoffin logiikkaan. Omronin mobiilirobotin kommunikointi logiikalle toteutettiin käyttäen Omron Call/Door Boxia. Logiikalle luotiin myös OPC UA -palvelin jatkokehittämistä varten. Toiminnallisen demon osuudessa roboteille luotiin ohjelmat, jotka käyttävät OPC UA -muuttujia logiikan kautta. Logiikan ja robottien ohjelmat toteutettiin siten, että aliohjelmien lisääminen ja käyttö on mahdollista.

Opinnäytetyön lopputuloksena oli toiminnallinen kokonaisuus, jossa robotit välittivät toisilleen viestejä OPC UA:n ja Beckhoffin logiikan avulla. Opinnäytetyön tuloksia tullaan hyödyntämään opetuskäytössä sekä laitteiden yhdistämisessä yläjärjestelmään.

Asiasanat: OPC UA, Beckhoff, Yaskawa, Omron

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Mechanical Engineering
Machine Automation

JÄÄSKELÄINEN, ALIINA:
Developing Data Transmission in a Robotized Production Environment Using the OPC UA Protocol
Bachelor's thesis 37 pages, appendices 13 pages
December 2023

The purpose of this thesis was to develop communication between two robots using platform-independent OPC UA. The thesis was a part of the development of the Tampere University of Applied Sciences' FieldLab environment. This study was carried out as a project, where the implementation of the OPC UA environment was planned and the operation of data transfer between devices was demonstrated.

An OPC UA server was implemented in a Yaskawa collaborative robot, and the server's client was integrated into Beckhoff's logic. Communication to the logic for the Omron mobile robot was implemented using the Omron Call/Door Box. An OPC UA server was also created for the logic for further development. In the functional demonstration part, programs were created for the robots that use OPC UA variables through logic. The logic and robot programs were implemented in such a way that the addition and use of subprograms are possible.

As a result of the thesis, a functional system was achieved in which the robots exchanged messages with each other using OPC UA and Beckhoff's logic. The outcomes of the thesis will be utilized for educational purposes as well as in integrating the devices into the upper-level system.

Key words: Beckhoff, OPC UA, Yaskawa, Omron

SISÄLLYS

LYHENTEET JA TERMIT

AMR	Autonominen mobiilirobotti.
AGV	Autonomisesti ohjattu ajoneuvo.
Cobotti	Yhteistyörobotti.
COM	Microsoftin Component Object Model, mahdollistaa ohjelmistojen tiedonvaihdon.
DCOM	Distributed COM. Tietojen vaihto tietokoneiden välillä.
HMI	Human macine interface. Käyttöliittymä.
HTTP	Hypertext Transfer Protocol. Verkkopalveluiden ja -selainten protokolla.
IIoT	Industrial Internet of Things, teollisuuden neljäs vallankumous.
OPC	OLE for Process Controll.
OPC UA	Open Platform Communication Unified Architecture.
PLC	Programmable Logic Controller. Ohjelmoitava logiikka.
SOAP	Simple Object Access Protocol. Protokolla XML viestien välitykseen.
TCP	Transmission Control Protocol. Internetin perusprotokolla.
XLM	Extensible Markup Language. Standardoitu rakenteellinen kuvauskieli.

1 JOHDANTO

Teollisuus 4.0, eli teollisuuden neljäs vallankumous, on muutosprosessi, jonka tavoitteena yhdistää digitalisaatio ja valmistustekniikat. Eri valmistajien automaatiojärjestelmien- ja laitteiden välinen kommunikaatio nousee täten jatkuvasti merkittävämmäksi osaksi. Tiedonsiirron tulee olla luotettavaa ja turvallista ja toteutettavissa helposti, ilman monia erillisiä laiteajureita. Yhtenä ratkaisuna tähän on OPC:n alustariippumaton kommunikaatioarkkitehtuuri, OPC UA.

Tämän opinnäytetyön tavoitteena on luoda kommunikaatioketju kahden eri valmistajan robotin välille ja toteuttaa opetuksessa hyödynnettävä OPC UA-ympäristö. Opinnäytetyön toimeksiantajana toimii Tampereen ammattikorkeakoulu ja työ on toteutettu osana Tampereen ammattikorkeakoulun FieldLabin kehittämistä. TAMK FieldLab on innovatiivinen oppimisympäristö, joka mahdollistaa erilaiset pilotoinnit ja sovellukset Teollisuus 4.0:n mukaisesti (Tampereen ammattikorkeakoulu n.d).

Tässä toiminnallisessa opinnäytetyössä esitetään robottien ja logiikan OPC UA -ympäristön rakentaminen sekä toteutetaan demonstraatio hyödyntäen OPC UA:ta. Toimeksiantajan tavoitteena on käyttää tulevaisuudessa OPC UA -ympäristöä linkkinä yläjärjestelmään sekä hyödyntää opinnäytetyötä osana opetusta.

2 OPC-STANDARDIT

2.1 Kehityksen historia

Teollisuusautomaatiossa ohjaus- ja automaatiojärjestelmät toteutetaan lähes poikkeuksetta PC- ja sovelluspohjaisesti. 90-luvulla oli tapana käyttää Windows-käyttöjärjestelmää alustana niin käyttöliittymille kuin ohjausjärjestelmille. Ongelmaksi muodostui kuitenkin useiden eri väylä- ja yhteysratkaisujen käyttö, joiden välillä tiedonsiirto oli mahdotonta ilman erillisiä ajureita. (Mahnke, Leitner & Damm 2009, 1.)

Käyttöliittymä- ja ohjaussovelluksia toimittavat yritykset huomasivat samanlaisia ongelmia tiedonsiirrossa. Fisher-Rosemount ja Rockwell Software perustivat työryhmän 90-luvun alkupuolella, jonka tavoitteena oli luoda laiteajureille standardi, joka mahdollistaa Plug and Play -tyyppisen pääsyn automaatiojärjestelmien dataan käytettäessä Windows -järjestelmää. (Mahnke, Leitner & Damm 2009, 1.)

Ensimmäinen OPC eli *OLE for Process Controll* – standardi julkaistiin 1996. OPC-standardien käyttökohteina ovat SCADA- ja HMI-järjestelmät, prosessinhallinta ja yläjärjestelmät (Mahnke, Leitner & Damm 2009, 1). Standardi noudattaa palvelin-asiakas-arkkitehtuuria. OPC Foundation vastaa standardin kehityksestä ja ylläpidosta (OPC Foundation 2023).

2.2 OPC Classic

Ensimmäinen OPC-standardi, OPC Classic, on Microsoft Windows -pohjainen. OPC Classic sisältää kolme eritelmää tietojen käyttämiseen: OPC Data Access (DA), OPC Alarms & Events (A&E) ja OPC Historical Data Access (HDA) (OPC Foundation 2023).

OPC DA on merkittävin OPC-rajapinta. Se tarjoaa alustan muuttujien lukemisen, kirjoittamiseen ja valvontaan. Sen pääasiallinen käyttötapa on datan siirto ohjauslaitteista käyttöliittymään. OPC A&E käsittää rajapinnan hälytys- ja tapahtumatie-

doille. Tämä tarkoittaa ilmoitusta käyttäjälle, kun jokin prosessiarvo muuttuu ehtojen yli. HDA määrittelee kyselymenetelmiä ja analytiikkaa sekä soveltaa sitä aiempaan dataan. (Mahnke, Leitner & Damm 2009, 3–5.) Jokainen standardiryhmä toimii omana itsenäisenä yksikkönään, eikä tiedonsiirtoa tapahdu standardien välillä. Tämä tarkoittaa esimerkiksi sitä, että OPC HDA ei kykene analysoimaan reaaliaikaista dataa.

Määrittelyjoukkojen tiedonsiirto tapahtuu Microsoftin COM/DCOM-teknologiaa hyödyntäen. Tämä tarkoittaa standardin olevan alustariippuvainen Microsoft Windows -käyttöjärjestelmästä. Lisäksi DCOM -teknologia aiheuttaa haasteita etäviestinnässä OPC:n kanssa: sitä ei voida käyttää Internet-kommunikoinnissa ja se on altis haavoittumaan. (Mahnke, Leitner & Damm 2009, 4, 8.)

