Iiro Kämäräinen

KUKA MXAUTOMATION-RAJAPINTA

Robotin ohjaus Siemensin S7-1500 -sarjan logiikalla

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutusohjelma
Huhtikuu 2018

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Huhtikuu 2018	**Tekijä/tekijät** Iiro Kämäräinen
Koulutusohjelma Tieto- ja viestintätekniikka		
Työn nimi KUKA MXAUTOMATION-RAJAPINTA. Robotin ohjaus Siemensin S7-1500 -sarjan logiikalla		
Työn ohjaaja Hannu Ala-Pöntiö		**Sivumäärä** 48 + 0
Työelämäohjaaja Pasi Kylmälä		

Opinnäytetyön tavoitteena oli tutustua KUKA.PLC mxAutomation -rajapintaan ja toteuttaa sitä hyödyntävä sovellusesimerkki Siemensin S7-1500 -sarjan logiikalla. Rajapinta mahdollistaa KUKA:n robotin ohjauksen ulkoisen ohjelmoitavan logiikan avulla. Opinnäytetyön toimeksiantajana oli kokkolalainen sähkö- ja automaatioalan yritys Apex Automation.

Sovellusesimerkki toteutettiin Apex Automationin omalla testirobotilla. Opinnäytetyön teoriaosassa on käyty läpi robotiikan perusteita sekä yleisiä asioita Siemensin TIA Portalista ja rajapinnan käyttöönotosta TIA Portal -ympäristössä. Käytännön osassa on kerrottu tehdystä sovellusesimerkistä sekä kohdatuista ongelmista.

Sovellusesimerkki toteutettiin Apex Automationin teknologiapäivää varten, jossa sitä oli tarkoitus käyttää rajapinnan toiminta-ajatuksen esittelyyn. Sovellusesimerkki toteutettiin tekemällä logiikkaohjelma sekä käyttöliittymä, joiden avulla oli mahdollista ohjata robottia, määrittää sen asetuksia ja lukea robottisolun diagnostiikkatietoja. Opinnäytetyö päättyy omaan pohdintaan sekä toteutetun sovellusesimerkin ja perinteisen robottiohjauksen vertailuun.

Asiasanat
Automaatio, ohjelmoitavat logiikat, robotiikka

ABSTRACT

Centria University of Applied Sciences	Date April 2018	Author Iiro Kämäräinen
Degree programme Information and communication technology		
Name of thesis KUKA MXAUTOMATION INTERFACE. Controlling a robot with a Siemens S7-1500 series PLC		
Instructor Hannu Ala-Pöntiö		**Pages** 48 + 0
Supervisor Pasi Kylmälä		

The aim of the thesis was to study the KUKA.PLC mxAutomation interface and to create an application example on a Siemens S7-1500 series programmable logic controller (later PLC) using the interface. The interface allows a KUKA robot to be controlled with an external PLC. The thesis was commissioned by Apex Automation, a company from Kokkola specialised in automation and electrical design.

The application example was carried out on Apex Automation's own test robot. In the theoretical part of the thesis the basics of robotics are explained. Basic information about Siemens's TIA Portal and how to use mxAutomation with it is also provided. In the practical part the application example is described and problems faced when designing it are addressed.

The application example was used to demonstrate the basic idea of the interface at a technology fair held by Apex Automation. The application example was implemented by creating a PLC program and an accompanying human-machine-interface to configure and control the robot and to access the diagnostics of the robot cell. The thesis is concluded with the authors thoughts on the subject and a comparison of the application example and the traditional approach to robot control.

Key words
Automation, PLCs, robotics

TIIVISTELMÄ
ABSTRACT
SISÄLLYS

KUVIOT

KUVAT

TAULUKOT

1 JOHDANTO

Opinnäytetyön tavoitteena oli tutustua robottivalmistaja KUKA:n kehittämään mxAutomation-rajapintaan ja sen hyödyntämiseen Siemensin S7-1500 -sarjan ohjelmoitavalla logiikalla. Rajapinnan tarkoituksena on mahdollistaa robotin liikkeiden ohjaus, asetusten määrittely sekä vikatietojen luku ulkoisen logiikkaohjaimen avulla (KUKA.PLC mxAutomation). Opinnäytetyö koostui rajapintaan tutustumisesta, sen hyödyntämisestä käytännön robottisovelluksessa sekä vertailusta perinteiseen robotin ohjaukseen.

Käytännön robottisovellus toteutettiin Apex Automationin koulutus- ja testauskäyttöön tarkoitetulla KUKA:n robotilla sekä Siemensin S7-1500 -sarjan logiikalla ja TP 1500 Comfort -paneelilla. Apex Automation on toteuttanut robotiikkaratkaisuja asiakkailleen mm. metalli- sekä puunjalostusteollisuudessa (Apex Automation 2017).

Opinnäytetyön teoriaosassa on kerrottu robotiikan perusteista. Tähän sisältyvät robotiikan ja teollisuusrobotin käsitteiden määrittely, lyhyt katsaus robotiikan historiaan, robotiikan kinematiikan ja tekniikan peruskäsitteiden määrittely, maininta turvallisuutta koskevista säädöksistä sekä rajapinnan ja työssä käytetyn robotin valmistajan KUKA:n esittely. Lisäksi teoriaosassa on kerrottu lyhyesti TIA Portalista ja sen ominaisuuksista, IEC 61131-standardista sekä HMI:n eli ihminen-kone-rajapinnan suunnittelusta. Teoriaosa päättyy rajapinnan asennusprosessin läpikäyntiin TIA Portal -ympäristössä.

Opinnäytetyön käytännön osassa on kerrottu toteutetun sovellusesimerkin vaatimista muutoksista käytetyn robottisolun laitteistoon sekä tehdystä logiikkaohjelmasta ja käyttöliittymästä. Logiikkaohjelman ja käyttöliittymän osalta on käyty läpi yksityiskohtaisesti sovelluksen rakennetta ja käytettyjä mxAutomation-kirjastolohkoja.

Opinnäytetyön toimeksiantoon sisältyi myös rajapinnan avulla toteutetun- sekä perinteisen robottiohjauksen vertailua. Tämä lähtökohta tarjosi ainutkertaisen mahdollisuuden lähestyä teollisuusrobottia ensimmäistä kertaa kahdesta hyvin erilaisesta näkökulmasta: erillisenä koneena, jolla on omat ominaisuutensa ja toimintalogiikkansa sekä toisaalta mxAutomationin kautta osana suurempaa kokonaisuutta, ulkoisen ohjaimen hallitsemana laitteena, jonka omat

ohjaustoiminnot ovat merkityksettömiä, jopa haitallisia korkeamman tason järjestelmän näkökulmasta.

Robottien määrä on Kansainvälisen robottiyhdistyksen tilastojen perusteella vakaassa kasvussa (International Federation of Robotics 2017b.). Robotics Finlandin Cristina Anderssonin mukaan teollisuudessa on tulevaisuudessa tavoitteena robotisoida kaikki mikä voidaan. Robotisointi nähdään myönteisenä tekijänä talouskasvulle sekä tuotantoteollisuuden palautumiselle Aasiasta. (Anteroinen 2015.) Robotisoinnin vaikutuksista työpaikkoihin kiistellään. Ennusteet vaihtelevat neutraaleista tai positiivisista vaikutuksista katastrofaaliseen massatyöttömyyteen jo lähitulevaisuudessa (International Federation of Robotics 2017a; Stark 2017).

Robottiohjauksen siirtäminen ohjelmoitavalle logiikalle tuo mukanaan monia etuja tuotantolaitoksiin, joissa käytetään jo logiikkaohjauksia muissa laitteissa. Yhteinen ohjausjärjestelmä mahdollistaa samojen ohjelmointityökalujen käytön ja vikatietojen päätymisen samaan paikkaan. Lisäksi integrointi mahdollistaa tietojen esittämisen samassa käyttöliittymässä, esimerkiksi yhdellä paneelilla. Se tekee myös robottiohjelmoinnista helpommin lähestyttävää, sillä useimmat alan insinöörit hallitsevat jo ohjelmoitavien logiikoiden käytön. (Wicks 2014.) Koska yhtenä robotiikan kehityksen haasteena voidaan pitää osaavan työvoiman puutetta (International Federation of Robotics 2017a; Kwang 2017), vastaa mxAutomation-rajapinnan kaltainen ratkaisu ajan tarpeita.

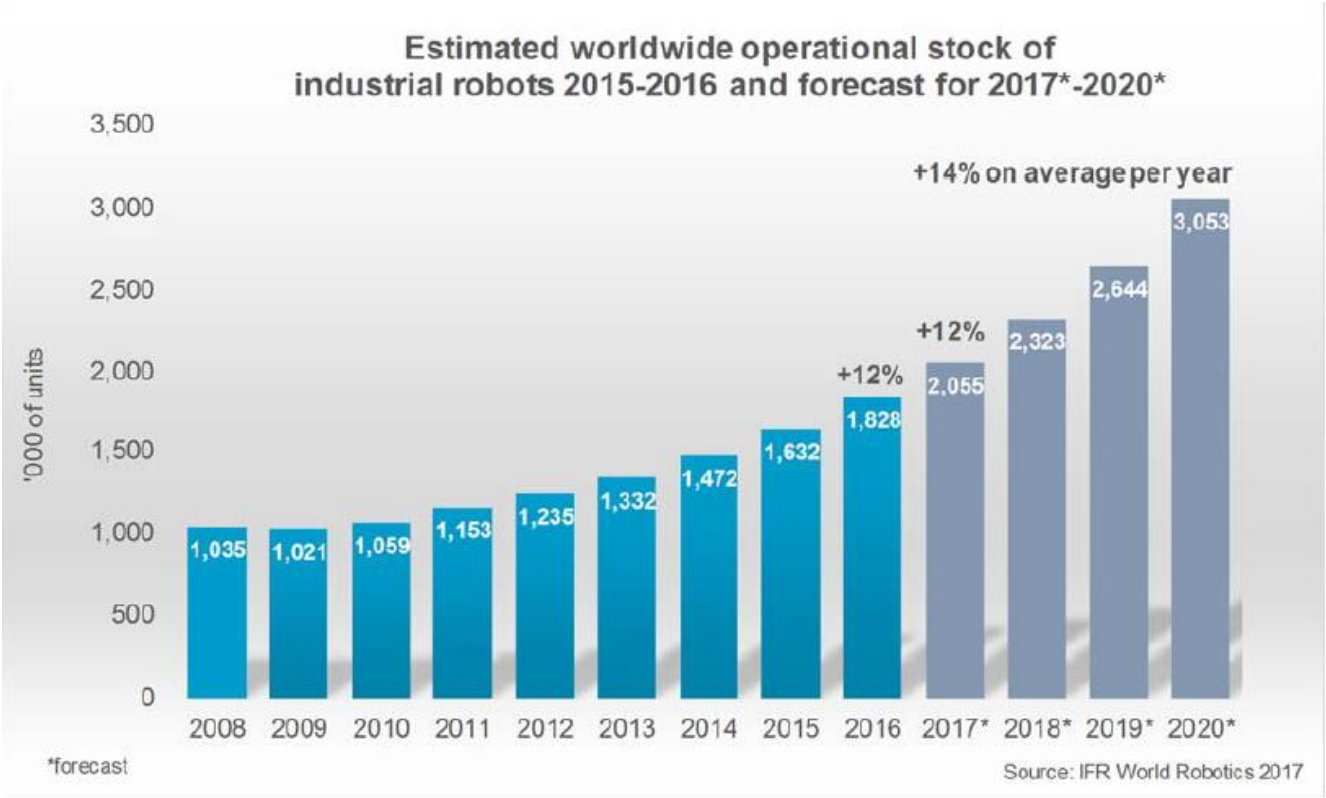
2 ROBOTIIKKA

Robotiikka määritellään ISO 8373-standardin mukaan robottien tutkimukseksi, sekä niiden suunnitteluksi, tuotannoksi ja käytöksi. Teollisuusrobotti taas luokitellaan standardin perusteella teollisuuden automaatiosovellukseen tarkoitetuksi automaattiohjatuksi, uudelleenohjelmoitavaksi ja monikäyttöiseksi käsittelylaitteeksi, joka koostuu kolmesta tai useammasta ohjattavissa olevasta liikenivelestä. Teollisuusrobotti voi olla paikoilleen kiinnitetty tai liikkuva. (ISO 8373:2012.) Kansainvälinen robotiikkayhdistys noudattaa standardin mukaista määritelmää (Industrial Robots). Myös esimerkiksi British Automation & Robot Associationin määritelmä on lähes vastaava. Toisaalta esimerkiksi Japanin Teollisuusrobottiyhdistyksen määritelmä sisältää myös ihmisen suoraan ohjaamat sekä sellaiset käsittelylaitteet, joita ei voida uudelleenohjelmoida. (British Automation & Robot Association.)

Teollisuusrobottien kehitys alkoi 1950-luvulla. Robotiikan syntymisen mahdollisti kauko- ja numeerisen ohjauksen kehitys. George C. Devol anoi vuonna 1954 patenttia kuljetuslaitteelle, jonka toiminta perustui opetettujen liikkeiden toistoon. (Asada 2005a, 1.) Devol ja Joseph Engelberger kehittivät ensimmäisen teollisuusrobotin vuonna 1959. Kaksi vuotta myöhemmin Unimate:n Unimation-teollisuusrobotti asennettiin ensimmäisenä General Motorsin Ternstedtin tehtaalle, jossa se lajitteli ja pinosi painevalettuja metalliosia. Vuonna 1969 General Motors otti käyttöön autoteollisuuden ensimmäisen robotteja hyödyntävän pistehitsauslinjan. Muita teollisuusrobotiikan merkkipaaluja olivat mm. vuonna 1974 ASEA:n kehittämä ensimmäisen täysin sähköinen, mikroprosessoriohjattu robotti sekä SCARA-robotin kehittäminen Japanissa 1970-luvun lopulla. (Lahden ammattikorkeakoulu 2016, 3; Robot History.)

Koko maailman osalta teollisuusrobottien määrä on kasvanut vakaasti 2010-luvulla ja vuosikymmenen viimeisiltä vuosilta odotetaan keskimäärin 14 % vuosittaista kasvua (KUVA 1). Uusia teollisuusrobotteja toimitetaan eniten Aasiaan. (International Federation of Robotics 2017b, 15–16) Suomessa teollisuuden robottikanta pieneni muutaman prosentin vuodessa 2010-luvun alussa, mutta Suomen robottiyhdistyksen mukaan lasku on pysähtynyt vuonna 2016 (Suomen robotiikkayhdistys 2016). Merkittäviä teollisuusrobottien valmistajia ovat mm. ABB, Fanuc, Motoman, Kawasaki ja KUKA (Lahden ammattikorkeakoulu 2016, 7). Teollisuusrobottien yleisimpiä käyttökohteita ovat auto-, elektroniikka- ja metalliteollisuus. (International Federation of

Robotics 2017b, 19). Yleisiä tehtäviä ovat mm. hitsaus (piste- ja kaarihitsaus) sekä kappaleiden lavaus- ja panostustehtävät (Lahden ammattikorkeakoulu 2016, 68–71).

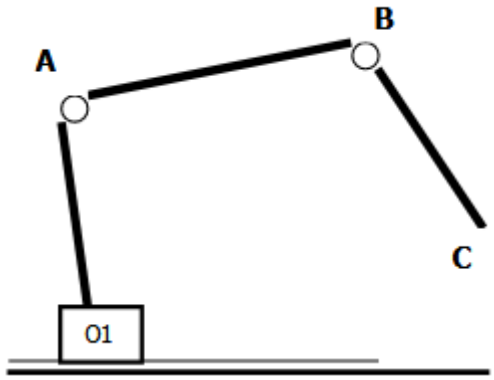


KUVIO 1. Arvio koko maailmassa käytössä olevien robottien määrästä 2015–16 ja ennuste vuosille 2017–2020 (International Federation of Robotics 2017b, 24)

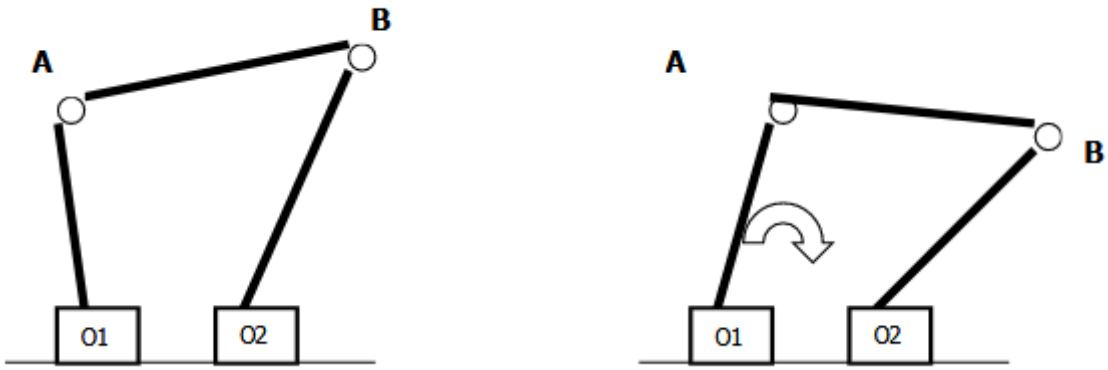
2.1 Teollisuusrobotin rakenne

Robotti koostuu manipulaattorista ja ohjausjärjestelmästä. Ohjausjärjestelmään kuuluvat yleensä robottiohjain, käsiohjelmointilaite (engl. teach pendant) sekä ulkoiset liitännät lisälaitteita ja tiedonsiirtoa varten. (KUKA 2015a, 17; Lahden ammattikorkeakoulu 2016, 44.) Manipulaattori koostuu toisiinsa liitetyistä tukivarsista, joiden suhteellinen liike toisiinsa nähden on yleensä suoraviivaista tai kiertyvää. Robottien rakenteet voidaan jakaa avoimen tai suljetun kinematiikan rakenteiksi sen perusteella, onko robotin tukivarret kytketty peräkkäin (KUVA 2)

vai rinnakkain (KUVA 3). Tukivarsien välisiä liikeniveliä kutsutaan vapausasteiksi. (Asada 2005b, 1; Lahden ammattikorkeakoulu 2016, 28–29.)



