SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Niklas Mehtälä

Varastoautomaatiojärjestelmän kehittäminen

Opinnäytetyö
Syksy 2024
Insinööri (AMK), Automaatiotekniikka

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Tutkinto-ohjelma: Insinööri (AMK), Automaatiotekniikka

Suuntautumisvaihtoehto: Sähköautomaatio

Tekijä: Niklas Mehtälä

Työn nimi alaotsikoineen: Varastoautomaatiojärjestelmän kehittäminen

Ohjaaja: Niko Ristimäki

Vuosi: 2024 Sivumäärä: 44 Liitteiden lukumäärä: 1

Työn tavoitteena oli kehittää Beckhoff-logiikalla ja servomoottoreilla toimiva, sekä TwinCAT3-ohjelmalla ohjelmoitu varastojärjestelmän pienikokoinen malli, jota voidaan käyttää tulevaisuudessa laboratoriotyönä jonkin kurssin toteutuksessa. Työ toteutettiin Seinäjoen ammattikorkeakoululle. Laite oli jonkin aikaa käyttämättömänä ja sille täytyi tehdä käyttöönotto ja kokeilla kaikkien akselien liikkuvuus ja turvarajojen toiminta.

Työn tarkoituksena oli suorittaa käyttöönotto ja kehittää laitteelle ohjausjärjestelmä, jolla laitetta voisi ohjata manuaalisesti tai käyttäen automaattiohjelmaa. Tuloksena saatiin toimiva varastoautomaatti, jota voidaan ajaa manuaalisesti ja automaattisesti laitteen HMI-paneelin käyttöliittymää käyttäen.

[1] Asiasanat: ohjelmointiympäristö, varastointi, ohjelmoitavat logiikat, servotekniikka

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Degree programme: Bachelor of Engineering, Automation Engineering

Specialization: Electric Automation

Author: Niklas Mehtälä

Title of thesis: Development of an automated storage system

Supervisor: Niko Ristimäki

Year: 2024 Number of pages: 44 Number of appendices: 1

The goal of the thesis project was to build a working small scale storage automation system by using servomotors, Beckhoff logic controllers and TwinCAT3-software. It was hoped that the device could be utilized in laboratory work in some laboratory courses. The thesis was done for Seinäjoki University of Applied Sciences. The device had been unused for many years, and it had to undergo commissioning and the mobility of all axes, and the operation of safety limits had to be tested.

The goal of the thesis was to perform the commissioning and to develop a control system for the device, which could be operated manually or using an automatic program. As the result, there was a working storage automation system, which could be operated manually and automatically through an HMI-interface.

[1] Keywords: programming environment, storage, programmable logic controllers, servo technology

SISÄLTÖ

Kuva-, kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

AS/RS	AS/RS (Automated Storaging/Retreaving system) on lyhenne automaattisesti toimivasta varastointijärjestelmästä.
BIOS	BIOS (Basic Input/Output System) on tietokoneen alhaisen tason ohjelmisto, joka toimii ensimmäisenä käyttöjärjestelmänä, kun tietokone käynnistetään.
HMI	HMI (Human-Machine Interface) tarkoittaa ihmisen ja koneen välistä rajapintaa.
HTML5	HTML5 on HTML (Hypertext Markup Language) -merkintäkielen uusi versio verkkosivujen ja käyttöliittymien suunnitteluun ja kehittämiseen.
JavaScript	Alun perin Netscapen kehittelemä pääosin verkkoympäristössä käytettävä ohjelmointikieli.
PLC	PLC (Programmable Logic Controller) on ohjelmoitava logiikkasäädin, jota käytetään teollisuuden automaation ja ohjausjärjestelmien ytimessä.

1 JOHDANTO

1.1 Työn taustaa

Työssä oli tarkoituksena kehittää pienoiskoon varastoautomaatiojärjestelmän uusi ohjaus käyttäen Beckhoff-logiikkaa ja ohjelmakirjastoja. Laitteen mekaniikka on valmistettu 2000-luvun alkupuolella ja siinä oli käytetty vuosien saatossa useita erilaisia ohjausjärjestelmiä. Tilajärjestelyiden vuoksi laite oli muutamia vuosia purettuna ja varastoituna myöhempää käyttöä varten. Laite koottiin uudelleen vuoden 2023 aikana käyttäen Beckhoff-logiikkaa. Nyt laiteelle oli tehtävä käyttöönotto ja liikkeenohjausta varten kehitettävä ohjelma. Työssä tilaajana toimi Seinäjoen ammattikorkeakoulu.

1.2 Työn tavoite

Työn tavoitteena oli kehittää Seinäjoen ammattikorkeakoululle toimiva pienoiskoon varastoautomaatiojärjestelmä, josta voitaisiin kehittää laboratoriotyö jonkin kurssitoteutuksen sisällöksi. Tarkoituksena oli tehdä varastoaseman fyysinen käyttöönotto, ohjausjärjestelmän ohjelmointi ja käyttöliittymän toteutus. Ohjelman tuli toteuttaa laitteelle manuaalisen käsiajon mahdollisuus ja automaattiajon ohjelma. Käsiajolla tuli pystyä ohjaamaan kaikkia laitteen liikkeitä ja toimintoja manuaalisesti käyttöliittymän kautta. Automaattiajon ohjelmalla voidaan viedä ja tuoda kappaleita ennalta määrätyille hyllypaikoille.

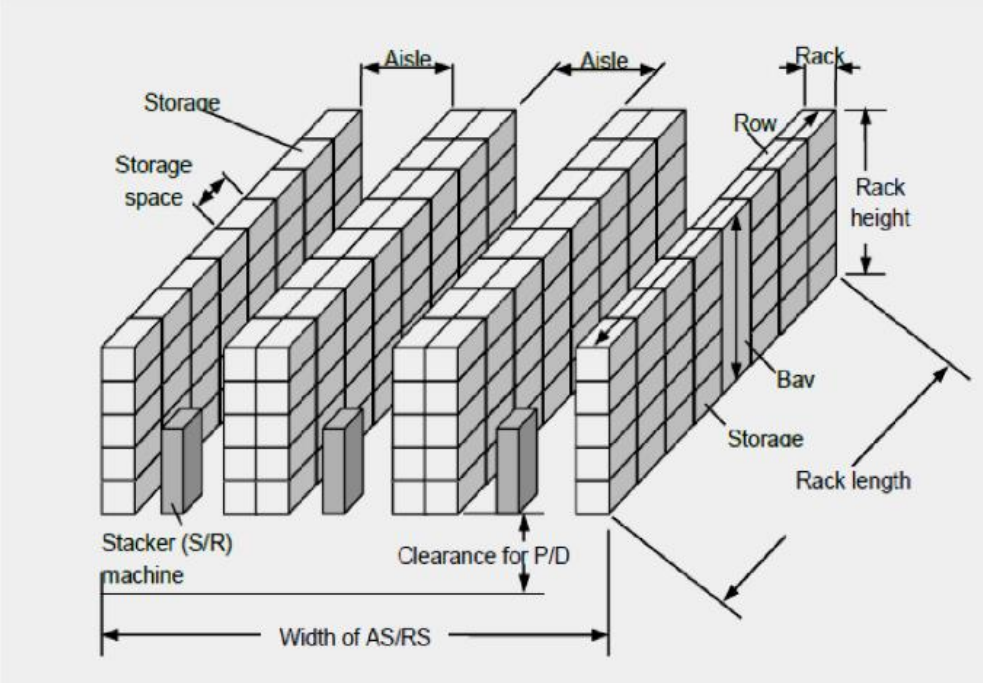
1.3 Työn rakenne

Työn rakenne koostuu teoriaosuudesta, käytännön toteutuksen osuudesta ja pohdinta-osuudesta. Ensin teoriaosuudessa kerrotaan yleisesti servomoottoreista ja EtherCAT:sta, lyhyesti TwinCAT 3 -ohjelmasta, HMI Engineering -työkalusta, HTML-kielestä, JavaScriptistä sekä Nielsenin heuristiikkasäännöistä. Laitteen käytännön toteutuksesta kerrotaan samassa järjestyksessä kuin ne ovat todellisuudessakin suunniteltu ja toteutettu aina laitteen turvaohjelmasta, automaattiohjelman valmistukseen. Lopuksi käydään läpi tuloksia ja pohditaan lopputulosta ja miten ne olisi voitu toteuttaa mahdollisesti paremmin.

2 TEORIA JA KÄYTETYT TEKNOLOGIAT

2.1 Varastoautomaatio

AS/RS-järjestelmät ovat pääasiallisesti teollisilla aloilla kehiteltyjä automaattisia varastointi- ja noutojärjestelmiä (Hameed ym., 2019, s. 1). Järjestelmissä sovellukset muuttuvat laajasti yksinkertaisista pienosavarastointi-/noutojärjestelmistä keskitettyihin järjestelmiin, joissa kokoonpano-, tuotanto- ja valmistusprosessit sijoittuvat niiden ympärille. Varastojärjestelmän valinta riippuu tilan koosta, tuotteiden painosta, varastointitoiminnan menetelmästä ja muista tekijöistä, jotka ovat kriittisiä automatisoitujen AS/RS-järjestelmien kehityksessä ja suunnittelussa. Kuviossa (kuvio 1) esitetään esimerkki AS/RS-järjestelmästä kokonaisuutena.



Kuvio 1. Esimerkki AS/RS-järjestelmästä (Hameed ym., 2019, s. 2).

Nykyaikaiset AS/RS-järjestelmät tarjoavat korkealuokkaista käyttövarmuutta monenlaisiin sovelluksiin (Kulwiec, i.a., s. 1). Ne ovat tärkeitä raaka-aineiden ja keskeneräisten tuotteiden välivarastoinnissa ja reitityksessä sekä varastonhallinnassa kaikenkokoisissa tiloissa.

2.2 Servomoottorit ja EtherCAT

Seuraavassa osiossa kerrotaan servomoottoreiden yleisestä toiminnasta verrattuna tavallisiin DC-moottoreihin. Osiossa käydään myös läpi Ethernetiin perustuvaa EtherCAT-protokollaa ja sen toimintaperiaatetta.

2.2.1 Servomoottorit

Grimmettin (2016, s. 102) mukaan servomoottorit ovat lähes samanlaisia kuin DC-moottorit, kuitenkin niissä on kriittisiä eroja. DC-moottori on yleensä suunniteltu pyörimään jatkuvasti 360 astetta annetulla nopeudella, kun taas servomoottorit on suunniteltu liikkumaan rajatun määrän astekulmia. Tiivistettynä, DC-moottoreiden tapauksessa halutaan yleensä, että moottorit pyörivät tasaisella nopeudella, joka voidaan säätää. Servomoottoreiden kohdalla taas tavoitteena on ohjata moottori tarkasti haluttuun asentoon.

2.2.2 EtherCAT

EtherCAT on Ethernetiin perustuva Beckhoffin kehittämä erittäin suorituskykyinen verkkoprotokolla (Advanced Motion Controls (AMC), i.a.). Ennen nopeita kenttäväyliä moottoreiden ja muiden laitteiden johdotukset oli vedettävä takaisin ohjaimelle. Perinteisessä moniakselisessa keskitetyssä ohjaimessa tämä saattoi tarkoittaa 50–100 terminaalia kaikille johdoille.

EtherCAT parantaa toimintatavallaan teollisen Ethernetin tunnettuja heikkouksia kuten tehoton väylän käyttö ja kytkinviive (Beckhoff, 2024). Tässä toimintatavassa yksi datakehys riittää usein siirtämään ohjaustiedot kaikkiin verkon solmuihin (Beckhoff, 2024). Ohjainlaite lähettää viestin, joka kulkee jokaisen liitetyn laitteen kautta. Kukin niin sanottu orjalaite käsittelee itselleen osoitetut tiedot ja lisää dataa kehykseen sen siirtyessä eteenpäin. Kehyksen viivästyminen johtuu laitteiston määritellyistä viiveajoista. Verkon tai haaran viimeinen solmu tunnistaa avoimen portin ja palauttaa datakehyksen tietoineen takaisin päälaitteelle.