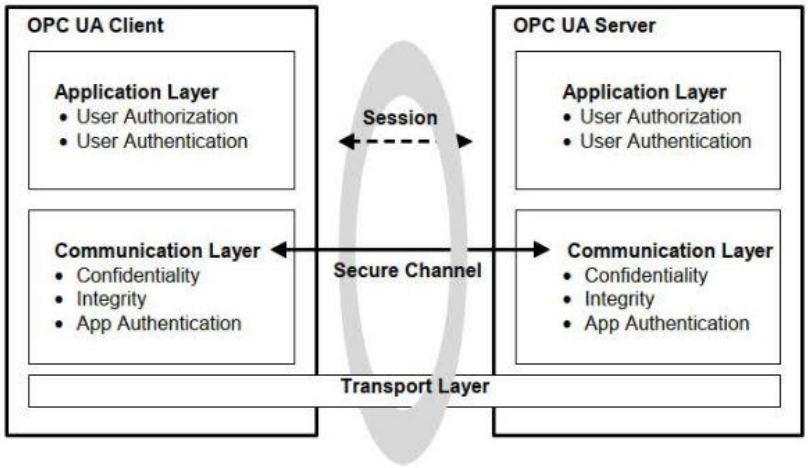
2.3 OPC UA

OPC Classic -määrittelyjen käyttö muuttui haasteelliseksi Windows-käyttöjärjestelmän kehittyessä, ja tuen poistuessa COM-teknologialle. Lisäksi tiedonsiirron luotettavuus oli kyseenalaistettavaa. Vuonna 2008 julkaistu OPC Unified Architecture (UA) on alustariippumaton, turvallinen ja vapaasti laajennettava standardi, joka käsittää OPC Classic -määrittelyt (OPC Foundation 2023).

IIoT eli Industrial Internet of Things on lisännyt automaatiojärjestelmien ja -laitteiden kytkentää verkkoon, mikä asettaa vaatimuksia tietoturvalle. OPC:n käyttö ulottuu mm. ERP- ja MES-järjestelmiin, minkä vuoksi yritysten informaatioteknologia tuli ottaa huomioon kehittäessä OPC UA:ta. OPC UA:n turvallisuusarkkitehtuuri mahdollistaa eri turvallisuustasojen määrittelyn: kenttälaitteistojen suojaukseen riittää useimmiten palomuuri, kun taas ylemmällä tasolla tietoverkkoyhteydet aiheuttavat uhkien kasvamisen myötä tarvitaan merkittävämpiä turvallisuustoimia. (Mahnke, Leitner & Damm 2009, 201; OPC Foundation 2023.)

OPC UA:n turvallisuusarkkitehtuuri sisältää mm. käyttäjän valvonnan, valtuutuksen, tietojen salauksen ja eheyden todentamisen (OPC Foundation 2023). Kuviossa 1 on esitetty OPC UA:n tietoturvan kolme suojauskerrosta: sovelluskerros,

viestintäkerros ja tiedonsiirtokerros. Sovelluskerros suojaa istunnon aikana siirrettyä dataa, viestintäkerros suojaa asiakkaan ja palvelimen välistä viestintää mm. sertifikaattien välillä ja tiedonsiirtokerros vastaa suojattujen tietojen siirrosta sekä vastaanotosta Socket-yhteyden kautta. (Mahnke, Leitner & Damm 2009, 234–238.)



KUVIO 1. OPC UA – tietoturva-arkkitehtuuri (OPC Foundation 2023).

OPC UA -standardi käyttää kahdenlaista tiedonsiirtomenetelmää: UA TCP ja OPC UA SOAP/HTTP. Nämä protokollat ovat olennaisia yhteyden luomisessa sekä tiedonvaihdossa palvelimen ja asiakkaan välillä. UA TCP on binääriprotokolla, joka käyttää TCP/IP:tä tiedonsiirtoon. Tämä tarkoittaa tiedonsiirron tapahtuvan binäärimuodossa. Internetin perusprotokollaa hyödynnetään TCP/IP-yhteyden luomiseksi ja ylläpitämiseksi. OPC UA SOAP/HTTP on verkkosovelluspalveluprotokolla, mikä perustuu World Wide Webin käyttämään HTTP-protokollaan. Ennen viestien lähettämistä ne salataan TLS-salausprotokollalla, mikä takaa tietoturvan ja yksityisyyden. (Mahnke, Leitner & Damm 2009, 198–199.)

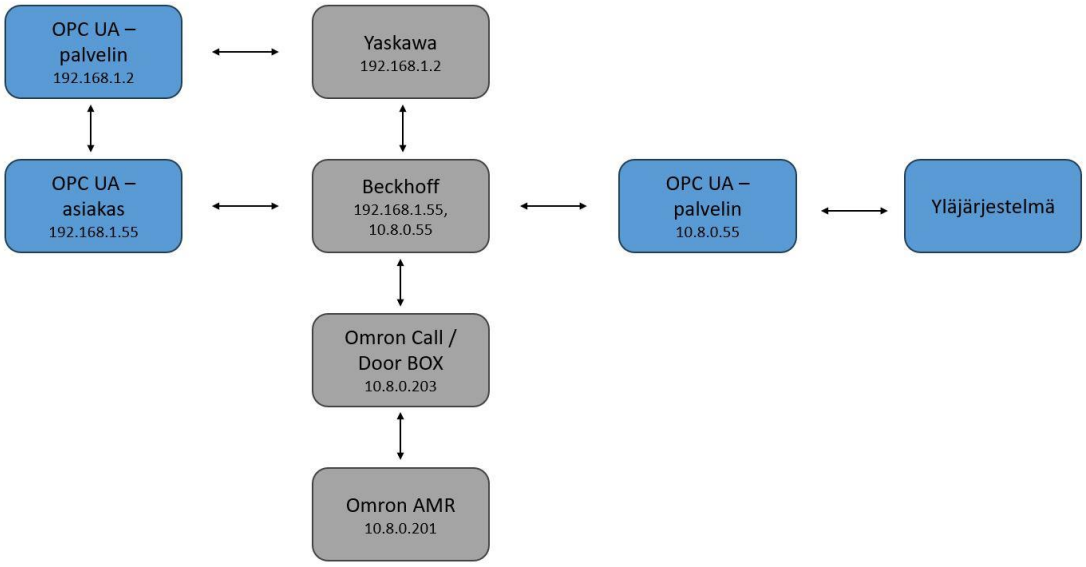
Tiedon mallinnus on OPC UA:n keskeinen periaate, ja se perustuu olio-ohjelmoinnin konseptiin. Tämä tarkoittaa, että OPC UA rakentaa tiedon hierarkkisesti ja ylläpitää yhtenäistä osoiteavaruutta. Yhtenäinen osoiteavaruus on tärkeä, sillä se mahdollistaa asiakkaalle kokonaisvaltaisen näkymän prosessista, jolloin erilaiset OPC-määrittelyt (DA, A&E, HDA) näkyvät asiakkaalle samalta palvelimelta. Tämä yhtenäisyys helpottaa tietojen seuraamista ja käyttöä, kun eri tiedonlähteet ja -tyypit ovat saatavilla samassa järjestelmässä ja samassa osoiteavaruudessa. (Mahnke, Leitner & Damm 2009, 10, 17.)

3 SUUNNITTELU

Tämän opinnäytetyön tavoitteena on toteuttaa opetuskäyttöön järjestelmä, jolla voidaan toteuttaa eri valmistajien laitteistojen välistä kommunikaatiota OPC UA -protokollan avulla. Suunnittelun lähtökohtana toimivat kaksi robottia, Yaskawan yhteistyörobotti ja Omronin autonominen mobiilirobotti. Kuvassa 1 on esitetty Yaskawan yhteistyörobotti. Lisäksi järjestelmään haluttiin logiikka ohjaamaan Yaskawan yhteistyörobottia, jotta saataisiin luotua yleisesti teollisuudessa nähtävä kokonaisuus. Logiikkapaketti valittiin Beckhoffin valikoimasta, sillä kyseisen merkin käyttö teollisuudessa on laaja. Omronin mobiilirobotti kytketään Omron Call/Door Boxin kautta logiikan I/O-kortteihin, jolloin myös Omronin ohjaus toteutuu logiikalla.