KUVA 1. Kolmen vapausasteen avoimen kinematiikan rakenne (Lahden ammattikorkeakoulu 2016, 29)



KUVA 2. Yhden vapausasteen suljetun kinematiikan rakenne (Lahden ammattikorkeakoulu 2016, 29)

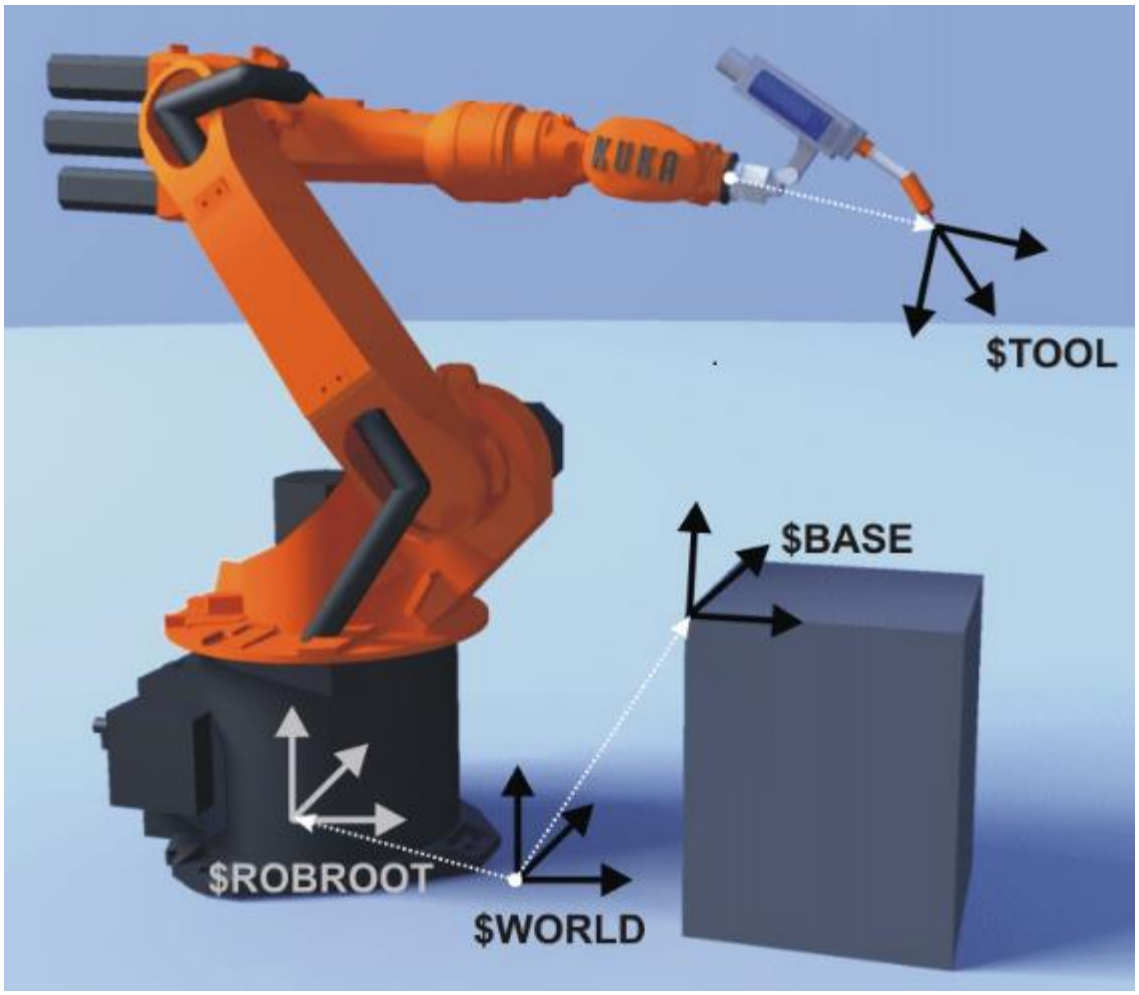
Yleisiä teollisuusrobottityyppejä ovat vapausasteiden perusteella määriteltynä suorakulmainen, sylinteri-, napakoordinaatti-, SCARA-, kiertyvänivelinen sekä rinnakkaisrakenteinen robotti (Asada 2005b; Crowder 1998; International Federation of Robotics 2017c, 33; Lahden ammattikorkeakoulu 2016, 7). Suorakulmaisen robotin kolme ensimmäistä vapausastetta ovat lineaarisia. Nurkista tuettua suorakulmaista robottia kutsutaan portaalirobotiksi (Lahden ammattikorkeakoulu 2016, 12). Sylinterirobotilla on yksi kiertyvä ja kaksi suoraviivaista liikeniveltä, napakoordinaattirobotilla taas kaksi kiertyvää ja yksi suoraviivainen liikenivel. SCARA-robotilla

on kolme kiertyvää, sekä yksi lineaarisen pystyliikkeen toteuttava liikenivel. Kiertyvänivelisen robotin kaikki vapausasteet ovat kiertyviä. (Asada 2005b; Crowder 1998; International Federation of Robotics 2017c; Lahden ammattikorkeakoulu 2016, 11–17.) Rinnakkaisrakenteisissa roboteissa kaksi tai useampia varsia on kytketty rinnakkain. (Asada 2005b; International Federation of Robotics 2017c, 33–34; Lahden ammattikorkeakoulu 2016, 11–17.)

2.2 Koordinaatistot

Koordinaatistolla tarkoitetaan sääntöjoukkoa, jonka perusteella pisteelle voidaan antaa koordinaatit, eli numeeriset arvot, jotka kuvaavat sen sijaintia (Sanastokeskus TSK ry, 13, 17). Robotin sijainnin määrittämiseen käytetään yleensä oikeankätisiä suorakulmaisia koordinaatistoja. Näitä ovat universaali- eli yleiskoordinaatisto sekä perus- ja työkalukoordinaatistot. (KUKA 2015a, 64–65; Lahden ammattikorkeakoulu 2016, 28–29.)

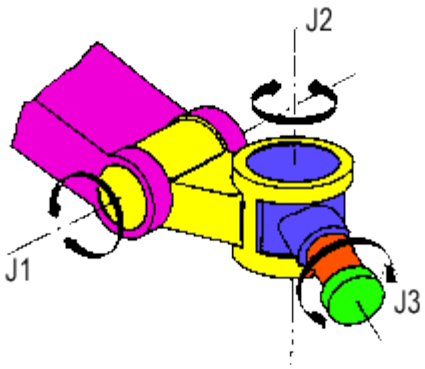
Universaali-, eli yleiskoordinaatisto on muuttumaton robotin ulkopuolinen koordinaatisto. Peruskoordinaatisto on universaalikoordinaatistoon perustuva vapaasti määriteltävissä oleva koordinaatisto. Sen avulla voidaan määrittää esimerkiksi työstökappaleita tai kiinnikkeitä. Työkalukoordinaatiston origo on oletuksena työkalulaipan keskipisteessä, mutta käyttäjä voi määrittää sen haluamansa työkalun keskipisteeseen. (KUKA 2015a, 64–65; Lahden ammattikorkeakoulu 2016, 28–29.) Kuvassa 4 esitetylle KUKA:n robotille on lisäksi määritelty ”$ROBROOT”-koordinaatisto, jonka origo sijaitsee aina robotin jalustan keskipisteessä. (KUKA 2015a, 64–65.)



KUVA 3. KUKA:n robotin koordinaatistot (KUKA 2015a, 64)

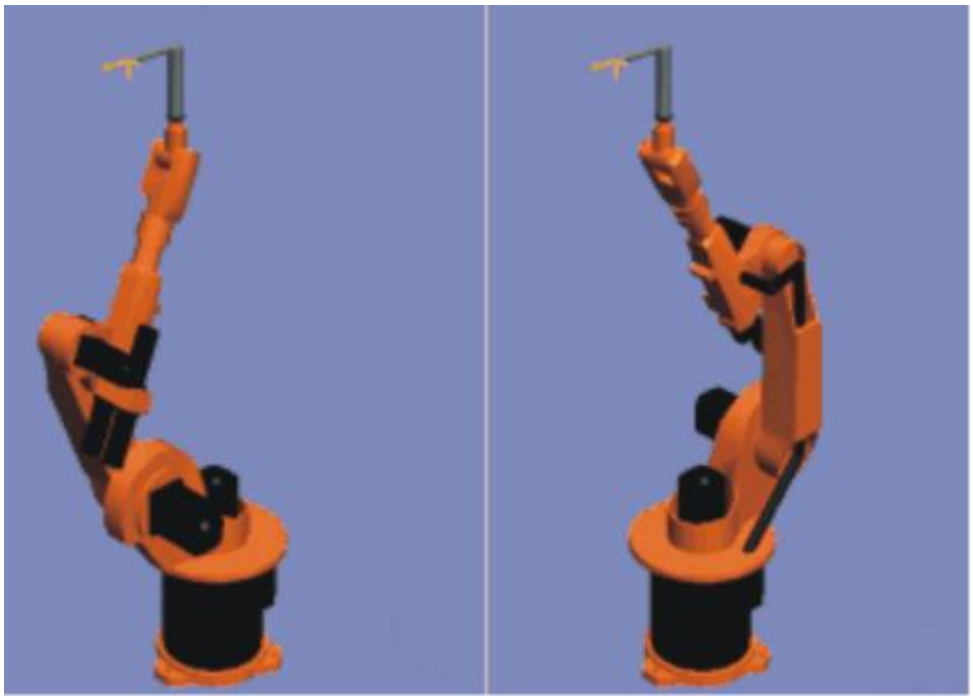
2.3 Muita parametreja

Robotin työkalun keskipisteen aseman lisäksi on kyettävä ohjaamaan työkalun orientaatiota. Orientaatiolla viitataan asentoon, jossa työkalu lähestyy työkappaletta. Orientaatiota säädetään robotin ”ranteella”, eli kolmella kauimmaisella akselilla robotin jalustasta katsottuna (KUVA 5). (Crowder 1998; Lahden ammattikorkeakoulu 2016, 39.) Suorakulmaisten koordinaatistossa näiden nivelten aseman muutos toteuttaa oikeankätiset kiertymät X-, Y- ja Z-akselien ympäri. Näitä voidaan kutsua esim. nimillä A, B ja C. (Siemens 2016, 49.) Liikkeistä voidaan käyttää myös lentodynamiikasta peräisin olevia termejä kallistus, kääntyminen ja nyökkäys (engl. roll, pitch & yaw) (Crowder 1998; Hall 2015; Lahden ammattikorkeakoulu 2016).



KUVA 4. Robotin ”ranne” (Crowder 1998)

”Status”- ja ”turn”-muuttujat mahdollistavat robotin tarkan aseman määrittämisen yksiselitteisesti. Robottiohjain ei ilman näitä tietoja kykene muuntamaan koordinaattien ja työkalun orientaation avulla esitettyä asemaa akselikulmiksi, sillä samat työkalukeskipisteen koordinaatit on mahdollista saavuttaa eri akselikulmien arvoilla (KUVA 6). ”Status” kuvaa robotin nivelten asemaa suhteessa robotin jalustaa lähinnä olevaan niveleen. ”Turn” taas kertoo nivelkohtaisesti, onko akselikulman astearvo negatiivinen vai positiivinen. (KUKA 2015a, 301-304; Siemens 2016, 51-54.)



KUVA 5. Sama työkalukeskipisteen asema eri nivelkulmilla saavutettuna (KUKA 2015a, 301)

Tarkkuusparametri on arvo, joka määrittelee erosuureen varsinaisen pisteen ja robotin liikeradan välille (KUKA 2015a, 279; Lahden ammattikorkeakoulu 2016, 46). Se kertoo robotille ympyrän halkaisijan, jonka verran robotti voi kiertää määritellyn pisteen. Esimerkiksi 10 mm:n tarkkuusmääritys pisteelle tarkoittaisi sitä, että robotti voi kiertää pisteen maksimissaan 10 mm:n etäisyydeltä. (Lahden ammattikorkeakoulu 2016, 46.) Robottiohjain käyttää tarkkuusparametria reitin optimointiin liittyvässä laskennassa (KUKA 2015a, 279; Lahden ammattikorkeakoulu 2016, 46).

2.4 Robotin ohjelmointi

Robotin ohjelmoinnilla tarkoitetaan robotin liikkeiden suoritusjärjestyksen- ja logiikan määrittelemistä. Ohjelmoinnilla voidaan myös määritellä robotin toimintaa ulkoisten muuttujien perusteella tai vikatilanteen sattuessa. Vanhoja ohjelmointitapoja olivat mm. sähkömekaanisten kytkentöjen ja rajakytkinten käyttö sekä johdattamalla ohjelmointi, jolloin robottiohjelma suoritettiin käsin ja akselien paikkatieto tallennettiin. Nykyään robotteja ohjelmoidaan yleensä yhdistämällä opetus ja tekstipohjainen ohjelmointi. (Lahden ammattikorkeakoulu 2016, 47–51.)

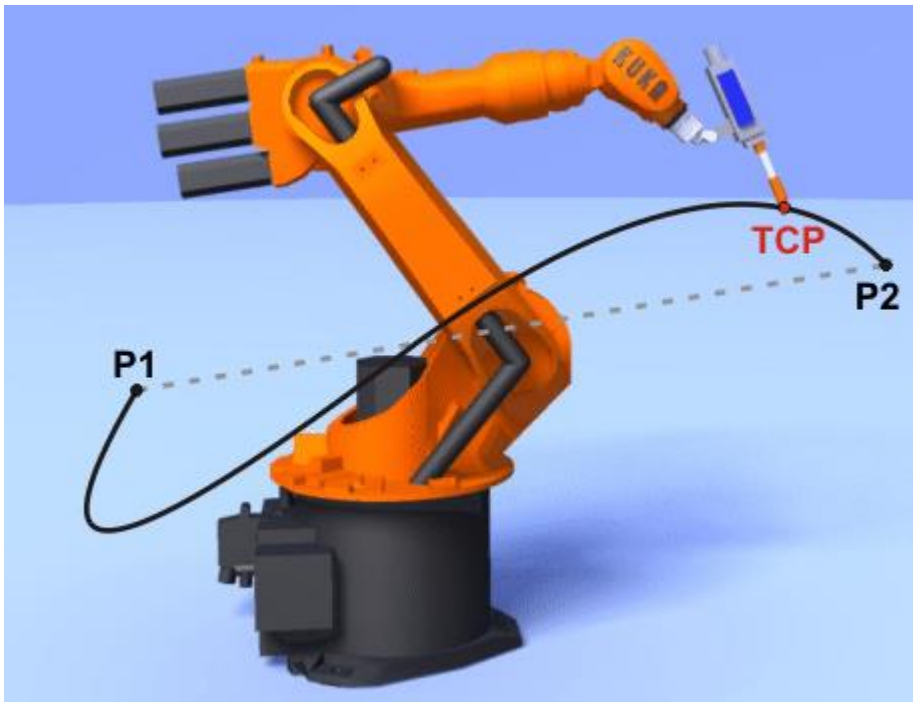
Opettamisella tarkoitetaan robotin ajamista käsin johonkin asemaan ja kyseisen pisteen tallennusta, tekstipohjaisella ohjelmoinnilla loogisten rakenteiden luomista tekstiä kirjoittamalla. Nykyisin robottien ohjelmointikielet ovat yleensä Pascal-kielen kaltaisia, mutta valmistajakohtaisia. Kielistä löytyy liikekäskyjen lisäksi mm. ulkoisiin tuloihin ja lähtöihin liittyviä ominaisuuksia sekä tavanomaiset logiikkarakenteet. (Lahden ammattikorkeakoulu 2016, 47–51.) Koska opettamalla ohjelmointi vaatii tuotannon pysäyttämistä, on suurempia muutoksia varten hyödyllistä käyttää etäohjelmointia, jolloin ohjelma kirjoitetaan ulkoisella tietokoneella (Lahden ammattikorkeakoulu 2016, 48, 51–52).

2.5 Liiketyypit

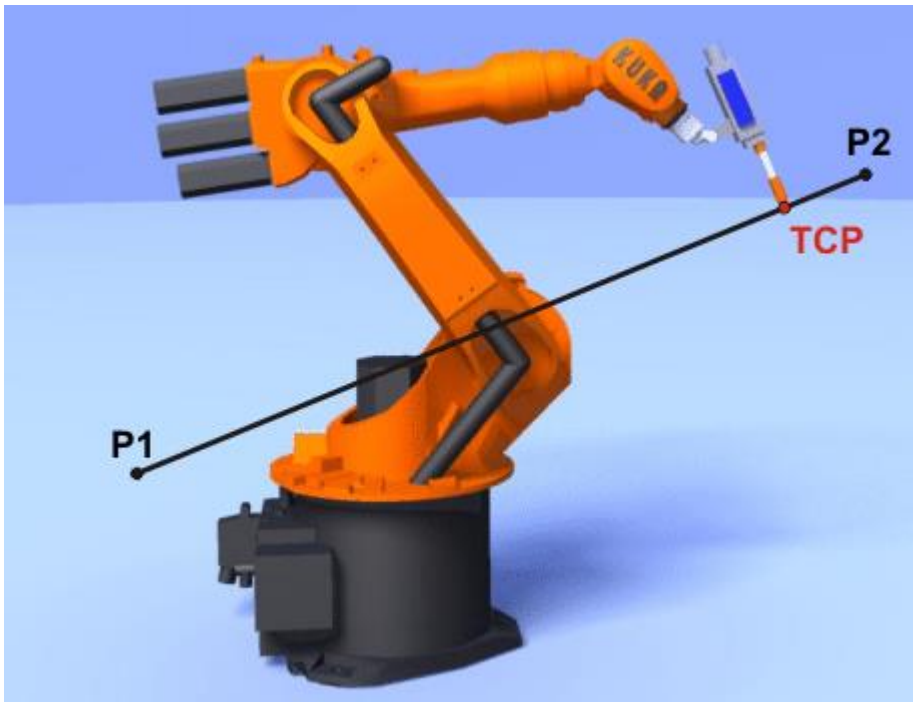
Robotin ohjaus liikekäskyllä siirtää robotin työkalun keskipistettä (TCP) pisteestä toiseen tietyn liiketavan mukaisesti. Liiketapoja ovat asemasta asemaan-, suoraviivainen, sekä ympyräliike

(KUKA 2015a, 277–278; Lahden ammattikorkeakoulu 2016, 50). Lisäksi on mahdollista toteuttaa ohjauspisteiden avulla määritetty käyrä (engl. Spline). Tällöin robotti suorittaa useiden suoraviivaisten ja ympyräliikkeiden muodostaman radan yhtenä liikekäskynä. (KUKA 2015a, 283.)

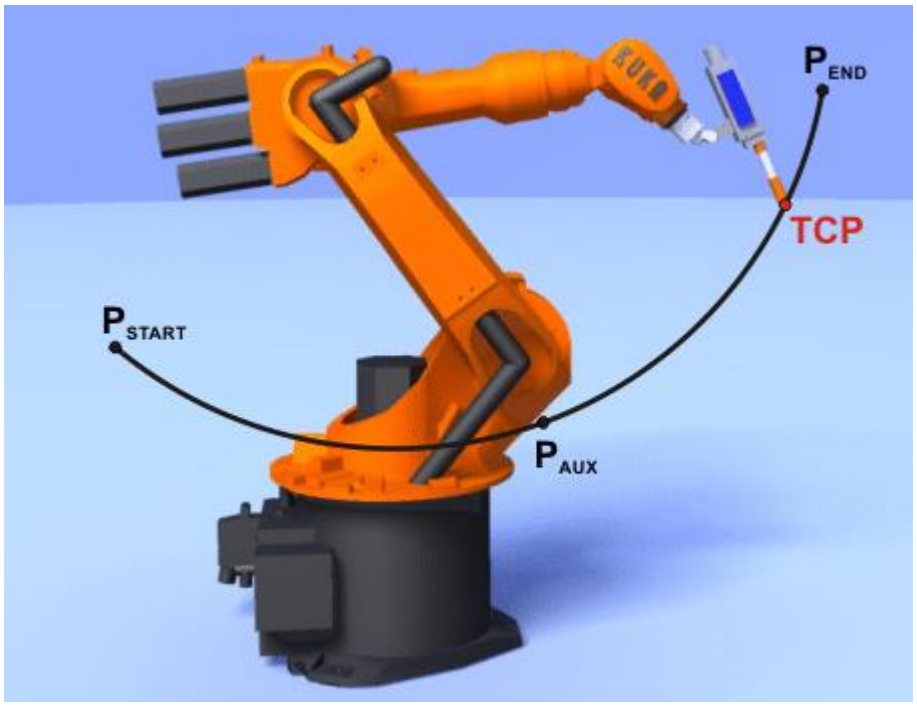
Asemasta asemaan-liike siirtää robotin työkalun keskipisteen nopeinta reittiä seuraavaan pisteeseen (KUVA 7). Jos robotin nivelet ovat kiertyviä, tämä ei vastaa suorinta reittiä. Suoraviivainen liike siirtää työkalun keskipistettä suorassa linjassa pisteeltä pisteelle (KUVA 8). Ympyräliikkeellä robotti siirtää työkalun keskipistettä apupisteen avulla muodostettua kaarta pitkin seuraavalle pisteelle (KUVA 9). (KUKA 2015a, 277–278.)



KUVA 6. Asemasta asemaan-liike (KUKA 2015a, 277)



KUVA 7. Suoraviivainen liike (KUKA 2015a, 278)



KUVA 8. Ympyräliike (KUKA 2015a, 278)

2.6 Turvallisuus

Robotiikan turvallisuutta käsitteleviä standardeja ja säädöksiä ovat mm. ISO 10218-1:2006 sekä konedirektiivi (2006/42 EY). ISO 10218-1:2006-standardissa on määritelty esimerkiksi kolmiasentoisen sallintapainikkeen (”kuolleen miehen kytkin”) käyttö sekä käsiohjauksen alennettu nopeus. (Lahden ammattikorkeakoulu 2016, 80, 87.)

