LAB University of Applied Sciences

Kamerajärjestelmän hyödyntäminen servo-ohjauksessa

Kuivalajittelun purkauslaitteessa

LAB-ammattikorkeakoulu
Insinööri (AMK), Konetekniikka
2023
Jan Lyly

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Jan Lyly	Opinnäytetyö, AMK	2023
	Sivumäärä 35	

Työn nimi

Kamerajärjestelmän hyödyntäminen servo-ohjauksessa

Tutkinto ja koulutusala

Konetekniikan insinööri (AMK)

Toimeksiantajaorganisaatio (jos opinnäytetyöllä on toimeksiantaja)

Jartek AI Oy

Tiivistelmä

Opinnäytetyön tavoitteena on kehittää purkureunojen automatisoitu ohjaus, joka on osa purkauslaitetta. Purkauslaite puolestaan purkaa lautoja kerros kerrokselta kohti kuivalajittelua. Kehitystyö toteutetaan Jartek AI Oy:n toimesta. Kehitystyössä on mukana kumppani, joka toimittaa kamerajärjestelmän ja sen ohjelmoinnin. Työn tavoitteena on saada purkauksesta hallittavampi ja tehokkaampi.

Purkureunojen automatiikan ohjaus toteutetaan Beckhoff-automaatiojärjestelmällä. Automaatio ohjelmoidaan TwinCAT3-sovelluksella, joka on integroitu Microsoft Visual Studion kehitysympäristöön.

Opinnäytetyön tekeminen on kehittänyt ohjelmointitaitoja, auttanut ymmärtämään sahateollisuudesta ja sen automatiikasta. Kehitystyö on edelleen käynnissä ja odottaa varsinaista testausta tuotannossa. On ollut mielenkiintoista olla osa uuden tuotteen kehitystä, kun tuote on vielä prototyyppivaiheessa.

Kehitystyön tavoitteen saavuttaminen auttaisi vähentämään purkauksen häiriötilanteita, jotka voivat aiheuttaa tuottavuuden laskua sahateollisuudessa. Kun purkausprosessi on hallittavampi ja tehokkaampi, laitos voi käsitellä enemmän materiaalia vähemmässä ajassa. Tämä johtaa parantuneeseen tuottavuuteen.

Asiasanat

purkauslaite, ohjelmointi, sahateollisuus

Abstract

Author(s)	Type of Publication	Published
Jan Lyly	Thesis, UAS	2023
	Number of Pages	
	35	

Title of Publication

Utilizing camera system in servo-control

Degree, Field of Study

Mechanical engineering (UAS)

Organisation of the client (if the thesis work is commissioned by another party)

Jartek Ai Oy

Abstract

The objective of this thesis is to develop an automated control system for the unpacking layer edges, which is part of the unloading machine. The unloading machine unloads boards layer by layer towards the dry sorting. The development work is carried out by Jartek AI Oy. Associate is also involved in the development work, providing the camera system and its programming. The goal of the project is to make the unloading process more manageable and efficient.

The control of the unpack layer edge automation is implemented with the Beckhoff automation system. The automation is programmed with the TwinCAT 3 application, which is integrated into the Microsoft Visual Studio development environment.

The thesis has developed programming skills and helped to understand the sawmill industry and its automation. The development work is still ongoing and awaits actual testing in production. It has been interesting to be a part of the development of a new product while it is still in the prototype phase.

Achieving the goal of the development work would help reduce disruption situations during unloading, which cause a decrease in productivity in the sawmill industry. When the unloading process is more manageable and efficient, the facility can handle more material in less time, leading to improved productivity.

Keywords

unloading machine, programming, sawmill

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoituksena on kehittää purkauslaitteen automatiikkaa, jonka avulla saadaan tehostettua sahateollisuuden tuotantoprosessia. Purkauslaitteella tarkoitetaan laitetta, joka automaattisesti purkaa sahalautoja kuivalajitteluun. Tuottavuutta kehitetään jatkuvasti prosessien eri osa-alueilla. Tässä opinnäytetyössä keskitytään purkauslaitteen automaation kehittämiseen.

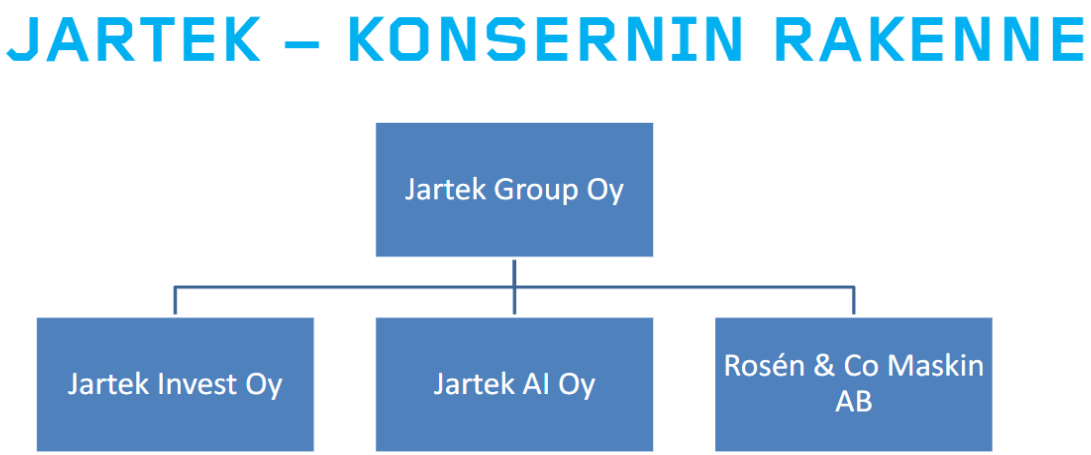
Opinnäytetyössä keskitytään erityisesti olio-ohjelmointimenetelmiin, joita hyödynnetään purkauslaitteen automaation kehittämisessä. Olio-ohjelmointi mahdollistaa joustavan ja modulaarisen ohjelmiston suunnittelun, mikä helpottaa laitteen toiminnan tarkkaa seurantaa ja muokkaamista tarvittaessa. Opinnäytetyön aikana kehitetään myös mittausdatan käyttöä ja kommunikaatiota laitteiden välillä.

Tämän opinnäytetyön tarkoituksena on tarjota yksityiskohtainen kuvaus purkauslaitteen automatiikan kehittämisestä. Työssä pyritään kehittämään luotettava ja tehokas automaatiojärjestelmä, joka vastaa sahateollisuuden vaatimuksia ja edistää sen kehitystä.

2 Toimeksiantaja ja toimiala

2.1 Jartek Oy

Jartek Oy on vuonna 1957 perustettu perheyritys, joka on erikoistunut sahateollisuuteen (Jartek a). Henkilöstön määrä on 62 vuoden 2023 alussa (Jartek b). Konserniin kuuluu Jartek Group Oy, Jartek Invest Oy, Jartek AI Oy ja tytäryhtiö Rosen & Co Maskin AB (Kuva 1).



Kuva 1. Konsernin organisaatiorakenne (Jartek b)

Yritys toimii projektitalona, joka on erikoistunut lajittelu- ja paketointilaitosten, puun lämpökäsittelylaitosten sekä jatkojalostusteknologian toimittamiseen sahateollisuuteen (Jartek a). Yritys suunnittelee tuotantokoneet ja -laitokset Lahden toimipisteellä. Projektit ja käyttöönotot hoidetaan asiakkaan luona sekä etänä.

2.2 Sahateollisuus

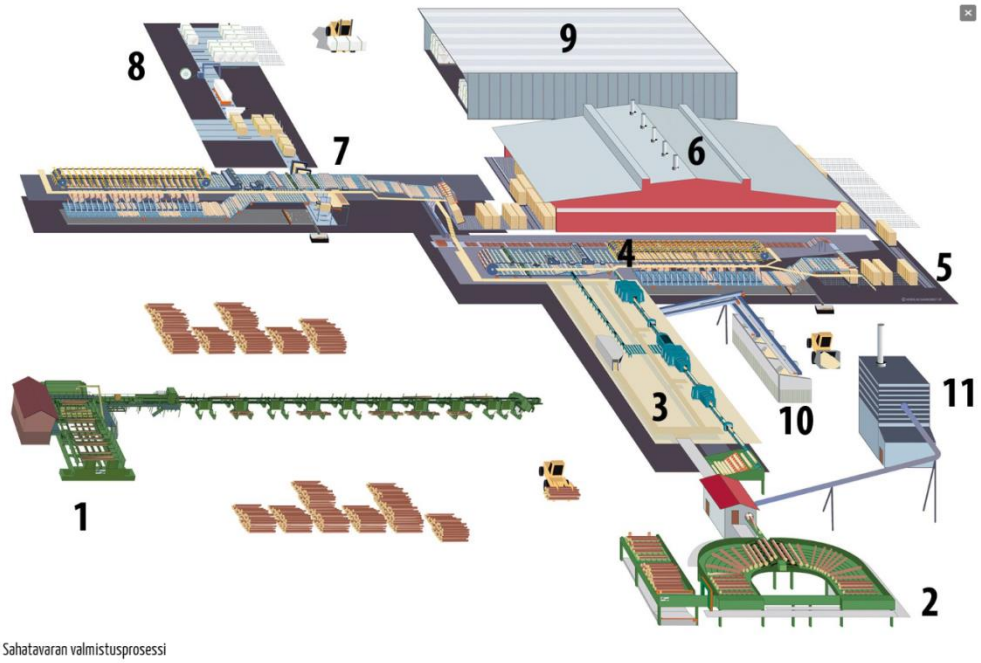
2.2.1 Sahalaitos yleisesti

Prosessi alkaa tukkien lajittelusta ja varastoinnista, jonka jälkeen tukit kuljetetaan kuorintaan. Kuoritut tukit siirretään sahalinjaan ja tukki valmistetaan sahatavaraksi. Tuore sahatavara menes dimensiolajittelulaitokselle, jota kutsutaan myös nimellä tuorelajittelu. Tuorelajittelussa sahatavara lajitellaan dimension (paksuus x leveys) mukaan lokeroihin.