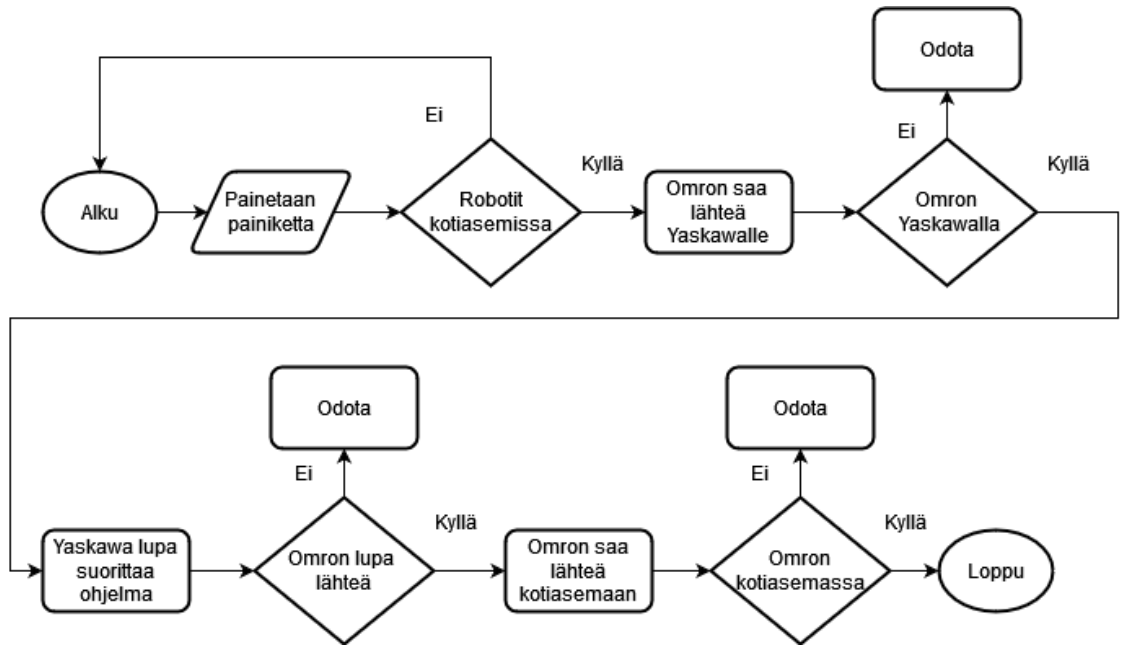
KUVA 1. Yaskawa HC10DTP -yhteistyörobotti.

OPC UA -palvelin toteutetaan Yaskawan robottiohjaimeen. Palvelimen asiakkaina toimii Beckhoffin logiikka. Tiedonsiirto palvelimen ja asiakkaan välillä tapahtuu Ethernet -kaapelin kautta. Beckhoffin logiikkaan luodaan myös OPC UA -palvelin, joka voidaan yhdistää tulevaisuudessa yläjärjestelmään. Kuviossa 2 on esitetty laitteiden ja OPC UA -ympäristön topologia IP-osoitteiden kanssa sekä yläjärjestelmän kytkeytyminen järjestelmään. Laitteistot ovat esitetty kuviossa harmaalla pohjalla, OPC UA -ympäristö sinisellä pohjalla.



KUVO 2. Topologia laitteistojen ja OPC UA -ympäristön välillä.

Kommunikaation demonstroimista varten luodaan toiminnallinen kokonaisuus laitteiden välille. Painiketta painettaessa logiikka antaa Omronille luvan siirtyä Yaskawan luokse. Omronin kuitatessa paikkansa Yaskawalla, saa Yaskawa suorittaa ohjelmansa. Yaskawan ohjelmakierron ollessa valmis, saa Omron luvan siirtyä takaisin aloitusasemaan. Muuttujat kirjoitetaan OPC UA -palvelimelle. Kuviossa 3 on esitetty vuokaavio toiminnallisen demon prosessista.



KUVIO 3. Prosessikaavio toiminnallisesta demosta.

4 LAITTEET JA KOMPONENTIT

Toiminnallisen järjestelmän laitteistoina toimivat robotit ja logiikka. Tässä luvussa on esitelty laitteistojen toimintaa ja ominaisuuksia.

4.1 Yaskawa HC10DTP

Yaskawa HC10DTP on kuusiakselinen yhteistyörobotti, jonka on suunniteltu toimimaan ihmisten rinnalla erilaisissa teollisuus- ja valmistusympäristöissä. Robotin hyötykuorma on 10 kg, ja siinä on korkea suojausluokitus (IP67). Cobotti on nopeasti konfiguroitavissa eri tehtäviin, sillä siinä on tarkka käsiohjaus ja se on helposti ohjelmoitavissa. Yhteistyörobotti voi liikkua yhteistyötilassa 1000 mm/s nopeudella, sillä varustelusta löytyy nopeasti kosketukseen reagoiva Power and Force Limiting –tekniikka. Cobotti pystytään lisäksi asettaa toimimaan ei-yhteistyötilassa jopa 2000 mm/s nopeudella. (Yaskawa America 2023.)

Robotissa on liitettynä YRC1000 mikro -robottiohjain. Yaskawan yhteistyörobotti on tullut Tampereen ammattikorkeakoulun FieldLab-kehitysympäristöön TRE-HID-hankkeen myötä.

4.2 Omron LD-90

Omron LD-90 on autonominen mobiilirobotti (AMR), joka on suunniteltu erilaisiin teollisuussovelluksiin. Verrattuna perinteisiin autonomisesti ohjattuihin ajoneuvoihin (AGV), Omron LD-90 ei vaadi tilojen muutoksia, kuten lattiamagneetteja. Omronin oma ohjelmisto (MobilePlanner) ja ohjaukset mahdollistavat robotin älykkään navigoinnin ihmisten ja ennalta suunnittelemattomien esteiden ympärillä, jotka voivat pysäyttää perinteiset AGV:t. (Omron Corporation 2023.)

Omron AMR on integroitavissa valmiisiin automaatiojärjestelmiin. Mobiilirobotissa on yksinkertainen ohjelmointi ja useita liitäntävaihtoehtoja viestintään. LD-90 sisältää turvajärjestelmän ja -laaserit, mikä mahdollistaa toimimisen ihmisen kanssa. (Omron Corporation 2023.) Kuvassa 2 on esitelty Omron LD-90 mobiilirobotti.

KUVA 2. Omron LD-90 mobiilirobotti (Omron Corporation 2023).

4.3 Beckhoff CX5130

Beckhoffin CX5130 on sulautettu järjestelmä, jossa on Intel Atom E3827 prosessori 1.75GHz kellopulssilla ja Windows 10 käyttöjärjestelmä. Logiikassa on kaksi itsenäistä Ethernet -liitäntää ja verkkokorttia, mikä mahdollistaa eri verkkoihin kytkeytymisen, ja niiden välillä kommunikoinnin. Muina liitäntöinä ovat neljä USB-porttia sekä DVI-I -portti. (Beckhoff 2023.)

Logiikan kommunikointi I/O-kortteihin tapahtuu E-bus -terminaalin avulla. Logiikkaan on liitetty EL1008 Input- ja EL2008 Outputkortit. Molemmat I/O-kortit ovat digitaalisia sekä niissä on kahdeksan kanavaa. PLC on esitetty kuvassa 3.

KUVA 3. Beckhoff CX5130-logiikka (Beckhoff 2023).

4.4 Omron Call/Door Box

Omron Call/Door Box on komponentti, jonka avulla on mahdollista ohjata Omron mobiilirobottia suoraan automaatiolaitteen I/O-kanavilla. Call/Door boxissa on neljä kytkintä, joista kaksi toimii mobiilirobotin Input-kanavina ja kaksi Output-kanavina. Call/Door Box voidaan yhdistää mobiilirobottiin joko käyttämällä langallista Ethernet-yhteyttä tai langatonta WiFi-yhteyttä. Kuvassa 4 on esitetty Omron Call/Door Box.

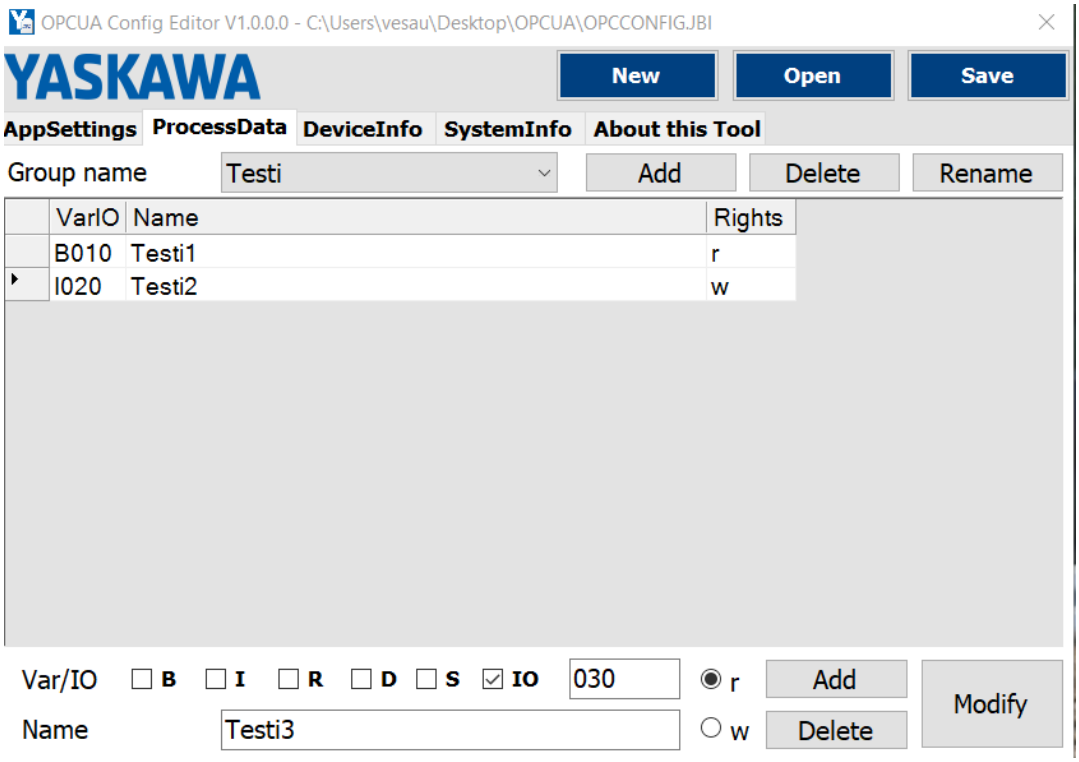
KUVA 4. Omron Door/Call Box.