Konedirektiivin perusteella säädetty valtioneuvoston asetus koneiden turvallisuudesta käsittelee koneiden turvallisuusvaatimuksia. Näitä ovat esimerkiksi pysäytyslaitteita koskevat säädökset. Asetuksen mukaan koneessa on oltava ohjauslaite, jolla kone voidaan pysäyttää turvallisesti sekä yksi tai useampi hätäpysäytyslaite vaaratilanteiden varalle. (Valtioneuvoston asetus koneiden turvallisuudesta 12.6.2008/400.)

2.7 KUKA

KUKA on Augsburgissa vuonna 1898 perustettu yritys. Se on yksi maailman johtavista automaatioratkaisuiden toimittajista. Yrityksen nimi oli alun perin lyhenne sähkeitä varten (Keller und Knappich Augsburg). Yritys on käynyt historiansa aikana läpi useita fuusioita sekä nimenmuutoksia. (The history of KUKA.) KUKA on ollut vastuussa monista robotiikan edistysaskelista, kuten maailman ensimmäisestä kuusiakselisesta sähkökäyttöisestä robotista (1973) sekä ensimmäisestä PC- eli mikrotietokonepohjaisesta robottiohjaimesta (1996). (Robot History; The history of KUKA).

2.7.1 KR Agilus-robotti

KR Agilus-sarja on KUKA:n pienikokoisten viisi- ja kuusiakselisten robottien sarja. Niiden toimintasäde on 500 ja 1100 mm:n ja nostokyky kolmen ja kymmenen kg:n välillä. Kaikki Agilus-sarjan robotit käyttävät KUKA:n KR C4-ohjainta ja KUKA smartPAD-käsiohjelmointilaitetta. Esimerkkinä sarjan nimeämiskäytännöstä opinnäytetyön sovellusesimerkissä käytetty malli KR 6 R900 sixx oli kuusiakselinen (sixx) robotti 900mm:n toimintasäteellä (R900) ja 6kg: nostokuormalla (6). (KUKA 2016.)

2.7.2 SmartPAD

KUKA smartPAD on KUKA:n valmistama käsiohjelmointilaite. Se on yhteensopiva kaikkien KR C4-ohjainta käyttävien robottien kanssa. SmartPAD:issa on käsiohjausta varten 6D-hiiri sekä kahdeksan paria käsiohjauspainikkeita, joiden avulla on mahdollista liikuttaa robottia joko akseli kerrallaan tai karteesisessa koordinaatistossa. (The KUKA smartPAD.)

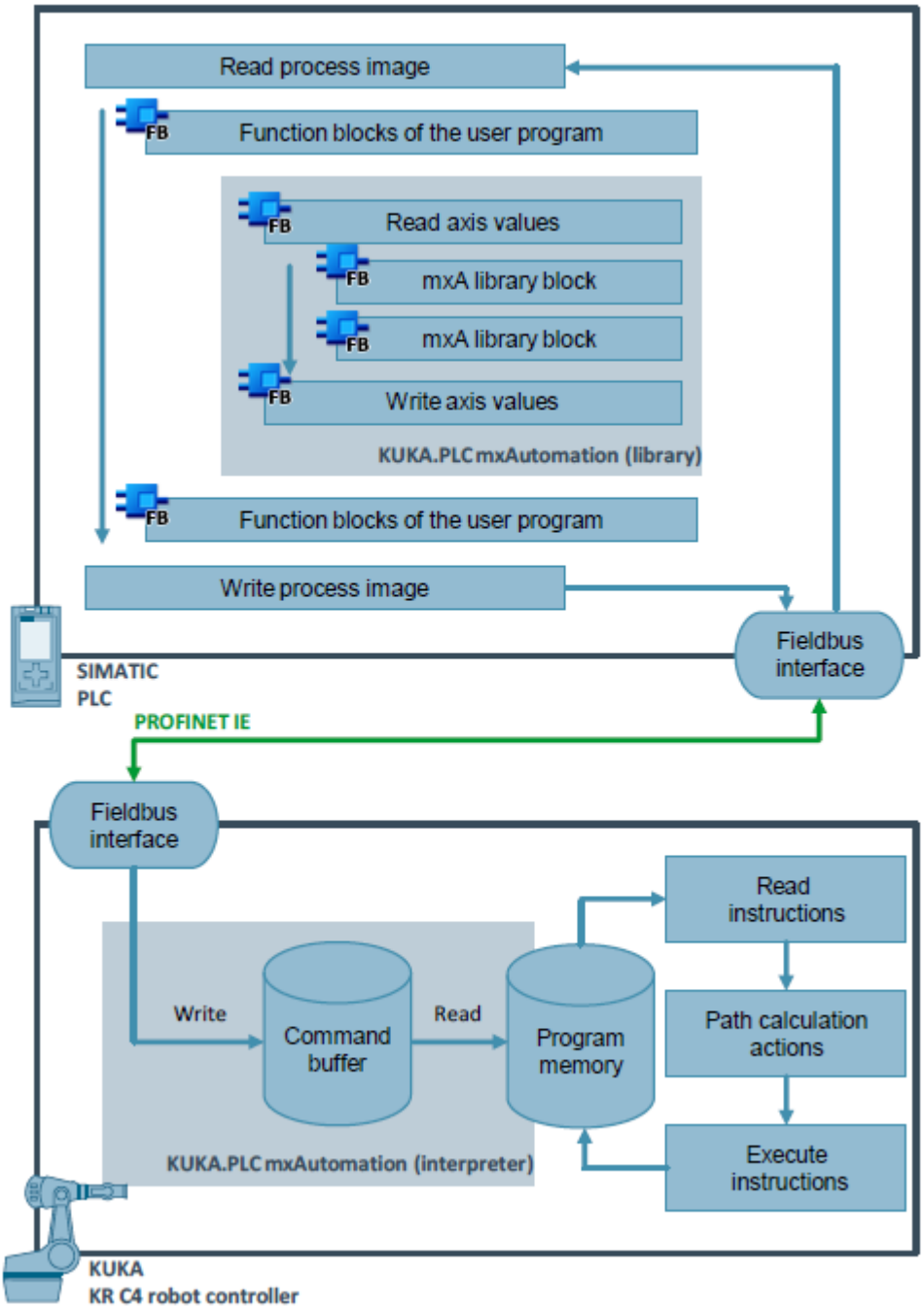
2.7.3 WorkVisual

WorkVisual on KUKA:n tarjoama sovelluspaketti, joka on tarkoitettu KUKA:n robottien asetusten määrittelyyn ja vianhallintaan. Robottiohjaimen asetukset voidaan ladata WorkVisualiin projektina ja WorkVisual-projektissa määritellyt asetukset on mahdollista viedä takaisin robotille. WorkVisualissa voidaan mm. määritellä kenttäväyläasetuksia, turvamäärityksiä sekä ohjelmoida robottia etänä. (KUKA 2014, 11–12.)

2.7.4 mxAutomation

mxAutomation on KUKA:n kehittämä rajapinta, joka mahdollistaa robotin ohjelmoinnin ulkoisella ohjelmoitavalla logiikalla. Rajapinnan tavoitteena on mahdollistaa KUKA:n robottien ohjaus ilman laajaa ymmärrystä robottiohjelmoinnista sekä parantaa tuotannon joustavuutta ja laajennettavuutta. Rajapinta mahdollistaa robotin ohjauksen ulkopuolisesta käyttöliittymästä, jolloin käsiohjelmointilaitetta ei tarvita. mxAutomation on yhteensopiva useiden ulkoisten ohjausjärjestelmien kanssa, mm. AllenBradley, Beckhoff ja Siemens (esim. TIA Portal). (KUKA.PLC mxAutomation.)

KUKA:n tarjoama KUKA.PLC mxAutomation-teknologiaobjekti koostuu kahdesta osasta: logiikkaohjelmaan tuotavasta lohkokirjastosta, joka sisältää robotin ohjaukseen vaaditut lohkot sekä robottiohjaimelle asennettavasta tulkitsijasta, joka ottaa vastaan logiikan käskyjä. Tulkitsija siirtää komennot robotin muistipuskuriin, josta ne voidaan suorittaa. (Siemens 2016, 12.) Rajapinnan toiminta-ajatus on esitetty kuviossa 1.



KUVIO 2. Yleiskuvaus KUKA.PLC mxAutomation funktioista (Siemens 2016, 11)

KUKA:n KUKA.PLC mxAutomation-rajapinnan versio 2.1 täyttää PLC Open-järjestön ”PLCopen Motion Control Part 4”-sertifikaatin kriteerit (KUKA.PLC mxAutomation). Lohkokirjastoihin sisältyy liikekäskyjä toteuttavia MC-toimintolohkoja, joiden toiminta vastaa täysin tai muita kirjastolohkoja enemmän PLCopen-standardin mukaista (KUKA 2015b, 94).

3 MXAUTOMATION TIA PORTALISSA

mxAutomation-rajapinnan käyttöönotto S7-1500 -sarjan logiikalla vaatii toimia sekä Siemensin TIA Portalissa, että robottiohjaimella. TIA Portaliin on tuotava mxAutomation-kirjasto, joka sisältää tarvittavat toiminto- ja datalohkot robottiohjauksen toteuttamiseksi. Lisäksi projektin laitteistoon on lisättävä KUKA:n robottiohjainta kuvaava teknologiaobjekti ja sille on määriteltävä tarvittava alue osoiteavaruudesta. (Siemens 2016, 15–20.) Robottiohjaimelle on asennettava mxAutomation-asetuspaketti sekä ProConOS-virtuaalilogiikka (KUKA 2015b, 15).

3.1 TIA Portal

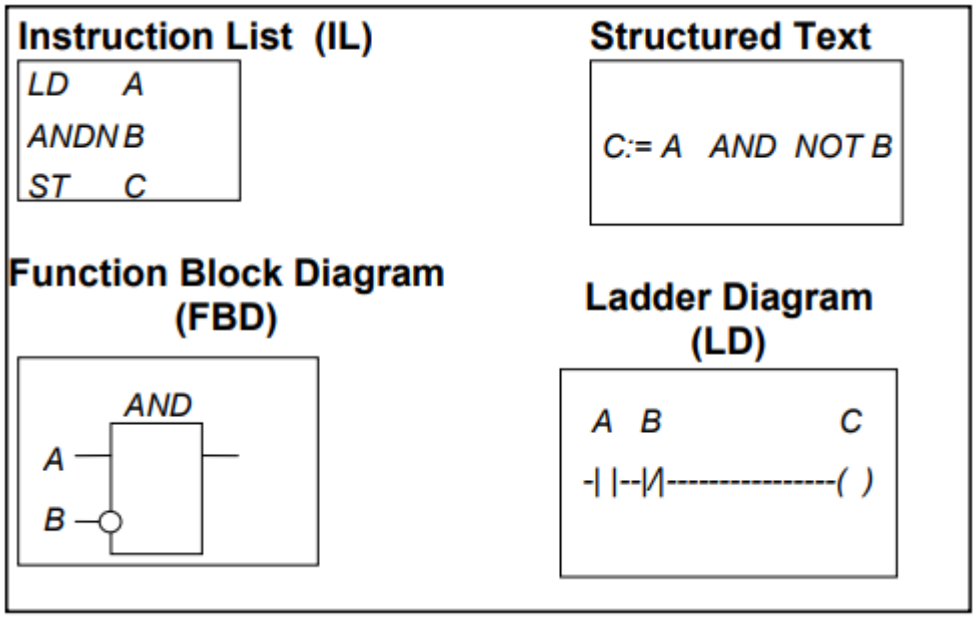
TIA Portal (Totally Integrated Automation Portal) on Siemensin tarjoama ohjelmointityökalu. Siihen sisältyvät STEP 7-logiikkaohjelmointityökalu, SIMATIC HMI-käyttöliittymien suunnitteluun tarkoitettu WinCC sekä taajuusmuuttajien parametrisointiin tarkoitettu StartDrive. TIA Portalin käytön eduiksi voidaan laskea mm. automaatio- ja käyttöliittymäsuunnittelun integrointi yhden ohjelmiston alle sekä yhtenäinen tietorakenne eri editorien välillä. (Siemens 2014a, 34–35.) Kaikki TIA Portal-projektin data tallennetaan projektin alle objekteina, jotka on järjestetty puumaiseen hierarkiaan laitteiden ja alustojen perusteella (Siemens 2014a, 36).

SIMATIC Step 7 -ohjelmisto sisältää työkalut mm. laitteiston konfigurointiin sekä ohjelmointiin IEC-61131 -standardikielillä: käskylista IL (nimellä STL), tikapuukaavio LAD, lohkokaavio FBD, graafinen sekvenssikaavio SFC (nimellä GRAPH) sekä strukturoitu teksti ST (nimellä SCL) (TIA Portal (Step 7)). WinCC:n avulla on mahdollista suunnitella käyttöliittymiä samassa sovellusnäkymässä niin pieniin paneeleihin kuin valvomojärjestelmiinkin. Tiedonjako TIA Portalin Step 7:n kanssa on pyritty tekemään mahdollisimman tehokkaaksi. (TIA Portal (WinCC TIA).)

3.1.1 Standardi-ohjelmointikielet

IEC 61131-3 on ohjelmointikieliä käsittelevä osa logiikkaohjaimiin sekä niiden liitännäislaitteisiin liittyvää IEC 61131 -standardia (TC1 - Standards). Standardissa on määritelty logiikka- eli

PLC-standardikieliksi käskylista (IL), strukturoitu teksti (ST), tikapuukaavio (LAD) sekä lohkokaavio (FBD). Lisäksi ohjelmien tekoon on määritelty graafinen sekvenssityökalu SFC. Ohjelmointikielet ovat keskenään yhteensopivia ja käytetty ohjelmointikieli voidaan valita tapauskohtaisesti. (IEC 61131-3: a standard programming resource.) Kuvassa 10 on esitetty sama looginen toiminto neljällä eri kielellä.



KUVA 9. Standardi-ohjelmointikielet (IEC 61131-3: a standard programming resource)

Standardissa on lisäksi määritelty mm. muuttujat ja tietotyypit sekä ohjelman järjestysyksiköt (POU). Järjestysyksiköitä ovat funktiot eli uudelleenkäytettävät loogiset toiminnot, toimintolohkot (FB), jotka sisältävät toiminnallisuuksien lisäksi dataa sekä edellä mainituista koostuvat ohjelmat. (IEC 61131-3: a standard programming resource.)

3.1.2 Tietotyypit ja muuttujat

Tietotyypit ovat tapa esittää parametreja niin, ettei yhteensopimattomien arvojen käsittelystä synny virheitä ohjelmassa. Tavallisia IEC 61131-3 standardin mukaisia tietotyyppejä ovat mm. Boolen muuttuja (boolean), kokonaisluku (integer) ja reaaliluku (real). Lisäksi tiedon esittämiseen voidaan käyttää näistä johdettuja tietotyyppejä. (IEC 61131-3: a standard programming resource.) Taulukossa 1 on esitetty TIA Portalin Step 7:n sisältämät standardi-tietotyypit.

	Description	S7 - 300/400	S7-1200	S7-1500
Bit data types	• BOOL • BYTE • WORD • DWORD	✓	✓	✓
	• LWORD	×	×	✓
Character type	• CHAR (8 bit)	✓	✓	✓
Numerical data types	• INT (16 bit) • DINT (32 bit) • REAL (32 bit)	✓	✓	✓
	• SINT (8 bit) • USINT (8 bit) • UINT (16 bit) • UDINT (32 bit) • LREAL (64 bit)	×	✓	✓
	• LINT (64 bit) • ULINT (64 bit)	×	×	✓
Time types	• TIME • DATE • TIME_OF_DAY	✓	✓	✓
	• S5TIME	✓	×	✓
	• LTIME • L_TIME_OF_DAY	×	×	✓

TAULUKKO 1. Standardinmukaiset perustietotyypit (Siemens 2014b, 83)

TIA Portal Step 7 tarjoaa lisäksi käytettäväksi mm. strukturoituja ja taulukko-tietotyyppejä sekä käyttäjän määrittelemiä PLC-tietotyypit. Strukturoitu tietotyyppi on muista tietotyypeistä koostuva tietorakenne. PLC-tietotyypit ovat strukturoituja tietotyyppejä, joiden rakenne on määritelty pysyvästi. Taulukko (Array) on tietorakenne, joka koostuu useista saman tietotyypin muuttujista. (Siemens 2014b, 61–63.)

Muuttujien IEC 61131-3 -standardin mukainen käyttötarkoitus on parametrien tallennus ohjelman sisäiseen käsittelyyn. I/O, eli tulo ja lähtö-osoitteiden linkitys muuttujiin helpottaa ohjelmien uudelleenkäyttöä muissa laitteissa. Muuttujat voivat olla staattisia tai globaaleja, eli käytettävissä paikallisesti tai koko ohjelman laajuisesti. (IEC 61131-3: a standard programming

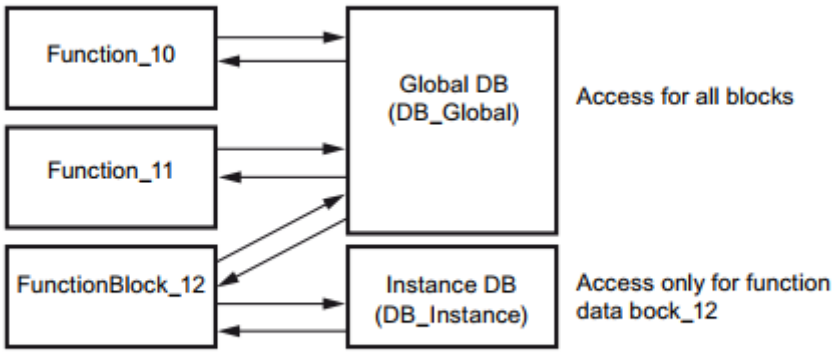
resource.) TIA Portalissa muuttujat esitetään tageina, jotka voivat olla PLC tageja tai globaaleja tai instanssi-datalohkon muuttujia. PLC tagit viittaavat suoraan I/O-osoitteeseen tai muistialueeseen, kun taas datalohkon muuttujat on määritelty datalohkossa. Globaalit muuttujat ovat käytettävissä koko logiikkaohjelmassa, kun taas instanssidatalohkon muuttujia käytetään lähinnä toimintalohkossa, jota varten datalohko on luotu. (Siemens 2014c, 1409–1410.)

3.1.3 Lohkot ja kirjastot TIA Portalissa

TIA Portalin Step 7 -logiikkaohjelmointityökalussa voidaan käyttää neljäntyyppisiä lohkoja: järjestyslohkoja, funktioita, toimintolohkoja ja datalohkoja (Siemens 2014b, 36). Kirjastojen avulla on mahdollista säilöä projektin elementtejä, esim. kokonaisia laitekuvia, ohjelmia, lohkoja, tageja tai tauluja. Kirjastot voidaan jakaa globaaleiksi ja projektikirjastoiksi. Globaalit kirjastot on tarkoitettu useiden projektien käyttöön, projektikirjastot projektin sisäiseen käyttöön. (Siemens 2014b, 67–69.)

Järjestyslohko (OB) toimii käyttäjän luoman ohjelman ja logiikan käyttöjärjestelmän rajapinnassa. Käyttöjärjestelmä voi suorittaa järjestyslohkon esim. käynnistyksen tai ohjelmakierron yhteydessä tai ohjelmakeskeytyksen perusteella. (Siemens 2014c, 1376.) Funktiot (FC) ovat ohjelmalohkoja, joilla ei ole omaa muistia. Funktioissa käytetyt parametrit on näin ollen linkitettävä todellisiin muuttujiin. Funktio suoritetaan, kun se kutsutaan toisessa ohjelmalohkossa. (Siemens 2014c, 1376-77.) Toimintolohkot (FB) ovat ohjelmalohkoja, joilla on oma muisti, johon lohko tallentaa tulo- ja lähtöparametrit (Siemens 2014c, 1376).