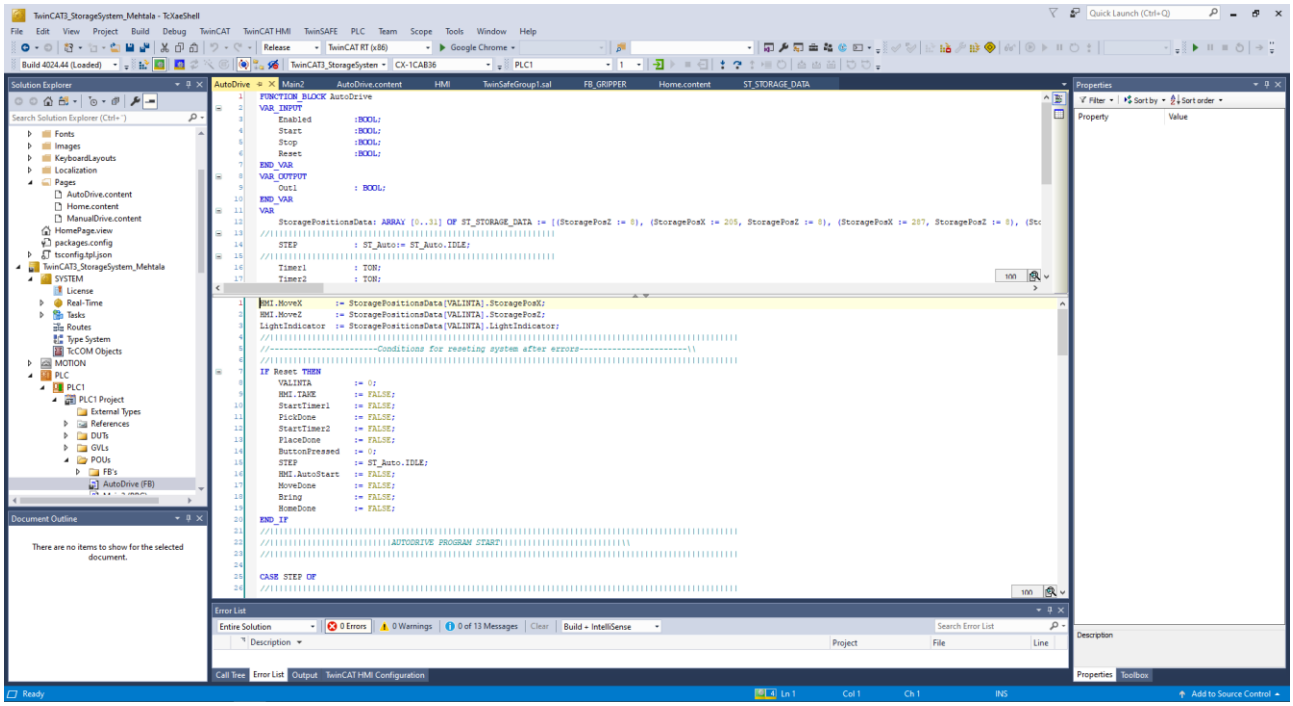
2.3 TwinCAT-ohjelmisto

Beckhoff kehitti ensimmäisen PC-pohjaisen koneen ohjaimen vuonna 1986 (Beckhoff, 2021, s. 2). Kymmenen vuotta ensimmäisestä mallista julkaistiin TwinCAT-ohjelmiston ensimmäinen versio, joka yhdisti kaikki automaation toiminnot yhdelle ohjelmistoalustalle. Tämä loi perustan PC-pohjaisten ohjainten teknologialle, johon yritykset automaatioalalla ympäri maailmaa luottavat.

TwinCAT on edelleen vakiintunut modulaarinen ohjelmistoalusta, jota käytetään toteuttamaan tehokkaita ja erittäin joustavia automaatioratkaisuja yli 30 eri teollisuusalalla ja yli 75 maassa ympäri maailmaa (Beckhoff, 2021, s. 2). Älykkään ohjauksen sisällyttäminen ohjelmistoon mahdollistaa joustavan suorittamisen PC-laitteistolla monenlaisissa suorituskykyluokissa.

Nykyaikaisten koneiden monimutkaisuuden hallitsemiseksi ja samalla turhan insinöörityön vähentämiseksi on suuntauksena käyttää modulaarisia ohjausohjelmistoja (Beckhoff, 2023, s. 8). Yksittäisiä toimintoja, kokoonpanoja tai laiteyksiköitä kutsutaan moduuleiksi. Näiden moduuleiden tulisi olla mahdollisimman itsenäisiä ja hierarkkisesti rakennettuja. Rakenteen tulisi olla toteutettu siten että alimmat moduulit ovat yksinkertaisimpia ja uudelleen käytettäviä elementtejä. Standardoitujen käyttöliittymien avulla ylimmät moduulit voidaan yhdistää monimutkaisemmiksi laiteyksiköiksi tai jopa kokonaisiksi toimiviksi laitteiksi. Ihanteellisesti yksittäiset moduulit voidaan ottaa käyttöön, laajentaa, skaalata ja käyttää uudelleen toisistaan riippumatta (mts.8).

2.3.1 Extended Automation Engineering XAE



Kuvio 2. TwinCAT3 -laajennus Visual Studiossa.

Yksi TwinCAT 3:n pääperiaatteista on ohjelmistosuunnittelun yksinkertaistaminen (Beckhoff, 2023, s. 8). Sen sijaan että rakennettaisiin omia erillisiä työkaluja, on hyödyllisempää integroida järjestelmä jo olemassa oleviin ohjelmistoympäristöihin. Kehitysalustanaan TwinCAT3 -ohjelmisto käyttää kuvion (kuvio 2) mukaista Microsoft Visual Studio® -ympäristöä. Ohjelman integroiminen Visual Studioon laajennuksena mahdollistaa käyttäjälle kestävän ja helposti laajennettavan alustan.

2.3.2 Extended Automation Runtime XAR

TwinCAT 3 Runtime tarjoaa reaaliaikaisen ympäristön, jossa TwinCAT-moduuleita voidaan ladata, suorittaa tai hallinnoida (Beckhoff, 2023, s. 9). Yksitäiset moduulit voidaan kehittää eri kääntäjillä, mikä mahdollistaa niiden riippumattoman ohjelmoinnin eri valmistajien tai kehittäjien toimesta. Lisäksi ei ole merkitystä, ovatko moduulit PLC-, NC-, CNC-tyyppisiä tai C-koodilla luotuja.

TwinCAT 3 Runtime -ympäristössä moduulit aktivoituvat syklisesti tehtävien kautta (Beckhoff, 2023, s.9). Yhdellä ohjaus-PC:llä voidaan pyörittää useita tehtäviä samanaikaisesti. Erilaiset moduulit (SPS, C/C++, MATLAB®) voivat kutsua toisiaan TwinCAT 3 -ympäristössä ja sovelluksen ohjelmistoarkkitehtuuri tarjoaa paljon mahdollisuuksia (Beckhoff, 2023, s. 9). Tämän ansiosta useita eri toiminallisuuksia sisältäviä moduuleita voidaan yhdistää yhdeksi kattavaksi ohjelmistoksi. Moduulien määrä, joita tehtävät voivat kutsua, on lähes rajaton. Jos koodin suoritus venyy liian pitkäksi, käyttäjä tulee kohtaamaan syklisiä ylityksiä. TwinCAT 3:ssa tehtävien määrä on teoriassa rajattu 65 000:een, mutta käytännössä se riippuu suorituslaitteen järjestelmäresursseista.

2.4 HMI Engineering

TwinCAT 3 HMI on tunnettuun Visual Studio -kehitysympäristöön integroitu työkalu (Beckhoff, 2017, s. 4). HTML5- ja JavaScript-pohjalta mahdollistetaan käyttäjälle alustasta riippumattomien ja nopeasti reagoivien käyttöliittymien kehittäminen ja suunnittelu. Näillä työkaluilla kehitetyt käyttöliittymät mukautuvat automaattisesti käytössä olevan näytön resoluutioon, suuntaan ja kokoon. Beckhoffin (2017, s. 6) mukaan TwinCAT HMI:ssä ”what-you-see-is-what-you-get” (WYSIWYG) -editori mahdollistaa graafisen käyttöliittymän konfiguroinnin ilman ohjelmointitaitoja.

2.5 HTML

Tässä osiossa esitellään HTML-kielen historiaa ja sen kehitystä HTML 1.0 -versiosta viimeisimpään HTML5-versioon. Osion tarkoituksena on alustaa lukijan tietämystä HTML5-kielestä, joka on työn käyttöliittymässä olennainen ja kriittinen tekijä.

2.5.1 Klassinen HTML

HTML ensimmäiset suunnitelmat ja toteutukset tehtiin vuosina 1990–1991 (Korpela, 2014, s. 28). Alkuvaiheessa kieli muuntui jatkuvasti selainten tekijöiden lisätessä omia ”tagejaan”, joista osa levisi muihinkin selaimiin, mutta jotkin jäivät selainkohtaisiksi (Korpela,

2014, s. 28). Korpelan (2014, s.28) mukaan kielen varhaisista vaiheista on jossain vaiheessa käytetty nimitystä ”HTML 1.0”. HTML-kieltä kehitettiin aluksi CERNissä ja myöhemmin vuonna 1995 laadittiin julkinen määrittely kielelle, spesifikaatio: IETF:n HTML 2.0. Tässä versiossa siitä jätettiin pois esimerkiksi ”section”-elementti, joka oli varhaisissa suunnitelmissa, mutta joka tuli selaimiin ja määrittelyihin vasta HTML5:ssä. Selainten tekijät jatkoivat laajennusten lisäämistä ja samaan aikaan kehiteltiin ajatusta HTML:n uudesta versiosta HTML 3.0, mutta uutta versiota ei koskaan julkistettu. Vuonna 1997 julkistettiin HTML 3.2, joka oli pääasiassa selaimiin valmiiksi tehtyjen laajennusten virallistamista. HTML 4.0 oli myös laajennusten virallistamista, mutta siinä oli joitakin teoreettisia uutuuksia, joita vähitellen lisättiin selaimiin (esimerkiksi ”object”-elementti). Vuonna 1999 siitä tehtiin hyvin lievästi muokattu versio HTML 4.01, joka on edelleenkin ’virallinen HTML’ (Korpela, 2014, s. 28).

HTML 3.2 -versiosta lähtien HTML:n määrittelyä hoiti webistä kiinnostuneiden yritysten ja laitosten muodostama organisaatio W3C (World Wide Web Consortium). Klassisen HTML:n kehitys lakkautettiin vuonna 1998.

2.5.2 XHTML

XHTML:n tausta on XML:ssä, joka on luonteeltaan yleinen merkkauskieli (Korpela, 2014, s. 28). Vuonna 1998 XML määriteltiin ensimmäisen kerran virallisesti ja XHTML vuonna 2000. XML:n yksinkertaisen rakenteen vuoksi HTML:n muuttaminen XML-pohjaiseksi, merkitsee rajoituksia ja hiukan pitempää merkkausta. Esimerkiksi HTML-tagista <input type=radio checked> tulee XHTML:ssä <input type=”radio” checked=”checked”/> (mts. 29). W3C kehitti HTML:ää XML-pohjaisena (XHTML 1.0), joka merkitsi myös HTML 4.01:n määrittelyä XML-pohjaisena. XHTML 1.1 laajensi aikaisempaa versiota hiukan, mutta lisäsi myös rajoituksia. XHTML 1.1 oli kuitenkin pääosin HTML 4.01 määriteltynä XML-pohjaisena ja moduloituna. XHTML 2.0 -nimellä suunniteltiin uutta versiota, jonka tarkoituksena oli kehittyä uudelta pohjalta lähteväksi versioksi, vapaana HTML:n sekavan historian tuottamista ongelmista. Tämä ei kuitenkaan ottanut tuulta siipiensä alle.

2.5.3 Uusi HTML

Useilla eri tahoilla ajateltiin HTML:n kehittämistä vanhalta pohjalta. XML:stä todettiin, että siitä ei ole verkkosivujen kieleksi HTML:n korvaajana (Korpela, 2014, s.29). Vuonna 2004 W3C:n järjestämässä kokouksessa ehdotettiin HTML:n laajentamista HTML 4:n ja selainten käytön perustalle (Korpela, 2014, s. 29). Enemmistö kuitenkin äänesti uuden version kehitystä uudelta pohjalta. Applen, Mozillan ja Operan työntekijöiden perustama WHATWG-yhteisö (Web Hypertext Application Technology Working Group) halusi kuitenkin kehittää HTML:ää vanhalta pohjalta. Yksi varhainen ajatus laajennuksista oli lomakkeiden toiminnallisuuden laajentaminen, Web Forms 2.0 ja toinen suunnitelma oli Web Applications 1.0. Myöhemmin ideat yhdistettiin ja tätä kutsuttiin nimityksellä HTML5.

2.5.4 HTML 5

HTML5 viittaa ns. HTML5-sovelluksiin, joissa sovelluksia toteutetaan JavaScriptillä selaimella tai tarkemmin selainmoottorilla (Korpela, 2014, s.29). Tässä HTML on tarpeellinen, sillä selain suorittaa JavaScript-koodia HTML-dokumenttiin liitettynä. Sovelluksissa ei kuitenkaan tarvita välttämättä HTML5:n mukana tulleita määrittelyjä, mutta HTML5-kielessä on paljon enemmän sovelluskehitystä tukevia rakenteita ja määrittelyjä verraten aikaisempiin versioihin.

2.6 JAVASCRIPT

Tämän osion tavoitteena on selvittää JavaScriptin historiaa lukijalle, jotta lukija voi ymmärtää työn käyttöliittymän tärkeitä osia, ja miten käyttöliittymä koostetaan käytettävään muotoon.

2.6.1 JavaScriptin historiaa

JavaScript on nykyään merkittävimpiä ohjelmointikieliä (Peltomäki, 2017, s. 1). Siitä julkaistiin ensimmäinen versio vuonna 1995 Netscape Navigator -selaimen yhteydessä. Kieli sai paljon kyseenalaista huomiota, koska erilaisia selaimia varten täytyi käytännössä luoda erikseen toteutuksia, jonka vuoksi osa luoduista sivuista ei toiminut kaikilla selaimilla

tarkoitetulla tavalla. ”Monille dynaaminen HTML muistutti kirosanaa ja esitettiin paljon mielipiteitä siitä, että suurin osa logiikasta täytyy kirjoittaa palvelinpuolelle” (Peltomäki, 2017, s. 1). JavaScript standardoitiin 1990-luvulla nimityksellä ECMAScript. 2000-luvulla tilanne teki muutosta. Asynkroniset pyynnöt selaimelta palvelimelle (Ajax-tekniikka) mahdollisti työasemaohjelmien tyyppiset käyttöliittymät. Apukirjastot, kuten jQuery, helpottivat selainten yhteensopivuutta, eikä enää tarvinnut luoda eri versioita useampia selaimia varten. ECMAScript-standardit antoivat kielelle monia tarpeellisia ja piirteitä, ja tämän vuoksi se sopi paremmin myös opetuskäyttöön ohjelmoinnin perustekursseille. HTML5 -version myötä JavaScript-kieli nousi entistä tärkeämpään asemaan, sillä lähes kaikki dynaamisuus vaatii JavaScript-ohjelmointi.