Lajittelun jälkeen saman dimension sahatavara menee rimoitettavaksi kuivaamokuormiksi (Kuva 2).

Kuva 2. Rimoituksesta tulleita kuivauskuormia

Kuivaus toteutetaan kuivaamokanavissa ja -kamareissa. Kuiva sahatavara yleensä viimeistellään lopullisiin mittoihin ja jossain paikoin viedään lämpökäsittelyyn. Kuivalajittelussa lajitellaan lokeroihin samaa laatua olevaa sahatavaraa (paksuus x, leveys x pituus, laatu). Lopuksi sahatavarat paketoidaan ja varastoidaan. Prosessissa syntyy sivutuotetta kuten haketta, purua ja kuorta. Kuorta käytetään pääosin polttoaineena esimerkiksi tehtaan omissa lämpölaitoksissa. (Varis 2017, 65–154.)



Kuva 3. Sahalaitos (Sahateollisuuskirja b)

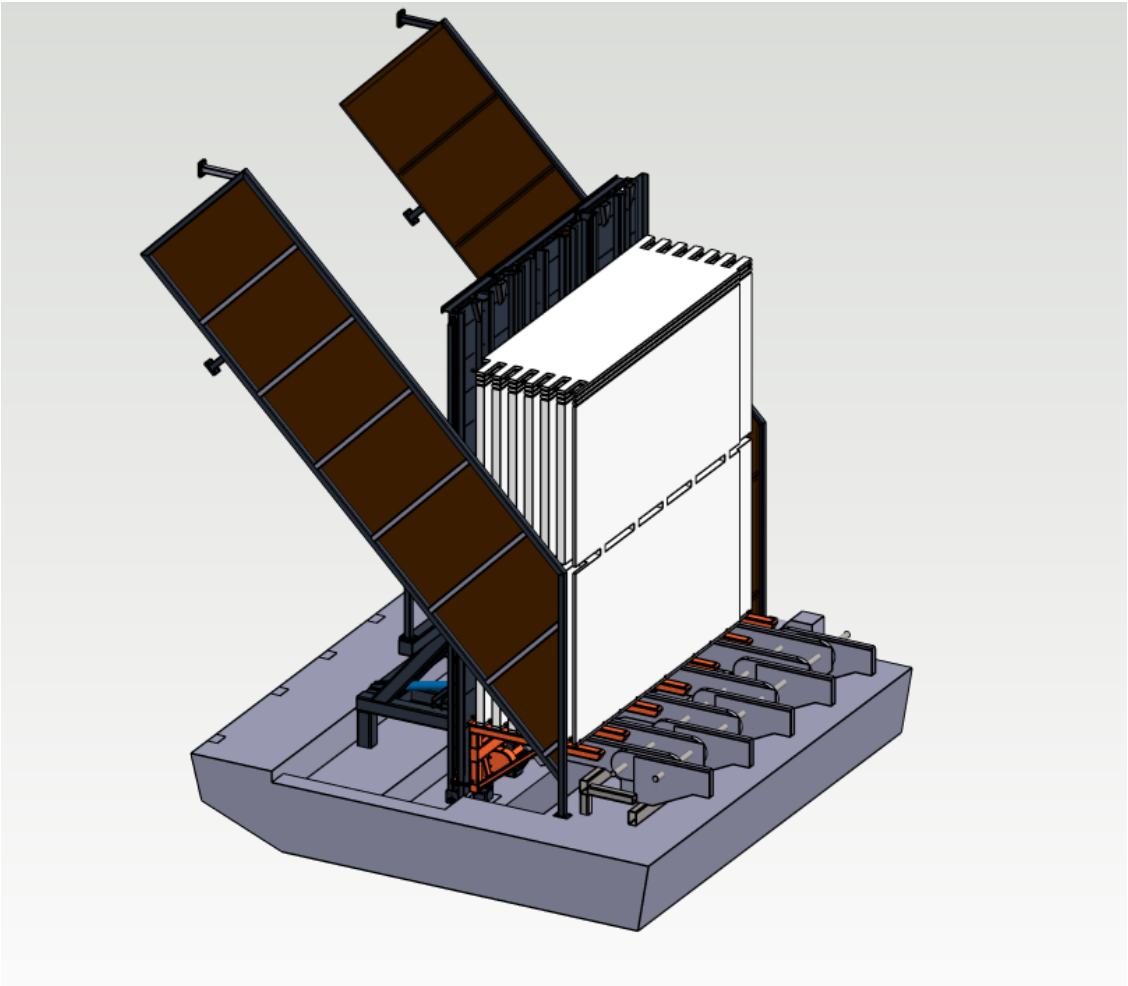
Sahalaitoksen eri vaiheet voidaan jaotella seuraaviin osakokonaisuuksiin (kuva 3):

- Tukkilajittelu ja varastointi (1)
- Kuorinta (2)
- Sahalinja (3)
- Dimensiolajittelu/tuorelajittelu (4)
- Rimoitus (5)
- Kuivaus (6)
- Kuivalajittelu (7)
- Pakkaus (8)
- Varasto (9)
- Siilo (10)
- Lämpölaitos (11) (Sahateollisuuskirja b)

2.2.2 Purkauslaite

Kuivaamolta tuleva kuivauskuorma varastoidaan varastokuljettimille väliaikaisesti, josta ne siirretään purkauslaitteen alkupäähän automaattisilla kuljettimilla, trukilla tai siirtovaunulla

riippuen laitoksesta. Väliaikaiset varastointimenetelmät vaihtelevat eri tehtaissa ja laitoksissa. Kuljettimilla olevat kuormat siirtyvät purkauslaitetta kohti automaattisesti tai käsiajolla. Kuivalajitteluun tuleva kuorma nostetaan purkaushissillä. Se toimii hydraulisesti, jossa on voimaa vaativat sylinterit kuorman painon takia. Hissi nostaa koko kuorman, jota kannattelevat sorkat. Hissi kallistuu ja nousee kerros kerrallaan ylöspäin kuorman nojatessa vasteeseen. Kuorman ylin kerros valuu lajittelulaitoksen ensimmäiselle syöttöpöydälle, ja kuorman kerrosten välissä olevat rimat ja välipuut tippuvat rimoille tarkoitetulle kuljettimelle. (Varis 2017, 148.) Purkauslaitteen toimintaa on havainnollistettu kuvassa 5.



Kuva 4. Purkauslaite kuorman kanssa (Jartek Oy)

Purkausprosessin katkeamatonta etenemistä voidaan saavuttaa käyttämällä apupurkauslaitetta. Sen tärkeimpänä tehtävänä on ottaa jäljelle jäänyt kuorma hallintaan ja purkaa se samalla, kun varsinaisen purkaushissin on tarkoitus kääntyä pystyasentoon ja laskeutua vastaanottamaan seuraava kuivauskuorma. Kun kuivauskuorma on nostettu purkukorkeudelle, apupurkaimen sorkat vetäytyvät tieltä, jonka jälkeen purkaushissi jatkaa nostamista kerros kerrokselta. (Varis 2017, 148.)

Kuivauskuormien välissä olevat epätasaisuudet voivat johtaa virheisiin ja ruuhkiin kuormien purkauksessa. Tämän ongelman minimoimiseksi on kehitetty purkureunoja, joita ohjataan servomoottoreilla (Kuva 21). Purkureunan avulla voidaan hallita kerrospurkausta tarkemmin, jotta virheiden ja ruuhkien riski pienenee. Nykyisin purkureunojen ohjaus tapahtuu käsiajolla, mutta opinnäytetyön tarkoituksena on ottaa käyttöön kamerajärjestelmän avustus purkureunojen ohjaamiseen automaattisesti. Tämä mahdollistaa entistä tarkemman ja luotettavamman purkamisen, sillä kamerajärjestelmän avulla voidaan saada tarkkoja mittatietoja ja ohjata purkureunaa sen mukaan. Automatisoitu purkureunan ohjaus on tärkeä kehitysaskel, sillä se mahdollistaa nopeamman ja tehokkaamman kuorman purkamisen. Kamerajärjestelmän avulla saadaan myös parempaa tietoa kuormista, mikä helpottaa kuormien käsittelyä ja hallintaa.