Toimintolohkon kutsua ohjelmassa kutsutaan instanssiksi. Jokaisen instanssin data täytyy tallentaa datalohkoon (DB). Instanssidatalohkolle on aina määritelty sitä vastaava toimintolohko. Instanssidatalohkon tietorakenne voidaan määritellä vain sitä vastaavassa toimintolohkossa. Multi-instansseilla viitataan datalohkoon, johon useampi toimintolohko tallentaa datansa. Lohkot voivat olla sisäkkäisiä, esim. toimintolohkossa kutsutaan ajastinta, joka tallentaa parametrinsa sitä kutsuneen lohkon instanssidatalohkoon. (Siemens 2014b, 43–44). Globaali datalohko on lohko, jonka data on kaikkien ohjelman lohkojen käytettävissä (KUVIO 2). Globaalin datalohkon tietorakenne voidaan luoda vapaasti. (Siemens 2014b, 44–45.)



KUVIO 3. Globaalit ja instanssidatalohkot (Siemens 2014c, 1379)

3.1.4 Turvaohjelmat

TIA Portal (V13 ja uudemmat) tukee S7-1500F -sarjan logiikoiden turvaohjelmointia SIMATIC Step 7 Safety asetuspaketin avulla. Tällöin turvaohjelma ajetaan normaalin ohjelman rinnalla. Turvaohjelmien tekoon voidaan käyttää LAD- ja FBD-kieliä. Turvaohjelman lohkot erottaa ohjelmaeditorissa keltaisesta merkkiväristä. (Siemens 2014b, 86–87.) Tiedonsiirrossa hyödynnetään PROFISafe:a, joka mahdollistaa turvatiedon siirron samassa väylässä, kuin tavallisen ohjelman tieto (PROFISafe-hybridiväylätekniikka). Jokaiselle turva-I/O -laitteelle on määriteltävä PROFISafe-osoite, joka on mahdollista asettaa joko laitteella tai TIA Portalissa (Siemens 2014b, 91).

3.1.5 HMI

HMI:llä (lyhenne engl. Human-Machine Interface) eli ihminen-kone-rajapinnalla viitataan käyttöliittymään, jonka kautta käyttäjä kommunikoi automaatiolaitteen kanssa (Monahan, Hoffman, Ferchak & Bishop 2016). Tärkeitä hyvän käyttöliittymäsuunnittelun ohjeita ovat mm. vain ohjatun prosessin kannalta oleellisen informaation esittäminen sekä hillitty värien käyttö (kirkkaat värit varattu hälytyksille ja muille poikkeustiloilla, normaalitilassa harmaata ja pastellivärejä) (Gruhn 2011; McDaniel 2016; Monahan ym. 2016). Myös hyvin jaoteltu ja yhdenmukainen käyttöliittymä parantaa käytettävyyttä (Gruhn 2011; Monahan ym. 2016). Lisäksi esimerkiksi painikkeiden yksiselitteiset muuttumattomat tekstit helpottavat käyttöä (Monahan ym. 2016).

3.2 Käyttöönotto robottiohjaimella

Robottohjaimella mxAutomation-rajapinnan käyttöönotto tehdään WorkVisualin avulla. Ensin mxAutomation asetuspaketti asennetaan WorkVisualiin ja nykyinen projekti ladataan robottiohjaimelta. Ladattuun projektiin lisätään sen jälkeen ProConOS sekä mxAutomation-asetuspaketti. (KUKA 2015b, 15.) ProConOS on sulautettuihin tai PC-pohjaisiin ohjausjärjestelmiin tarkoitettu virtuaalilogiikka (KW Software, Inc). Seuraavaksi lisätään projektiin käytettyä kenttäväylää vastaava katalogielementti. Lopuksi muokattu projekti viedään takaisin robottiohjaimelle. (KUKA 2015b, 15.)

3.3 Robottiohjain TIA Portalissa

Jotta logiikka saa yhteyden robottiin, on robottiohjain esiteltävä TIA Portalissa PROFINET IO -laitteena. Projektiin on asennettava robottiohjainta vastaava GSDML-laitekuvaustiedosto. Tämän jälkeen se voidaan lisätä projektin laitteistokokoonpanoon raahaamalla. Robottiohjain vaatii 256 tuloa ja lähtöä. (Siemens 2016, 15–16.) Ylimääräisten toimintojen, kuten KUKA SafeOperationin käyttö saattaa vaatia lisätulojen ja -lähtöjen määrityksen (Siemens 2016, 16, 43–44). Robotti yhdistetään projektin muuhun laitteistoon PROFINET:in kautta määrittelemällä sille IP-osoite eli yksilöivä numeerinen laitetunnus laitteistokokoonpanossa (Siemens 2016, 17).

3.4 mxAutomation-lohkokirjasto

Jotta KUKA:n robottia on mahdollista ohjata Siemensin S7-1500 -sarjan logiikalla on TIA Portaliin tuotava KUKA.PLC mxAutomation -lohkokirjasto. Lohkokirjastoon on koottu robotin ohjaukseen tarvittavia lohkoja. Logiikkaohjelmassa kutsutut lohkot siirretään robotin muistipuskuriin, josta tulkitsija suorittaa lohkoa vastaavan funktion. (Siemens 2016, 11–12.) Kun kirjasto on avattu TIA Portalissa globaalina kirjastona, voidaan sen sisältämät lohkot ja datatyypit kopioida omaan projektihierarkiaan raahaamalla (Siemens 2016, 18–19).

3.4.1 Kirjaston sisältö

Lohkokirjasto sisältää toiminto- ja datalohkoja sekä funktioita, jotka voidaan jakaa seuraaviin ryhmiin: hallinnolliset funktiot, liikeohjelmointi, ohjelmansuorituksen ohjaus, keskeytys-ohjelmointi, polkukohtaiset kytkintoiminnot, diagnostiikkafunktiot sekä erikoisfunktiot. (Siemens 2016, 7–8.)

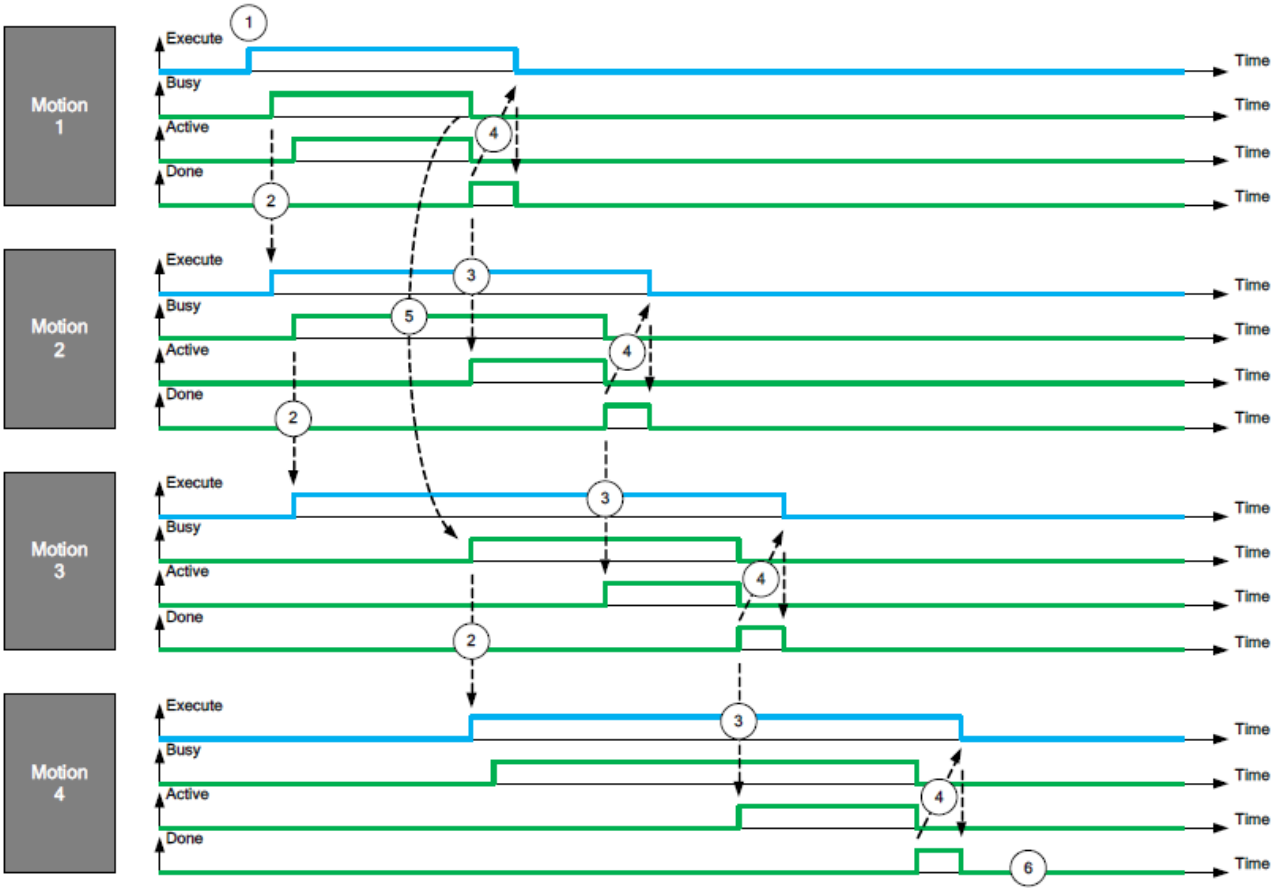
Hallinnolliset funktiot sisältävät robotin käynnistykseen, tilan valvontaan, työkalu- ja alustatietojen, oheislaitteiden ja rajakytkinten tilanlukuun ja -kirjoittamiseen. Liikeohjelmoinnin avulla voidaan ohjata robotin liikkeitä. Ohjelmansuorituksen ohjauksella kyetään keskeyttämään ja uudelleen käynnistämään tai perumaan ohjelman suoritus. (Siemens 2016, 7–8.) Keskeytysohjelmoinnilla viitataan keskeytyksien käyttöön ohjelmassa, jolloin ohjelman suoritus jatkuu keskeytysvektorin osoittamasta paikasta (Järvinen & Mikkonen 2010, 34). Polkukohtaisilla kytkintoiminnoilla tarkoitetaan liikeratojen tai etäisyyksien perusteella aktivoituvia kytkintoimintoja. Diagnostiikkafunktioihin kuuluvat virheviestien, diagnostiikkatietojen ja robotin tilan luku. Erikoisfunktioita ovat esimerkiksi jarrutuskoe sekä järjestelmämuuttujien luku ja kirjoitus. (Siemens 2016, 8.)

3.4.2 Yleiset tulot ja lähdöt

”ExecuteCmd”-tulon asettaminen siirtää lohkon robotin suorituspuskuriin, jos siellä on tilaa. Tulon nollaaminen johtaa lohkon poistamiseen puskurista, mikäli se ei ole jo suorituksessa. Tällöin myös kaikki lohkon lähdöt nollautuvat. ”Busy”-lähtö kertoo, että lohko on siirretty puskuriin tai että siirto on käynnissä. ”Active”-lähtö kertoo lohkon olevan suorituksessa ”Done”-lähtö kertoo suorituksen olevan valmis. ”Error”-lähtö kertoo virheestä lohkon suorituksessa robotilla, ”ErrorID”-lähtö sisältää virhettä kuvaavan vikatunnuksen. ”Aborted”-lähtö kertoo suorituksen perumisesta. (KUKA 2015b, 20–21.)

Liikekäskyjen suoritus on mahdollista ketjuttaa. Tämä tapahtuu asettamalla seuraavan lohkon ”ExecuteCmd”-tulo edellisen lohkon ”Busy”-lähdön nousevalla reunalla (KUVIO 3). (Siemens

2016, 13.) Lohkokirjaston ”MC”-alkuiset liikekäskylohkot poikkeavat toimintalogiikaltaan muista kirjastolohkoista. (KUKA 2015b, 22.)



KUVIO 4. Signaalisekvenssi toimintolohkojen ketjutettuun suoritukseen (Siemens 2016, 14)

3.5 mxAutomation-rajapinnan käyttö logiikkaohjelmassa

Hyödynnettäessä mxAutomation-kirjastolohkoja logiikkaohjelmassa on huomioitava muutamia säännönmukaisuuksia ohjelmarakenteessa. Ensimmäisen suoritettavan kirjastolohkon on aina oltava robotilta dataa rajapinnan sisäiseen käsittelyyn lukeva ”KRC_ReadAxisGroup” ja viimeisen lohkon on oltava ”MxADBRobots”-datalohkon datan robotille kirjoittava ”KRC_WriteAxisGroup”. (Siemens 2016, 20–21.) Nämä lohkot tulee kutsua vain kerran (paitsi jos käytössä useita robotteja) (KUKA 2015b, 31).

Robotti on käynnistettävä robottiohjaimen ulkoisen automaattiohjauksen tilabittejä hallinnoivan ”KRC_AutomaticExternal”-kirjastolohkon avulla ja robottiohjaimen on oltava ”EXT”-tilassa (ulkoinen ohjaus). KUKA:n dokumentaation mukaan ”KRC_AutomaticExternal”-lohko voidaan aktivoida automaattisesti suorittamalla ”KRC_AutoStart”-kirjastolohko, kun taas Siemens tarjoaa sovellusesimerkissään oman käynnistyssekvenssin, jonka toiminta on kuvattu sovellusesimerkin dokumentaatiossa. (KUKA 2015b, 30–31; Siemens 2016, 21–23.) Tiedonsiirto rajapinnan yli onnistuu vasta, kun rajapinta on alustettu suorittamalla ”KRC_Initialize”-kirjastolohko (Siemens 2016, 23). Kirjastolohko vertailee tulkitsijan ja lohkokirjaston versionumeroita ja ilmoittaa niiden yhteensopivuudesta ”Done”-bittinsä avulla. (KUKA 2015b, 31; Siemens 2016, 23).

4 CASE: SOVELLUSESIMERKKI APEX-MESSUILLE

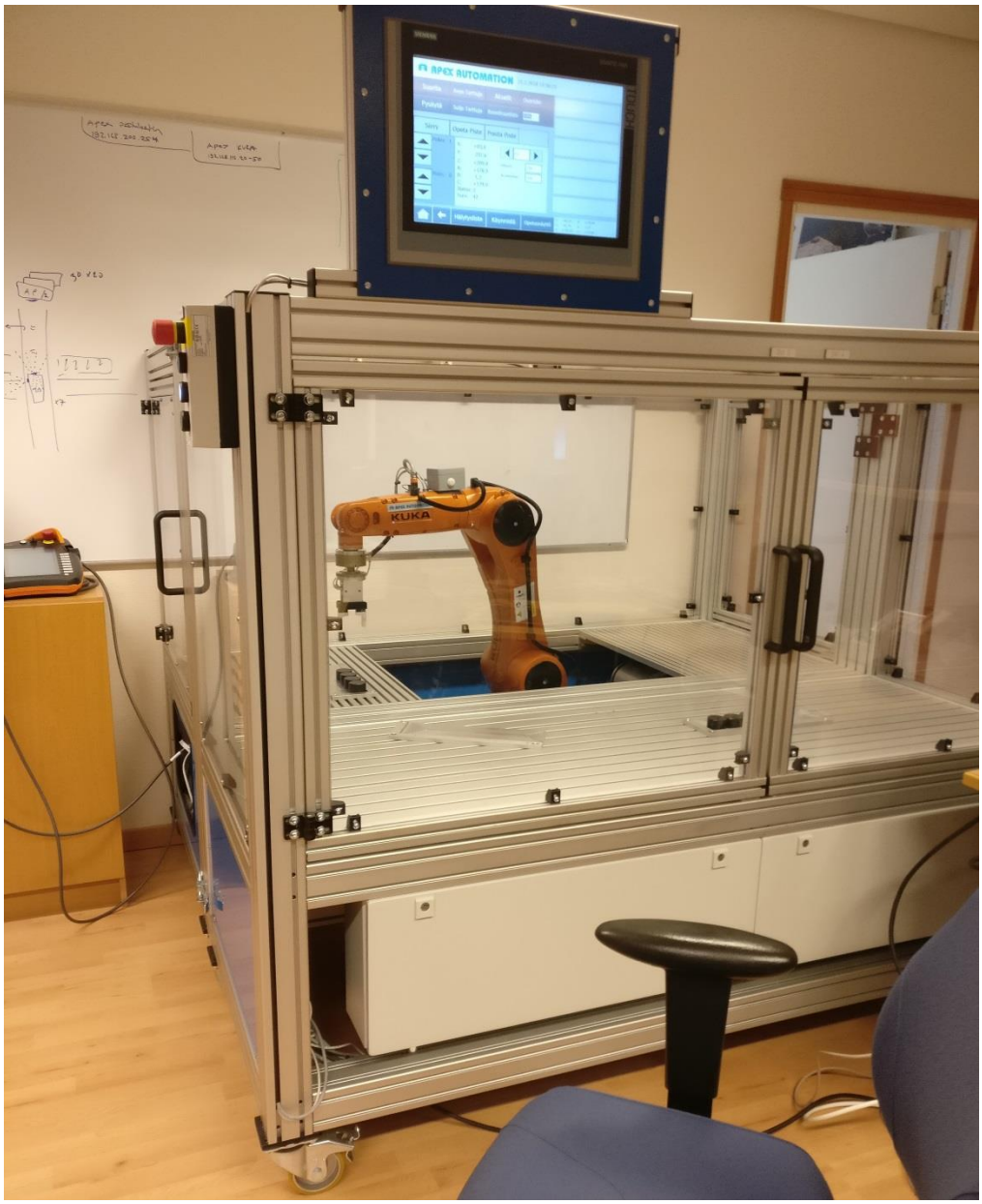
Opinnäytetyön toimeksiantoon kuului rajapintaan tutustumisen lisäksi sovellusesimerkin toteuttaminen Apex Automationin järjestämää teknologiapäivää varten. Kyseessä oli asiakkaille ja yhteistyökumppaneille suunnattu kutsutapahtuma, joka koostui luennoista ja esityksistä. Tavoitteena oli saada toimiva sovellusesimerkki valmiiksi tätä tapahtumaa varten, jotta sitä voitaisiin käyttää rajapinnan toiminta-ajatuksen esittelyyn. Sovellusesimerkkiä varten tuli toteuttaa rajapintaa hyödyntävä logiikkaohjelma, sekä käyttöliittymä HMI-paneelille. Käyttöliittymästä käsin oli tarkoitus kyetä ohjaamaan robottia, muuttamaan sen asetuksia sekä lukemaan robotin ja robottisolun muun laitteiston vikatietoja.

4.1 Apex Automation

Työn toimeksiantaja Apex Automation on vuonna 1993 perustettu automaatio- ja sähkösuunnittelua sekä asiantuntijapalveluita tarjoava kokkolalainen osakeyhtiö (Apex Automation 2017). Apex Automation on alansa johtava toimija alueella. Apex Automationin erityisalaa ovat kone- ja sähköturvallisuus. Yritys työllistää yli 50 henkilöä. Apex Automationin asiakkaat ovat mm. teollisuuden ja energia-alan yrityksiä sekä laite- ja järjestelmäntoimittajia. Yrityksen liikevaihto vuonna 2017 oli 5,2 miljoonaa euroa.