2.6.2 JavaScript nykyään

Nykyään JavaScript-kieltä voi käyttää myös palvelinpuolen toteuttamiseen (Peltomäki, 2017, s.1). Palvelinpuolelle JavaScript-ohjelmat rakennetaan Node.js-ympäristöä hyödyntäen. Se sisältää JavaScript-kielen tuen, joka on rakennettu Chromium-projektin ja Google Chromen käyttämän V8-selainmoottorin varaan.

2.7 Nielsenin heuristiset säännöt

Tässä osiossa selitetään Jacob Nielsenin 10 heuristiikan sääntöä, joiden avulla työn käyttöliittymää suunniteltiin ja kehitettiin käyttäjälle ystävällisemmäksi.

2.7.1 Järjestelmän tilan näkyvyys

Käyttöliittymän tulisi aina pitää käyttäjä informoituna tapahtuvista asioista, sopivan palautteen avulla sopivan ajan kuluessa (Nielsen, 2024). Tämä tarkoittaa, että käyttäjä tietää koko toimintaprosessin ajan, mikä laitteen tai prosessin tila on. Esimerkiksi, kun ajetaan liikeakselia positiiviseen suuntaan, käyttäjälle ilmoitetaan reaaliaikaista paikkatietoa käyttöliittymässä ja/tai painettavassa painikkeessa syttyy valo, joka indikoi painikkeen olevan aktiivinen.

2.7.2 Järjestelmän ja reaalimaailman yhtenevyys

Käyttöliittymän tulisi puhua käyttäjän kieltä (Nielsen, 2024). Järjestelmän tulisi käyttää sanoja, lauseita ja käsitteitä, jotka ovat tuttuja käyttäjälle. Tietoa on esitettävä luonnollisessa ja loogisessa järjestyksessä seuraten tosimaailman käytäntöjä. Vertauskuvien käyttö on myös hyvä tapa auttaa tapahtumien hahmottamisessa.

2.7.3 Käyttäjän hallinta ja vapaus

Käyttäjät yleensä suorittavat toimintoja vahingossa (Nielsen, 2024). Käyttäjälle on tärkeää osoittaa selkeät poistumisreitit vahinkojen varalta. Käyttäjällä on annettava kontrollin tunne.

2.7.4 Johdonmukaisuus ja standardit

Käyttäjien ei pitäisi joutua ihmettelemään tarkoittavatko eri sanat, tilanteet tai toimenpiteet samoja asioita (Nielsen, 2024). Jakobin lain mukaan suurin osa ihmisten ajasta kuluu sellaisten digitaalisten tuotteiden parissa, joita ei ole tuotettu itsenäisesti. Kokemukset toisista tuotteista luovat käyttäjille odotuksia tuotteen toimivuudesta. Johdonmukaisuuden puuttuminen voi lisätä käyttäjille kognitiivista kuormitusta, silla se pakottaa heitä oppimaan jotain uutta.

2.7.5 Virheiden ehkäisy

Hyvä virheen ilmaisu on tärkeää, mutta parhaiten suunnitellut tuotteet ennaltaehkäisevät ongelmia (Nielsen, 2024). On poistettava virhealttiit tilanteet tai tarkistettava ne ja tarjottava käyttäjille vahvistusvaihtoehto ennen kuin he sitoutuvat toimintaan. On olemassa kahta erilaista virhettä:

- Lipsahdus
- Virhe

Lipsahdukset ovat huolimattomuudesta johtuvia tahattomia virheitä, kun taas virheet ovat tietoisia virheitä, jotka pohjautuvat käyttäjän mentaalimallin ja suunnittelun ristiriitaisuuksiin (Nielsen, 2024).

2.7.6 Tunnistaminen muistamisen sijaan

Käyttäjän muistin rasittamista tulee minimoida tekemällä elementit, toiminnot ja optiot näkyviksi (Nielsen, 2024). Käyttäjän ei pitäisi joutua muistamaan tietoa käyttöliittymän osasta toiseen. Informaation tulisi olla mahdollisimman läpinäkyvää tai helposti löydettävissä, jos tarve niin vaatii. Ihmisillä on rajallisesti lyhytaikaista muistia. Käyttöliittymä, jossa on selkeä näkymä, vähentää kognitiivista taakkaa käyttäjältä.

2.7.7 Joustavuus ja tehokkuus

Kun oikotiet ovat piilossa noviisikäyttäjältä, nopeuttavat oikopolut ammattilaisen toimintaa, mutta käyttöliittymä palvelee silti molempia käyttäjiä (Nielsen, 2024). Käyttäjälle tulee antaa mahdollisuus räätälöidä usein käytettyjä toimintoja. Joustavat prosessit voi toteuttaa monella tavalla, joten käyttäjille voi tarjota mahdollisuus valita mieluisensa.

2.7.8 Esteettinen ja minimalistinen suunnittelu

Käyttöliittymän ei tulisi tarjota informaatiota, joka ei ole hyödyllistä tai usein tarvittua (Nielsen, 2024). Jokainen ylimääräinen tieto käyttöliittymässä kilpailee hyödyllisen tiedon kanssa ja vähentää hyödyllisen tiedon merkitystä. Tämä ei kuitenkaan tarkoita, että käyttöliittymästä täytyy valmistaa mielikuvitukseton ja yksiulotteinen. Suunnittelussa tulee pitää sisältö ja visuaalinen visio keskitettynä tärkeisiin asioihin.

2.7.9 Käyttäjiä autetaan tunnistamaan, diagnosoimaan ja toipumaan virheistä

Virheestä tulisi viestiä selkeällä kielellä, osoittaa virhe tarkasti ja rakentavasti ehdottaa ratkaisu (Nielsen, 2024). Virheilmoitusten tulisi olla näkyvillä visuaalisten tehosteiden ja toimintojen avulla, jotta käyttäjän on helppo huomata ne.

2.7.10 Ohjeet ja dokumentaatio

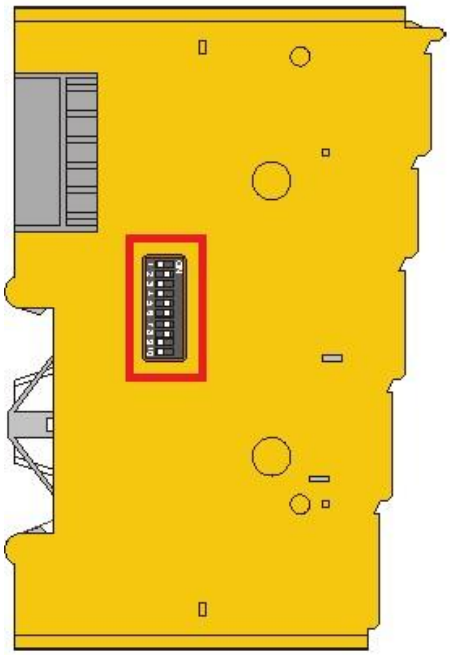
Parasta on, jos järjestelmä ei tarvitse enempiä selityksiä. Kuitenkin saattaa olla tarpeellista tarjota dokumentaatiota auttamaan käyttäjää ymmärtämään, kuinka toteuttaa tehtävät (Nielsen, 2024). Dokumentaation ja avun pitäisi olla helposti löydettävissä ja keskittyä käyttäjän tehtävään. Avun tulee olla tiivis ja listata vain tarpeelliset ja varmat askeleet tehtävän toteutumiseen.

2.8 TwinSAFE

Tässä osiossa esitellään työssä käytetyt turvakortit ja kerrotaan lukijalle mitä roolia kortit toimittavat. Työssä käytettiin vain kolmea turvakorttia laitteen koon ja vähäisen vaaran vuoksi.

2.8.1 Turvakortit

Laitteessa käytettiin neljäkanavaista EL1904-input-turvakorttia sekä nelikanavaista EL2904-output-turvakorttia. Näiden korttien signaalien ohjaamiseen käytettiin EL6900-turvakommunikkatiomoduulia. Laitteen turvapiirissä käytettiin kahdennetuilla kytkimillä varustettua turvakytkintä. Turvakytkimeltä johdot kytkettiin EL1904-turvakortin 1 ja 2 kanaviin, joista fyysiset kytkinsignaalit voidaan syöttää TwinCAT3-ohjelmalle virtuaalisiksi tulomuuttujiksi. EL2904-turvakortin ykköskanavalta lähtevät johdot vietiin AX5801-lisäkortille, joka turvakytkintä painettaessa kytkee servo-ohjaimessa pysäytystilan aktiiviseksi. Pysäytystila aktivoituu 100 millisekunnissa ja voidaan nollata pois vain turvakytkimen palauduttua perusasentoonsa ja hälytyksen manuaalisella poistolla käyttöpaneelin ”Reset”-painikkeella.



Kuvio 3. Turvakanavan osoitteen määrittäminen (Beckhoff, 2023, s. 41).

EL1904- ja EL2904-terminaalit mahdollistavat erilaisten turvasensoreiden ja -toimilaitteiden yhdistämisen. Niitä ohjataan EL6900 TwinSAFE -logiikka terminaalilla (Beckhoff, 2023, s. 18). Tämä logiikkaterminaali toimii yhdyskäytävänä TwinSAFE tulo- ja lähtöterminaalien välillä. Se mahdollistaa joustavan, helpon ja edullisen turvaohjausjärjestelmän konfiguroinnin. Jokaiselle TwinSAFE-turvaterminaalille on määritettävä kuvion (kuvio 3) osoittamasta paikasta terminaalille turvakanavan osoite fyysisillä DIP-kytkimillä. Näitä osoitteita on yhteensä 1023 erilaista, näitä voidaan määrittää kuvion (kuvio 4) mukaisesti.

DIP switch										Address
1	2	3	4	5	6	7	8	9	10	
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	1
OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
ON	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
OFF	OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	4
ON	OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	5
OFF	**ON**	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	6
ON	**ON**	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	OFF	7
...	...	...	...	...	...	...	...	...	...	...
ON	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	1023

⚠ WARNING

TwinSAFE address

Each TwinSAFE address may only be used once within a network / a configuration!
The address 0 is not a valid TwinSAFE address!

Kuvio 4. Turvaosoitteen määrittäminen (Beckhoff, 2023, s. 41).

Kuten kuviosta (kuvio 4) voidaan todeta, jokaista turvaosoitetta voidaan käyttää vain kerran.

2.8.1.1 EL1904

EL1904 TwinSAFE -komponentti on digitaalinen syöttöliityntä, joka on suunniteltu antureille, joissa on potentiaalivapaat kontaktit 24 V DC:n käyttöjännitteelle (Beckhoff, 2023b, s. 19). Moduulissa on neljä vikasietoista tuloa. Signaalit kulkevat FSoE:n (FailSafe over EtherCAT) kautta TwinSAFE-komponentille, jossa niitä voidaan vertailla. EL1904:ssä on turvallisuusparametreja, jotka mahdollistavat toiminnallisuuden mukauttamisen erityisesti kuhunkin turvallisuusorientoituneeseen vaatimukseen (mts.20). Turvallisuusparametrit siirretään syöttöliittimeen TwinSAFE-komponentilta, kun turvasovellus käynnistetään. Tämä yksinkertaistaa huoltotoimenpiteitä, koska komponentti voidaan yksinkertaisesti vaihtaa (Beckhoff, 2024).

Kuvio 5. EL1904-turvatulokortti (Beckhoff, 2023b, s.1).

Kuvion (kuvio 5) mukaiseen turvamoduuliin kytkettiin kahdennetun turvakytkimen johtimet moduulin I1- ja I3- sekä I2- ja I4-liittimiin, jotka muodostavat moduulin kanavat 1 ja 2. Liittimiin I5 ja I7 kytkettiin AX5801-lisäturvakortilta takaisinsyöttöä varten johtimet, joiden avulla voidaan välittää signaalia, jos servo-ohjaimella on jotain ongelmia.

2.8.1.2 EL2904

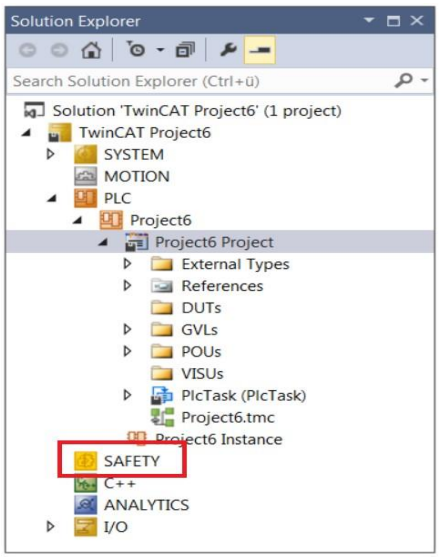
EL2904 on turvalähtöterminaali, jossa on digitaalisia lähtöjä toimilaitteisiin liittymistä varten, joilla on maksimissaan 0,5 ampeerin käyttövirta (Beckhoff, 2023a, s. 19). Yksinkertaisesti selitettynä EL2904 on samankaltainen turvakortti kuin EL1904, erottavana tekijänä on se, että EL2904 toimii signaalin lähettäjänä, kun taas EL1904 toimii signaalin vastaanottajana.