Kuva 5. Purkauslaite Koskisen Oy (Sahateollisuuskirja c)

3 PC-pohjainen ohjelmointi teollisuudessa

3.1 Beckhoff

3.1.1 Yritysesittely

Beckhoff on yritys, joka toimittaa avoimia automaatiojärjestelmiä, jotka perustuvat pc-pohjaiseen ohjausjärjestelmään. Kattavaan tuotevalikoimaan kuuluu teollisuus pc:t, I/O ja kenttäväyläkomponentit, liikenneohjaustuotteet, automatisointiohjelmistot, kaapittomat ohjaukset automaatioon ja kuvankäsittelyn laitteistot. Tuotelinjat toimivat yksittäisinä komponentteina tai ryhmäksi yhdistettyinä ohjausjärjestelmänä. (Beckhoff a.)

Kuva 6. Tuotevalikoima (Beckhoff a)

3.1.2 Teollisuustietokoneet ja ratkaisut

Tietokoneiden kehitys kasvaa eksponentiaalisesti kehittyvien puolijohdepiirien mukana. Tietokonepiirit ovat tulossa tehokkaimmiksi, pienemmiksi ja halvemmiksi. Teollisuustietokoneet yleistyvät yhä enemmän automaatioalalla. Teollisuustietokoneet voivat kattaa valvontahallinnan, minkä PLC:t tarjoavat. Lisäksi teollisuustietokoneilla voidaan käsitellä enemmän työmääriä, kuten HMI, portit, tekoälysovellukset. Teollisuustietokoneet voivat suorittaa nämä työmäärät tehokkaiden suorituskykykiihdyttimien, kuten GPU:ien, TPU:ien, VPU:ien ja NVMe SSD:ien ansiosta. Toimintojen yhdistäminen vähentää ohjausjärjestelmän laitteiston kokoa tehdasalueella. Teollisuustietokoneet voivat suorittaa samat tehtävät kuin PLC:t, mutta käyttöjärjestelmän ansiosta niillä voidaan ajaa erilaisia sovelluksia ja ohjelmia, joita ei ole saatavilla PLC:lle. Kuitenkin teollisuustietokoneen käyttöjärjestelmä,

kuten Windows ja Linux, ovat alttiita kyberhyökkäyksille, mutta nykyiset virustorjuntaohjelmat ja palomuurit ovat kehittyneet tarpeeksi vähentämään tätä riskiä. (Candtsolution 2022.) VPN-suojausta käytetään laajasti laitetoimittajien ja asiakkaiden välisessä yhteydessä.

3.1.3 C60xx -teollisuus Pc

Beckhoffin C60xx-sarjan ultra-kompakteja teollisuustietokoneita käytetään laajasti erilaisissa teollisuussovelluksissa, kuten automaatiojärjestelmissä, valvontajärjestelmissä ja data-analytiikassa. C60xx-tietokoneet ovat pienikokoisia, mikä mahdollistaa niiden asentamisen ahtaaseen tilaan, ja niissä on useita laajennusmahdollisuuksia ja erilaisia liitäntöjä.

C60xx-tietokoneet ovat modulaarisia ja skaalautuvia, mikä tarkoittaa, että niitä voidaan laajentaa tarpeen mukaan erilaisilla lisämoduuleilla. Tämä tekee niistä erittäin joustavia ja soveltuvia erilaisiin käyttötarkoituksiin. C60xx-tietokoneet ovat myös suunniteltu kestämään haastavia teollisuusympäristöjä, kuten tärinää, pölyä ja kosteutta.

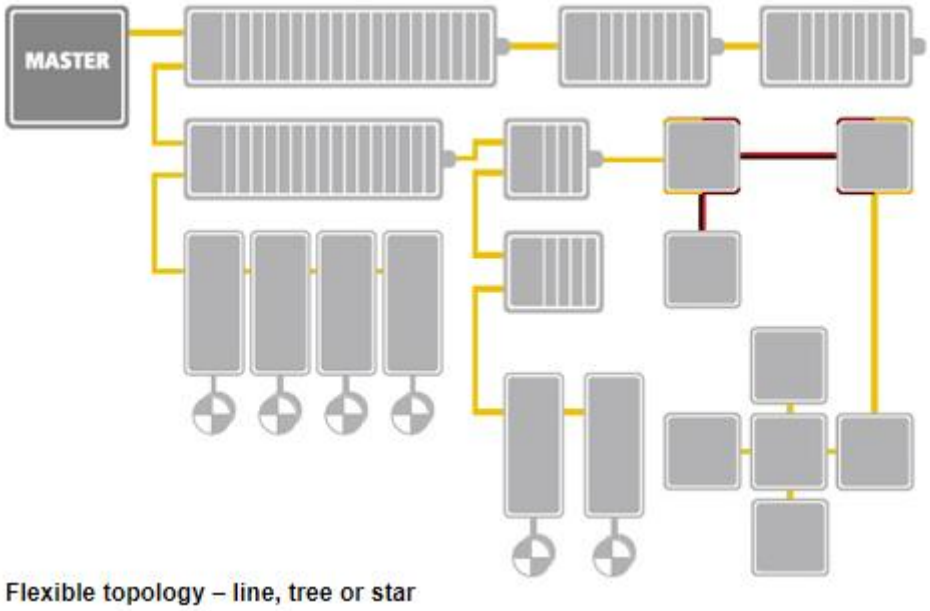
C60xx-tietokoneiden tekniset ominaisuudet vaihtelevat mallin mukaan. Niissä on useita erilaisia prosessoreita, kuten Intel Core i3/i5/i7, Celeron ja Atom, ja ne käyttävät erilaisia käyttöjärjestelmiä, kuten Windowsia, Linuxia ja Beckhoffin omaa TwinCAT-järjestelmää. C60xx-tietokoneissa on myös useita liitäntöjä, kuten Ethernet, USB, HDMI ja DisplayPort, ja ne tukevat erilaisia teollisuusväyliä, kuten PROFIBUSia ja EtherCATia. (Beckhoff b.)

Kuva 7. C6030 IPC (Beckhoff e)

3.1.4 EtherCAT

Beckhoff käyttää EtherCAT-verkkotekniikkaa. Synkronointinopeus voidaan toteuttaa nanosekunnin tarkkuudella. Nopeus on etu kaikissa sovelluksissa, joissa kohdejärjestelmää ohjataan tai mitataan väyläjärjestelmän kautta. Nopeiden reaktioaikojen avulla voidaan vähentää odotusaikoja prosessivaiheiden välillä, mikä parantaa sovelluksen tehokkuutta. Optimaalisesti käytettynä EtherCAT:n suorituskyky johtaa parantuneeseen tarkkuuteen, suurempaan suorituskykyyn. EtherCAT-sovelluksissa laitteen rakenne määrittää verkon topologian, ei päinvastoin. Perinteisissä teollisuuden Ethernet-järjestelmissä on rajoituksia, kuinka monta kytkintä ja keskittimiä voidaan ketjuttaa, mikä rajoittaa verkon kokonaisrakennetta. Koska EtherCAT ei tarvitse kytkimiä tai keskittimiä, tällaisia rajoituksia ei ole. EtherCAT on lähes rajaton verkon topologian suhteen. Linja-, puu-, tähti- ja näiden yhdistelmät ovat mahdollisia, kuten kuvassa 8 havainnollistetaan. Automaattisen linkin havaitsemisen ansiosta solmuja ja verkko-osia voidaan irrottaa käytön aikana ja liittää uudelleen, jos master-tuki mahdollistaa tämän toiminnon. Linjatopologiaa laajennetaan rengastopologiaksi kaapelin turvallisuuden varmistamiseksi. Tämä turvallisuusvaatimus ei vaadi muuta kuin toisen Ethernet-portin master-laitteelta, ja slave-laitteet tukevat kaapelin turvallisuutta jo valmiiksi. Tämä mahdollistaa laitteiden vaihdon käytön aikana.

EtherCAT-teknologia helpottaa konfiguraatiota, diagnostiikkaa ja ylläpitoa, jotka kaikki ovat järjestelmän kustannuksiin vaikuttavia tekijöitä. EtherCAT voidaan määrittää automaattisesti osoitteiden määrittämiseksi, mikä poistaa tarpeen manuaaliselle konfiguraatiolle. Alhainen väyläkuormitus ja peer-to-peer-fysiikka parantavat elektromagneettista häiriönsietoa. Verkko havaitsee mahdolliset häiriöt tarkasti niiden tarkan sijainnin avulla, mikä vähentää merkittävästi vianmääritykseen tarvittavaa aikaa. Käynnistyksen yhteydessä verkko vertaa suunniteltua ja todellista rakennetta havaitakseen mahdolliset poikkeamat. EtherCAT:n suorituskyky auttaa myös järjestelmän konfiguroinnissa poistamalla tarpeen verkon säätämiselle. Suuren kaistanleveyden ansiosta on mahdollista siirtää lisätietoja TCP/IP:n ohella. EtherCAT ei perustu TCP/IP:hen, joten MAC-osoitteita tai IP-osoitteita ei ole tarvetta hallita tai IT-asiantuntijoiden konfiguroida kytkimiä ja reitittimiä. (EtherCAT Technology Group a.)