4.2 Lähtötilanne ja laitteistomuutokset

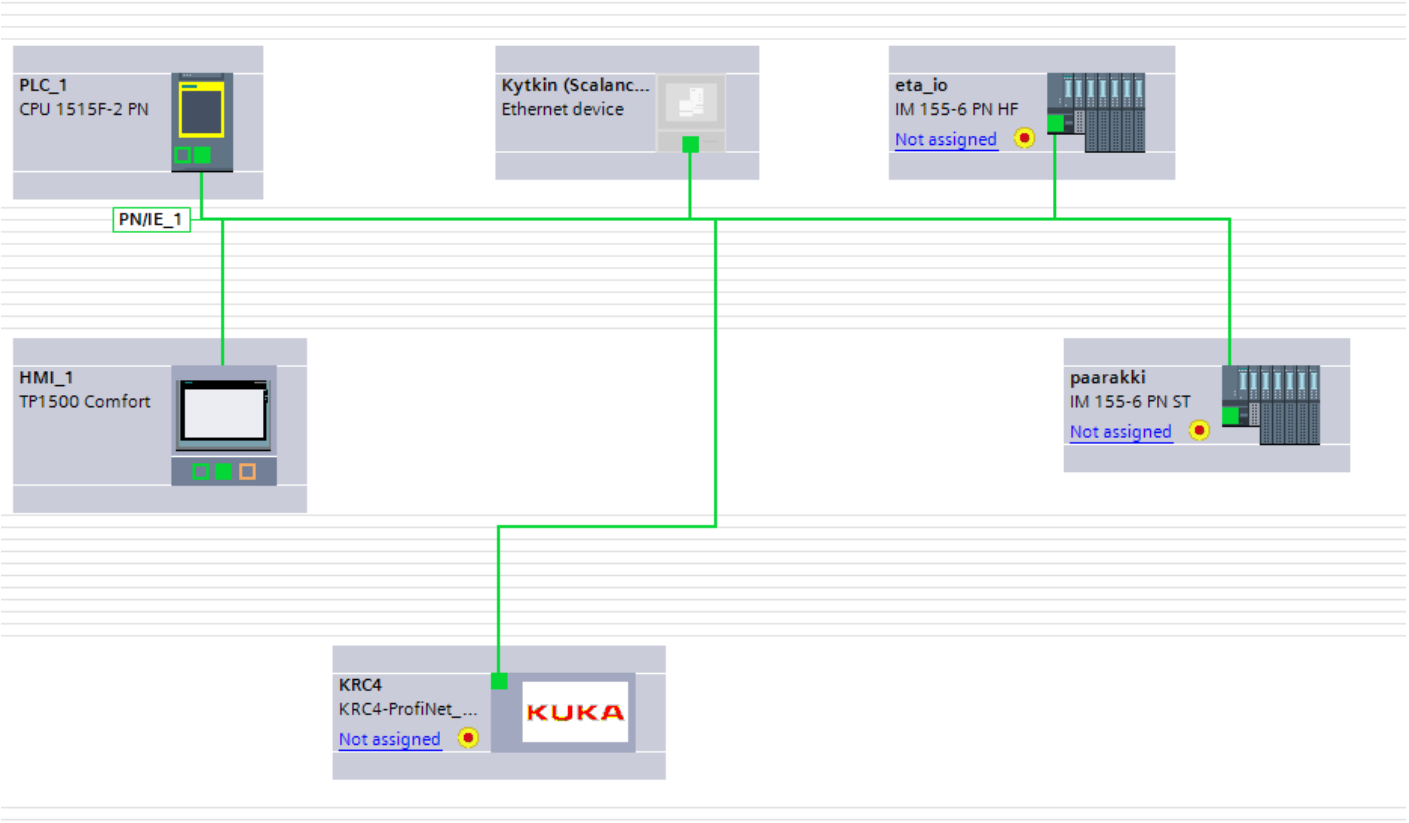
Robotin ympärille oli rakennettu kahtia jaettava ja liikuteltava robottisolu. Robottisolu koostui itse robotin lisäksi suojahäkistä, t-ura -pöydästä sekä turvalaitteistosta (KUVA 11). Soluun oli toteutettu eri sarjan Siemensin logiikalla turvaratkaisu. Myös tarvittavat I/O-kytkennät oli tehty valmiiksi, sillä solu oli ollut aiemmin toiminnassa ilman rajapintaa.



KUVA 10. Robottisolu laitteistomuutosten jälkeen

Robottisolun turvaohjelma sekä ulkoinen käynnistys oli toteutettu Siemensin ET 200SP -logiikalla. Sen tilalle vaihdettiin S7-1515F-2 CPU eli keskusyksikkö sekä IM 155-6 PN ST -I/O-moduuli, sillä uuteen logiikkaan ei voitu suoraan kytkeä ET 200SP:n I/O-kortteja. Solun I/O:ta oli hajautettu lisäksi IM 155-6 PN HF -I/O-moduulille. Itse I/O-kortteja ei tarvinnut vaihtaa, mikä vähensi uudelleenkytkentöjen tarvetta. Robottisoluun liitettiin myös Siemensin TP1500 Comfort -paneeli käyttöliittymän toteuttamiseksi. Uudet laitteet liitettiin järjestelmän PROFINET-väy-

lään. Väylälaitteet oli yhdistetty Siemensin Scalance XB-500-kytkimellä. Muutokset järjestelmän sähkökaapelointiin merkittiin sähkökuviin puhtaaksi piirtoa varten. Järjestelmän PROFINET-laitteet käyvät ilmi kuvasta 12.



KUVA 11. Järjestelmän PROFINET-laitteet

4.3 Ohjelmistosuunnittelu

Logiikkaohjelman suunnittelussa lähdettiin liikkeelle vain tarpeellisten ominaisuuksien sisällyttämisestä. Verrattuna esimerkiksi Siemensin tarjoamaan esimerkkisovellukseen, sovellusesimerkistä tehtiin hyvin pelkistetty. Lohkojen ja muuttujien nimeämiskäytännöt pohjautuivat Siemensin S7-1200/S7-1500 -ohjelmointityylioppaaseen, sillä looginen ja toistuva tyyli helpotti projektinhallintaa ohjelmointityön osalta. Käytetyt kirjastolohkot erottaa niiden nimen alkuosasta ”KRC” (PLCopen-standardin mukaisten lohkojen osalta ”MC”).

Ohjelmaa varten TIA Portal -projektiin luotiin aluksi funktio ”mxAutomation_MAIN, joka ajettiin projektin ”Main”-järjestyslohkossa. Tämän funktion sisällä ajettiin robotin ja logiikan välistä tiedonsiirtoa hallinnoivat kirjastolohkot ”KRC_ReadAxisGroup” ja ”KRC_WriteAxisGroup” sekä näiden välissä ”MxAutomationDemoApplication”-lohko, jonka sisälle koko demosovellus rakennettiin. mxAutomation_MAIN-funktio suoritettiin vain, mikäli robotille oli sallittu ulkoinen ohjaus, eli se oli ”EXT”-tilassa. Muulloin robotin liikkeen salliva ”MOVE_ENABLE”-bitti pakko ohjattiin päälle suoraan I/O-osoitteen avulla, jotta robottia voitiin haluttaessa ohjata käsiohjelmointilaitteella.

Demosovellukseen tarvittavia muuttujia varten luotiin projektiin sekä omia strukturoituja tietotyyppejä, että omia globaaleja datalohkoja. Sovellusesimerkin sisäiset muuttujat tallennettiin ”MxAutomationDemoApplication”-toimintolohkon instanssi-datalohkoon. Paneelilta tulevat ohjaukset sekä suurin osa paneelilla esitetyistä muuttujista tallennettiin ”InstHMI_IO”-lohkoon tai niille luotiin lohkoon välimuuttuja. Paneelilla esitetty liikepiste, sovellusesimerkkiä varten tallennetut poimintapisteet sekä tallennetun liikepisteen tietojen nollauksessa käytetty oletusarvoinen struktuurimuuttuja ja niiden ”UDT_POINTDATA”-muotoiset pistekohtaista dataa sisältävät struktuurimuuttujat tallennettiin ”InstPoints”-lohkoon.

Opetetut liikepisteet ja niiden oheistiedot tallennettiin ”InstPrograms”-lohkoon. Tähän lohkoon luotiin lisäksi osoittimet valitulle ohjelmalle ja liikepisteelle, sekä ohjelmakohtaiset taulukot, joihin tallennettiin pistekohtaisesti tieto siitä, onko indeksinumeroltaan vastaavaan liikepisteeseen tallennettu asematieto. Näin estettiin se, ettei robotti ohjelmaa suorittaessaan pyri ajamaan ”tyhjään” positioon (yleiskoordinaatiston nollapiste sijaitsee robotin jalustan keskipisteessä, joten sitä kohti ajaminen saattaa johtaa robotin tarttujan törmäämisen robotin runkoon ennen akselikulmien ohjelmallisten maksimiarvojen saavuttamista).

4.3.1 Rajapinnan alustus

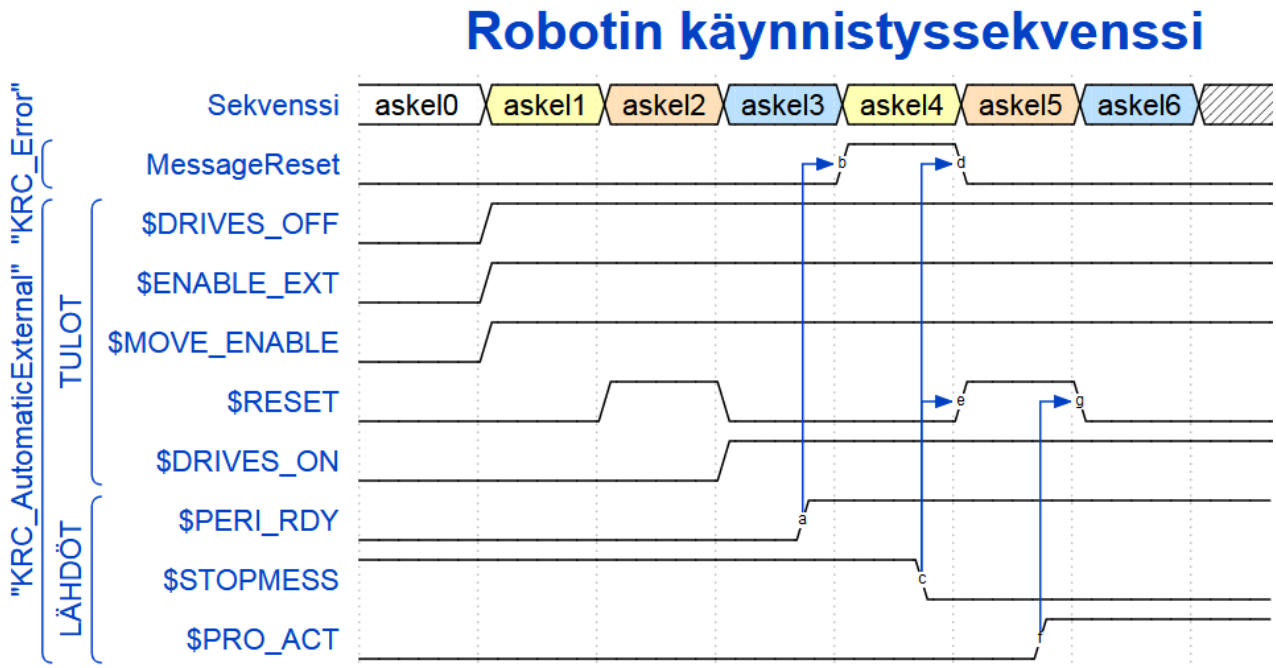
Sovelluksen ohjelmakierron aluksi oli ajettava alustuslohko ”KRC_Initialize”. Tämä lohko vertailee logiikan ja robottiohjaimen sovellusversioita ja se on ajettava jokaisella käynnistyskerralla ennen muiden funktioiden kutsumista (KUKA 2015b, 33–34; Siemens 2016, 23–24). Alustuksen onnistumisesta, eli kirjastoversioiden yhteensopivuudesta ilmoittavan ”Done”-bitin tieto vietiin ohjelmassa sisäiseen välimuuttujaan. Välimuuttujan ”epätosi”-arvo johti toimintolohkon

suorituksen päättymiseen ”RET”-käskyllä, jolloin alustuslohkon onnistunut suoritus oli ennakkoehto ohjelman suorituksen etenemiselle.

Rajapinnan alustuksen onnistumiseksi on rajapinnan asennus robottiohjaimelle suoritettava tarkasti ohjeita noudattaen. Rajapinnan alustuksen epäonnistuminen estää muiden kirjastolohkojen toiminnan. Myös vianmääritys on tällöin hyvin hankalaa. Koska asetuspakettia ei ollut aluksi asennettu oikein robottiohjaimelle, kuluivat ensimmäiset päivät sovellusesimerkin suunnittelusta ja testauksesta tämän ongelman ratkomiseen. Ongelma ratkesi poistamalla ja uudelleen asentamalla rajapinta robottiohjaimelle ohjeiden mukaisesti.

4.3.2 Robotin käynnistys

Robotin käynnistäminen rajapinnan avulla oli toteutettava ohjaamalla ulkoisen ohjauksen tilabittejä hallinnoivan ”KRC_AutomaticExternal”-lohkon tuloja kuviossa 4 esitetyn sekvenssin mukaisesti. Tämä toteutettiin ohjelmoimalla sekvenssi, joka ohjaa sisääntulo-bittejä päälle tai pois ulostulo-bittien tilojen perusteella. Kuviossa on esitetty käytetyt tilabitit sekä merkitty nuolilla reuna-aktiiviset tilan muutokset.



KUVIO 5. Robotin käynnistyssekvenssi

Robotin käynnistys siihen tarkoitetun ”KRC_AutoStart”-lohkon avulla ei onnistunut. Myöskään Siemensin sovellusesimerkissä kuvatun sekvenssin käyttö ei tuottanut tulosta. Toimivaan lopputulokseen päädyttiin ”yritys ja erehdys” -periaatteella: etsimällä oikea kombinaatio KUKA:n ja Siemensin dokumentaatiota hyödyntäen saatiin robotti lopulta käynnistymään. Tuntematto-maksi jääneestä syystä ”KRC_AutomaticExternal”-lohkon ”CONF_MESS”-bitin tilan muuttaminen ei tuottanut odotettua tulosta, joten virheiden kuittaus oli suoritettava ajamalla sekvenssin yhteydessä ”KRC_Error”-kirjastolohko ja korvaamalla ”CONF_MESS”-tulon ohjaus tämän lohkon ”MessageReset”-tulolla. ”KRC_Error”-lohkon ulostuloja hyödynnettiin lisäksi robotin tilan ilmaisemiseen paneelilla, sekä käynnistyssekvenssin alustuksen yhteydessä.

4.3.3 Robotin aseman lukeminen ja ohitusnopeuden määritys

Robotin aseman lukemiseen tarkoitetut kirjastolohkot ja robotin ohitusnopeuden määrittelyyn tarkoitettu toimintolohko koottiin lohkon ”RobotConfig” alle. Robotin asema luettiin sekä akselikohtaisesti ”KRC_ReadActualAxisPos”-kirjastolohkolla, että aktiivisessa koordinaatistossa ”KRC_ReadActualPosition”-kirjastolohkolla. Lisäksi ”RobotConfig”-lohkon sisällä ajettiin ”ChangeOverride”-toimintolohko. Tämä lohko pakotti näytöltä muokattavissa olevan ”override”-ohitusnopeusmuuttujan arvon 1–100% välille ja suoritti tämän jälkeen kirjasto-lohkon ”KRC_SetOverride”, joka tallensi sille syötetyn arvon robotin todelliseksi ohitusnopeudeksi. Mikäli ohitusnopeus on määritelty nollaksi, ei liikekäskyjen suorituksen estävä ”Collective Fault”-signaali välttämättä nollaudu virheen kuittauksen tai uudelleenkäynnistyksen jälkeen.

4.3.4 Ohjelmointinäytön taustafunktiot

Osa käyttöliittymän toteutukseen vaadituista toiminnoista toteutettiin logiikkaohjelmassa, sillä niiden toteuttaminen WinCC:n avulla olisi ollut huomattavasti vaikeampaa tai mahdotonta. Ohjelmointinäytön osalta nämä taustafunktiot koottiin ”ProgrammingScreen”-toimintolohkon sisälle. Näitä olivat ohjelman ja liikepisteen valintaan käytettyjen nuolinäppäimien ohjauslohko, käsiajon tilan muuttamiseen tarkoitettu lohko, lohkot liiketyypin, tarttujan ohjauskomennon ja koordinaatiston pistekohtaiseen valintaan sekä lohkot joilla voitiin muuttaa pistekohtaisesti liikepisteen koordinaatteja, tarkkuusparametria, kiihtyvyyttä sekä nopeutta ja pisteen nimeä. Lisäksi ”ProgrammingScreen”-lohkon sisällä ajettiin liikepisteen opettamiseen ja poistamiseen

tarkoitetut lohkot sekä lohko, joka haki ohjelma- ja pisteindeksien osoittaman liikepisteen tiedot näytölle.

4.3.5 Liikkeiden ohjaus

Liikkeiden ohjaus toteutettiin ”MotionControl”-lohkon sisällä. Lohkon sisällä kutsuttiin kirjastolohko ”KRC_Jog” käsiajon ohjaukseen, lohko ”GripperControlMan” tarttujan käsiohjaukseen painonapeilla, lohko ”CopyCurrentProgram”, joka siirsi näytöltä valitun ohjelman datan välimuuttujaan suoritusta varten, lohko ”DemoCalculatePoints”, joka laski demo-ohjelmaa varten tarvitut liikepisteet kolmen vakiopisteen perusteella sekä lohko ”MotionCmds”, jonka sisällä suoritettiin varsinaiset liikekäskyt. Liikekäskyjen toteutusta varten lohkokirjastossa on ”KRC_Move”-alkuisia sekä PLCopen -standardin mukaisia ”MC_Move”-alkuisia toimintolohkoja.

”MotionCmds”-lohko sisälsi ohjelman suorituksen ohjauksen sekä sekvenssirakenteen, jonka avulla oli mahdollista ajaa joko tai yksittäinen robotin valittuun liikepisteeseen siirtävä liikekäsky tai ohjelma, joka suorittaa liikekäskyjä ketjutetusti. Ohjelmia oli viisi (0-4) käyttäjän ohjelmoitavissa olevaa sekä kaksi valmiiksi tallennettua (pino ja pyramidi). Valmiiksi tallennetut demo-ohjelmat kasasivat robottisolun sisällä olleista kiekoista joko pinon tai pyramidin käyttäjän valinnan mukaan. Mikäli käyttäjä vaihtoi rakennelmaa kesken ohjelman suorituksen, kasasi robotti seuraavalla ohjelmakierrolla uuden rakennelman.

Sekvenssiin oli toteutettu ristiin lukitukset yksittäiselle liikkeelle sekä käyttäjän tallentaman ja valmiiksi tehdyn ohjelman suoritukselle. Näin pyrittiin toteuttamaan liikkeen ohjaus niin, ettei liikekäskyjä tai muita robotin liikkeen ohjaukseen liittyviä lohkoja kutsuttaisi kuin yhdestä paikasta. Liikkeenohjaussekvenssiä varten luotiin myös toimintolohko ”GripperControlAuto”, joka toteutti tarttujan käsiohjausta vastaavan toiminnon ylemmän tason lohkon kutsumana ilman painonappeja.