Kuvio 6.EL2904-turvalähtökortti (Beckhoff, 2023a, s. 1).

3 SOVELLUS

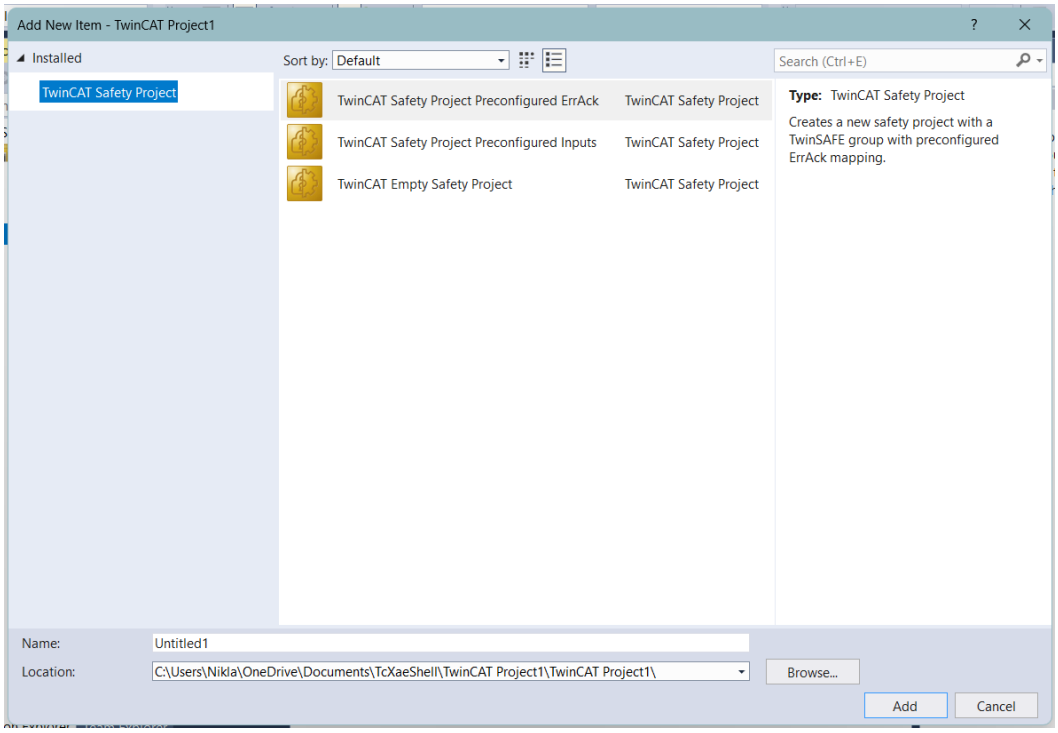
3.1 TwinSAFE-projektin luonti

Turvaohjelma luotiin TwinCAT-ohjelmassa kuvion (kuvio 7) osoittamasta ”SAFETY”-sarakkeesta. Sarakkeesta painamalla hiiren oikealla näppäimellä, avautuu valikko, josta voidaan lisätä uusi TwinSAFE-projekti valitsemalla optio ”Add new item”.



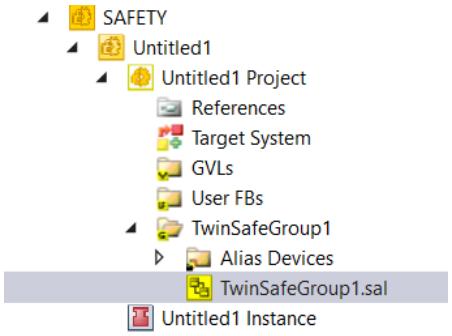
Kuvio 7. TwinCAT 3 solution explorer.

Kun optio on valittu, avautuu kuvion (kuvio 8) mukainen ikkuna, josta valitaan optio ”TwinCAT Safety Project Preconfigured ErrAck”. Kyseinen vaihtoehto luo uuden turvaprojektin, jossa on valmiiksi konfiguroidut esiasetukset. Käyttäjän ei tarvitse itse joka kerta kyseisiä asetuksia tehdä.



Kuvio 8. Uuden TwinCAT-turvaprojektin luonti.

Kun projekti on nimetty ja tallennettu käyttäjän valitsemaan tiedostosijaintiin, voidaan TwinCAT-ohjelman Solution explorer -valikosta löytää kuvion (kuvio 9) luotu TwinSAFE-projektipuu. Koska valittiin esimääritelty projekti, on projektipuuhun luotu valmiiksi ”TwinSafeGroup1.sal”-tiedosto, johon turvaohjelma ohjelmoidaan.

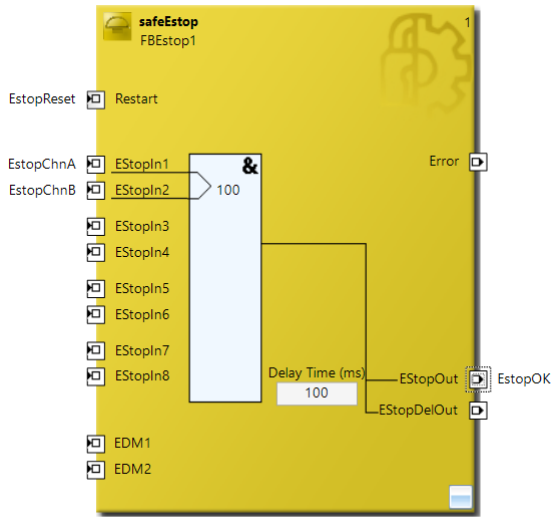


Kuvio 9. TwinSAFE-projektipuu

3.1.1 TwinSAFE ohjelmointi

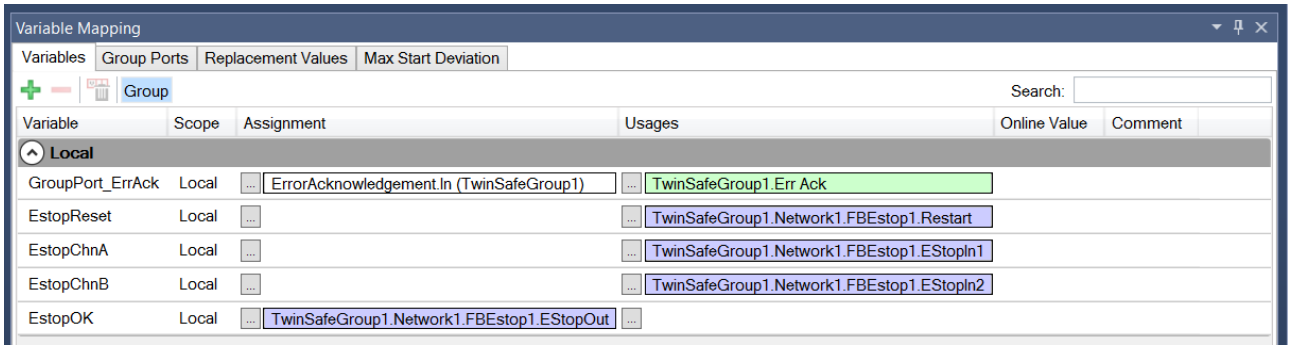
Turvaohjelman ohjelmointi aloitettiin selvittämällä kaikki tarvittavat fyysiset tulo- ja lähtötiedot turvakorteille, jotta voitiin olla varmoja, mitä turvalaitteita oli kytketty kullekin kanavalle. Kun kaikki tarvittavat tiedot olivat selvitetty, voitiin avata turvaohjelman editointityökalu ja

lisätä tarvittavat toimilohkot. Työssä käytettiin kahdennettua hätäseis-painiketta, joka tarkoittaa, että painikkeelle viedään kahdet parit johtoja kahdelle erilliselle rinnakkain toimivalle kytkimelle. Tällä voidaan lisätä järjestelmän toimivuutta. Rinnakkaisjärjestelmä vikaantuu vain, jos kaikki sen lohkot/kytkimet vikaantuvat.



Kuvio 10. SafeEstop-turvalohko.

Tässä työssä tarvittiin ainoastaan toimilohko hätäseis-painikkeen aktivointia varten. Kuvion (kuvio 10) mukainen SafeEstop-turvalohko lisättiin ohjelmaan ja EStopIn1- ja EStopIn2-tuloihin kytkettiin kahdennetun turvakytkimen digitaaliset tulomuuttujat. Näin ollen, kun hätäseis-painike painetaan pohjaan, molemmat kytkimet aukeavat ja molempien kanavien signaalit katkeavat, jolloin hätäseistila aktivoituu ja kaikki meneillään olevat prosessit keskeytyvät, kunnes hätäseis-piiri kuitataan. SafeEstop-toimilohkolle lisättiin myös digitaalinen nollauspainike, jolla hätäseis-tila voidaan kuitata oletetun vaaratilanteen jälkeen. EstopOut-lähtöön liitettiin lähtömuuttuja, jonka kautta hätäseis-signaali kulkee laitteen servo-ohjaimelle aktivoiden hätäseis-tilan. EL1904-tulokortin kolmannelle kanavalle kytkettiin myös takaisinkytkentäliittimet servo-ohjaimen AX5801-lisäturvakortilta, joka antaa tiedon PLC:lle, että tila on normalisoitunut.



Kuvio 11. Muuttujien liittäminen fyysisiin tulo- ja lähtöportteihin.

Kun turvaohjelma oli ohjelmoitu, täytyi vielä liittää digitaaliset ohjausmuuttujat fyysisiin tulo- ja lähtöportteihin. Tämä onnistui kuvion (kuvio 11) mukaisesta ”Variable Mapping”-ikkunasta. Kuviossa (kuvio 11) nähdään ohjelmaan liitetyt muuttujat sinisellä pohjalla, ne vastaavat fyysisiä tuloja tai lähtöjä, kun taas tyhjiin kohtiin liitetään digitaaliset ohjausmuuttujat ohjelmallista toimintaa varten. Muuttuja listassa tulo- ja lähtömuuttujat erotellaan sen mukaan, kummassa sarakkeessa muuttuja on. Jos muuttuja on ”Usages”-sarakkeessa, muuttuja on tällöin tulo, mutta jos muuttuja on ”Assignment”-sarakkeessa, on se silloin lähtömuuttuja. Kun muuttujat on liitetty oikeisiin tulo-/lähtöportteihin, voitiin ohjelmaa testata. Testausten ja pienten ongelmien jälkeen ohjelma voitiin todeta toimivaksi.

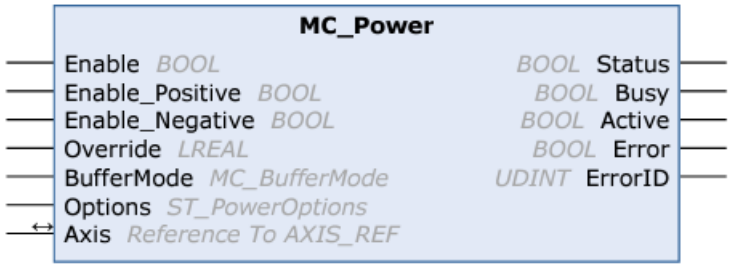
3.2 Pääohjelman lohkot

Pääohjelma toteutettiin FBD (Function Block Diagram) -lohkoilla, joiden avulla pystytään ohjaamaan fyysisen laitteen liikkeitä.

Käytettyjä toimilohkoja työssä olivat:

- Mc_Power
- Mc_MoveAbsolute
- Mc_Jog
- Mc_Reset
- Mc_Home.

3.2.1 Mc_Power

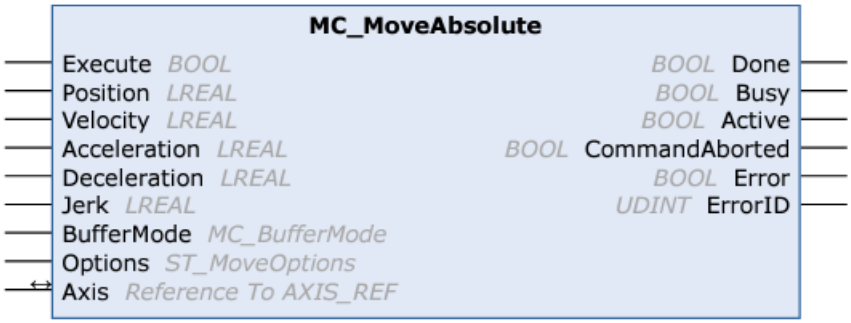


Kuvio 12. Mc_power-lohkon tulot ja lähdöt (Beckhoff, 2024, s. 17).