Kuva 8. Linja, puu ja tähti topologiat (Ethercat Technology Group b)

3.1.5 TwinCAT 3

TwinCAT on Beckhoffin kehittämä ohjelmisto- ja ohjausjärjestelmä, joka mahdollistaa PC-pohjaisen automaation suorittamisen. Sovellus koostuu useista eri moduuleista, joista jokainen vastaa tietystä toiminnallisuudesta. Erilaisia moduuleita ovat esimerkiksi PLC-ohjelmointi, liikkeenohjaus, näytteenotto, signaalinkäsittely ja visualisointi. TwinCAT on avoin järjestelmä, joka tukee eri ohjelmointikieliä, kuten:

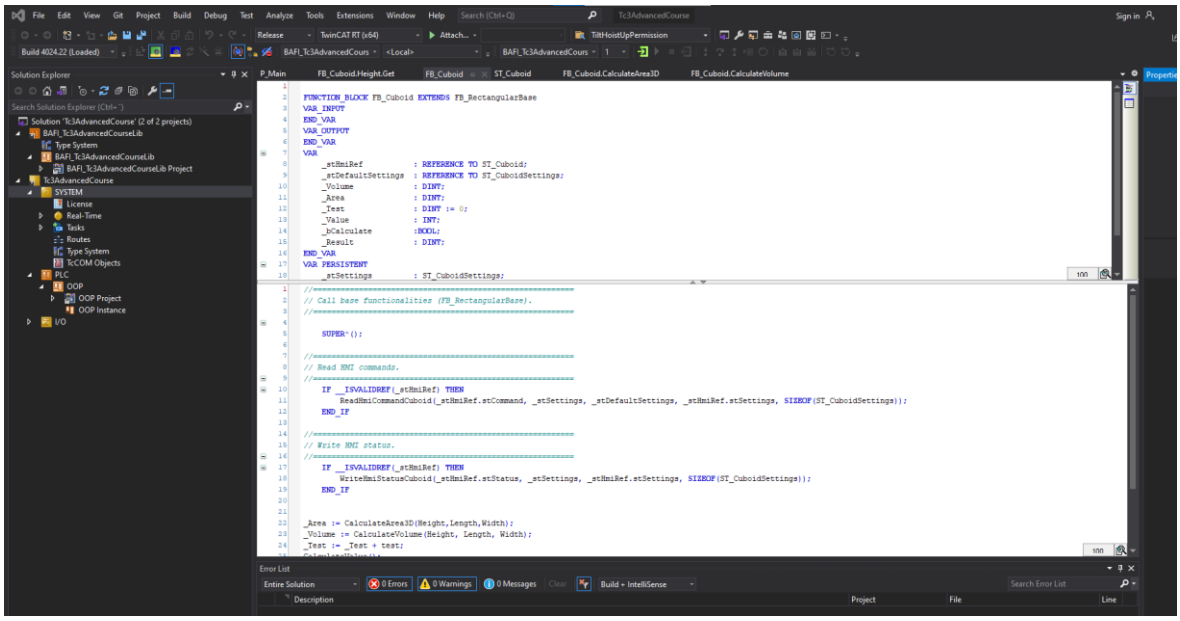
- IEC 61131-3 -ohjelmointikieli, joka on yleisesti käytetty standardi automaatiosovelluksissa. IEC 61131-3 tarjoaa neljä ohjelmointikieltä: ladder-diagram (LD), function-block-diagram (FBD), structured-text (ST) ja sequential-function-chart (SFC).
- C++ -ohjelmointikieli, joka on yleinen ohjelmointikieli laajemmassa ohjelmistokehityksessä. C++ mahdollistaa kehittäjille suuremman vapauden ja joustavuuden ohjelmistojen kehittämisessä.
- Matlab- ja Simulink ohjelmointikieli, joka on erityisesti suunniteltu matemaattisten mallien kehittämiseen ja simulointiin.

TwinCAT mahdollistaa reaaliaikaisen suorituksen ja sen avulla voidaan toteuttaa monimutkaisia automaatiosovelluksia. TwinCAT-järjestelmä koostuu PC-pohjaisesta ohjausyksiköstä ja erilaisista I/O-moduuleista, jotka on kytketty järjestelmään. Ohjausyksikkö suorittaa

ohjelmat ja antaa käskyt I/O-moduuleille, jotka ohjaavat laitteita ja keräävät antureilta tulevaa dataa. (Beckhoff f.)

Beckhoffin TwinCAT 3-ohjelmistoa voidaan integroida Microsoft Visual Studio-kehitysympäristöön, joka tarjoaa lisää ominaisuuksia ohjelmistokehitykselle (Kuva 9). Tämä mahdollistaa TwinCAT 3-ohjelmistojen kehittämisen ja testaamisen Microsoft Visual Studion ympäristössä, joka sisältää monipuolisia työkaluja ohjelmistokehittäjille.

TwinCAT 3-järjestelmässä on mahdollista hyödyntää versionhallintaa. Tämä mahdollistaa usean käyttäjän samanaikaisen työskentelyn saman sovellusohjelman parissa. Versionhallinnassa on myös mahdollista siirtää vain omat muutokset tai luodut komponentit, jolloin koko ohjelmakokonaisuutta ei tarvitse ladata versionhallintaan. Lisäksi ohjelma voidaan palauttaa aikaisempiin versioihin, mikä varmistaa ohjelman varmuuskopioinnin. Tämä on tärkeä ominaisuus, joka auttaa välttämään mahdollisia virheitä ja vahinkoja ohjelmoinnin aikana.

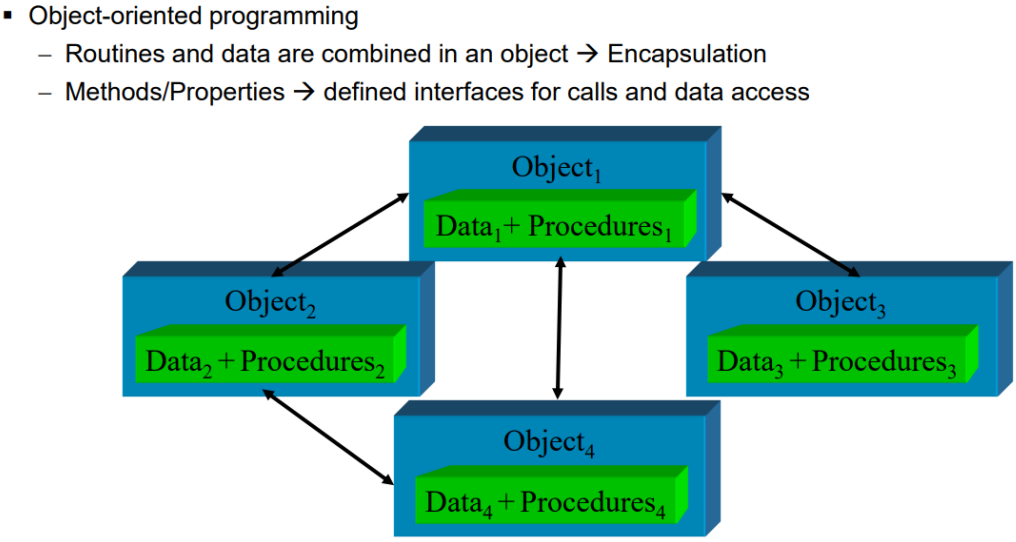


Kuva 9. TwinCAT 3 integroitu Microsoft Visual Studio kehitysympäristöön

3.2 Olio-ohjelmointi

Olio-ohjelmoinnin (OOP) arkkitehtuuri on ohjelmistojen suunnittelumalli, joka perustuu ohjelman rakentamiseen joukkona objekteja, jotka vuorovaikuttavat keskenään suorittaakseen tehtäviä. OOP tarjoaa useita etuja, kuten modulaarisen suunnittelun, ylläpidon helppouden ja koodin uudelleenkäytettävyyden. OOP-arkkitehtuuri on suunniteltu helpottamaan koodin uudelleenkäyttöä, mikä mahdollistaa ohjelmien rakentamisen olemassa olevista

ohjelmistokomponenteista. Tämä voi johtaa nopeampiin kehitysaikoihin ja alhaisempiin kustannuksiin. OOP-arkkitehtuurissa objektit kommunikoivat keskenään metodien ja viestien välityksellä. Jokaisella objektilla on oma data ja se voi keskustella muiden objektien kanssa lähettämällä viestejä pyytäen tiettyjä toimintoja (Kuva 10). Viestit vastaanottaa kohdeobjekti, joka suorittaa pyydetyt toiminnot ja palauttaa tuloksen. OOP-arkkitehtuuri helpottaa myös kapselointia, joka tarkoittaa objektin kykyä pitää sen data ja metodit yksityisinä ja piilossa muilta objekteilta. Kapselointi tarjoaa datan turvallisuuden ja estää muiden objektien pääsyn tai muokkaamisen ilman asianmukaista valtuutusta. (Graham 2019, 25-28.) Olio-ohjelmoinnin historia juontaa juurensa Simula-kielen kehittämiseen 1960-luvulla, mutta se tuli laajasti käyttöön vasta 1980-luvulla Smalltalk-kielen myötä. Nykyään olio-ohjelmointi on yksi suosituimmista ohjelmointiparadigmoista. (Graham 2019, 2-3).