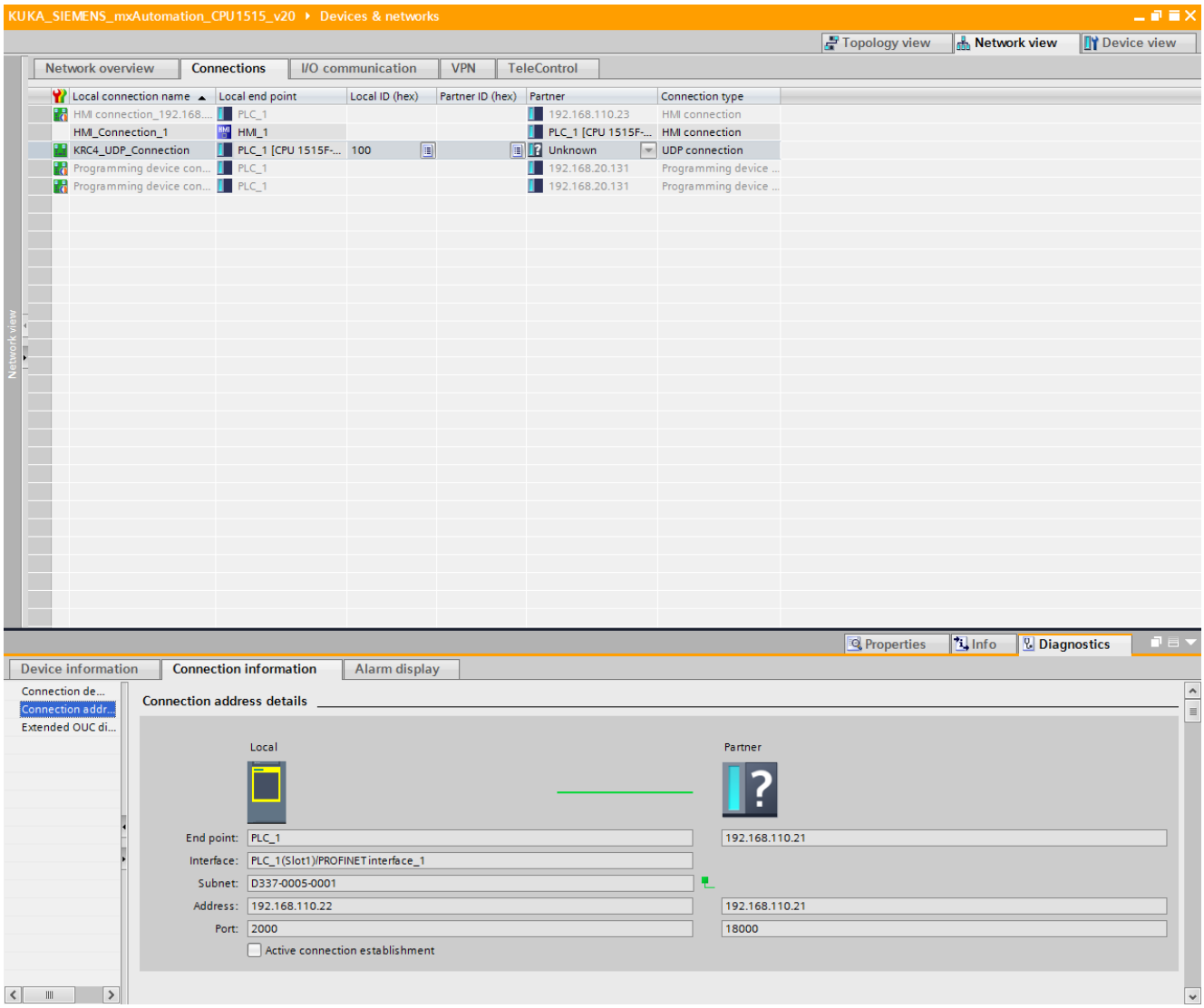
”MotionCmds”-lohkon sisällä toteutettiin kolme eri pysäytystyyppiä robotille. Ohjelman suorituksen lopetus poisti kaikki liikekäskyt (sekä tarttujan ohjauksen) suorituksesta välittömästi ja ajoi liikepuskurin tyhjentävän kirjastolohkon ”KRC_Abort”. Pysäytyspyynnön saatuaan robotti

suoritti nykyisen ohjelmakierron loppuun, jonka jälkeen se siirtyi ohjelman lähtöasemaan asemasta-asemaan -liikkeellä (ensimmäiseen liikepisteeseen). Hätäpysäytyksen jälkeen robotti jatkoi ohjelman suoritusta ”aloita”-painikkeella (tai ovikotelon painonapilla).

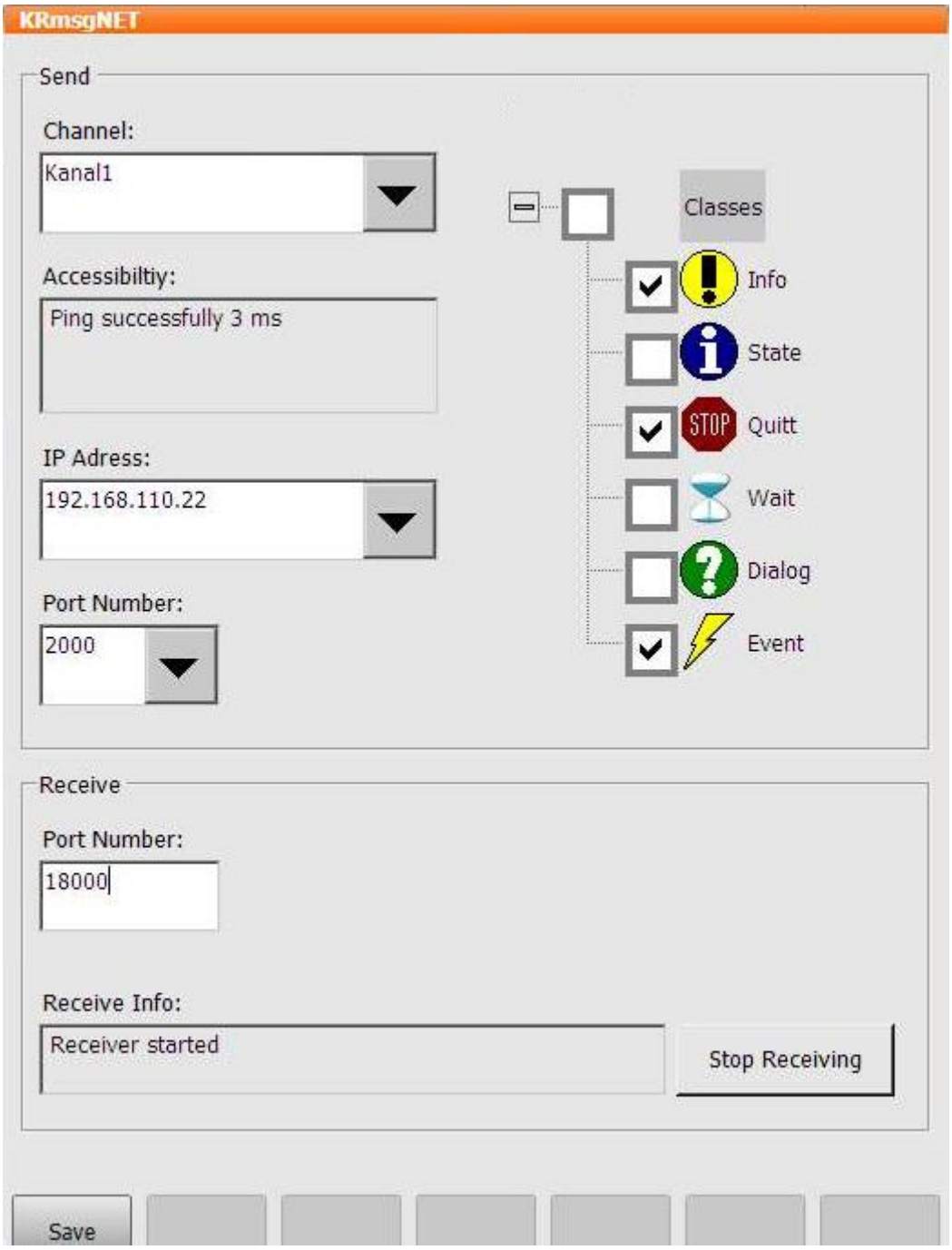
Liikkeiden suorituksen ketjutus ”KRC_Move”-alkuisten kirjastolohkojen avulla tuotti aluksi hankaluuksia. Toimiva sekvenssirakenne syntyi helpommin ”MC_Move”-alkuisten PLCOpen-standardilohkojen avulla. Tämä johtui lohkojen suoritusta seuraavien bittien käyttäytymiseroista. Myös ”KRC_Interrupt”-jarrutuslohkon käyttö aiheutti ongelmia, sillä jarrutuksen ollessa päällä mitkään muut komennot (edes puskurin tyhjentävä ”KRC_Abort”) eivät toimi. Tämä lohko jätettiin lopulta pois sovellusesimerkistä, sillä sitä ei katsottu tarpeelliseksi.

4.3.6 UDP-kommunikointi

Robotin diagnostiikkaviestit (hälytykset ym.) tuotiin logiikkaohjelmaan UDP-protokollan mukaisen tiedonsiirron avulla. Tätä varten luotiin toimintolohko ”GetAlarmsWithUDP”, jonka sisällä kutsuttiin Siemensin valmiit kommunikointilohkot ”TUSEND” ja ”TURCV” tiedon lähettämistä ja vastaanottamista varten. UDP-yhteys määriteltiin TIA Portalin laitteistokonfiguraatio-ikkunassa (KUVA 13). Lisäksi robotin käsiohjelmointilaitteelta oli sallittava viestien lähetys eteenpäin KRmsgNET-palvelun asetuksia muuttamalla (KUVA 14). UDP-kommunikointia käytettäessä on noudatettava sekä Siemensin, että KUKA:n ohjeita kommunikoinnin alustamisesta ja oltava tarkkana IP-osoitteiden ja porttinumeroiden kanssa. Väärä porttinumero robottiohjaimen lähetysparametreissa johti sovellusesimerkkiä tehtäessä tuntien turhaan työhön vianetsinnän osalta.



KUVA 12. UDP-yhteys TIA Portalissa



KUVA 13. KRmsgNET-asetukset

4.3.7 Ovikoteloiden painikkeet ja valot

Robottisolun ovien viereen oli asennettu jo aiemmin painikekotelot. Painikekoteloissa oli hätä-pysäytyspainikkeet sekä kolme painonappia (sininen, valkoinen ja vihreä). Sinistä painiketta

oli hyödynnetty turvaohjelmassa kuittauspainikkeena. Valkoinen painike oli aiemmin ollut ulkoisen ohjauksen pysäytyspyyntöpainike ja vihreä painike ulkoinen ohjelman käynnistyspainike.

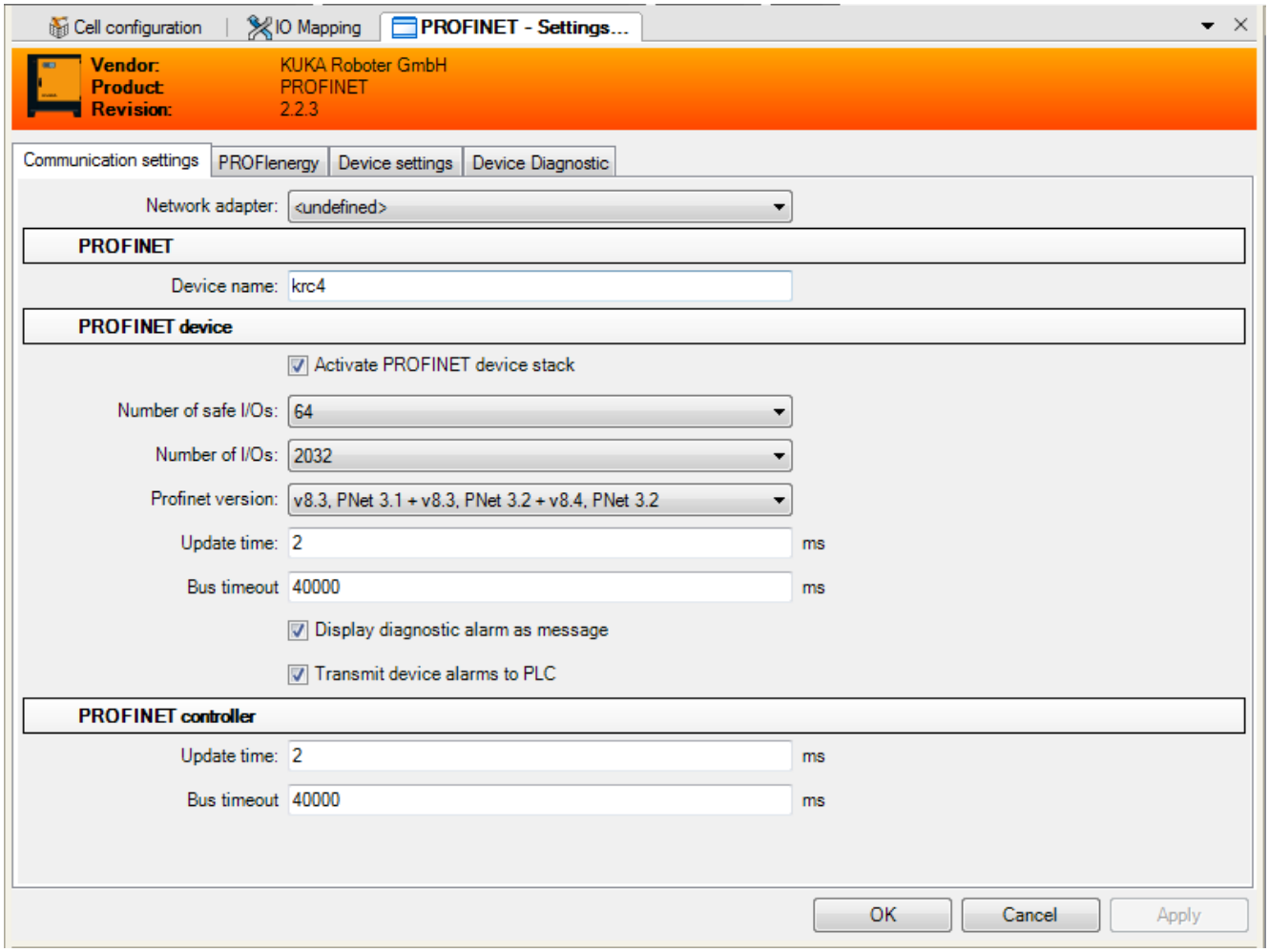
Painikkeita käytettiin samoihin tarkoituksiin myös sovellusesimerkin yhteydessä. Ohjelman suoritus käynnistyi (tai jatkui) vihreällä painikkeella. Valkoinen painike asetti ohjelmaan pysäytyspyynnön. Vihreän painikkeen valo vilkkui, mikäli robotti oli käynnissä, mutta ohjelma (tai yksittäinen liike) ei ollut suorituksessa ja paloi jatkuvasti suorituksen aikana. Valkoisen painikkeen valo vilkkui silloin, kun pysäytyspyyntö oli asetettu ja ohjelma edelleen suorituksessa ja paloi jatkuvasti, kun robotti oli pysäytetty pysäytyspyynnön avulla. Toinen painallus pysäytyspyynnön ollessa aktiivinen perui sen.

4.4 Turvasuunnittelu

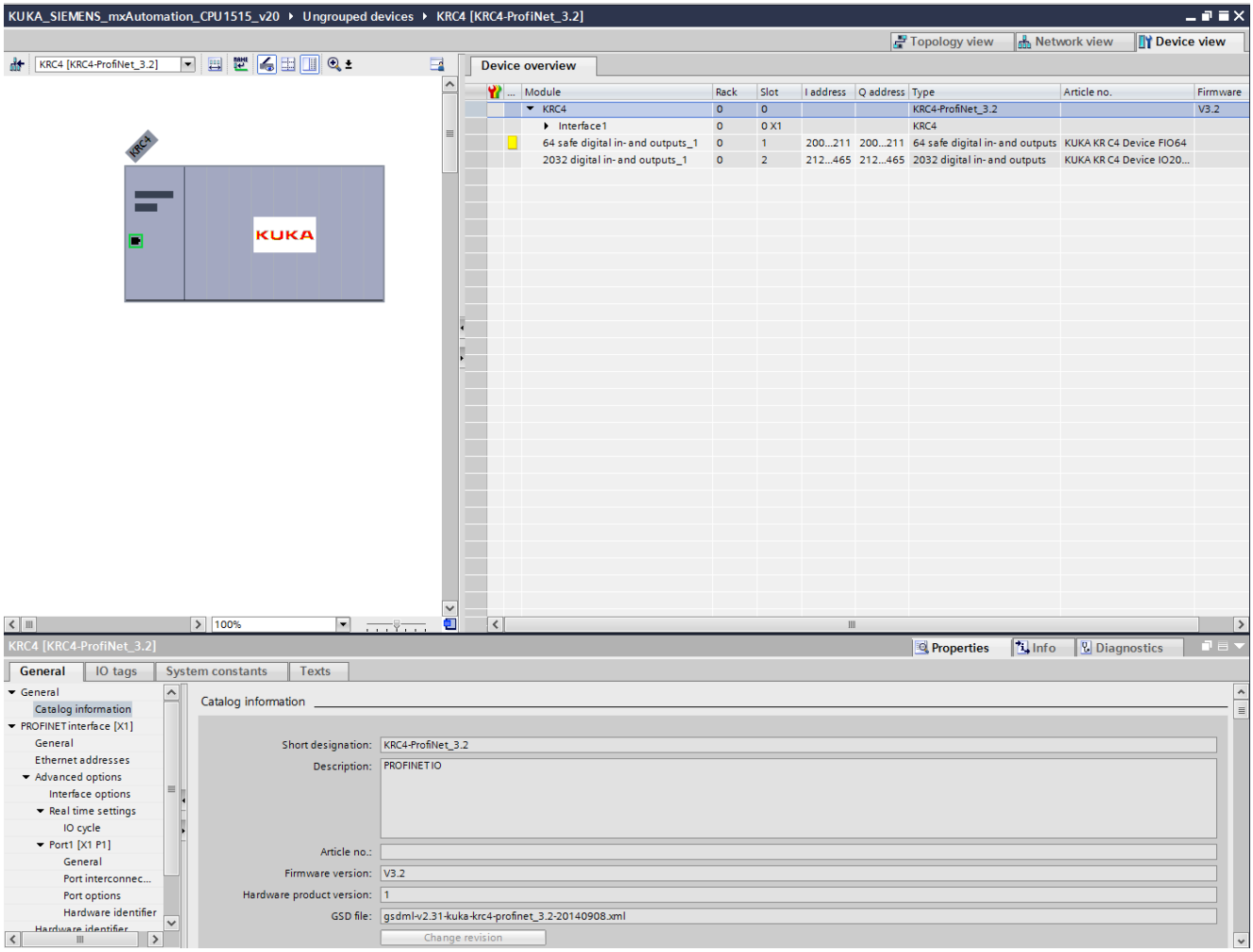
Robottisolulle aiemmin toteutettu turvaohjelma päätettiin säilyttää mahdollisimman muuttumattomana. Turvaohjelmassa oli toteutettu hälytykset ja kuittaukset törmäyssuojalle, robottisolun ovien rajakytkimille, moduulien väliselle rajakytkimelle sekä hätäpysäytys-piirille. Sovellusesimerkki rakennettiin vanhan turvaohjelman päälle. Kaikki muu vanhalle logiikalle toteutettu ohjelmisto poistettiin. Projektin I/O-määritykset säästettiin ja niitä käytettiin ominaisuuksien uudelleentoteutukseen mxAutomationin ja käyttöliittymän avulla. Tällaisia ominaisuuksia olivat esim. ovikoteloiden painikkeiden valojen ohjaukset sekä törmäyslaipan turvapiirin ohitus.

4.4.1 mxAutomation ja PROFISafe

Sovellusesimerkin turvaohjelmassa hyödynnettiin PROFISafe-kommunikointia. Asennettaessa mxAutomation-rajapintaa WorkVisualissa, tuli PROFINET-asetuksissa muuttaa turva I/O:n lukumäärä 64:n (KUVA 15). Turvatulot- ja lähdöt määriteltiin myös TIA Portalissa robottiohjaimen laitekuvatiedostolle (KUVA 16). Mikäli PROFISafe:a ei voida tai haluta käyttää, on robotin turvatoiminnot mahdollista integroida turvaohjelmaan hyödyntämällä robottiohjaimen X11-liityntärajapintaa (Siemens 2016, 43–44).