5 OPC UA YMPÄRISTÖN LUOMINEN

5.1 OPC UA palvelin Yaskawa

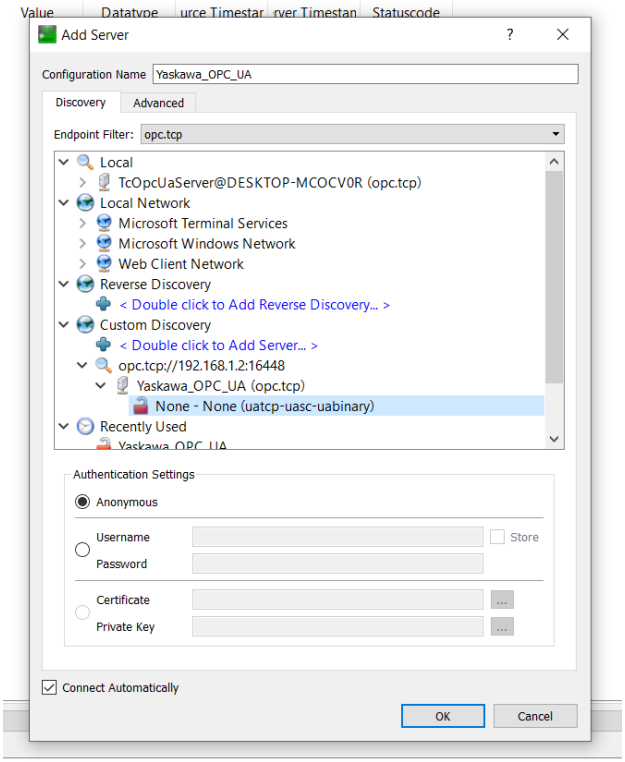
OPC UA -palvelin toteutetaan kokoonpanossa Yaskawan robottiohjaimeen. Palvelimen luonti aloitetaan lataamalla muistitikulta OPCUA.out -tiedosto robotin käsiohaimen MotoPlus aplication -valikon juureen. Palvelin luo oletuksena yhteyden muodostamiseen tarvittavan varmenteen. Palvelin on käytettävissä lisenssin aktivoimisen jälkeen.

Jotta palvelimesta voidaan nähdä muuttujia ja niiden tiloja, tulee luoda muuttujat. Robottiohjain lukee muuttujatiedoston JSON-tiedostona. Muuttujien luonti tapahtuu helpoiten Yaskawan OPC UA Config -editorilla. Editoriin nimetään ryhmä, johon muuttujat halutaan luoda. Muuttujille asetetaan numerot, tyypit ja oikeudet, jonka jälkeen tiedosto tallennetaan ’OPCCONFIG.JBI’ nimellä. Editori tallentaa tiedoston JSON-muotoon. Tiedosto siirretään muistitikulla Yaskawan robottiohjaimeen, jolloin muuttujat luodaan palvelimelle. Kuvassa 5 on esitetty muuttujien luonti OPC UA Config -editorilla.



KUVA 5. Muuttujalistaus OPC UA Config -editorilla.

Palvelimen testaamiseen on olemassa useita kaupallisia sovelluksia. Tässä opinnäytetyössä käytettään Unified Automationin UaExpert -ohjelmistoa, joka yhdistyy palvelimeen luoden testiasiakkaan. OPC UA -palvelimeen yhdistetään kirjoittamalla palvelimen IP-osoite ja sen perään porttinumero. Palvelimen IP-osoite on sama, kuin Yaskawan robottiohjaimen IP. Porttinumero Yaskawan luomiin palvelimiin on 16448. Tämän jälkeen valitaan saatavilla oleva palvelimen päätepiste. Valittavana on suojattu tai suojaamaton päätepiste, mikä riippuu palvelimen asetuksista. Kuvassa 6 on esitetty polku palvelimen yhdistämiseen. Yaskawan OPC UA -palvelimen muuttujat JSON-tiedostona on esitetty liitteessä 1.



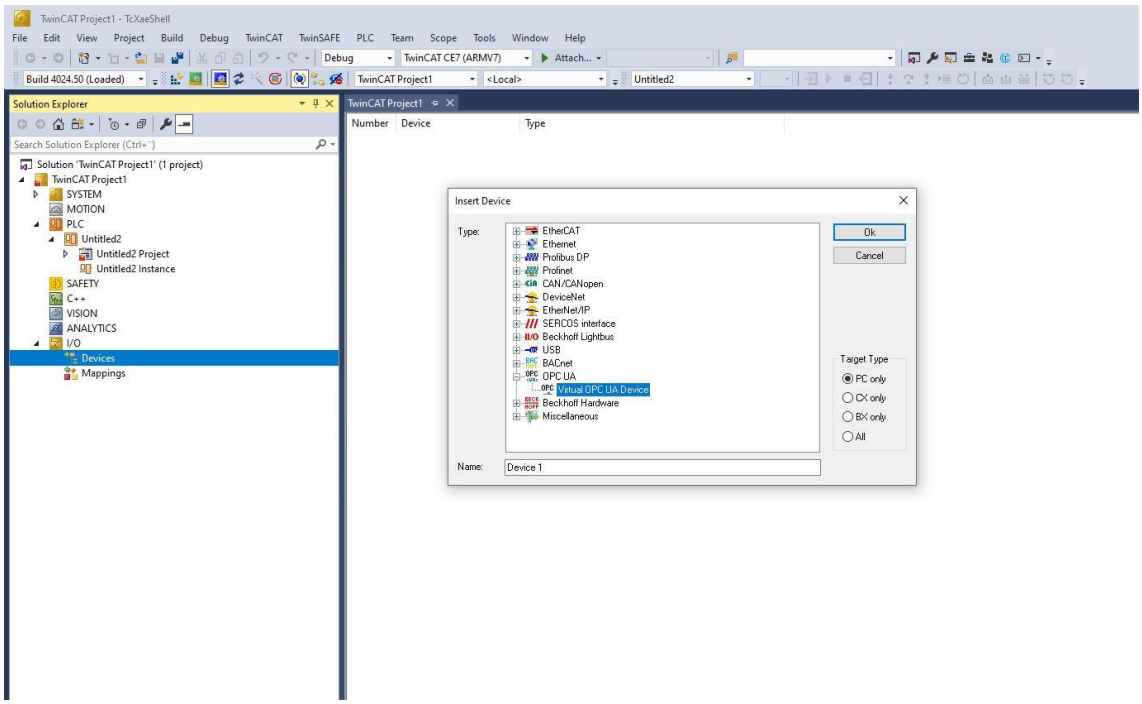
KUVA 6. Palvelimeen yhdistäminen käyttäen UaExpert-sovellusta.

5.2 OPC UA asiakkuus Beckhoff

Beckhoffin logiikoita ohjelmoidaan TwinCAT 3 XAE -ympäristöllä. XAE on integroitu Microsoft Visual studioon, jonka ansioista useiden eri ohjelmointikielien käyttö on mahdollista. Beckhoffin logiikkaan sekä ohjelmoinnissa käytettävälle tietokoneelle tulee asentaa OPC UA- laajennus, jotta OPC UA -palveluiden käyttö on mahdollista. OPC UA -asiakkuus voidaan toteuttaa kahdella eri tavalla, käyt-

täen joko Client I/O-virtuaalilaitetta tai luomalla PLCopen -rajapinta, jossa käytetään erilaisia toimintalohkoja yhteyden määrittämiseen. Tässä opinnäytetyössä OPC UA -asiakkuus on toteutettu käyttäen Client I/O-virtuaalilaitetta.