Kuva 10. Olio-ohjelmointi (Patolinna 2023)

3.3 TwinCAT 3 Olio-ohjelmointi

3.3.1 Yleistä

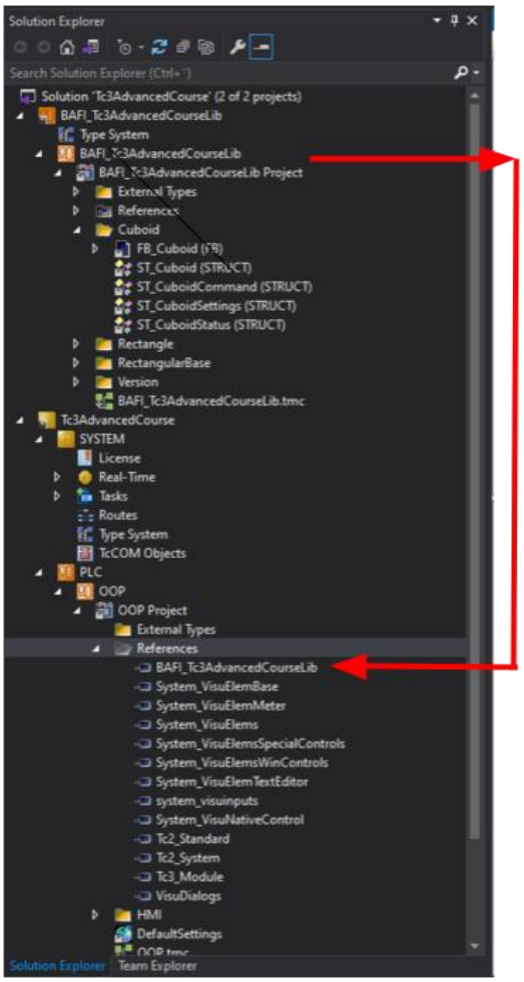
Jakamalla ohjelmisto objekteihin voidaan kehittää selkeä, hyvin rakennettu sovellus. Näin sovellus ja yksittäiset elementit ovat helposti ymmärrettäviä ja helposti laajennettavia. Ohjelmointiobjektien uudelleenkäyttö säästää aikaa ja kustannuksia sovellusten kehityksessä ja ylläpidossa. (Beckhoff c.)

3.3.2 Kirjasto

Kirjasto on PLC-projekti TwinCAT3:ssa, joka sisältää erilaisia uudelleenkäytettäviä objekteja eri projekteja varten, kuten

- ohjelmia
- funktiolohkoja ja funktioita
- rajapintoja
- käyttäjän määrittelemiä tietotyyppejä
- globaaleja muuttujia
- vakioita
- parametrilistoja
- visualisointeja
- tekstialueita ja kuvia.

Kirjaston objektien käyttämiseksi se on lisättävä viittauksena TwinCAT PLC -koneprojektiin, kuten kuvassa 11 on esitetty. Kun kirjasto on lisätty, siinä määritellyt objektit ovat käytettävissä esimerkiksi ’’koneprojektissa’’. Kirjastoprojekti voi myös viitata muihin kirjastoihin, ja tällöin kaikki viitatut kirjastot lisätään ’’koneprojektin’’ Kirjasto-sijaintipaikkoihin. On myös mahdollista asentaa useita versioita samasta kirjastoprojektista, mikä voi olla hyödyllistä, kun eri koneprojektit vaativat eri kirjaston versioita. (Patolinna 2023.)



Kuva 11. Kirjaston viittaus projektissa

3.3.3 Funktiolohko

Funktiolohko (Function block) on ohjelmointikäsite, joka edustaa luokkaa (Class) Olio-ohjelmoinnissa (Patolinna 2023). Luokka on ohjelmoinnin käsite, joka kuvaa tietyn tyyppisten objektien yhteisiä piirteitä ja toimintoja. Luokan avulla voidaan luoda samankaltaisia objekteja, jotka sisältävät samoja ominaisuuksia ja käyttävät samoja metodeja. Tämä lisää koodin uudelleenkäyttöä ja helpottaa sen ylläpitoa. (Graham 2019, 45–46.) Funktio lohko mahdollistaa datan vaihdon muuttujien avulla

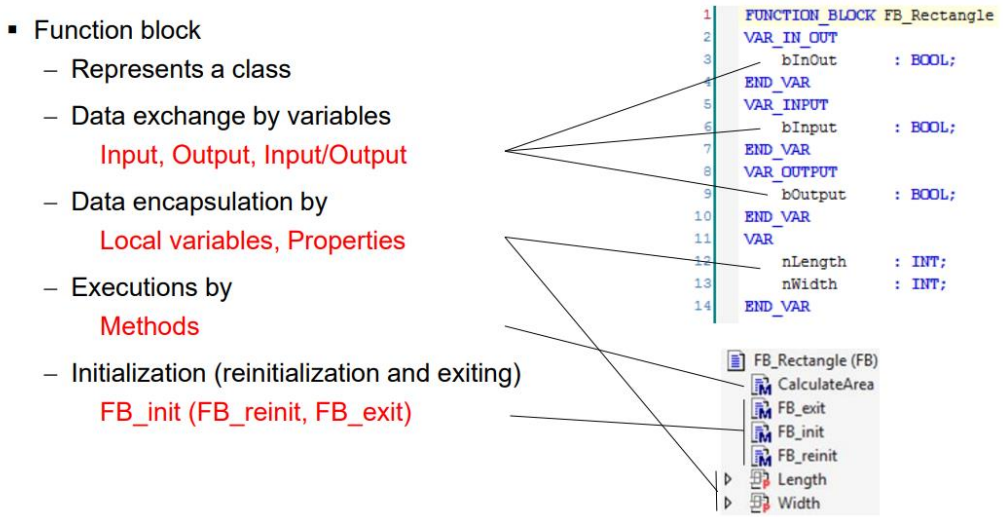
- input
- output
- input/output

Datan kapselointi toteutetaan käyttämällä paikallisia muuttujia ja ominaisuuksia. Funktiolohkon toimintaa määritellään sen metodeilla.

FB_init-lohkoa käytetään funktiolohkon alustamiseen, kopioimiseen ja lopettamiseen. FB_reinit-lohkoa käytetään automaattisesti ennen ensimmäistä funktiolohkon esiintymistä. FB_reinit-lohkoa käytetään myös automaattisesti ennen funktiolohkon esiintymän kopioimista (online-muutos), jos FB_reinit on toteutettu.

Alustus, uudelleen alustus ja lopetus toteutetaan:

- FB_init
- FB_reinit
- FB_exit (Kuva 12)



Kuva 12. Luokan datan vaihto (Patolinna 2023)

Funktiolohkoa kutsutaan yleensä objektin tai luokan kautta, joka sisältää kyseisen funktion. Objekti tai luokka toimii funktiolohkon isäntänä. Kutsuttaessa funktiolohkoa sen isännän kautta, luodaan uusi instanssi funktiolohkosta. Tämä instanssi sisältää kaikki funktiolohkon määrittelemät muuttujat ja toimii kopiona alkuperäisestä funktiolohkosta. (Beckhoff c.)