Toimilohkoa käytetään kytkemään akselille ohjelmallinen lupa (Beckhoff, 2024, s. 17). Lohkossa on kuvion (kuvio 12) mukaisia vasemmalla puolella olevia tuloja ja oikealla puolella olevia lähtöjä, jotka voidaan määrittää erillisten muuttujien ohjattaviksi. Lohkon sisään on merkitty, mitä mikäkin tulo tai lähtö merkitsee. Axis-tulo määrittää, mitä laiteen liikerataa ohjataan. Enable antaa kyseiselle liikeakselille luvan liikkua, ja Enable_Positive- sekä Enable_Negative-tulot määrittävät voiko liikettä suorittaa positiiviseen ja/tai negatiiviseen suuntaan. Override-tulo tarjoaa mahdollisuutta liittää laitteelle kiinteiden liikenopeuksien ns. ylikirjoittaminen. Override-tulon ollessa 100 tarkoittaa se sitä, että kiinteäksi parametroidut liikenopeudet ovat 100 prosenttisia ja sitä voidaan ohjata 0–100 prosentin välillä esimerkiksi fyysisellä potentiometrillä tai käyttöliittymän digitaalisella potentiometrillä. BufferMode on lisätulo, jolla voidaan lisätä toiminnallisuuksia lohkolle. Mc_Buffered-parametrilla voidaan määrittää, pysähtyykö liike heti kun laitteen lupa (Enable) liikkua loppuu vai suoritetaanko meneillään oleva toiminto loppuun ennen pysähdystä. Mc_Power-lohkoon ei todellisuudessa ole lisätty Options-tuloa, vaikka se toimilohkossa näkyykin.

Lohkon lähdöt kertovat erilaista dataa lohkon tilasta (Status). Busy-lähtö on totuusarvomuuttujatyyppinen lähtö, joka pysyy aktiivisena niin kauan kuin liikeakselilla on lupa liikkua. Active-lähtö kertoo, jos liiketoimintoja suoritetaan ja on aktiivisena vain liiketoiminnon ajan. Error-lähtö kertoo, jos lohkossa on virhetila tosi/epätosi-typpisesti, ja ErrorID antaa virhekoodin vikatilalle.

3.2.2 Mc_MoveAbsolute

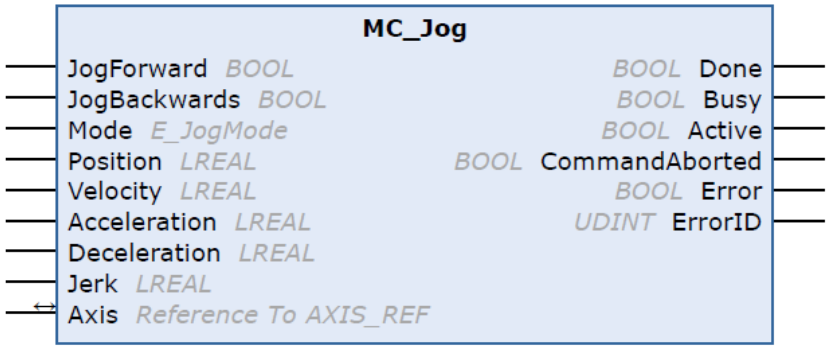


Kuvio 13. Mc_MoveAbsolute-lohkon tulot ja lähdöt (Beckhoff, 2024, s. 60).

MC_MoveAbsolute-toimintolohko aloittaa akselin paikoituksen määriteltyyn absoluuttipisteeseen ja tarkkailee akselin liikettä koko liiketoiminnon ajan (Beckhoff, 2024, s. 60). Tässäkin lohkossa tulee määrittää Axis-tulolle, mitä liikerataa halutaan ohjata. Kuvion (kuvio 13) mukaisilla lähdöillä ja tuloilla määritellään, miten liiketoiminnot toteutetaan. Execute-tulo on totuusarvomuuttujatyyppinen tulo, jolla liiketoiminto kytketään aktiiviseksi. Position-tulo on LREAL-tyyppinen tulo, jossa kerrotaan matkan etenemä absoluuttisesti nollapisteestä laskien. Velocity-tulo määrittää liiketoiminnon nopeuden ja sen aputuloina toimivat Acceleration- ja Deceleration-tulot, jotka määrittävät kiihtyvyyden (Acceleration) ja hidastuvuuden (Deceleration). Jerk-tulo määrittää, millä voimalla akseli lähetetään liikkeelle. Jos meneillään on jo jokin muu liiketoiminto, voidaan bufferMode-tulolla määrittää, toteutetaanko haluttu liiketoiminto vasta edellisen päätyttyä vai keskeytetäänkö meneillään oleva toiminto ja aloitetaan haluttu uusi liiketoiminto. Options-tulo on datastruktuuri, joka sisältää harvemmin käytettyjä parametrejä (mts. 61).

Lähdöt tässäkin lohkossa kertovat olennaisia tietoja lohkon tilasta. Done-lähtö kertoo, kun liiketoiminto kyseisellä akselilla on päättynyt. Busy-lähtö on aktiivinen niin kauan kun liikettä jatketaan. Active-lähtö indikoi, että liiketoiminto on suoritettu (Beckhoff, 2024, s. 61). CommandAborted-lähtö on aktiivinen, jos liiketoimintoa ei voitu suorittaa kokonaan (esimerkiksi hätäseis painettu). Error-lähtö kertoo, jos toimilohko on vikatilassa, ja ErrorID-lähtö antaa vialle vikakoodin.

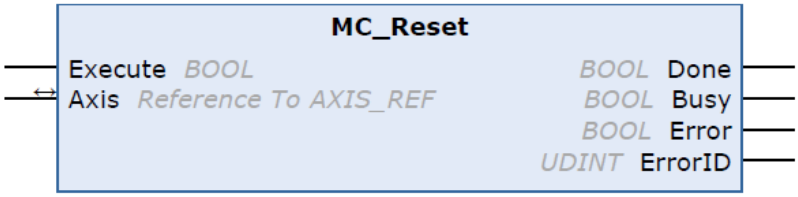
3.2.3 Mc_Jog



Kuvio 14. Mc_Jog-toimilohkon tulot ja lähdöt (Beckhoff, 2024, s. 93).

Mc_Jog-toimilohko mahdollistaa akselin liikuttamisen manuaalisilla painikkeilla (Beckhoff, 2024, s. 93). Kuvio (kuvio 14) kertoo, millaisia ja minkä nimisiä tuloja ja lähtöjä toimilohko sisältää. JogForward- ja JogBackwards-tulot ovat akselin positiivisen ja negatiivisen suunnan ohjausta varten. Mode-tulolle voidaan määritellä, miten liikettä toteutetaan (esimerkiksi jatkuva liike tai nykäyksittäin etenevä liike). Tässä lohkossa Position-tulo ei kerro liikettä nollapisteen perusteella, vaan matkan, joka liikutaan, kun käytetään nykäyksittäin etenevää liikettä (MC_JOGMODE_INCHING). Velocity määrittää liikenopeuden, Acceleration liikkeen kiihtyvyyden ja Deceleration liikkeen hidastuvuuden, sekä Jerk liikkeelle lähdön voiman. Kuten aikaisemmin selitetyissä lohkoissa, tulee tällekin lohkolle määrittää ohjattava liikerata. Tämän lohkon lähdöt ovat identtiset Mc_MoveAbsolute-toimilohkon lähtöihin.

3.2.4 Mc_Reset

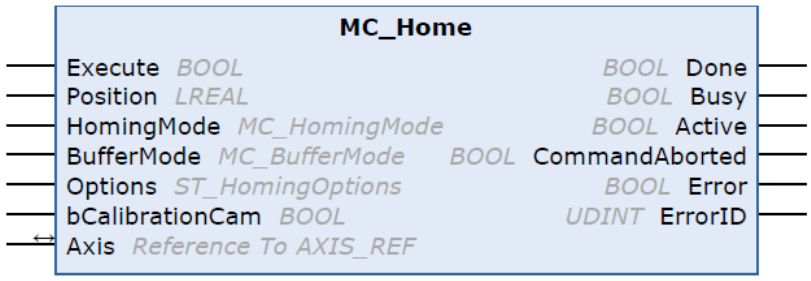


Kuvio 15. Mc_Reset-toimilohko lähtöineen ja tuloineen (Beckhoff, 2024, s. 18).

Tätä toimilohkoa käytetään resetoimaan NC-akseli (Beckhoff, 2024, s. 18). Kuviosta (kuvio 15) voidaan huomata, että tällä toimilohkolla ei tarvita useita parametrejä. Toimilohkolle määritetään Axis-tulolle resetoitava NC-akseli ja Execute-tulo toteuttaa resetoinnin.

Lähdöistä saadaan näytettyä tietoa toimilohkon tilasta. Done-tulo on aktiivinen, kun resetointi on valmis. Busy-tulo ilmaisee, että lohko on juuri käytössä. Error-tulo ilmaisee vikatilan ja ErrorID-tulo antaa vialle vikakoodin.

3.2.5 Mc_Home



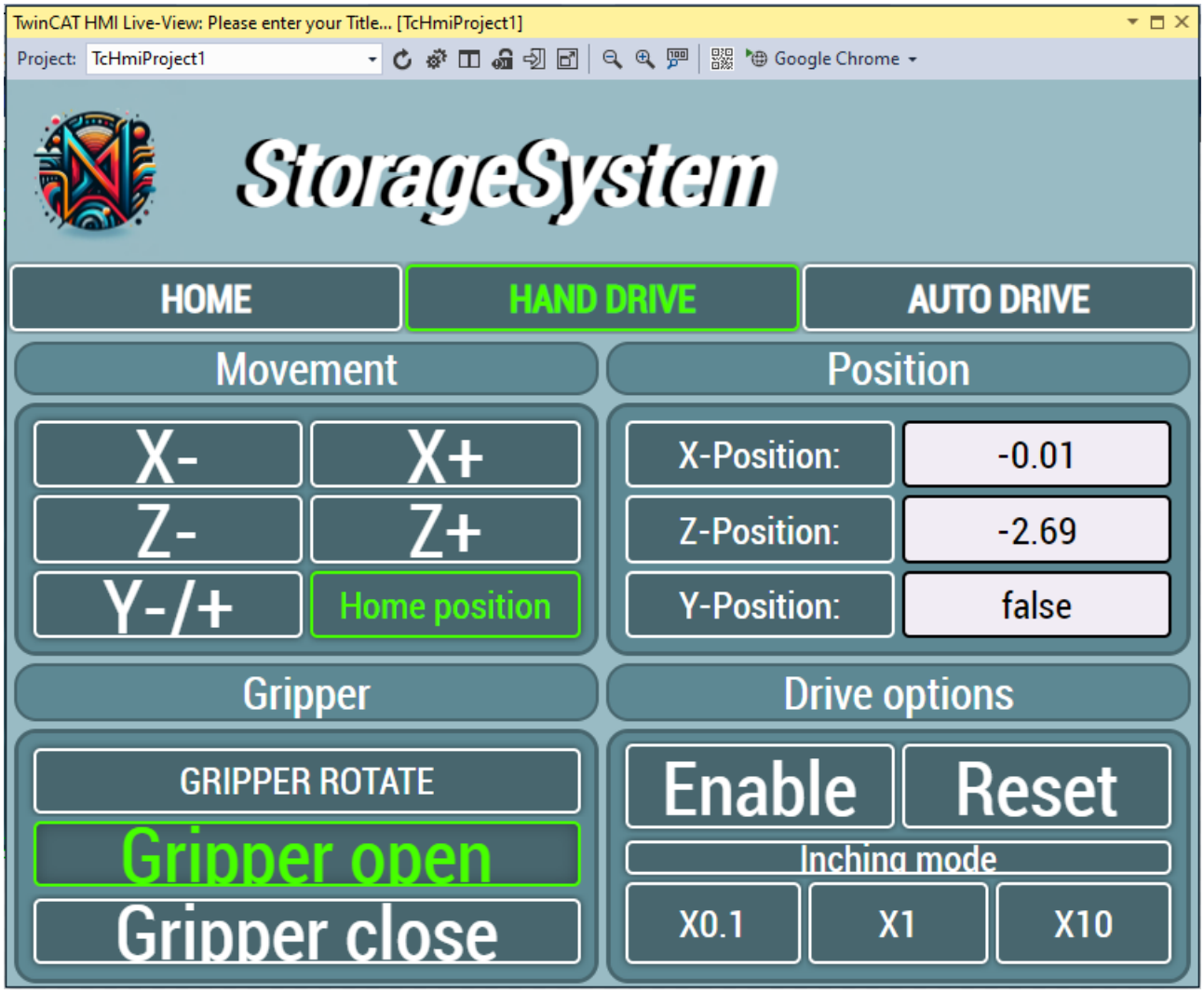
Kuvio 16. Mc_Home-toimilohko (Beckhoff, 2024, s. 91).

Akselin referenssiin ajo toteutetaan Mc_Home-toimilohkolla (Beckhoff, 2024, s. 91). Mc_Home on yksi niistä toimilohkoista, joille täytyy määrätä akseli, jota se ohjaa. Execute-tulo toteuttaa refenssiin ajon. Position-tulolle määrätään piste, johon referenssi asetetaan. Tämän jälkeen kaikki liikkeet mitoitetaan tästä aiemmin asetetusta pisteestä. Position-tulolle voidaan antaa myös DEFAULT_HOME_POSITION-optio. Tämä valinta on määritetty TwinCAT-ohjelman järjestelmämanagerissa. HomingMode-tulolla määritetään, miten referenssiin ajo toteutetaan. Tulolle syötetään MC_HomingMode-kirjaston MC_Default-Homing-, MC_Direct-, MC_ForceCalibration- tai MC_ResetCalibration-muuttuja. MC_DefaultHoming käynnistää oletusreferenssiin ajon, MC_Direct asettaa referenssipisteen suoraan ilman liiketoimintoja, MC_ForceCalibration pakottaa toimilohkon ”axis is calibrated”-tilaan ilman liiketoimintoja sekä referenssi pisteen asettamista. MC_ResetCalibration resetoi kalibrointitilan ilman liiketoimintoja sekä referenssipisteen asettamista. BufferMode ei ole Mc_home-toimilohkolle aktiivinen toiminto. Options-tulo on käytössä vain, jos MC_Direct-toimintoa käytetään. bCalibrationCam-tulolle voidaan määrittää kytkin, anturi tai muu vastaavanlainen laite, jolla voidaan antaa totuusarvomuuttuja-signaalia toimilohkolle. Kun Signaali aktivoituu tai sammuu, referenssipiste asetetaan pisteeseen, jossa laite fyysisesti sijaitsi signaalin aktivoituessa tai sammuessa. Toimilohkon lähdöt toimivat samalla tavalla kuin Mc_MoveAbsolute-toimilohkon lähdöt.