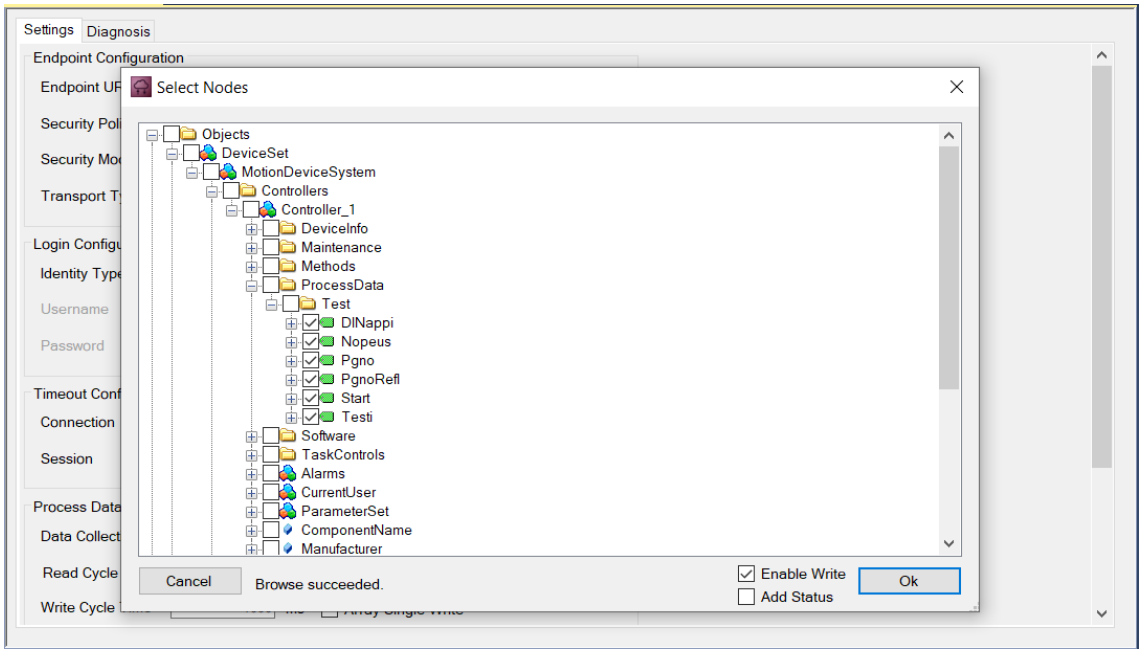
Asiakkaan tekeminen aloitetaan luomalla uusi TwinCAT XAE Projekti, jonka jälkeen lisätään uusi Standard PLC -projekti. PLC-projektin asetuksista tulee aktivoida TMC File -kohta. Tämän jälkeen valitaan projektipuun I/O valikon alta kohta Devices ja lisätään uusi laite. Ohjelmaan avautuu valintaikkuna, jossa on esitettynä mahdolliset lisättävät laitetyypit. OPC UA -valikon alta valitaan Virtual OPC UA Device, jolloin ohjelma luo virtuaalisen OPC UA -laitteen. Kuvassa 7 esitettynä I/O valintaikkuna.



KUVA 7. I/O laitteiden valintaikkuna, Virtual OPC UA Device valittuna.

I/O Devices -valikon alle ilmestyy Device 1 (OPC UA Virtual), johon lisätään uusi objekti. Valintaikkunasta valitaan OPC UA Client [module], jolloin OPC UA -asiakas on luotu. Projektipuusta valittaessa Client#1 saadaan auki asiakkaan asetukset. Endpoint URL -kohtaan kirjoitetaan alkuun opc.tpc:// ja yhdistettävän palvelimen IP-osoite ja porttinumero. Samassa välilehdessä voidaan asettaa palvelimen ja asiakkaan välisen yhteyden suojaus.

Samalta välilehdeltä löytyy myös Add Nodes -painike, jota painamalla avautuu valikko. Kuvassa 8 on esitetty valikko ja polku, josta pääsee valikoimaan asiakkaalle siirrettävät muuttujat. Mikäli palvelimen muuttujia halutaan kirjoittaa, tulee valikosta raksittaa kohta Enable Write.



KUVA 8. OPC UA palvelimelta valittavien muuttujien polku ja valitut muuttujat.

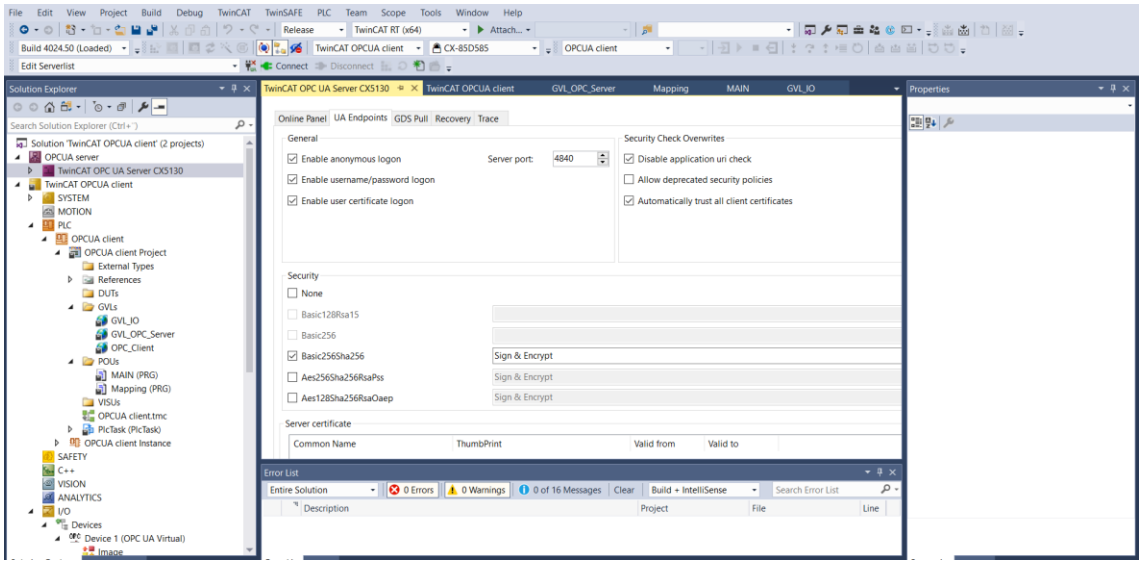
Muuttujalista saadaan luotua painamalla Client#1 asetuksista Create Plc Code -kohtaa, jolloin logiikan GVLs -kohdan alle tulee globaali muuttujalista, joka sisältää tarvittavat attribuutit ja määrittelyt. Input- ja outputmuuttujien linkkaukset tapahtuvat automaattisesti. Beckhoff OPC UA -asiakkuuden muuttujalistaus on esitetty liitteessä 2.

5.3 OPC UA palvelin Beckhoff

OPC UA -palvelin luodaan samaan projektiin kuin asiakaskin. Koska OPC UA -lisäosa on jo asennettuna logiikkaan, ei tässä tapauksessa sitä tarvitse asentaa.

OPC UA -palvelin luodaan uutena TwinCAT Connectivity -projektina, joka sisällytetään ohjelmaan. Connectivity-projektiin lisätään OPC UA Server -objekti. Näyttöön avautuu ikkuna, jossa on erinäisiä välilehtiä liittyen palvelimen tilaan ja asetuksiin. UA Endpoints -välilehdeltä voidaan määrittää palvelimen portti, pää-

tepiste ja suojaukset. Tässä tapauksessa porttina käytetään 4840: laa, ja suojaustasoksi valitaan Basic256Sha256. Kuvassa 9 on esitetty palvelimen asetuksia.



KUVA 9. Beckhoffin OPC UA-palvelimen asetukset.

Jotta palvelin saadaan pyörimään logiikkaan, tulee aktivoida TwinCAT OPC UA Configuration -työkalupalkki. Työkalupalkin avulla saadaan määritettyä päätepiste, jota palvelimen halutaan käyttävän. Työkalupalkkiin muodostuu vetovalikko, josta valitaan suojaustaso. Painamalla Connect-painiketta saadaan muodostettua yhteys palvelimeen. Add device type -painiketta painamalla muodostetaan logiikalle pääsy palvelimen Data Access -määrittelyihin. Tässä projektissa on asetettuna vain yksi logiikka, joten näyttöön tulee automaattisesti tämän logiikan tiedot.

OPC UA -palvelimeen halutaan lisätä muuttujia PLC:n ohjelmasta. Tätä varten luodaan logiikalle uusi globaali muuttujalista, jossa määritetään palvelimen muuttujat. Muuttujille tulee asettaa attribuutti OPC.UA.DA := 1, jolloin muuttujat aktivoituvat OPC UA -palvelimen Data Access -määrittelyihin. Kun muuttujat ovat määritelty, on palvelin valmiina käytettäväksi. Beckhoff OPC UA -palvelimen muuttujalistaus on esitetty liitteessä 3.