TwinCAT-ohjelmistoympäristössä funktiolohkoja voidaan käyttää monipuolisesti olioperustaisen ohjelmoinnin toiminnallisuuksien toteuttamiseen. Esimerkkejä näistä ovat

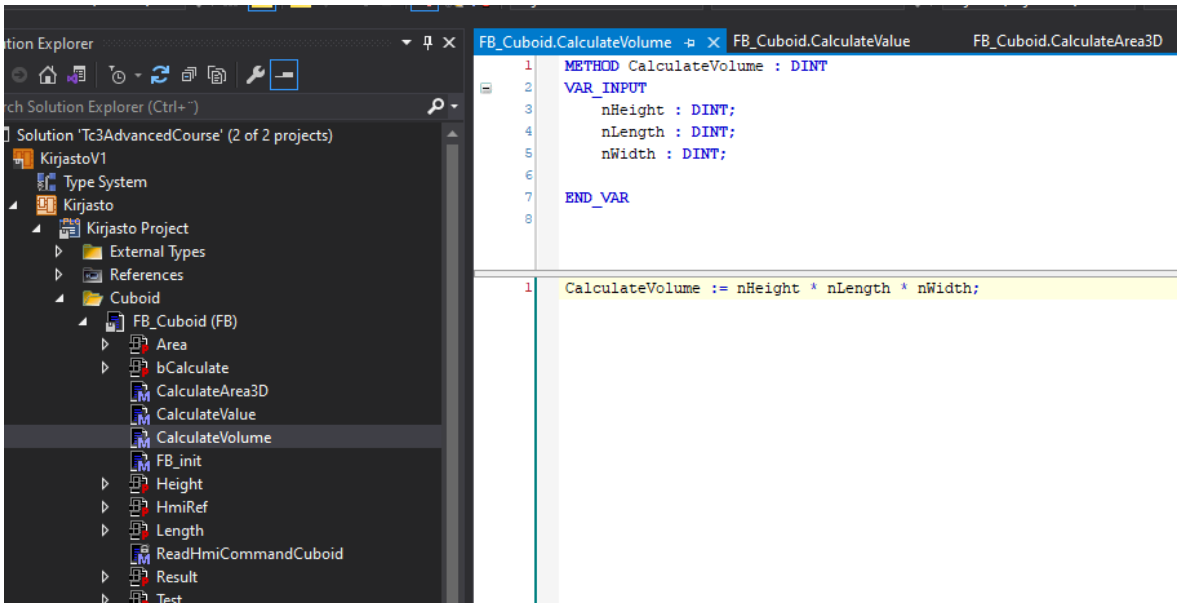
- metodit (object Method)
- ominaisuudet (object Property)
- rajapintojen toteuttaminen (implementing an interface)
- funktiolohkon laajentaminen (extending a function block)

3.3.4 Metodi

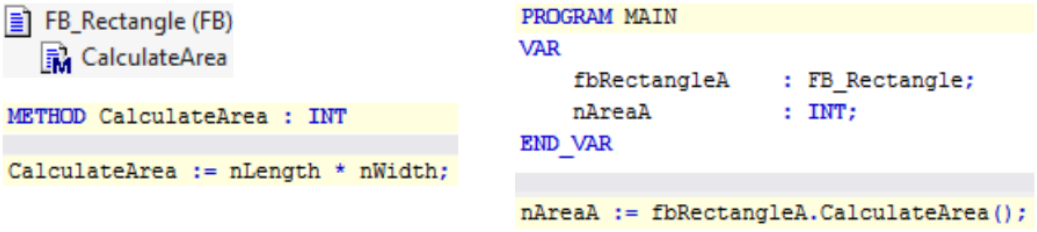
Olio-ohjelmoinnissa metodi on ohjelmoidun toiminnon kokoelma, joka on määritelty luokan osana ja on käytettävissä kaikille kyseisestä luokasta luoduille oliolle. Kukin olio voi kutsua metodia, joka suoritetaan olion kontekstissa. Tämä mahdollistaa metodin uudelleenkäytön useassa oliossa, jotka on luotu samasta luokasta.

Oliot pysyvät itsenäisinä toisistaan koko sovelluksen suorituksen ajan. Jos jokin olio kutsuu metodia, kyseinen metodi voi käyttää vain sille tiedossa olevia tietoja, mikä varmistaa tietojen eheyden sovelluksessa. Luokka voi sisältää useita metodeja. Jokainen niistä voidaan kutsua luokan kautta, vaikka ne toimisivatkin riippumattomasti toisistaan. Metodien määrittely, viittaus ja kutsuminen eroavat toisistaan eri olioperustaisissa kielissä, mutta perusperiaatteet ovat yleensä samat. Kun luokka ja sen metodit on määritelty, sovellus voi viitata luokkaan joko suoraan samasta tiedostosta tai osoittamalla luokkaan, jossa se sijaitsee. Esimerkiksi sovellus voi tuoda paketin tai moduulin, joka sisältää luokan, ja käyttää sen metodeja sovelluskoodissa. Sovellus voi sitten luoda olioita, jotka perustuvat luokkaan ja käyttää sen metodeja. (Sheldon 2023.)

TwinCAT-ohjelmistossa metodien sisällä määritellyt muuttujat ovat tilapäisiä ja olemassa vain sen aikaa, kun metodia suoritetaan. Nämä muuttujat ja funktiopalikat tallennetaan tyypillisesti pinoon, joka on muistin osa ja jota käytetään tilapäiseen tallennukseen ohjelman suorituksen aikana. TwinCAT alustaa kaikki metodissa määritellyt muuttujat ja funktiopalikat jokaisen metodin kutsun yhteydessä. Toisin sanoen kaikki aikaisempien metodikutsujen tallentama tieto häviää ja muuttujat palautuvat oletusarvoihinsa. Tämä käyttäytyminen on yleistä ohjelmointikielissä ja sen tarkoitus on varmistaa, että metodit suoritetaan johdonmukaisesti ja ennakoitavasti. Alustamalla kaikki muuttujat ja funktiopalikat jokaisen kutsun yhteydessä, TwinCAT auttaa välttämään virheitä tai odottamatonta käyttäytymistä, joka voisi tapahtua, jos vanhentunut tieto aiemmista metodikutsuista saisi jatkua. (Beckhoff c.)