KUVA 14. PROFINET-asetukset WorkVisualissa



KUVA 15. Robottiohjaimen turva-I/O TIA Portalissa

4.4.2 mxAutomation ja turvasuunnittelu

mxAutomation-rajapintaa käytettäessä ohitetaan joitakin robotin turvaominaisuuksia. Koska robottiohjain on tällöin aina ”EXT”-tilassa (ulkoinen ohjaus), ei sallintakytkintä tarvitse käyttää käsiajon yhteydessä. Käsiajonopeutta ei ole myöskään rajoitettu. Nämä toiminnot on kuitenkin mahdollista toteuttaa ohjelmallisesti, mikäli niitä halutaan käyttää. Toisaalta rajapintaa hyödynnettäessä ”EXT”-tilan turvatoiminnot ovat kokoajan käytössä. Turvalaitteiston on oltava toiminnassa ja kuitattuna, jotta robotin liikuttaminen olisi mahdollista. Tästä syystä esim. robotin operointi turva-alueen sisältä ei ole mahdollista, kuten tavallisesti robotin ollessa ”T1”-tilassa.

Koska robotti oli liitetty PROFISafe-turvaväylään, laukaisi turvapiirin katkeaminen myös robotin turvapysäytyksen, joka oli kuitattava robottiohjaimella. Turvapiirin aiheuttamat virheet kuitattiin ajamalla uudelleen robotin käynnistyssekvenssi. Tällöin robotin liikekäskyjen suorituksen estävä ”Collective fault”-vikasignaali (jota ilmaisi ”KRC_AutomaticExternal”-kirjastolohkon ”STOPMESS”-lähtö) voitiin nollata, jolloin hälytys kuittaantui ja robotti saatiin uudelleen toimintavalmiuteen.

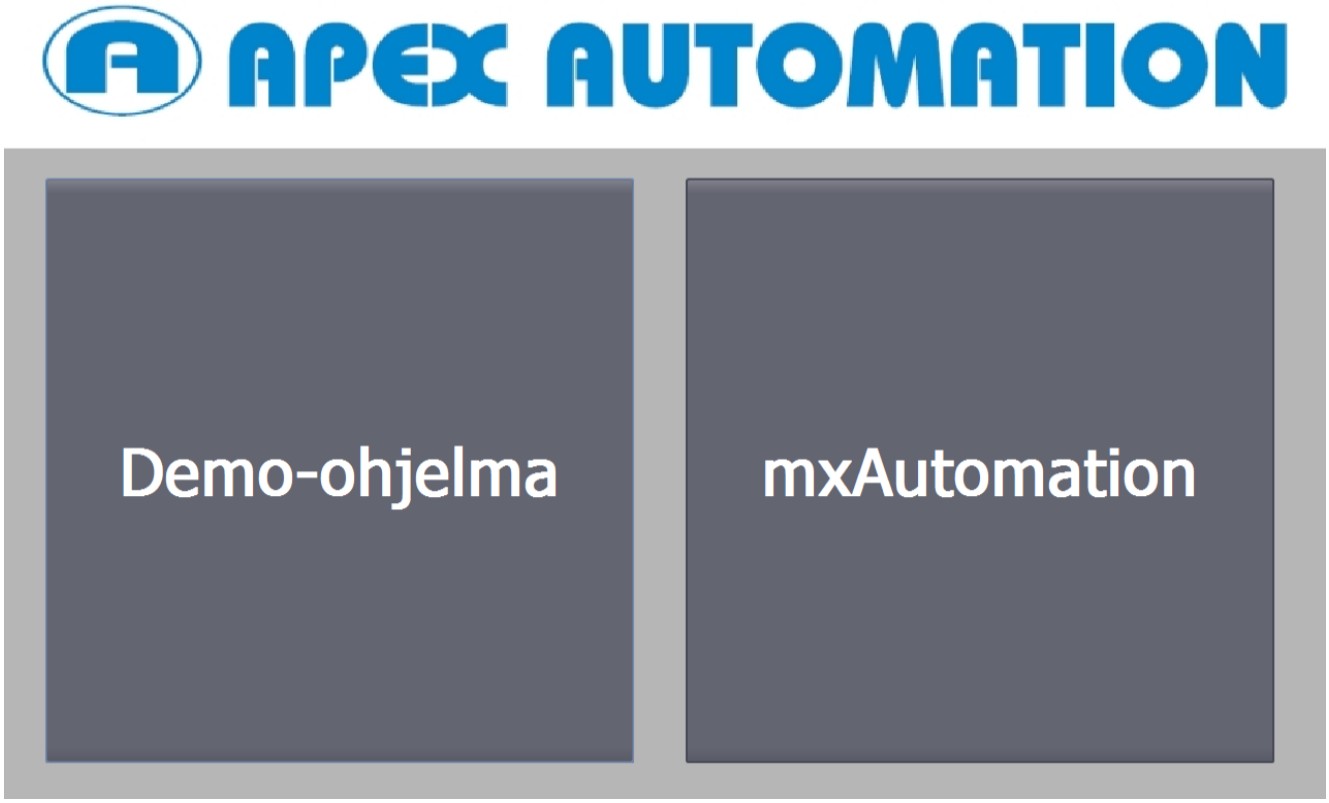
4.5 HMI-suunnittelu

Paneelisuunnittelun osalta tavoitteena oli yksinkertainen ja helppokäyttöinen käyttöliittymä, joka mahdollisti robotin käynnistyksen, käsiohjauksen, online-ohjelmoinnin sekä diagnostiikan ja oleellisten bittien tilatietojen seurannan. Lisäksi huomioitiin yleisiä hyvän näyttösuunnittelun ohjeita. Näytölle tehdyistä painikkeista tehtiin riittävän suuria, jotta niihin oli helppo osua ja painikkeiden teksteistä ja grafiikkakuvista pyrittiin tekemään yksiselitteisiä. Käyttöliittymässä navigoinnista pyrittiin tekemään yksinkertaista niin, että kaikkiin oleellisiin näyttöihin pääsi suoraan kaikissa näytöissä muuttumattomana pysyvästä alapalkista ja myös edelliseen näyttöön paluu onnistui erillisellä alapalkin painikkeella.

Ylä- ja alapalkit toteutettiin muuttumattomina pohjina ja niitä käytettiin aloitusnäyttöä lukuun ottamatta kaikissa näytöissä. Yläpalkkiin sijoitettiin Apex Automationin logo, päivämäärä ja kellonaika sekä kuittaamatonta hälytystä ja hätäpysäytystä kuvaavat tekstit ja grafiikkaelementit. Alapalkkiin sijoitettiin painikkeet aloitusnäyttöön, edelliseen näyttöön sekä huolto- ja ohjelmointinäyttöihin siirtymistä varten. Lisäksi alapalkkiin oli sijoitettu ”käynnistä”-painike, josta robotti saatiin käyttövalmiuteen.

4.5.1 Aloitusnäyttö

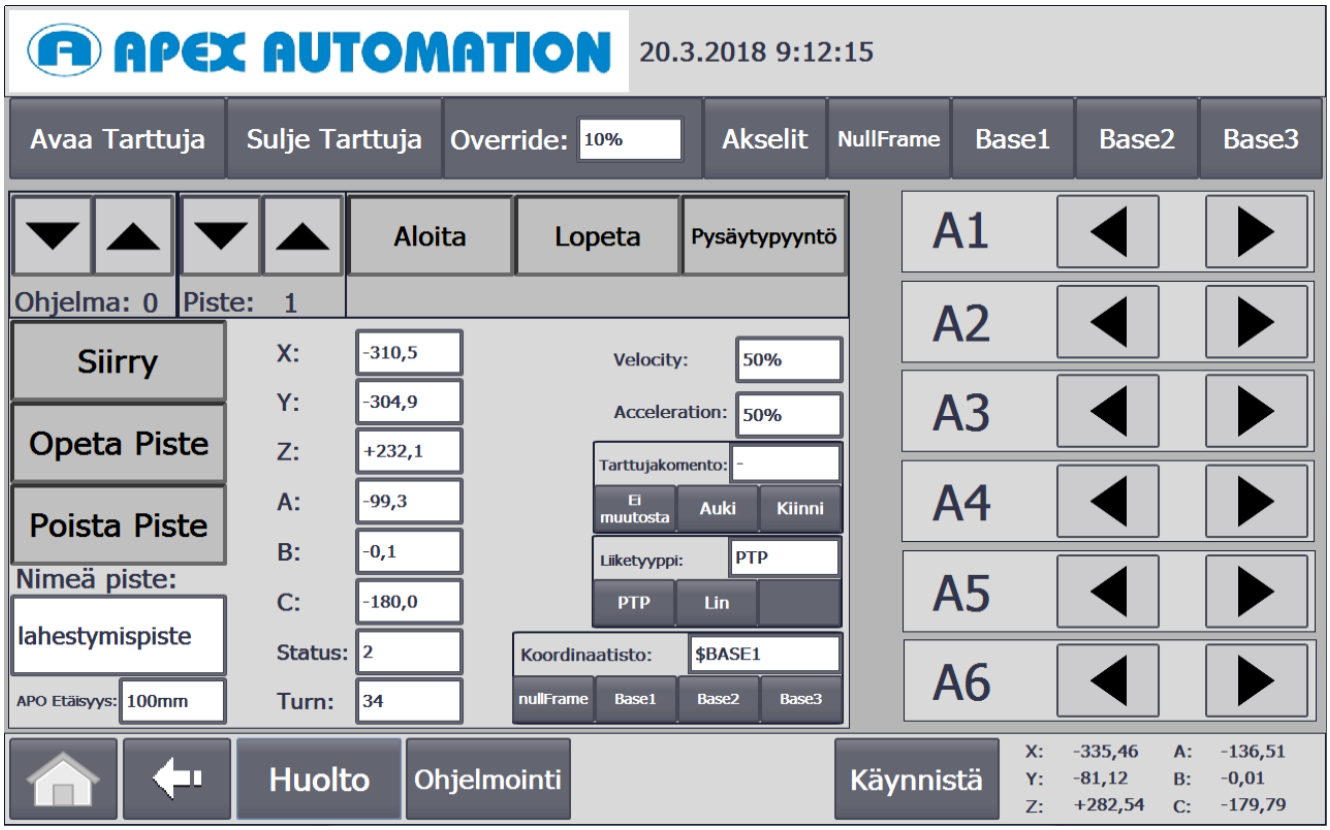
Aloitusnäyttöön (KUVA 17) sijoitettiin painikkeet ”mxAutomation” sekä ”Demo-ohjelma”. ”mxAutomation”-painike vei käyttäjän ohjelmointinäyttöön. ”Demo-ohjelma”-painikkeella siirryttiin esittelyä varten tehdyn valmiin ohjelman suorituksen ohjaukseen tarkoitettuun näyttöön. Aloitusnäytön yläreunaan sijoitettiin koko ruudun levyinen Apex Automationin logo.



KUVA 16. Aloitusnäyttö

4.5.2 Ohjelmointinäyttö

Käyttöliittymään luotiin ohjelmointinäyttö, jolta käsin oli mahdollista ohjata robottia sekä opettaa sille ohjelman liikeradan muodostavia liikepisteitä (KUVA 18). Ohjelmointinäytön osat jaoteltiin selkeästi niin, että robotin käyttäminen näytöltä olisi helppoa ja yksiselitteistä. Ylävasemmalle koottiin painikkeet tarttujan avaamiseen ja sulkemiseen sekä ohitusnopeuden säätö. Näiden alapuolelle sijoitettiin ikkunaan ryhmiteltynä robotin online-ohjelmointiin tarvittavat painikkeet ja syöttö- sekä näyttökentät. Näytön oikeassa laidassa oli kuusi paria ohjauspainikkeita käsiohjausta varten sekä painikkeet käsiohjauksen koordinaatiston valintaan (akselikohtainen sekä koordinaatistot NULLFRAME, BASE1, BASE2 tai BASE3). Käsiohjauspainikkeiden selitysteksti sekä painikkeen toimintaa kuvaavat symbolit toteutettiin niin, että ne muuttuivat vastaamaan käsiohjauksen tilaa (akselikohtainen tai koordinaatisto).

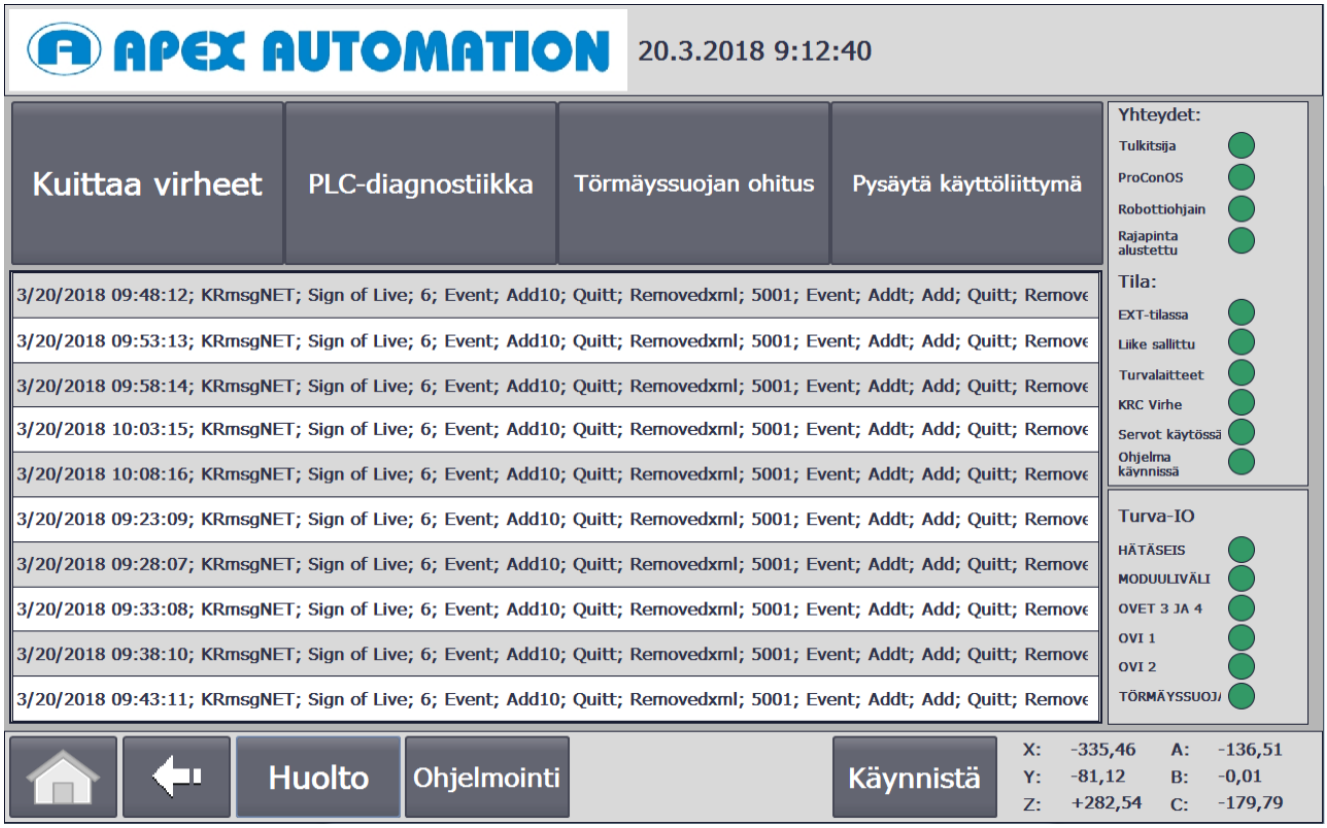


KUVA 17. Ohjelmointinäyttö

Ohjelmointia varten ryhmitellyssä ikkunassa oli ylhäällä vasemmalla nuolinäppäimet ohjelman sekä liikepisteen valintaan. Näiden oikealla puolella oli sijoitettu ohjelman suoritukseen liittyvät painikkeet aloita, lopeta ja pysäytyspyyntö ohjelman (eli pisteiden muodostaman liikeradan ja pistekohtaisten työkalutoimintojen) suorittamiseen, suorituksen perumiseen ja pysäytyspyynnön asettamiseen sekä näiden alla tekstikenttä, josta ilmeni mikä ohjelman liikepisteistä on suorituksessa. Ikkunan vasemmassa laidassa oli painikkeet liikepisteeseen siirtymiseen sekä liikepisteen opettamista ja poistamista varten. Lisäksi vasemmassa laidassa oli kentät liikepisteen nimen sekä tarkkuusparametrin määrittelyä varten. Näiden oikealla puolella oli kentät liikepisteen koordinaattien esittämistä ja muokkausta varten, sekä liikepisteen ”status”- ja ”turn”-muuttujien esittämiseen tarkoitetut kentät. Oikeassa laidassa oli kentät pistekohtaisen nopeuden ja kiihtyvyyden määrittelyyn sekä liikepisteen tarttujakomennon (ei toimintoa, avaa, sulje), liiketyypin (vain asemasta asemaan -ja lineaariliikkeet) ja koordinaatiston valintaan.

4.5.3 Huoltonäyttö

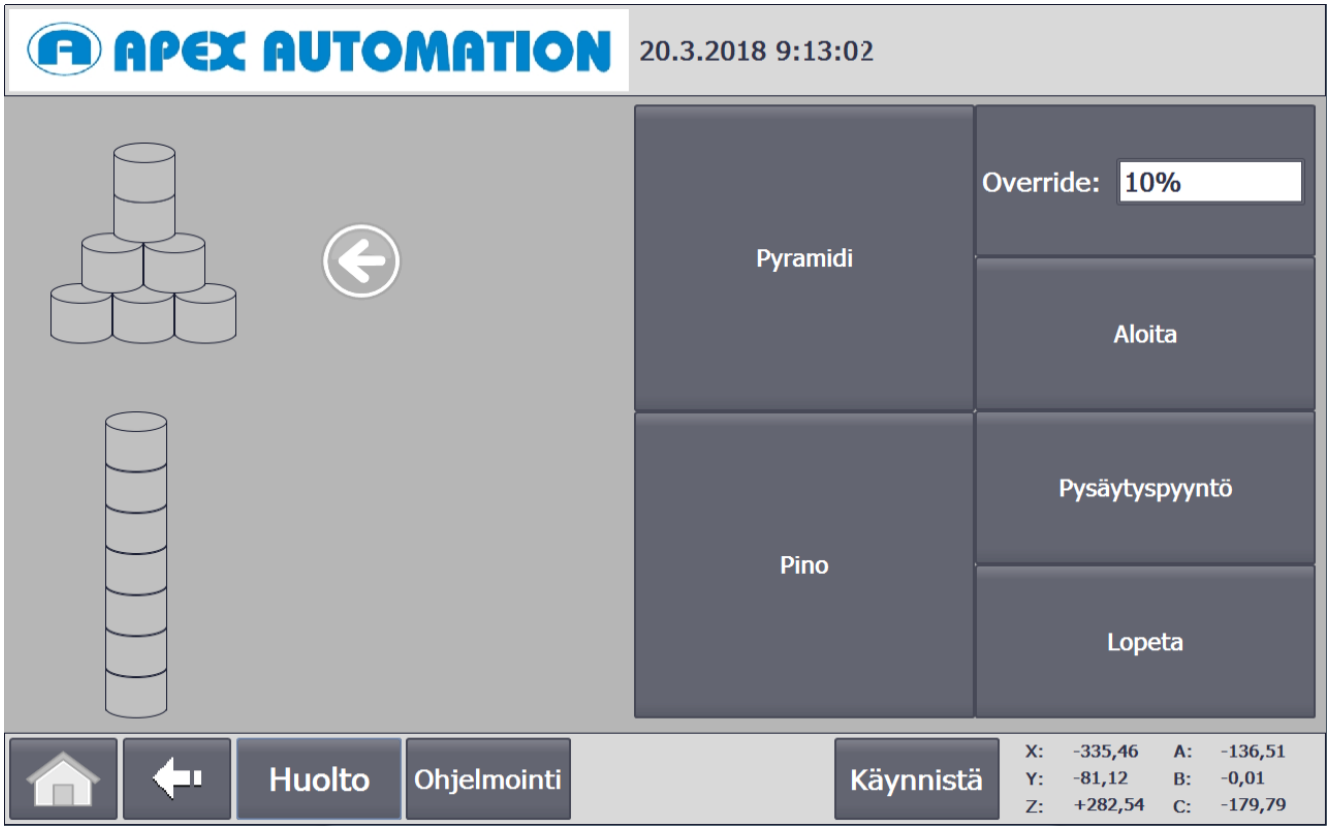
Huoltonäytölle (KUVA 19) oli pyritty tuomaan kaikki robottisoluun liittyvät vika- ja hälytystiedot. Huoltonäytön vasempaan yläkulmaan oli koottu painikkeet robottiohjaimen hälytysten kuittaukselle, logiikan diagnostiikkanäkymän sisältävään ikkunaan siirtymiselle sekä törmäyssuojan ohitukselle ja käyttöliittymän pysäyttämiselle (eli poistumiselle ”runtime”-tilasta paneelin Windows-työpöydälle). Näiden alapuolelle oli tehty taulukko, johon tuotiin näkyville robottiohjaimen hälytys- ja tilaviestit. Huoltonäytön oikeaan reunaan oli koottu kaksi aluetta, joissa oli värein indikoivin palloin esitetty robotin tilabittejä sekä turva-I/O:n bittien tilat. Pallojen väritys oli sellainen, että vikatila ilmaistiin kirkkaan punaisella, normaalitila harmaanvihreällä.