6 TOIMINNALLINEN DEMO

Tässä luvussa esitellään toiminnallisen demon ohjelmoinnin toteutusta.

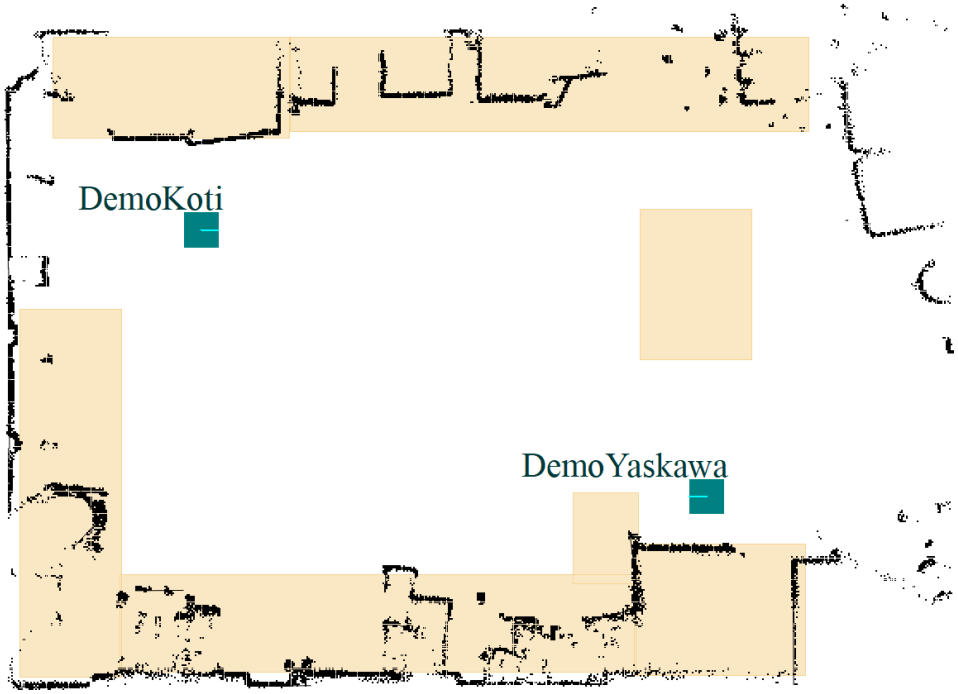
6.1 Beckhoff

Toiminnallisen demon logiikkaohjelma toteutetaan samaan projektiin, jossa on OPC UA -ympäristön toteutus. Ensimmäisenä luodaan muuttujalista, jossa määritellään ohjelmassa tarvittavat muuttujat ja niiden tyypit. Osa muuttujien tiedoista on tarpeen siirtää OPC UA -palvelimelle. Tämä tehdään luomalla siirto-ohjelma, jossa linkitetään kaksi eri muuttujaa logiikan muuttujalistasta sekä palvelimen muuttujalistasta. Näin kaksi muuttujaa saavat saman arvon. Siirto-ohjelma luetaan logiikan pääohjelman aluksi. Pääohjelman alussa tulee myös asettaa OPC UA -asiakkaan kirjoitusoikeudet sallituksi.

Ohjelma jaetaan kahteen aliohjelmaan: Omronin ohjaukseen sekä Yaskawan ohjaukseen. Omronin ohjelmassa määritetään, milloin mobiilirobotti saa liikkua maalipisteisiinsä. Yaskawan ohjauksessa määritetään, milloin yhteistyörobotin ohjelma saa alkaa. Yaskawan ohjaus logiikalla mallintaa myös tapaa, jolla robottiohjaukset toteutetaan teollisuudessa. Logiikan ohjelmat ja muuttujalista ovat esitettyinä liitteessä 4.

6.2 Omron

Omronin mobiilirobottia ohjelmoidaan Omron Mobile Planner -sovelluksella. Ohjelman teko aloitetaan luomalla tilasta kartta, jossa mobiilirobotti liikkuu. Tämä tapahtuu ajeluttamalla mobiilirobottia käsiohjaimen kanssa, jolloin robotti laserskannaa ympäristöä. Tämän jälkeen tilakartasta poistetaan ylimääräinen kohina pois sekä määritellään mobiilirobotin kielletyt alueet. Mobiilirobotille merkitään maalipisteet, joihin mobiilirobotin tulee liikkua. Kartan siistimisen ja maalipisteiden luomisen jälkeen kartta tallennetaan robotille. Kuvassa 10 on esitetty Omronin mobiilirobotin kartta, jossa kielletyt alueet ovat keltaisella pohjalla, ja maalipisteet ovat merkattu sinisillä neliöillä.



KUVA 10. Mobiilirobotin kartta Mobile Planner –sovelluksessa.

Mobiilirobotin ohjelma rakennetaan paikkapisteiden ja Omron Call/Door Boxin tulojen ja lähtöjen avulla. Ohjelma luodaan Route-välilehteen. Jotta mobiilirobotin liikkuminen ympäristössä ei aiheuta yllättäviä tilanteita, luodaan Macro-komennot robotin sisääntulojen lukemiseen. Näin ollen mobiilirobotti kertoo, milloin se lähtee paikkapisteestään liikkumaan. Mobiilirobotti ilmoittaa paikkansa Call/Door Boxin lähtöjen kautta. Mobiilirobotin ohjelma on esitetty liitteessä 5.

6.3 Yaskawa

Yaskawan yhteistyörobotti ohjelmoidaan suoraan käyttäen robottiohjainta. Demoa varten luodaan uusi ohjelma. Robotille opetetaan kaksi paikkapistettä, joiden välillä robotti liikkuu. Ohjelma toteutetaan hyödyntäen while true – looppia, jolloin robotilla on valmius suorittaa vaadittu toiminto aina ohjelman ollessa käynnissä. Robotin ohjelma tarkistaa OPC UA -palvelimelta muuttujia, jotka määrittävät, milloin robotti saa liikkua. Yaskawan ohjelma on esiteltynä liitteessä 6.

7 POHDINTA

Opinnäytetyön tarkoituksena oli luoda kokonaisuus, jossa eri valmistajien laitteistot kommunikoivat keskenään ja tehdä siitä selonteko, jota voidaan hyödyntää opetuskäytössä. Työssä tutustuttiin OPC UA -standardin toimintaan ja mahdollisuuksiin, suunniteltiin kokonaisuus, jolla OPC UA -ympäristö voitiin toteuttaa sekä toteutettiin toiminnallinen demo OPC UA -ympäristön toiminnan kuvaamiseksi.

Työn lopputuloksena oli valmis OPC UA -ympäristö toimeksiantajan tarpeet huomioiden. Toiminnallisen demon ohjelmat ovat toteutettu siten, että ohjelmien jatkokehittäminen on helposti toteutettavissa. Haasteena oli suora kommunikointi Yaskawalta Omronille, sillä Bool-tyyppisen muuttujan luominen OPC UA -palvelimelle ei ollut mahdollista. Näin ollen Beckhoffin logiikassa ei voitu asettaa muuttujia suoraan toisiinsa, vaan oli tarpeellista kirjoittaa ohjelmaan näiden muuttujien välinen yhteys.

Jatkotutkimusaiheena olisi Beckhoffin OPC UA -palvelimen yhdistäminen yläjärjestelmään, sekä toiminnallisen demon kehittäminen käyttöjärjestelmällä ja useammalla ohjelmalla. Jatkotutkimusaiheet ovat oleellisia nopeasti kehittyvällä teknologian alalla, jotta FieldLab-tila pystyy vastaamaan opetuksen ja työelämän muuttuviin tarpeisiin.

LÄHTEET

Beckhoff Automation. 2023. Beckhoff New Automation Technology. Viitattu 16.9.2023. https://www.beckhoff.com/fi-fi/

Manual Embadded-PC - CX51x0. n.d. Beckhoff. Käyttöohje. Viitattu 10.8.2023. https://download.beckhoff.com/download/Document/ipc/embedded-pc/embedded-pc-cx/cx5100_en.pdf

LD Platform Peripherals. n.d. Omron Corporation. Käyttöohje. Viitattu 17.9.2023. https://assets.omron.com/m/5f7d3ae7376db288/original/LD-Platform-Peripherals-User-s-Guide.pdf

Mahnke, W., Leitner, S & Damm, M. 2009. OPC Unified Architecture. Berliini: Springer.