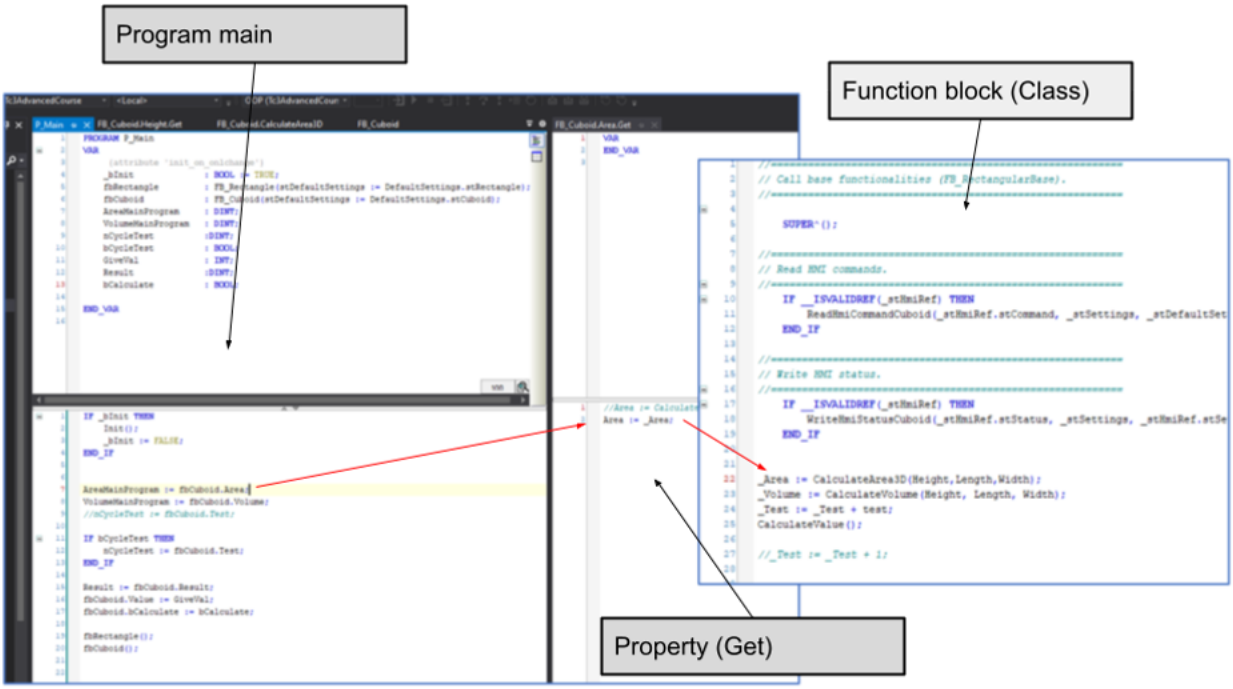
Kuva 13. Tilavuuslaskumetodi



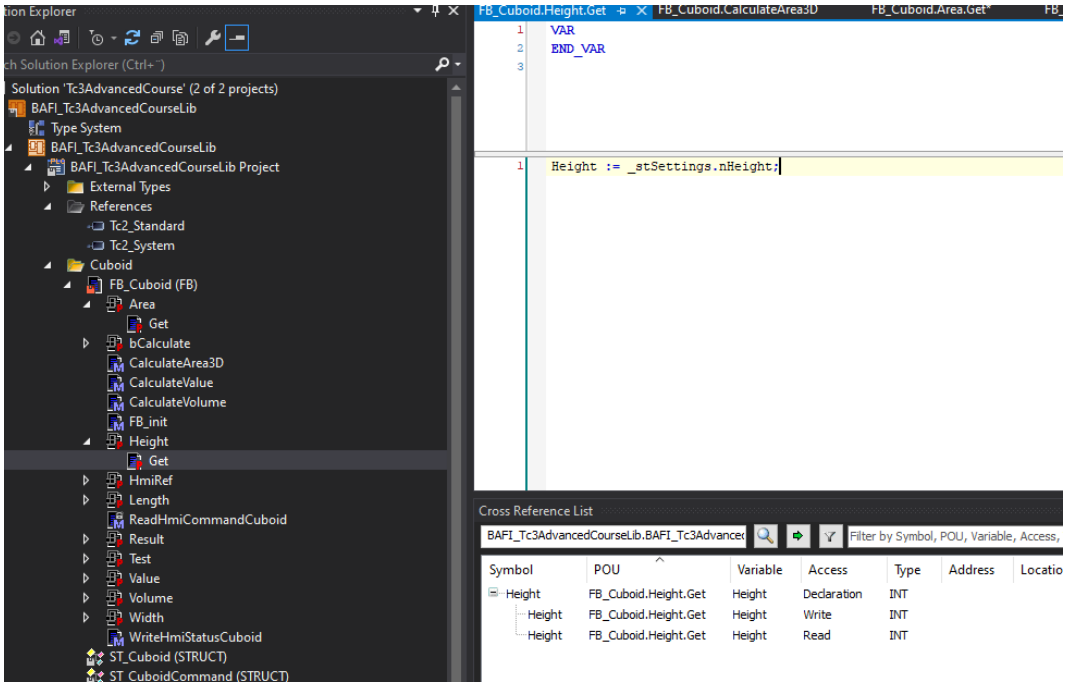
Kuva 14. Metodin kutsuminen Program main:ssa (Patolinna 2023)

3.3.5 Ominaisuus

Funktio-ohjelmointilohkojen ominaisuudet (property) mahdollistavat luokan paikallisiin muuttujiin pääsyn Get- ja Set-toimintojen avulla. Get-toiminto voi palauttaa luokan paikallisen muuttujan arvon, kun taas Set-toiminto voi kirjoittaa luokan paikalliseen muuttujaan. Lisäksi Get-toiminnolla voidaan suorittaa funktio, joka palauttaa arvon, kun propertya kutsutaan. Kuvassa 15 havainnollistetaan propertyn käyttöä ja kuinka muuttujan arvoa voidaan hyödyntää kuvan esimerkin mukaan.



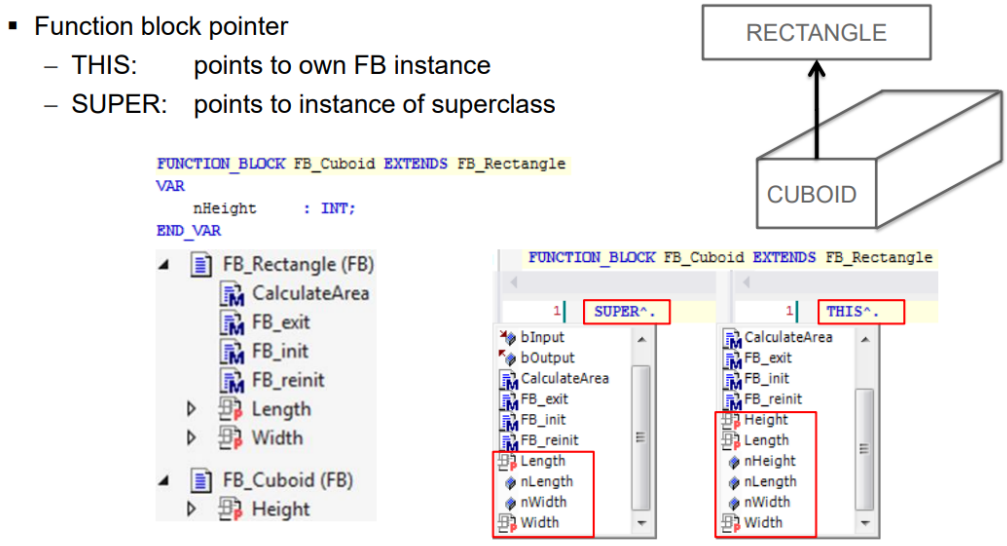
Kuva 15. Propertyn käyttö luokassa



Kuva 16. Property (Get)

3.3.6 Perintä

Funktiolohkon laajentaminen perustuu olioperustaisen ohjelmoinnin käsitteeseen periytyvyydestä (inheritance). Johtuva funktiolohko laajentaa perusfunktio-lohkoa tätä tarkoitusta varten ja perii perusfunktio-lohkon ominaisuudet ja toiminnallisuudet ja lisäksi sen omat ominaisuudet ja toiminnallisuudet. (Beckhoff c.)

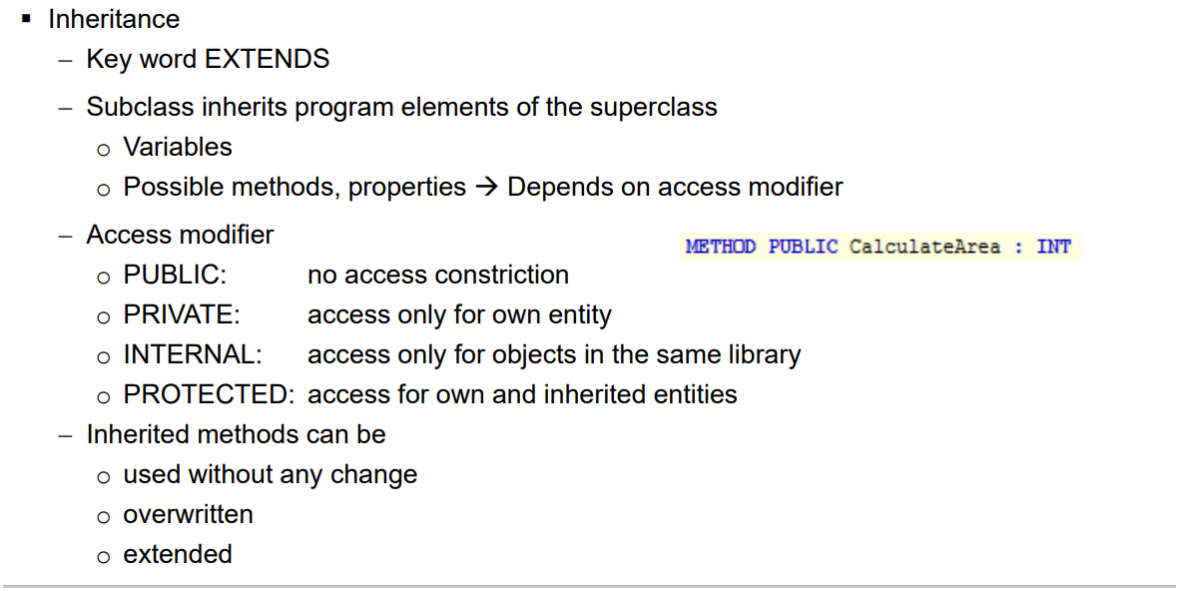


Kuva 17. Toisen funktiolohkon perintä (Patolinna 2023)

Perintä (inheritance) on ohjelmoinnin konsepti, jossa aliluokka (subclass) perii yläluokan (superclass) ohjelmointielementtejä. Aliluokka käyttää yläluokan muuttujia ja mahdollisia metodeja ja ominaisuuksia. Mahdollisuus käyttää yläluokan metodeja ja ominaisuuksia aliluokassa riippuu siitä, millainen pääsymerkintä (access modifier) yläluokan metodilla tai ominaisuudella on.

Pääsymerkinnällä voi rajoittaa metodeihin ja ominaisuuksiin pääsyä aliluokassa. Public-pääsymerkinnän omaavat metodit ja ominaisuudet ovat kaikkien käytettävissä. Private-pääsymerkinnän omaavat metodit ja ominaisuudet ovat saatavilla vain saman olion sisällä. Internal-pääsymerkinnän omaavat metodit ja ominaisuudet ovat saatavilla vain saman kirjaston objekteille. Protected-pääsymerkinnän omaavat metodit ja ominaisuudet ovat käytettävissä vain omassa ja perityssä entiteetissä.

Perityt metodit voidaan käyttää sellaisenaan, korvata tai laajentaa. Aliluokan nimeämiseen käytetään EXTENDS-avainsanaa, ja aliluokka ei saa sisältää muuttujia, joiden nimet ovat samat kuin yläluokassa. Ainoa poikkeus on VAR_TEMP-muuttujan käyttö, joka voi esiintyä sekä ylä- että aliluokassa. Yläluokan elementteihin voidaan viitata aliluokassa käyttämällä SUPER-osoitinta. (Patolinna 2023.)



Kuva 18. Perintä (Patolinna 2023)

3.3.7 Rajapinta

Rajapinta (interface) on ohjelmoinnin työkalu, jota käytetään olioperustaisessa ohjelmoinnissa. Rajapinta kuvaa joukon metodien ja ominaisuuksien prototyyppejä, joita eri luokat voivat toteuttaa. Prototyyppi tarkoittaa tässä yhteydessä sitä, että metodit ja ominaisuudet sisältävät vain määritykset, mutta eivät varsinaista toteutusta.