3.3 Käsiajo

Käsiajon ohjelma toteutettiin pääohjelman toimilohkoja käyttäen. Käsiajon liikekomennot toteutettiin MC_Jog-toimilohkoilla ja tarttujan varren liikettä ohjattiin totuusarvomuuttuja-tyyppisellä lähtömuuttujalla. Tarttujan kiinni ja auki -toiminnoille oli omat lähtömuuttujat, joilla tarttujaa ohjattiin. Tarttujan päätä voitiin myös halutessa kääntää 180 astetta omalla lähtömuuttujalla. Kaikki tarttujan toiminnot olivat paineilmalla ohjattuja. Käsiajon käyttöliittymään lisättiin painike liikeratojen referenssiin vientiä varten. Tämä toiminto toteutettiin MC_Home-toimilohkolla. Painiketta painaessa laite lähtee hitaasti viemään x-akselin kelkkaa negatiiviseen suuntaan, jossa induktiivinen anturi toteuttaa referenssipisteen nollauksen havaittuaan kelkan metallisen rungon. Servomoottorit olivat vanhoja ja jarruttomia, eikä työssä käytetty enkooderia. Tätä moottoreiden jarruttomuutta hyödynnettiin z-akselin nollauksessa. Kun servo-ohjaimesta suljetaan virta, z-akselin kelkka lähtee painonsa vuoksi laskemaan alaspäin kohti x-akselin kelkkaa, jonka mukana z-akseli kulkee. Seuraavan kerran, kun laite kytketään päälle, laite ei muista enää sijaintiaan ja asettaa kaikki nollat sijaitsemaansa paikkaan. Näin z-akselia ei tarvitse koskaan erikseen ajaa referenssiin. Käyttöliittymään lisättiin myös painikkeita nykäyksittäistä ajoa varten. Kun Inching-toiminto on aktiivinen, voidaan käyttöliittymästä määrittää, halutaanko liikkua kymmenesosa millimetriä, millimetri tai kymmenen millimetriä kerrallaan. Inching-toiminto saadaan aktiiviseksi MC_Jog-toimilohkon tuloista.

3.3.1 käsiajon käyttöliittymä



Kuvio 17. Käsiajon käyttöliittymä.

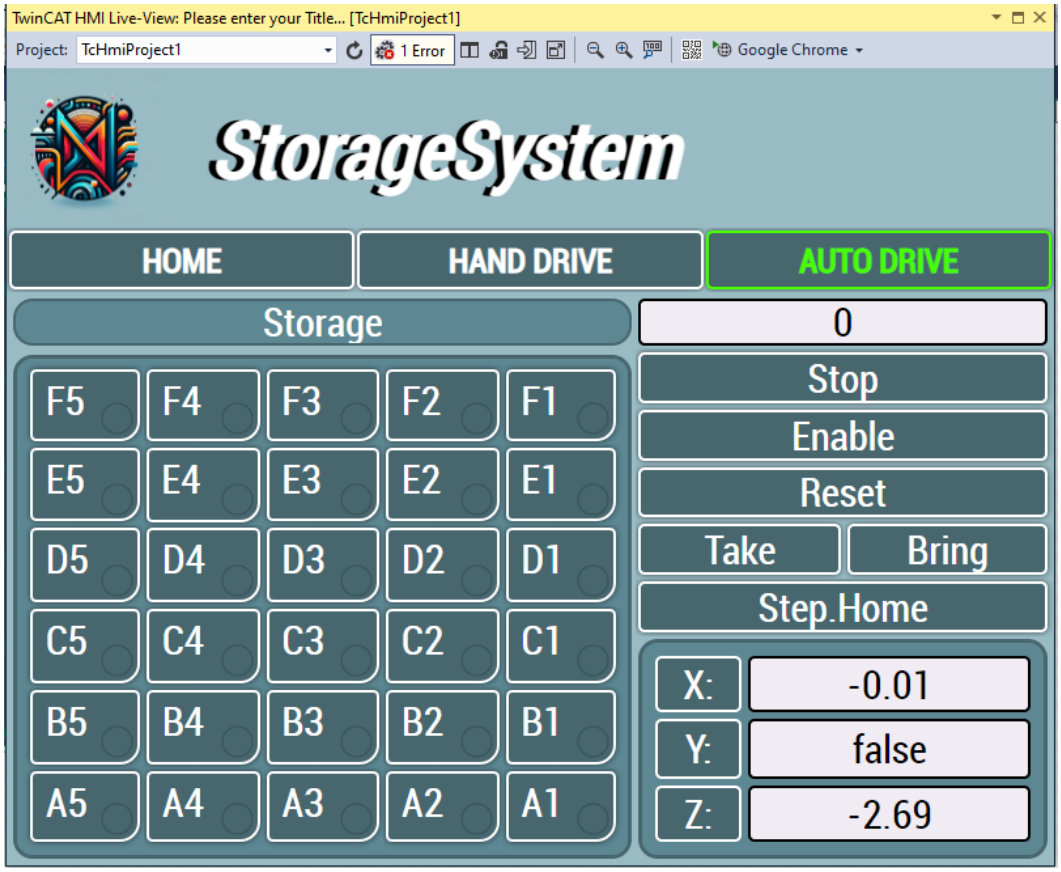
Käsiajoa varten käyttöliittymään tehtiin kuvion (kuvio 17) mukainen ”HAND DRIVE”-sivu. Sivulle lisättiin neljä eri toimintoaluetta: ”Movement”, ”Position”, ”Gripper” ja ”Drive options”. ”Movement”-lohkosta käyttäjä voi toteuttaa vapaata liikeohjausta laitteen X-, Y- ja Z-akseleille. ”Movement”-lohkosta käyttäjä voi myös suorittaa referenssiin ajon ”Home position”-painiketta painamalla. ”Position”-lohkosta käyttäjälle ilmaistaan laitteen reaaliaikaista sijaintia referenssipisteeseen nähden. ”Gripper”-lohkossa käyttäjä voi määrittää tarttujan kiinni tai auki, sekä pyörittää tarttujan päätä 180-astetta ympäri. ”Drive options”-lohkosta voidaan ohjata erilaisia optioita, kuten virhetilan resetointi ”Reset”-painikkeella tai servo-ohjaimen aktivointi/deaktivointi ”Enable”-painikkeella. Samassa lohkossa käyttäjä voi myös aktivoida nykäyksittäisen ajo tilan ”Inching mode”-painikkeella ja määrittää nykäyksen

koon aktivoimalla ”X0.1”-, ”X1”-, ”X10”-painikkeen. Riippuen aktivoidusta painikkeesta laite kulkee nykäyksittäin 0,1 millimetriä, 1 millimetri tai 10 millimetriä kerrallaan.

Käsiajon käyttöliittymää suunniteltaessa hyödynnettiin teoriaosiossa esiteltyjä Nielsenin heuristiikkasääntöjä (s. 17–19). Käyttöliittymästä yritettiin kehittää käyttäjälle informoiva, helppokäyttöinen ja selkokielisesti ymmärrettävä, jotta käyttäjällä on mahdollisimman vähän opeteltavaa. Käyttöliittymästä yritettiin jättää pois kaikki ylimääräinen toiminnallisuus, jotta käyttäjän ei tarvitsisi muistaa liikaa asioita ja toimintoja käyttöliittymästä. Nielsenin heuristiikkasääntöjä soveltaen käyttöliittymän esteettisyyteen käytettiin paljon aikaa.

3.4 Automaattiajo

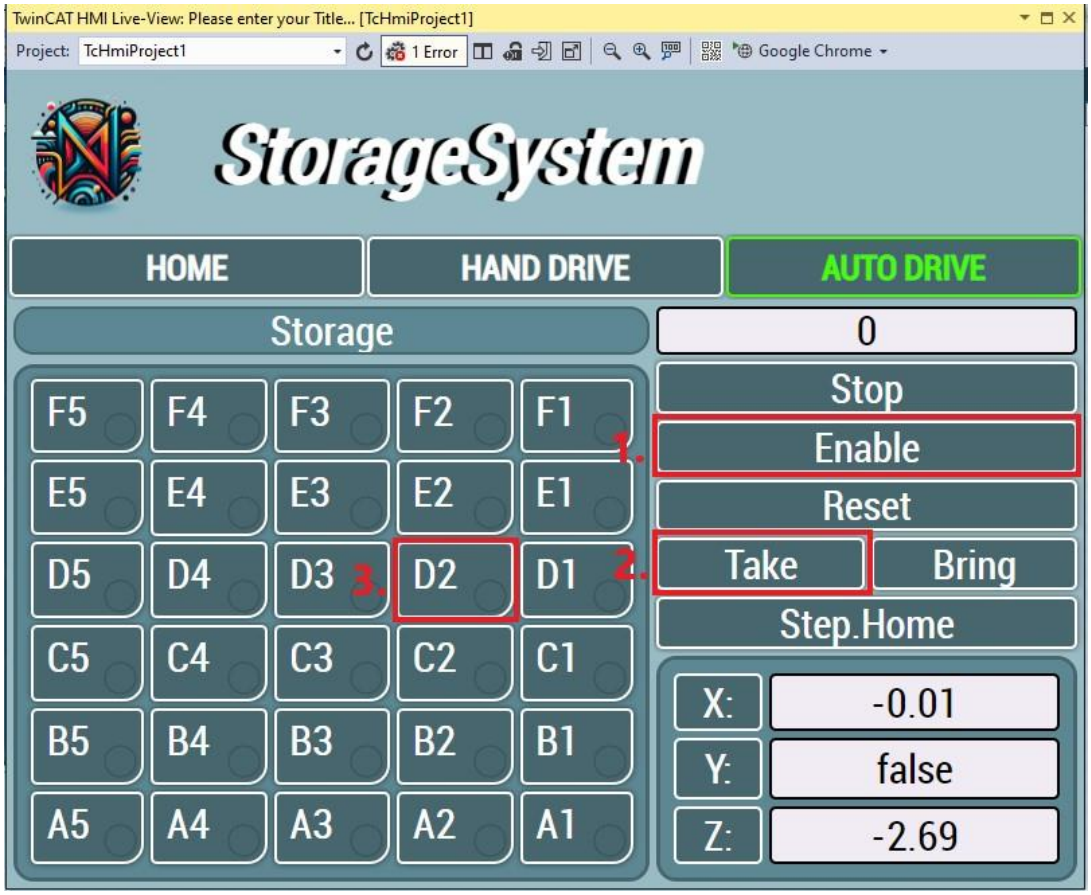
Automaattiajon ohjelma toteutettiin ”structured text”-kielellä ja se toimii käyttöliittymän kautta. Ohjelmassa voidaan määrittää, halutaanko hyllyyn viedä kappaleita vai halutaanko sieltä palauttaa kappaleita. Varastossa oli yhteensä kolmekymmentä hyllypaikkaa, jotka eroteltiin kerroksittain ja riveittäin. Joka kerroksessa oli yhteensä viisi paikkaa, numeroituna yhdestä viiteen, ja jokainen kerros oli merkitty aakkosilla alkaen alimmasta hyllystä. Kerroksia varastossa oli yhteensä kuusi kappaletta. Painettaessa jonkin hyllypaikan painiketta ohjelma hakee paikan koordinaatit listasta, jossa on jokaiselle hyllypaikalle x- ja z-koordinaatit sekä totuusarvomuuttujatieto merkkivalolle.



Kuvio 18. Automaattiohjelman käyttöliittymä.

Automaattiohjelmaa varten luotiin kuvion (kuvio 18) mukainen käyttöliittymä, josta käyttäjä voi ohjailla laitteen automaattiajoa.