Omron Corporation. 2023. Omron. Viitattu 16.9.2023. https://www.omron.com

OPC Foundation. 2023. OPC Foundation. Viitattu 14.8.2023. https://opcfoundation.org/

Tampereen ammattikorkeakoulu. n.d. TAMK FieldLab. Viitattu 19.10.2023. https://sites.tuni.fi/fieldlab/

Yaskawa America. 2023. Yaskawa America. Viitattu 10.9.2023. https://www.yaskawa.com/

Yaskawa OPC UA Server Installation Manual. n.d. Yaskawa. Käyttöohje. Viitattu 8.6.2023. Vaatii käyttöoikeuden. https://www.yaskawa.com/

LIITTEET

Liite 1: Yaskawa OPC UA -palvelin muuttujalistaus 1 (2)

```
/JOB
//NAME OPCCONFIG
//POS
///NPOS 0,0,0,0,0,0
//INST
///DATE 2023/10/14 11:56
///ATTR SC,RW
NOP
'{
'"AppSettings": {
'"CertStore": "0",
'"UserAccess": {
'"Anonymous": "1",
'"Custom": "0"},
'"Security": {
'"None": "1",
'"Basic128Rsa15": "0",
'"Basic256": "0",
'"Basic256Sha256": "0"},
'"Network": {
'"Port1": "1",
'"Port2": "0"},
'"UTC": "0",
'"DisableMethods": "0",
'"ProcessData": {
'"OPCUA ": {
'"B011": {
'"Name": "Pgno",
'"Rights": [{
'"Access": "w"}]},
'"B012": {
'"Name": "PgnoRefl",
'"Rights": [{
'"Access": "r"}]},
'"B013": {
'"Name": "Start",
'"Rights": [{
'"Access": "w"}]},
'"B014": {
'"Name": "Yaskawa_ready",
'"Rights": [{
'"Access": "r"}]},
'"B015": {
'"Name": "Omron_at_home",
'"Rights": [{
'"Access": "w"}]},
```

'"B016": {
'"Name": "Omron_at_yaskawa",
'"Rights": [{
'"Access": "w"}]},
'"B017": {
'"Name": "Omron_to_yaskawa",
'"Rights": [{
'"Access": "w"}]},
'"B018": {
'"Name": "Omron_to_home",
'"Rights": [{
'"Access": "w"}]},
'"B019": {
'"Name": "Button_1 ",
'"Rights": [{
'"Access": "w"}]}},
'"Beckhoff": {}}},
'"VendorSettings": {
'"DeviceInfo": {
'"Controller": {
'"CSProductCode": "JZRCR-BCS02-6
"',
'"CSSerialNr": "1A2068019950002"
'},
'"R1": {
'"CSSerialNr": "R21301-505-1"}}}}
END

Liite 2: Beckhoff OPC UA -asiakkuuden muuttujalistaus

```
1 VAR_GLOBAL
2
3 // OPC UA Client muuttujat
4
5 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Inputs^Status^Connected'}
6 bConnected AT %I* : BOOL ;
7 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Inputs^Status^ReadBusy'}
8 bReadBusy AT %I* : BOOL ;
9 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Inputs^Status^KeepAlives'}
10 bKeepAlives AT %I* : BOOL ;
11 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Inputs^Status^StmState'}
12 nStmState AT %I* : BYTE ;
13 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Outputs^Control^ResetStm'}
14 bResetStm AT %Q* : BOOL ;
15 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Outputs^Control^Execute'}
16 bExecute AT %Q* : BOOL ;
17 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^Outputs^Control^Write Enable'}
18 bWrite_Enable AT %Q* : BOOL ;
19
20 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.MotionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Button_1^Inputs^Value'}
21 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_ProcessData_OPCUA_Button_1 AT %I* : BYTE ;
22 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.MotionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Button_1^Outputs^Value'}
23 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_ProcessData_OPCUA_wButton_1 AT %Q* : BYTE ;
24
25 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.MotionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Omron_at_home^Inputs^Value'}
26 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_ProcessData_OPCUA_Omron_at_home AT %I* : BYTE ;
27 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.MotionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Omron_at_home^Outputs^Value'}
```

```
28 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_at_home AT %Q* : BYTE ;
29
30 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Om-
ron_at_yaskawa^Inputs^Value'}
31 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_Omron_ AT %I* : BYTE ;
32 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Om-
ron_at_yaskawa^Outputs^Value'}
33 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_at_yaskawa AT %Q* : BYTE ;
34
35 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Omron_to_home^In-
puts^Value'}
36 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_Omron_to_home AT %I* : BYTE ;
37 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Om-
ron_to_home^Outputs^Value'}
38 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_to_home AT %Q* : BYTE ;
39
40 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Om-
ron_to_yaskawa^Inputs^Value'}
41 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_Omron_to_yaskawa AT %I* : BYTE ;
42 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Om-
ron_to_yaskawa^Outputs^Value'}
43 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_to_yaskawa AT %Q* : BYTE ;
44
45 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Pgno^Inputs^Value'}
46 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_Pgno AT %I* : BYTE ;
47 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Pgno^Outputs^Value'}
48 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wPgno AT %Q* : BYTE ;
49
50 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.PgnoRefl^Inputs^Val-
ue'}
```

```
51 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_PgnoRefl AT %I* : BYTE ;
52
53 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Start^Inputs^Value'}
54 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_Start AT %I* : BYTE ;
55 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Start^Outputs^Value'}
56 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wStart AT %Q* : BYTE ;
57
58 {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client#1^DeviceSet.Mo-
tionDeviceSystem.Controllers.Controller_1.ProcessData.OPCUA.Yaskawa_ready^In-
puts^Value'}
59 DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_Yaskawa_ready AT %I* : BYTE ;
60
61 END_VAR
```

Liite 3: Beckhoff OPC UA -palvelin muuttujalistaus

```
1 {attribute 'qualified_only'}
2 VAR_GLOBAL
3 // Beckhoff OPC UA Serverin muuttujat
4
5 {attribute 'OPC.UA.DA' := '1'}
6 ServerOPCPgnoRefl : BYTE ;
7
8 {attribute 'OPC.UA.DA' := '1'}
9 ServerOPCPgno : BYTE ;
10
11 {attribute 'OPC.UA.DA' := '1'}
12 ServerOPCStart : BYTE ;
13
14 {attribute 'OPC.UA.DA' := '1'}
15 ServerOPCYaskawaReady : INT ;
16
17 {attribute 'OPC.UA.DA' := '1'}
18 ServerOPCOmronAtYas : BYTE ;
19
20 {attribute 'OPC.UA.DA' := '1'}
21 ServerOPCOmronAtHome : BYTE ;
22
23 {attribute 'OPC.UA.DA' := '1'}
24 ServerOPCOmronToYas : BYTE ;
25
26 {attribute 'OPC.UA.DA' := '1'}
27 ServerOPCOmronToHome : BYTE ;
28
29 {attribute 'OPC.UA.DA' := '1'}
30 ServerOPCButton_1 : BYTE ;
31
32 END_VAR
```

Liite 4: Beckhoff muuttujalista ja ohjelmat

Global Variable List: GVL_IO

```
1 {attribute 'qualified_only'}
2 VAR_GLOBAL
3
4 // OPC UA muuttujat
5 OPCClientStatus : BYTE ;
6 OPCPgnoRefl : BYTE ;
7 OPCStart : BYTE ;
8 OPCPgno : BYTE ;
9 OPCYaskawaReady : INT ;
10 OPCOmronAtYas : BYTE ;
11 OPCOmronAtHome : BYTE ;
12 OPCOmronToYas : BYTE ;
13 OPCOmronToHome : BYTE ;
14 OPCButton_1 : BYTE ;
15
16 // I/O muuttujat
17 Button_1 AT %I* : BOOL ;
18
19 // Omron AMR IO muuttujat
20 OmronAtYaskawa AT %I* : BOOL ;
21 OmronAtHome AT %I* : BOOL ;
22 OmronToYaskawa AT %Q* : BOOL ;
23 OmronToHome AT %Q* : BOOL ;
24
25 END_VAR
26
```