KUVA 18. Huoltonäyttö

4.5.4 Demo-ohjelma

Demo-ohjelmaa varten luotiin oma näyttö, josta oli mahdollista valita robotin kasaama rakennelma (KUVA 20). Lisäksi näytöllä oli ohjelman suorituksen ohjaukseen tarkoitetut painikkeet ”Aloita”, ”Pysäytyspyyntö” sekä ”Lopeta”. Nämä painikkeet toimivat, kuten ohjelman suorituksen ohjauspainikkeet ohjelmointiruudulla. Näytölle luotiin kiekoista muodostuvaa pinoa ja pyramidia muistuttavat kuviot. Kuviot toteutettiin niin, että yksittäiset kiekot vaihtoivat väriä taustaväristä vaaleampaan harmaan sävyyn osoittamaan rakennelmassa olevien kiekkojen määrää. Näytölle luotiin valittua rakennelmaa osoittavat nuolet, jotka vilkkuivat, mikäli toinen rakennelma, tai sen purku, oli vielä kesken.



KUVA 19. Demo-ohjelman ohjausnäyttö

5 POHDINTA

Tärkeä osa opinnäytetyön toimeksiantoa oli rajapinnan sekä sen avulla tehdyn sovellusesimerkin vertailu perinteiseen robottiohjaukseen. Koska robotin käsiohjelmointilaite ja robottiohjelmoinnin perusteet tulivat väkisinkin tutuiksi rajapinnan käyttöönoton ja sovellusesimerkin suunnittelun yhteydessä, oli eroja ja yhtäläisyyksiä helppo löytää. Vertailua helpotti myös se, että luotu käyttöliittymä muistutti monilta osin perinteistä robotin käsiohjelmointilaitetta, sillä samat perusohjaustoiminnot oli kyettävä toteuttamaan myös rajapintaa hyödynnettäessä.

5.1 Vertailu perinteiseen robottiohjaukseen

Verrattuna perinteiseen robottiohjaukseen mxAutomationin avulla toteutettu sovellusesimerkki tarjosi erilaisen näkökulman. Sovelluksesta kävi ilmi vain pieni osa mxAutomationin eduista. Toisaalta demosovelluksen kohdalla ilmenneet puutteet johtuivat lähinnä siitä, ettei tavoitteena ollut rakentaa täydellisesti robottiohjaimen korvaavaa, vaan ainoastaan rajapinnan esittelyyn sopiva järjestelmä.

mxAutomation-rajapintaa hyödynnettäessä on huomioitava se, että robotin ohjaus vaatii edelleen ymmärrystä robotiikasta. Robottiohjelman toteuttaminen logiikalla mxAutomation-kirjastolohkojen avulla on mahdotonta, mikäli suunnittelija ei ymmärrä lohkoihin liittyvien muuttujien ja ohjaustoimintojen merkitystä. Vaikka robotin käsiohjelmointilaitetta ei valmiin mxAutomation-sovelluksen käyttöönoton jälkeen varsinaisesti tarvita, noudattaa itse robotti samoja lainalaisuuksia, kuin perinteisellä robottiohjauksella toteutettuna.

5.1.1 Edut

Monet rajapinnan edut, kuten useiden robottien ohjauksen tuonti yhdelle näytölle eivät olleet mahdollisia toteuttaa tai niitä ei koettu tarpeellisiksi opinnäytetyön kannalta. Mahdollisuus koota vikatiedot yhteen paikkaan oli yksi selkeistä eduista, joita sovellusesimerkin yhteydessä ilmein. Sekä robottiohjaimen, että muiden logiikkaan liitettyjen laitteiden vikatiedot saatiin rajapinnan ansiosta helposti samalle paneelille. Vikatietojen hyödyntäminen logiikkaohjelmassa oli

varsin helppoa. Myös muiden tietojen luku robotilta oli helppoa, mikä kannusti niiden hyödyntämiseen sekä ohjelmoinnissa, että visualisoinnissa.

Robotin ohjaaminen Siemensin 15 tuumaiselta Comfort-paneelilta oli helppoa ja kosketusnäyttöjen yleisyyden takia tällainen ohjaustapa saattaisi tehdä robotin käsiohjauksesta (ja näin ollen myös online-ohjelmoinnista) helposti lähestyttävää suunnittelijoille, jotka syystä tai toisesta haluavat välttää robotin käsiohjelmointilaitteen käyttöä. Vaikka käsiohjelmointilaitteen 6D-hiiri tarjosi subjektiivisesti miellyttävämmän ohjauskokemuksen, ei näytön ohjauspainikkeiden käyttö hävinnyt silmämääräisen paikoitustarkkuuden perusteella käsiohjelmointilaitteen painonapeille.

5.1.2 Kehitysmahdollisuudet

Sovellusesimerkistä jätettiin pois joitain ominaisuuksia, joiden lisääminen olisi parantanut sen hyödyllisyyttä mahdollisissa käytännön sovelluksissa. Nämä ominaisuudet rajattiin pois, sillä niitä ei koettu tarpeellisiksi toteuttaa opinnäytetyön yhteydessä. Mitään robottiohjelmoinnin tai robotin asetusten määrittelyyn liittyvää toimintoa ei kuitenkaan jätetty pois sen takia, että se olisi ollut mahdotonta toteuttaa. Rajaukset määriteltiin pääosin kahden muuttujan avulla: ominaisuuden tarpeellisuus ja toteutuksen vaatima ohjelmointityö.

Robotin ohjaus- ja ohjelmointitoiminnot käyttöliittymässä toteutettiin esittelytarkoitukseen. Esim. ehtorakenteiden toteutusta ei katsottu tarpeelliseksi online-ohjelmoinnissa, sillä se olisi mxAutomationia hyödynnettäessä järkevämpää toteuttaa logiikalla (kuten sovellusesimerkin demo-ohjelman tapauksessa tehtiinkin). Ohjelmarakenteiden tekeminen suoraan logiikkakoodiin voidaan nähdä etuna esimerkiksi silloin, kun yksi suunnittelija vastaa robotin lisäksi esim. turvaohjelman tekemisestä, sillä ohjelmointityökalua ei tarvitse vaihtaa kesken suunnittelutyön.

Usean robotin solun avulla olisi voitu demonstroida keskitettyä ohjausta, esim. toteuttamalla jokin synkronoitu robottisovellus, jonka tilatietoja olisi voitu tuoda näytölle. Robotin toimintaa liittyviä sisäisiä parametreja olisi muutenkin voinut hyödyntää enemmän käyttöliittymää rakennettaessa. Tällöin rajapinnan toiminta-ajatus olisi voitu tuoda paremmin esille visuaalisen esityksen kautta.

5.2 Yleisiä huomioita

mxAutomation-rajapinnan käyttöönotto ei onnistunut suoraan ohjeita noudattamalla, sillä ohjeiden ja rajapinnan todellisen toiminnan välillä oli ristiriitoja. Tämä saattaa olla tapauskohtaista, mutta sovellusesimerkin tapauksessa hidasti merkittävästi työn etenemistä. Yleisesti ottaen rajapinnan toiminta vastasi kuitenkin sitä, mitä siltä odotettiin. Kaikki sovellusesimerkin yhteydessä ilmenneet poikkeavuudet sekä käytetyt ratkaisumallit on pyritty esittämään edellisen pääluvun asiaankuuluvissa alaluvuissa.

Sovellusesimerkin kaltaisen järjestelmän suunnittelu ja toteutus käytännön sovellusta varten vaikuttaisi olevan suhteellisen yksinkertaista, sillä suurin osa ajasta kului viankorjaukseen sekä rajapinnan testaukseen, ei varsinaiseen ohjelmointityöhön. Toisaalta testikäyttöön tarkoitettu toimintaympäristö rajasi pois sellaisia tekijöitä, jotka saattaisivat esim. tuotantolaitoksessa vaikeuttaa ja hidastaa järjestelmäsuunnittelua sekä testausta ja käyttöönottoa. Myös järjestelmän laajuuteen tehdyt rajaukset perustuivat lähinnä testikäytön vaatimuksiin. Todellinen käytännön sovellus saattaisi vaatia huomattavasti monimutkaisempia ohjelmarakenteita.

6 YHTEENVETO

Opinnäytetyön tavoite, eli rajapintaan tutustuminen ja sen käyttöönotto täyttyi lähes odotusten mukaisesti. Vaikka sovellusesimerkki ei ollut täysin valmis teknologiapäivään mennessä, oli se esittelykelpoinen. Kaikki halutut ominaisuudet saatiin lopulta toteutettua. Kohdatut ongelmat laajensivat ymmärrystä rajapinnan toiminnasta. Kohdattujen ongelmien ratkominen vaati myös lisäperehtymistä robotiikkaan sekä TIA Portaliin toimintaympäristönä. Tämä vahvisti ennestään niitä osaamisalueita, joita työstä suoriutuminen vaati.

Robotiikan perusteet oli pakko ymmärtää rajapinnan hyödyntämiseksi, joten opinnäytetyö toimi myös robotiikan peruskurssina. Koska robottiohjelmointi ja robotiikan ymmärrys ovat taitoja, joista on hyötyä automaation alalla, luo opinnäytetyö aiheesta uusia mahdollisuuksia työelämään siirryttäessä. Myös TIA Portal -osaaminen vahvistui sovellusesimerkin suunnittelun yhteydessä. Tutuiksi tulivat niin laitteistokonfiguraation määrittely, kuin ohjelmisto- ja HMI-suunnittelukin. Kyky hyödyntää Siemensin diagnostiikkaa vianmääritykseen on myös taito, joka liittyy oleellisesti automaatiojärjestelmien suunnitteluun ja toteutukseen TIA Portalin avulla.

mxAutomation-rajapinnan edut ja kehitysmahdollisuudet ilmenivät pääosin hyvin tehdystä sovellusesimerkistä. Puuttumaan jäi joitain ominaisuuksia, jotka eivät olleet fyysisesti mahdollisia tai kannattavia toteuttaa käytössä olleella laitteistolla. Yhteisen ohjelmointialustan edut tulivat ilmeisiksi jopa yksinkertaisen sovellusesimerkin suunnittelun yhteydessä. Robotilta haluttua tietoa oli helppo lukea ohjelmaan ja hyödyntää siinä. Tämä kannusti myös tiedon visuaaliseen esittämiseen näytöllä.

Käytännössä mxAutomation-rajapinta vaikuttaisi tarjoavan lähes kaikki oleelliset robottiohjelmointiin sekä robotin asetusten määrittelyyn vaaditut toiminnot. Rajapinnan hyödyntäminen käytännön robottisovelluksissa vaikuttaa siis täysin mahdolliselta. Rajapinnan soveltamisen hyödyt ja haitat ovat todennäköisesti hyvin tapauskohtaisia.

LÄHTEET

Anteroinen, S. 2015. Robottistrategia vaiheessa. Prometalli. 2/2015. 14–21.

Apex Automation. 2017. Saatavissa: https://www.apexautomation.fi. Viitattu 13.2.2018.

Asada, H. 2005a. Introduction to Robotics, Chapter 1. Saatavissa: https://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter1.pdf. Viitattu 8.2.2018.

Asada, H. 2005b. Introduction to Robotics, Chapter 3. Saatavissa: https://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter3.pdf. Viitattu 8.2.2018.

British Automation & Robot Association. Robots. Saatavissa: http://www.bara.org.uk/introduction-to-industrial-robots.html. Viitattu 7.2.2018.

Crowder, R. 1998. The Industrial Robot. Saatavissa: http://www.southampton.ac.uk/~rmc1/robotics/arirobot.htm. Viitattu 12.2.2018.

Gruhn, P. 2011. Human Machine Interface (HMI) Design: The Good, The Bad, and The Ugly (and what makes them so). Saatavissa: http://www.kirp.chtf.stuba.sk/moodle/pluginfile.php/61474/mod_resource/content/2/hmi_rules.pdf. Viitattu 13.3.2018.

Hall, N. 2015. Aircraft Rotations. Saatavissa: https://www.grc.nasa.gov/www/k-12/airplane/rotations.html. Viitattu 12.2.2018

International Federation of Robotics. 2017b. Executive Summary World Robotics 2017 Industrial Robots. Saatavissa: https://ifr.org/downloads/press/Executive_Summary_WR_2017_Industrial_Robots.pdf. Viitattu 7.2.2018.

International Federation of Robotics. Industrial Robots. Saatavissa: https://ifr.org/industrial-robots. Viitattu 7.2.2018.

International Federation of Robotics. Robot History. Saatavissa: https://ifr.org/robot-history. Viitattu 8.2.2018.

International Federation of Robotics. 2017a. The Impact of Robots on Productivity, Employment and Jobs. Viitattu 1.3.2018. Saatavissa: https://ifr.org/img/office/IFR_The_Impact_of_Robots_on_Employment.pdf.

International Federation of Robotics. 2017c. World Robotics 2017 Industrial Robots. Introduction. Saatavissa: https://ifr.org/downloads/press/WR_Industrial_Robots_2017_Chapter_1.pdf. Viitattu 14.2.2018.

ISO. 8373:2012. Robots and robotic devices — Vocabulary. 2012. Saatavissa: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en. Viitattu 7.2.2018.

Järvinen, H. & Mikkonen, T. 2010. Sulautettu ohjelmointi. ”Esipainos”. Saatavissa: http://www.cs.tut.fi/~sulo/pruju/pruju.pdf. Viitattu 22.2.2018.

KUKA. 2016. Industrial robotics_small robots. Saatavissa: https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/kuka_pb_kleinroboter_en.pdf Viitattu: 12.2.2018.

KUKA. KUKA.PLC mxAutomation. Saatavissa: https://www.kuka.com/en-us/products/robotics-systems/software/hub-technologies/kuka,-d-,plc-mxautomation. Viitattu 9.2.2018.

KUKA. 2015a. KUKA System Software 8.3. Operation and Programming Instructions for System Integrators. Saatavissa: http://www.wtech.com.tw/public/download/manual/kuka/krc4/KUKA%20KSS-8.3-Programming-Manual-for-SI.pdf. Viitattu 14.2.2018.

KUKA. 2015b. S7 Library. For KUKA.PLC mxAutomation 2.1. KUKA:n toimittama pdf-asiakirja.

KUKA. The history of KUKA. Saatavissa: https://www.kuka.com/en-us/about-kuka/history. Viitattu 9.2.2018.

KUKA. 2014. WorkVisual 3.1. Saatavissa: http://www.wtech.com.tw/public/download/manual/kuka/krc4/KUKA%20WorkVisual%203.1.pdf. Viitattu 14.2.2018.

KUKA. The KUKA smartPAD: simply more freedom. Saatavissa: https://www.kuka.com/en-hu/products/robotics-systems/robot-controllers/smartpad. Viitattu 14.2.2018.

Kwang, T. 2017. The future of production: IoT, AI, robotics, wearables and 3D printing. Saatavissa: https://www.enterpriseinnovation.net/article/future-production-iot-ai-robotics-wearables-and-3d-printing-51643031. Viitattu 8.2.2018.

KW Software, Inc. ProConOS. Saatavissa: https://www.plantautomation.com/doc/proconos-0001. Viitattu 22.2.2018.

Lahden ammattikorkeakoulu. 2016. Robotiikan yleinen osa, luennot. Saatavissa: http://miniweb.lpt.fi/automaatio/opetus/luennot/pdf_tiedostot/Robotiikka_yleinen.pdf. Viitattu 8.2.2018.

McDaniel, C. 2016. Design Tips to Create a More Effective HMI. Saatavissa: https://automation.isa.org/2016/05/design-tips-for-more-effective-hmi/. Viitattu 12.3.2018.

Monahan, B., Hoffmann, K., Ferchak, A. & Bishop, N. 2016. 11 principles to guide HMI design for critical drilling equipment. Saatavissa: http://www.drillingcontractor.org/11-principles-guide-hmi-design-critical-drilling-equipment-40154. Viitattu 13.3.2018.
PLCopen. IEC 61131-3: a standard programming resource. Saatavissa: http://www.plcopen.org/pages/tc1_standards/downloads/intro_iec.pdf. Viitattu: 13.3.2018.

PLCOpen. TC1 – Standards. Saatavissa: http://www.plcopen.org/pages/tc1_standards/. Viitattu: 13.3.2018.

Sanastokeskus TSK ry. 2014. Geoinformatiikan sanasto (TSK 45). 3. laitos. Helsinki: Sanastokeskus TSK ry.

Siemens. 2016. Controlling a KUKA Industrial Robot Using a SIMATIC S7-1500. Saatavissa: https://cache.industry.siemens.com/dl/files/123/109482123/att_867904/v1/109482123_S7-1500_KUKA_mxAutomation_DOKU_v11_en.pdf. Viitattu 18.3.2018.

Siemens. PROFISafe-hybridiväylätekniikka. Saatavissa: http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/kone_ja_prosessiturvallisuus_seka_atex/koneturvallisuus/simatic_turva_automaatio/profisafe_hybridivaylatekniikka.htm. Viitattu 13.3.2018.

Siemens. 2014b. Programming Guideline for S7-1200/S7-1500. Saatavissa: http://www1.siemens.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/simatic_s71200/programming-guideline-for-s71200-s71500_2014-09_en.pdf. Viitattu 13.3.2018.

Siemens. 2014a. SIMATIC S7-1500. Getting Started. Saatavissa: https://support.industry.siemens.com/cs/attachments/71704272/s71500_getting_started_en-US_en-US.pdf?download=true. Viitattu 23.2.2018.

Siemens. 2014c. STEP 7 Basic V13 SP1. System Manual. Saatavissa: http://www1.siemens.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/simatic_s71200/manualy/gsg_step7-basic-v10-5_2014-12_en.pdf. Viitattu 13.3.2018.

Siemens. TIA Portal (Step 7). Saatavissa: http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/ohjelmistot/tia_portal_step7.htm. Viitattu 13.3.2018.

Siemens. TIA Portal (WinCC TIA). Saatavissa: http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/kayttoliittymat/ohjelmistot/tia_portal_wincc.php. Viitattu 13.3.2018.

Stark, H. 2017. As Robots Rise, How Artificial Intelligence Will Impact Jobs. Saatavissa: https://www.forbes.com/forbes/welcome/?toURL=https://www.forbes.com/sites/haroldstark/2017/04/28/as-robots-rise-how-artificial-intelligence-will-impact-jobs/. Viitattu 5.3.2018.

Suomen Robotiikkayhdistys. 2016. Teollisuuden robottikannan väheneminen on pysähtynyt. Saatavissa: https://roboyhd.fi/2016/09/30/teollisuuden-robottikannan-vaheneminen-on-pysahtynyt/. Viitattu 7.2.2018.

Valtioneuvoston asetus koneiden turvallisuudesta 12.6.2008/400. Saatavissa: https://www.finlex.fi/fi/laki/ajantasa/2008/20080400. Viitattu 14.2.2018.

Wicks, M. 2014. PLC-based vs. proprietary robotic controls. Saatavissa: https://www.controleng.com/single-article/plc-based-vs-proprietary-robotic-controls/. Viitattu 8.2.2018.