3.4.1 Hyllyn täyttö

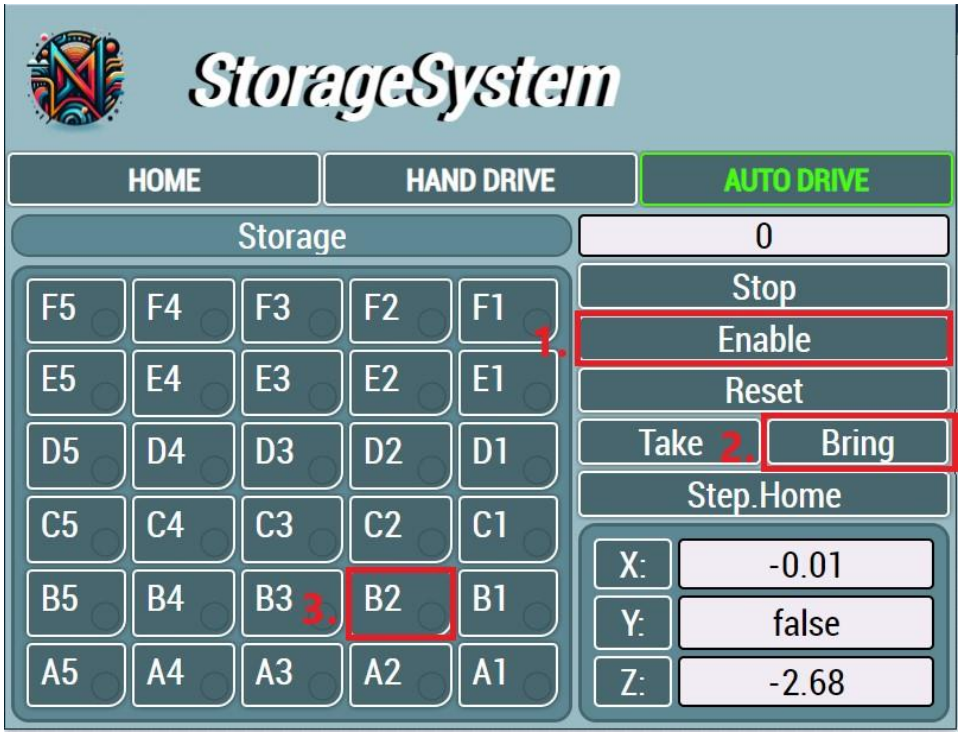


Kuvio 19. Automaattiohjelman vientitoiminto.

Automaattiohjelmassa hyllyn täyttö toimii kuvion (kuvio 19) osoittamassa järjestyksessä. Kun hyllyyn ollaan viemässä kappaleita, tulee laitteen servo-ohjain olla aktivoituna ja referenssiin ajo suoritettuna. Kun servo-ohjain on aktivoituna, ilmoitetaan se muuttamalla ”Enable”-painikeen valkoiset osat vihreäksi. Referenssiin ajo voidaan suorittaa ”HAND DRIVE”-sivulta ”Home position”-painikkeella (kuvio 18). Kun servo-ohjaimella on lupa toimia, aktivoidaan seuraavaksi ”Take”-painike, joka täyttää yhden ehdon kappaleen hyllyyn vientiä varten. Viimeisenä valitaan vapaa hyllypaikka ja aktivoidaan hyllypaikan painike. Kun kaikki kolme ehtoa on täyttynyt, ohjelma alkaa toteuttamaan toimintoja CASE-tyyppisen ohjelmarakenteen mukaisesti. Ohjelmassa on määritetty ehto, joka estää hyllyn ylitäyttämisen. Kun jokin hyllypaikka on täytetty, ei siihen paikkaan voida enää viedä kappaletta ennen kuin kyseisestä paikasta on haettu edellinen kappale pois. Jos hyllypaikassa on jo kappale, informoidaan se käyttöliittymässä hyllypaikan painikkeen merkkivalolla, joka muuttuu vihreäksi, kun hyllyssä on kappale.

3.4.2 Hyllyn tyhjennys

Kun hyllyssä on kappaleita, se ilmoitetaan vihreällä merkkivalolla kyseisen hyllypaikan painikkeessa. Kun merkkivalo palaa, ei sinne voida silloin viedä kappaleita ennen kuin paikka on tyhjä. Kun hyllystä haetaan kappaleita, tulee kuvion (kuvio 20) korostamien painikkeiden olla aktiivisia. Ensimmäisenä ”Enable”-painikkeella annetaan servo-ohjaimelle signaali käynnistyä ja valmistautua toteuttamaan liikkeitä. Jos ”Enable”-painike ei ole aktiivisena, se tulee aktivoida ja suorittaa referenssiin ajo tarkan paikoituksen saavuttamiseksi. Toisena valitaan ”Bring”-painike, kun halutaan palauttaa hyllystä kappaleita. Kolmantena halutaan valita hyllypaikka, jonka merkkivalo palaa vihreänä. Kun nämä ehdot täyttyvät, laite alkaa toteuttamaan laitteen CASE-tyyppistä ohjelmarakennetta tietyssä järjestyksessä



Kuvio 20. Automaattiohjelman hakutoiminto.

4 TULOKSET JA POHDINTAA

4.1 Työssä ilmenneitä haasteita

Tässä osiossa esitetään työn aikana ilmenneitä haasteita ja niihin löytyneitä ratkaisuja

4.1.1 Ongelma 1

Ensimmäisenä ongelmaksi muodostui, että laboratoriotilan A140.1 tietokoneilla ei ollut automaattisesti asennettuna TwinCAT3-ohjelmaa, jolla työn ohjelmointiosuus toteutetaan. Tähän ratkaisu saatiin ottamalla yhteys IT-tukeen, josta asennus suoritettiin noin viikon kuluessa.

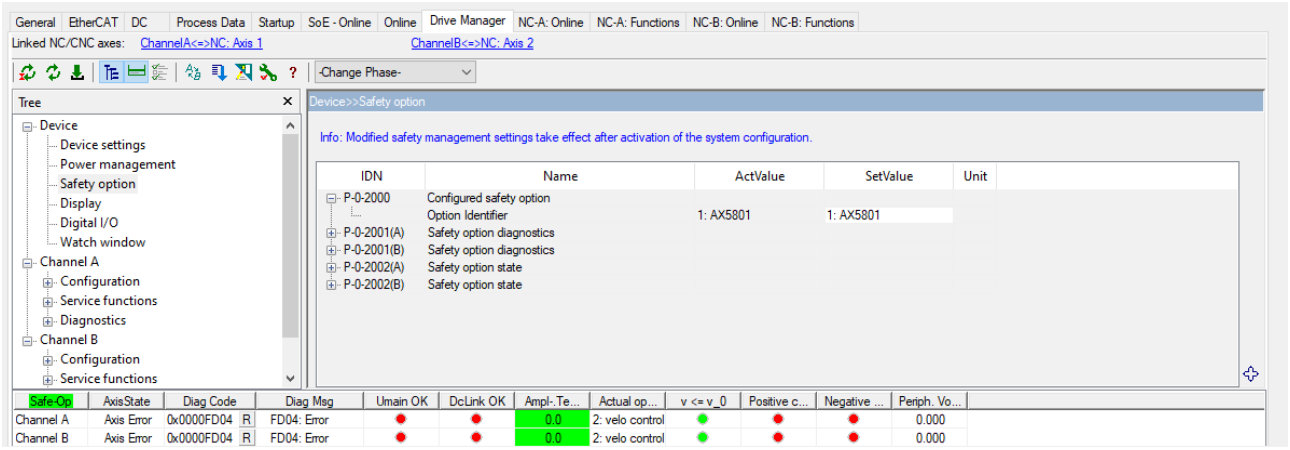
4.1.2 Ongelma 2

Seuraavaksi ongelmana olivat kuviossa (kuvio 21) esitetyt virheilmoitukset. Nämä virheilmoitukset ilmenivät työssä käytettävän Beckhoff CX5120 -sarjan servo-ohjaimen liikeratojen testausten alussa. Laitetta ohjelmoitiin TwinCAT3-ohjelmalla, jossa laitteen fyysistä toimivuutta voidaan testata ”Motion”-osiossa online-ohjauksella kaikilla akseleilla erikseen. Tämä kuitenkin vaatii ohjelmalta sen, että ohjelma ei havaitse yhtään virheilmoituksia online-tilaan siirryttäessä. Virheilmoituksiin ei verkosta löytynyt hyödyllistä apua, joten Beckhoff tekniseen tukeen oltiin yhteydessä.

1. Error 5.3.2024 17.14.02 314 ms | (65535): 'Drive 9 (AX5201-0000-0011)' (1008): SoE (Chn A) - Emergency (Hex: fdd4, 00, 'f6 00 00 00 00').
2. Error 5.3.2024 17.14.02 318 ms | (65535): 'Drive 9 (AX5201-0000-0011)' (1008): 'PREOP to SAFEOP' failed! Error: 'check device state for SAFEOP'. AL Status '0x0012' read and '0x0004' expected. AL Status Code '0x0045 - MBX SoE'
3. Error 5.3.2024 17.14.02 318 ms | (65535): 'Drive 9 (AX5201-0000-0011)' (1008): state change aborted (requested 'SAFEOP', back to 'PREOP').

Kuvio 21. Servo-ohjaimen virheilmoitukset.

Kuvion (kuvio 21) osoittamiin kahteen ensimmäiseen virheilmoitukseen saatiin osaratkaisu Beckhoffin teknisestä tuesta. Servo-ohjaimessa käytettiin AX5801-lisäkorttia, joka tuli kuvion 2 mukaisesti parametrisoida TwinCAT3-ohjelmassa servo-ohjaimen ”Drive Manager”-sivulta aktiiviseksi (kuvio 22).



Kuvio 22. AX5801-kortin parametrisointi ohjaimen drive manageriin.

Kun lisäinformaatiota odotettiin teknisestä tuesta, aloitettiin työssä käytettävän HMI-paneelin asennus. Paneelille täytyi johdottaa uusi virtakaapeli, joka saatiin Seinäjoen ammattikorkeakoulun sähkötekniikan laboratoriosta A130.1. Näyttöpäätteen adapterille oli alkuperäisesti kytketty sininen miinusjohto ja punainen plusjohto 24 voltin jännitesyöttöä varten, mutta johdot olivat liian lyhyet. Uusi kaapeli oli 3-johtimista kumikaapelia, jossa oli ruskean värinen plusjohdin, sinisen värinen miinusjohdin ja keltavihreä maadoitusjohdin.

Myöhemmin Beckhoff Automation otti yhteyttä ja hetken kommunikoinnin jälkeen teknisestä tuesta lähetettiin asiantuntija katsomaan, mikä laitteessa voisi olla vikana. Asiantuntija saapuessa muutamia päiviä myöhemmin löysi hän muutamia kriittisiä ongelmia, joita ei ollut huomattu aikaisemmin. Ensimmäiseksi asiantuntija huomautti, että servomoottorit ovat vanhoja ja tämän vuoksi ne täytyin käydä parametroimassa manuaalisesti TwinCAT-ohjelmaan. Tämän jälkeen laitteen liikeradat toimivat, mutta ne liikkuivat käänteisessä suunnassa, jolloin ne täytyi invertoida, jolloin liikkeet toimivat halutun suuntaisesti. Toisena asiantuntija huomasi tarkastaessaan kytkentöjä, että servo-ohjaimen 24 voltin jännitesyötölle syötettiin yksi punainen +24 voltin johdin ja yksi sininen maadoitusjohdin. Ongelma oli sivutettu aikaisemmin, sillä oli luultu, että laitteelle on johdotettu punainen plus johdin ja sininen miinus johdin, niin kuin normaalissa DC-sähköpiirissä. Kuitenkin jos maadoitusjohdin olisi kytketty standardoidulla keltavihreällä johtimella, olisi voitu huomata, että kytkennöissä oli jotain vikaa ja asiasta olisi otettu selvää. Teknisentuen asiantuntija esitti servo-ohjaimen ohjekirjasta, että laite tarvitsee toimiakseen kaksi +24 voltin syöttökaapelia ja yhden maadoituskaapelin. Näiden muutosten ja korjausten jälkeen laite saatiin operoitavaan kuntoon ja voitiin aloittaa ohjelmien kehitys.

4.1.3 Ongelma 3

Kun johdotukset oli tehty ja kytketty oikein sähkökeskukseen, ajateltiin että HMI-paneelissa olevaa integroitua PC-järjestelmää voitaisiin käyttää laitteen ohjelmalliseen ohjaukseen, jolloin sähkökeskuksessa oleva CX5120-sarjan PLC voitaisiin jättää pois ja korvata EK1100-kenttäväylä moduulilla. Kun komponentit oli vaihdettu, ei laitetta kuitenkaan saatu yhdistettyä. Tässä vaiheessa ilmeni työn kolmas ongelma. CP6201-sarjan HMI-paneelissa käytettiin vanhaa ”Windows CE 6.0” -järjestelmää, joka ei tue uudempaa TwinCAT3-ohjelmaa vaan olisi vaatinut vanhemman TwinCAT2-ohjelman käyttöä. Tämä ei lähtökohtaisesti ollut vaihtoehto, joten päädyttiin tilanteeseen, jossa aikaisemmin mainittu CX5120-sarjan PLC oli kytkettävä takaisin käyttöön. Kuitenkin PLC:n asennuksen jälkeen huomattiin, että laite käynnistyy suoraan BIOS:n eikä Windowsiin niin kuin normaalisti kuuluisi. Ongelmaksi havaittiin, että komponentti vaihdoksissa PLC:n muistikortti oli korruptoitunut. Ratkaisu tähän ongelmaan oli vaihtaa toisesta samanlaisesta PLC:stä muistikortti.

4.2 Tuloksia

Työlle määritettyyn tavoitteeseen päästiin ja aikaan saatiin toimiva varastojärjestelmä, jolla voidaan viedä tai tuoda kappaleita käyttäjän määräämille hyllypaikoille. Heti työn aloituksesta lähtien työssä ilmeni haasteita, jotka yksittäin vaikuttivat pieniltä, mutta yhdessä niistä koostui kokonaisuus haasteita, jotka estivät työn etenemistä suuntaan tai toiseen. Nämä haasteet saatiin kuitenkin ratkaistua ja lopullinen tulos on tekijän mielestä tyydyttävästi toteutettu.