POU: Mapping

```
1 PROGRAM Mapping
2 VAR
3 END_VAR
4
1 // OPC UA client muuttujien linkkaukset
2 GVL_IO. OPCYaskawaReady := OPC_Client .DeviceSet_MotionDeviceSystem_Control-
lers_Controller_1_ProcessData_OPCUA_Yaskawa_ready;
3 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wStart := GVL_IO . OPCStart;
4 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wPgno := GVL_IO . OPCPgno;
5 GVL_IO . OPCPgnoRefl := OPC_Client .DeviceSet_MotionDeviceSystem_Control-
lers_Controller_1_ProcessData_OPCUA_PgnoRefl;
```

```
6 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_at_home := GVL_IO . OPCOmronAtHome;
7 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_at_yaskawa := GVL_IO . OPCOmronAtYas ;
8 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_to_home := GVL_IO . OPCOmronToHome ;
9 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wOmron_to_yaskawa := GVL_IO . OPCOmronToYas ;
10 OPC_Client.DeviceSet_MotionDeviceSystem_Controllers_Controller_1_Pro-
cessData_OPCUA_wButton_1 := GVL_IO . OPCButton_1 ;
11
12 // Byte <-> bool linkkaukset
13 IF ( GVL_IO . OmronAtHome ) THEN
14 GVL_IO . OPCOmronAtHome := 1 ;
15 ELSE
16 GVL_IO . OPCOmronAtHome := 0 ;
17 END_IF
18
19 IF ( GVL_IO . OmronAtYaskawa ) THEN
20 GVL_IO . OPCOmronAtYas := 1 ;
21 ELSE
22 GVL_IO . OPCOmronAtYas := 0 ;
23 END_IF
24
25 IF ( GVL_IO . Button_1 ) THEN
26 GVL_IO . OPCButton_1 := 1 ;
27 ELSE
28 GVL_IO . OPCButton_1 := 0 ;
29 END_IF
30
31 IF ( GVL_IO . OmronToHome ) THEN
32 GVL_IO . OPCOmronToHome := 1 ;
33 ELSE
34 GVL_IO . OPCOmronToHome := 0 ;
35 END_IF
36
37 IF ( GVL_IO . OmronToYaskawa ) THEN
38 GVL_IO . OPCOmronToYas := 1 ;
39 ELSE
40 GVL_IO . OPCOmronToYas := 0 ;
41 END_IF
42
43
44
45 // OPC UA server muuttujien linkkaus
46
47 OPC_Server . ServerOPCPgno := GVL_IO . OPCPgno ;
48 OPC_Server . ServerOPCPgnoRefl := GVL_IO . OPCPgnoRefl ;
```

```
49 OPC_Server . ServerOPCStart := GVL_IO . OPCStart ;
50 OPC_Server . ServerOPCButton_1 := GVL_IO . OPCButton_1 ;
51 OPC_Server . ServerOPCOmronAtHome := GVL_IO . OPCOmronAtHome ;
52 OPC_Server . ServerOPCOmronAtYas := GVL_IO . OPCOmronAtYas ;
53 OPC_Server . ServerOPCOmronToHome := GVL_IO . OPCOmronToHome ;
54 OPC_Server . ServerOPCOmronToYas := GVL_IO . OPCOmronToYas ;
55 OPC_Server . ServerOPCYaskawaReady := GVL_IO . OPCYaskawaReady ;
```

POU: MAIN

```
1 PROGRAM MAIN
2 VAR
3 bFirsttime : BOOL := TRUE ;
4 // Esitellään FB:t
5 OmronControl : FB_OmronControl ( ) ;
6 YaskawaControl : FB_YaskawaControl ( ) ;
7 END_VAR
8
1 // Sallitaan OPC -clientin muuttujien kirjoitus
2 OPC_Client . bWrite_Enable := TRUE ;
3
4 // Muuttujien linkkausohjelma
5 Mapping ( ) ;
6
7 // Robottien ohjelmakierrot
8 OmronControl ( ) ;
9 YaskawaControl ( ) ;
10
```

POU: FB_OmronControl

```
1 FUNCTION_BLOCK FB_OmronControl
2 VAR_INPUT
3 END_VAR
4 VAR_OUTPUT
5 END_VAR
6 VAR
7 PendingTask : BOOL := FALSE ;
8 LastYaskawaReady : BOOL := FALSE ;
9 END_VAR
10
1 // Uuden tehtävän anto Omronille
2 IF ( OmronAtDestination AND NOT PendingTask ) THEN
3 IF ( GVL_IO . Button_1 AND NOT GVL_IO . OmronAtYaskawa ) THEN
4 SetOmronTask ( EOmronTasks . TO_YASKAWA ) ;
5 END_IF
6 IF ( GVL_IO . OPCYaskawaReady = 1 AND NOT LastYaskawaReady AND NOT
GVL_IO . OmronAtHome ) THEN
```

```
7 SetOmronTask ( EOmronTasks . TO_HOME ) ;
8 END_IF
9 END_IF
10
11 // Nousva reunan tarkistus
12 LastYaskawaReady := GVL_IO . OPCYaskawaReady <> 0 ;
13
14 // Odottavan tehtävän nollaus
15 IF ( NOT OmronAtDestination ) THEN
16 GVL_IO . OmronToHome := 0 ;
17 GVL_IO . OmronToYaskawa := 0 ;
18 PendingTask := FALSE ;
19 END_IF
20
```

Method: SetOmronTask

```
1 METHOD PRIVATE SetOmronTask : BOOL
2 VAR_INPUT
3 Task : EOmronTasks ;
4 END_VAR
5
1 CASE Task OF
2 EOmronTasks . TO_HOME :
3 GVL_IO . OmronToHome := 1 ;
4
5 EOmronTasks . TO_YASKAWA :
6 GVL_IO . OmronToYaskawa := 1 ;
7 END_CASE
8
9 PendingTask := TRUE ;
10
```

POU: FB_YaskawaControl

```
1 FUNCTION_BLOCK FB_YaskawaControl
2 VAR_INPUT
3 END_VAR
4 VAR_OUTPUT
5 END_VAR
6 VAR
7 OmronArrived : BOOL := FALSE ;
8 LastOmronAtYaskawa : BOOL := FALSE ;
9 END_VAR
10
1 // OPC Start ja Pgno nollaus
```

```
2 IF ( GVL_IO . OPCYaskawaReady = 0 ) THEN
3 GVL_IO . OPCStart := 0 ;
4 GVL_IO . OPCPgno := 0 ;
5 END_IF
6
7 // Aseta OmronArrived vain silloin kun Omron saapuu Yaskawalle
8 IF ( GVL_IO . OmronAtYaskawa AND NOT LastOmronAtYaskawa ) THEN
9 OmronArrived := TRUE ;
10 END_IF
11 LastOmronAtYaskawa := GVL_IO . OmronAtYaskawa ;
12
13 // Omron Yaskawalla, Yaskawa valmiina
14 IF ( OmronArrived ) AND ( GVL_IO . OPCYaskawaReady = 1 ) THEN
15 GVL_IO . OPCPgno := 1 ;
16 OmronArrived := FALSE ;
17 END_IF
18
19 // Ohjelman aloituksen ristilukot
20 IF ( GVL_IO . OPCPgnoRefl = GVL_IO . OPCPgno AND GVL_IO . OPCPgnoRefl = 1
AND GVL_IO . OmronAtYaskawa ) THEN
21 GVL_IO . OPCStart := 1 ;
22 END_IF
23
24
```

Liite 5: Omron mobiilirobotin ohjelma

Routes:

Demo

DemoKoti // Robotti kotipisteeseen

CustomOutputOn (BOX_DO1) // Robotti kotona ON

CustonInputsCheckAll (-1.0, Input_to_yas.., BOX_DI2) // Tarkistus DI2, lupa siirtyä Yaskawalle

CustomOutputOff (BOX_DO1) // Robotti kotona OFF

DemoYaskawa // Robotti Yaskawalle

CustomOutputOn (BOX_DO2) // Robotti Yaskawalla ON

CustonInputsCheckAll (-1.0, Input_to_home, BOX_DI1) Tarkistus DI1, lupa siirtyä kotiin

CustomOutputOff (BOX_DO2) // Robotti Yaskawalla OFF

DemoKoti // Robotti kotipisteeseen

CustomOutputOn (BOX_DO1) // Robotti kotona ON

Macros:

Input_to_home

sayInstant (Lähden kotiin, Normal, True, 0.0)

Input_to_home

sayInstant (Menen Yaskawalle, Normal, True, 0.0)

Liite 6: Yaskawan robotin ohjelma

```
ROBOT JOB - A-OPCUA
1       Start Job
2       LinearMove P075 Speed= 100.0 (mm/sec) // Robotti kotipisteeseen
3       Set B014 1 // Asetetaan Yaskawa_Ready == 1
4       While (B010=1) // While True loop
5               Set B11 B12 // Asetetaan PgnoRefl Pgno:n arvoksi
6               If (B013=B011 And B011 != 0) Then // Robotin ohjelman
        käynnistymisen ehdot
7                       Set B014 0 // Yaskawa_Ready == 0
8                       LinearMove P076 Speed= 100.0 (mm/sec)
        // Robotti Omronille
9                       LinearMove P075 Speed= 100.0 (mm/sec)
        // Robotti kotipisteeseen
10              EndIf
11              Set B014 1 // Asetetaan Yaskawa_Ready == 1
12      EndWhile
13      End Job
```