Työssä oli aiemmin aikomuksena, että laitteelle kehitetty käyttöliittymä lisättäisiin fyysiselle käyttöliittymäpaneelille, mutta työn loppuvaiheilla havaittiin ohjelmistojen versio-ongelma. Ohjelmointi oli aloitettu TwinCAT3 -ohjelman 1.12-versiolla ja laiteen ohjaimessa oli TwinCAT3 -ohjelman 1.10-versio. Microsoftin parametrien vuoksi version niin sanottu ”huonontaminen” ei ole mahdollista, mikä olisi tarkoittanut, että ohjelmat olisi täytynyt ohjelmoida uudelleen vanhemmalla versiolla tai ohjaimelle olisi voitu ladata verkosta uudempi versio TwinCAT 3 XAR:sta. Ohjain ei välttämättä tue enää 1.10 jälkeen julkaistuja versioita, koska 1.10 on viimeinen versio, joka tukee ”Windows 7 CE” -käyttöjärjestelmää. Ongelmasta todettiin, että sitä voisi tarjota projektityönä jollekin projektikurssin ryhmälle

tulevaisuudessa. Tällä tavalla käyttöliittymä voitaisiin tulevaisuudessa liittää käyttöliittymä-paneelille.

4.3 Pohdintaa

Käyttöliittymää suunniteltaessa olisi voitu soveltaa enemmän Nielsenin heuristiikkaa ja käyttöliittymätyökalun toiminnoista olisi voinut ottaa enemmän selvää, että lopputulos olisi voinut olla hiukan yksinkertaisempi. Lopputuloksesta kuitenkin tekijän mielestä on saatu riittävän yksinkertainen, että TwinCAT-ohjelmistoon vähänkin perehtynyt voi ymmärtää, mitä käyttöliittymän toiminnoissa tapahtuu. Itse automaattiohjelman ohjelmointi olisi myös voitu toteuttaa monella tavalla, joista moni olisi voinut olla tehokkaampi toteutustapa, mutta tekijän tapa ohjelmoida oli toimivin hänelle itselleen.

LÄHTEET

Advanced Motion Controls (AMC). (i.a.). *EtherCAT Servo Drives.* Google scholar. https://www.a-m-c.com/experience/technologies/network-communcation/ethercat/

Beckhoff. (22.9.2023a). *Operating instructions for EL2904.* https://www.beckhoff.com/en-en/support/download-finder/search-result/?download_group=35673111&download_item=35676178

Beckhoff. (22.9.2023b). *Operating instructions for EL1904.* https://www.beckhoff.com/fi-fi/support/download-finder/search-result/?download_group=35673106&download_item=35675115

Beckhoff. (11.12.2023). *TwinCAT 3: Product overview*. https://download.beckhoff.com/download/document/automation/twincat3/Product_overview_EN.pdf

Beckhoff. (2024). *EtherCAT – the Ethernet Fieldbus.* https://www.beckhoff.com/fi-fi/products/i-o/ethercat/

Beckhoff. (2017). *TwinCAT HMI: Responsive and platform-independent.* https://www.beckhoff.com/en-en/support/download-finder/search-result/?download_group=61372263&download_item=61372272

Beckhoff. (2021). *TwinCAT 3: The flexible software solution for PC-based control.* https://www.beckhoff.com/en-en/support/download-finder/search-result/?download_group=29234383&download_item=29269233

Beckhoff. (4.3.2024). *TE1000 TwinCAT 3 PLC Library: Tc2_MC2.* https://download.beckhoff.com/download/Document/automation/twincat3/TwinCAT_3_PLC_Lib_Tc2_MC2_EN.pdf

Grimmet, R. (2016). *Getting Started with Raspberry Pi Zero*. Packt Publishing.

Hameed, H. M., Al Amry, K. A. & Rashid, A. T. (16.11.2019). *The Automatic Storage and Retrieval System: An Overview*. International Journal of Computer Applications (0975 – 8887). Volume 177. Google Scholar. https://faculty.uobasrah.edu.iq/uploads/publications/1611233629.pdf

Korpela, J. (2014). *HTML5-käsikirja.* Docendo.

Kulwiec, R. (i.a.). *RELIABILITY OF AUTOMATED STORAGE AND RETRIEVAL SYSTEMS (AS/RS) A WHITE PAPER.* Google Scholar. http://www.pnkreis.com/images/column_1448348195/asrswhitepaper3%20mhi.pdf

Nielsen, J. (24.4.2024). *10 usability heuristics for user interface design*. https://www.nngroup.com/articles/ten-usability-heuristics/

Peltomäki, J. (2017). *JavaScript-kieli: Uudet ominaisuudet.* BoD – Book on Demand.

LIITTEET

Liite 1. TwinCAT 3 Autodrive -ohjelmakoodi

Liite 1. TwinCAT 3 Autodrive-ohjelmakoodi

```
FUNCTION_BLOCK AutoDrive

VAR_INPUT

        Enabled: BOOL;

        Start: BOOL;

        Stop: BOOL;

        Reset: BOOL;

END_VAR

VAR_OUTPUT

        Out1: BOOL;

END_VAR

VAR

        StoragePositionsData: ARRAY [0..31] OF ST_STORAGE_DATA := [(StoragePosZ := 8), (StoragePosX := 205, StoragePosZ := 8), (StoragePosX := 287, StoragePosZ := 8), (StoragePosX := 370, StoragePosZ := 8), (StoragePosX := 451, StoragePosZ := 8), (StoragePosX := 535, StoragePosZ := 8), (StoragePosX := 205, StoragePosZ := 93), (StoragePosX := 287, StoragePosZ := 93), (StoragePosX := 370, StoragePosZ := 93), (StoragePosX := 451, StoragePosZ := 93), (StoragePosX := 535, StoragePosZ := 93), (StoragePosX := 205, StoragePosZ := 180), (StoragePosX := 287, StoragePosZ := 180), (StoragePosX := 370, StoragePosZ := 180), (StoragePosX := 451, StoragePosZ := 180), (StoragePosX := 535, StoragePosZ := 180), (StoragePosX := 205, StoragePosZ := 266), (StoragePosX := 287, StoragePosZ := 266), (StoragePosX := 370,
```

StoragePosZ := 266), (StoragePosX := 451, StoragePosZ := 266), (StoragePosX := 535, StoragePosZ := 266), (StoragePosX := 205, StoragePosZ := 353), (StoragePosX := 287, StoragePosZ := 353), (StoragePosX := 370, StoragePosZ := 353), (StoragePosX := 451, StoragePosZ := 353), (StoragePosX := 535, StoragePosZ := 353), (StoragePosX := 205, StoragePosZ := 428), (StoragePosX := 287, StoragePosZ := 428), (StoragePosX := 370, StoragePosZ := 428), (StoragePosX := 451, StoragePosZ := 428), (StoragePosX := 535, StoragePosZ := 428), (StoragePosZ := 12)];

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

STEP: ST_Auto:= ST_Auto.IDLE;

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

Timer1: TON;

Timer2: TON;

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

StartTimer1: BOOL;

StartTimer2: BOOL;

PickDone: BOOL;

PlaceDone: BOOL;

MoveDone: BOOL;

Bring: BOOL;

HomeDone: BOOL;

LightIndicator: BOOL;

```
//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

        VALINTA: INT;

        ButtonPressed: INT;



END_VAR



HMI.MoveX := StoragePositionsData[VALINTA].StoragePosX;

HMI.MoveZ := StoragePositionsData[VALINTA].StoragePosZ;

LightIndicator:= StoragePositionsData[VALINTA].LightIndicator;

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//----------------------Conditions for reseting system after errors----------------------\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

IF Reset THEN

        VALINTA:= 0;

        HMI.TAKE:= FALSE;

        StartTimer1:= FALSE;

        PickDone:= FALSE;

        StartTimer2:= FALSE;
```

```
            PlaceDone:= FALSE;

            ButtonPressed:= 0;

            STEP:= ST_Auto.IDLE;

            HMI.AutoStart:= FALSE;

            MoveDone:= FALSE;

            Bring:= FALSE;

            HomeDone:= FALSE;

END_IF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//||||||||||||||||||||||||||||AUTODRIVE PROGRAM START||||||||||||||||||||||||||||\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||



CASE STEP OF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//-----------------------START PICKUP STEP----------------------\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

            ST_Auto.IDLE:

            IF HMI.Enable AND ButtonPressed <> 0 AND HMI.TAKE THEN
```

```
                    STEP:= ST_Auto.HOME;

            ELSIF HMI.Enable AND ButtonPressed <> 0 AND HMI.BRING THEN

                    STEP:= ST_Auto.MOVEPOS;

                    Bring:= TRUE;

            END_IF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//----------------------RETRIVING THE ITEM----------------------\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

            ST_Auto.PICKBOX:

            VALINTA:= 0;

 // Condition: If box in the place indicated by "VALINTA"-variable... jump to case "IDLE".

            IF StoragePositionsData[VALINTA].LightIndicator AND HMI.TAKE THEN

                    HMI.TAKE:= FALSE;

                    STEP:= ST_Auto.IDLE;

// Condition: If box not in the place indicated by "VALINTA"-variable... continue program.

            ELSE

                    HMI.AutoStart:= TRUE;

                    Timer1(IN:= StartTimer1, PT:= T#2S);
```

```
                HMI.TAKE:= FALSE;

        IF HMI.AutoStart AND HMI.MoveX = 0 AND HMI.MoveZ = 8 THEN

                HMI.YInOut := TRUE;

                HMI.AutoStart:=FALSE;

                StartTimer1:= TRUE;

        IF HMI.YInOut AND NOT HMI.AutoStart AND Timer1.Q THEN

                        HMI.GripperClose:=TRUE;

                        HMI.GripperOpen:= FALSE;

                        StartTimer1:= FALSE;

                        Timer1.IN:= FALSE;

        IF hmi.GripperClose AND NOT StartTimer1 AND NOT Bring THEN

                        hmi.YInOut:= FALSE;

                        PickDone:= TRUE;

                        StoragePositionsData[VALINTA].LightIndicator:=
FALSE;

                        STEP:= ST_Auto.MOVEPOS;

        ELSIF hmi.GripperClose AND NOT StartTimer1 AND Bring THEN

                        hmi.YInOut  := FALSE;
```

```
                                        STEP:= ST_Auto.HOME;

                END_IF

                END_IF

                END_IF

                END_IF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//----------------------MOVING TO POSITION----------------------\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

                ST_Auto.MOVEPOS:

                IF PickDone AND NOT PlaceDone AND ButtonPressed <> 0 THEN

                                VALINTA:= ButtonPressed;

                IF NOT StoragePositionsData[VALINTA].LightIndicator THEN

                                HMI.AutoStart:= TRUE;

                                PickDone:= FALSE;

                                MoveDone:= TRUE;

                                StoragePositionsData[VALINTA].LightIndicator:= TRUE;

                                STEP:= ST_Auto.PLACEBOX;

                ELSE
```

```
                    STEP:= ST_Auto.IDLE;

            END_IF

            ELSIF ButtonPressed <> 0 AND bring THEN

                    VALINTA:= ButtonPressed;

                    HMI.AutoStart:= TRUE;

            IF HMI.xDone AND HMI.zDone THEN

                    StoragePositionsData[VALINTA].LightIndicator:= FALSE;

                    STEP:= ST_Auto.PICKBOX;

                    HomeDone := TRUE;

            END_IF

            END_IF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//----------------------PLACING THE ITEM TO STORAGE----------------------\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

            ST_Auto.PLACEBOX:

            Timer2(IN:= StartTimer2, PT:= T#2S);

            IF MoveDone AND hmi.zDone AND hmi.xDone OR HomeDone AND
hmi.zDone AND hmi.xDone THEN
```

```
                HMI.YInOut := TRUE;

                HMI.AutoStart:= FALSE;

                MoveDone:= FALSE;

                StartTimer2:= TRUE;

                HMI.MoveZ:= HMI.MoveZ + 5;

END_IF

IF HMI.Y_OUT AND HMI.YInOut AND Timer2.Q THEN

                HMI.GripperClose:= FALSE;

                HMI.GripperOpen:= TRUE;

                Timer2.IN:= FALSE;

                StartTimer2:= FALSE;

                HMI.MoveZ:= StoragePositionsData[VALINTA].StoragePosZ;

END_IF



IF HMI.GripperOpen AND HMI.Y_OUT THEN

                HMI.YInOut:= FALSE;

                STEP:= ST_Auto.IDLE;

ELSIF HMI.GripperOpen AND HMI.Y_IN THEN
```

```
                    Bring:= FALSE;

                    HomeDone:= FALSE;

                    STEP:= ST_Auto.IDLE;

            END_IF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//----------------------Moving to home position----------------------\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

            ST_Auto.HOME:

            IF HMI.Enable AND ButtonPressed <> 0 AND HMI.BRING THEN

                    VALINTA:= 31;

                    Bring:= FALSE;

                    HMI.BRING:= FALSE;

                    HMI.AutoStart:= TRUE;

                    STEP:= ST_Auto.PLACEBOX;

            ELSIF HMI.Enable AND ButtonPressed <> 0 AND HMI.TAKE THEN

                    VALINTA:= 0;

                    HMI.AutoStart:= TRUE;

                    PickDone:= FALSE;
```

```
IF HMI.xDone AND HMI.zDone THEN

STEP:= ST_Auto.PICKBOX;

END_IF

END_IF

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

//||||||||||||||||||||||||||AUTODRIVE PROGRAM END||||||||||||||||||||||||||\\

//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

END_CASE
```