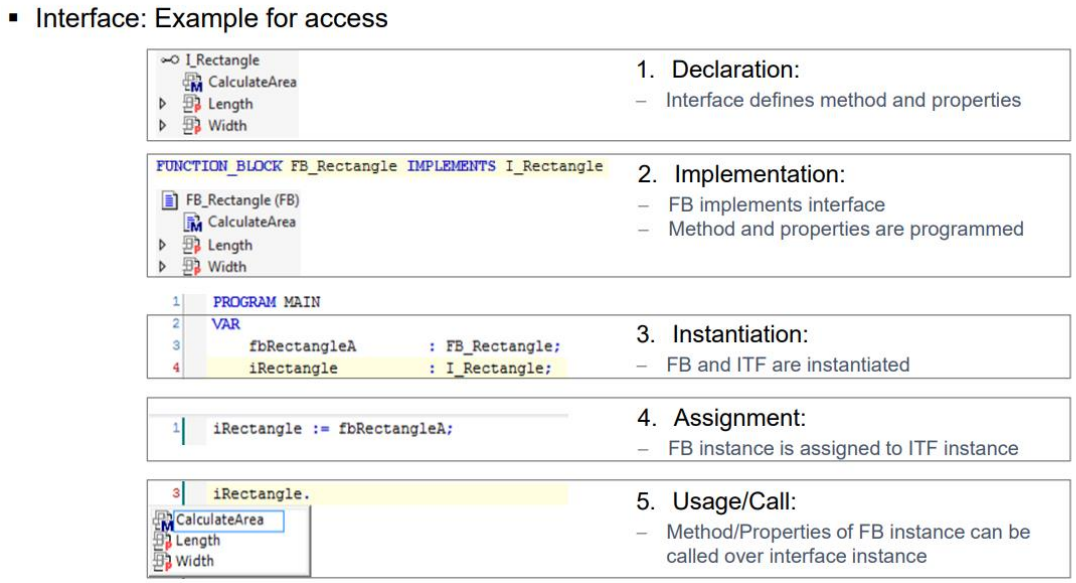
Rajapinnan avulla voidaan määritellä yhteiset ominaisuudet ja käyttää erilaisia luokkia tai funktiolohkoja hyödyntäen näitä ominaisuuksia samalla tavalla. Esimerkiksi jos useilla eri funktiolohkoilla on sama ominaisuus, voidaan tämä ominaisuus siirtää rajapintaan, jolloin funktiolohkot voivat toteuttaa sen rajapinnan avulla.

Rajapintaan voidaan lisätä metodi- ja ominaisuusobjekteja, jotka ovat rajapinnan määrittelemiä prototyyppejä. Näiden objektien toteutus on jätetty luokille, jotka toteuttavat rajapinnan. Rajapinnan käyttö auttaa välttämään tarpeetonta koodin toistoa ja parantaa ohjelman ylläpidettävyyttä. (Bechoff c.)

Rajapintojen hyödyt:

- Varmistus olemassa olevien elementtien tilasta: käyttämällä rajapintoja voidaan varmistaa, että tiettyjä elementtejä, kuten muuttujia tai funktioita, on olemassa tietyissä funktiopalikoissa.
- Voidaan käyttää useita FB-instansseja samanaikaisesti. Tämä mahdollistaa toimintojen tehokkaan jakamisen useiden laitteiden välillä.

- Laiteabstraktio, helppo laitteistokomponenttien vaihtaminen: Liittymien avulla laitteistokomponentteja voidaan käsitellä yleisellä tasolla, jolloin helppo vaihtaminen on mahdollista.
- Tietojen ja toimintojen välittäminen tai kapselointi muihin FB:iin: Liittymien avulla tietoja ja toimintoja voidaan helposti siirtää ja käyttää eri FB-instansseissa. (Patolinna 2023.)



Kuva 19. Rajapinnan käyttö (Patolinna 2023)

4 Kamerateknologia automaatiossa

Tuotantolaitoksien kapasiteetin kasvaessa tarvitaan entistäkin vauhdikkaampia ja vaikuttavampia tekniikoita prosessin optimointiin ja laadun varmistamiseen. Samanaikaisesti tulee kovenevia vaatimuksia seurattavuudelle ja tuoteturvallisuudelle.

Konenäkösovellus on paras tapa vastata näihin vaatimuksiin, sillä se mahdollistaa tauottoman prosessinohjauksen ja laadunvalvonnan suurissa kapasiteeteissa. Vaikkakin konenäköjärjestelmiä on käytetty jo pitkään, nopeus on usein ollut niiden heikkoutena.

Perinteinen konenäköjärjestelmä koostuu erillisestä PLC-ohjaimesta ja konenäkö-PC:stä, joiden välinen kommunikaatio aiheuttaa merkittävää viivettä. Se tekee niistä sopimattomia nopeiden prosessien ohjaamiseen. Mittaukset on kuitenkin tehtävä ilman prosessin keskeyttämistä.

Reaaliaikaisella PC-pohjaisella automaatioratkaisulla viiveet ovat kuitenkin minimaalisia, koska itse PLC-ohjaus ja kuvan analysointi suoritetaan samalla prosessorilla, omilla ytimillä. Esimerkiksi käsky hylätä viallinen tuote tuotantolinjalta voidaan ohjata millisekunneissa.

Kameraohjelmisto antaa nykyään käyttäjälle vapauden valita kameran ja optiikan, koska ohjelmistot usein tukevat standardia Ethernet-pohjaista GigE-rajapintaa, joka on yhteensopiva kaikkien kameravalmistajien kanssa. Konenäkö on monipuolinen ja tehokas tapa automatisoida monia teollisuuden toimintoja, kuten laadunvalvontaa, viivakoodien ja QR-koodien lukemista, hahmon- ja värin tunnistamista sekä kappaleiden lukumäärän laskentaa ja mittauksia.

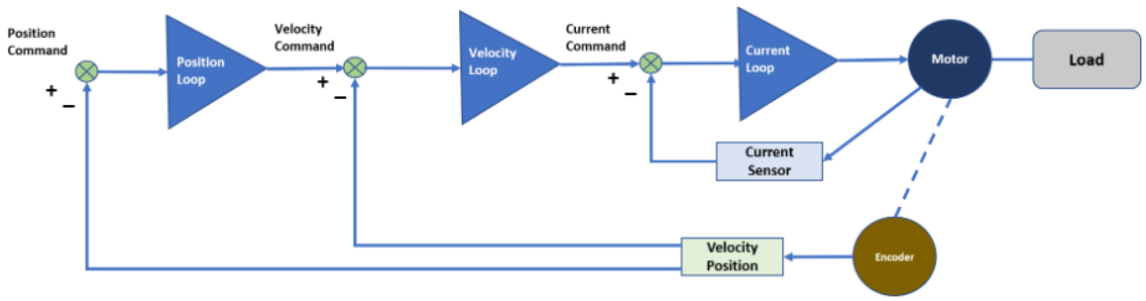
Konenäköjärjestelmä tuottaa runsaasti tietoa, jota voidaan hyödyntää IoT-ympäristössä. Tämä mahdollistaa koneoppimisen käytön, jossa tekoälyn algoritmit jatkuvasti oppivat ja optimoivat prosesseja. Koneoppiminen on samankaltaista kuin ihmisten opettaminen esimerkiksi kuvien avulla. (Suhonen 2023.)

5 Servomoottorit

Servomoottorit ovat tärkeitä osia monissa automaatiojärjestelmissä ja ne tarjoavat tarkkaa ja nopeaa liikettä. Servomoottorin toiminta perustuu periaatteeseen, jossa moottorin pyörimisnopeutta ohjataan tarkasti ulkoisella signaalilla (Kuva 20).

Kun servomoottorin käynnistää, moottori ottaa vastaan ohjaussignaalin. Ohjaussignaali ohjaa moottorin roottorin pyörimisnopeutta. Tämä ohjaussignaali välitetään moottorin ohjausyksiköltä. Ohjausyksikkö lukee ohjaussignaalin ja määrittelee sen perusteella, kuinka paljon virtaa tarvitaan, jotta moottori pyörisi halutulla nopeudella.

Servomoottorit ovat yleensä tehokkaita, koska niiden ohjausyksiköt pystyvät mukauttamaan virtaa nopeasti moottorin pyörimisnopeuden mukaan. Tämä tarkoittaa sitä, että servomoottorit voivat tuottaa korkeaa vääntömomenttia ja nopeaa kiihtyvyyttä, mikä on erityisen tärkeää monissa automaatiojärjestelmissä. (Kollmorgen Experts 2020.)



Kuva 20. Servo-ohjauksen toiminta (Kollmorgen Experts 2023)

6 Purkureunojen automatisoitu ohjaus

6.1 Aloitus

Hankkeen tavoitteena on automatisoida purkureunojen ohjausjärjestelmä, jotta häiriötilanteet voidaan minimoida ja purkauksesta saadaan hallitumpi. Purkauksen häiriöt johtuvat kuorman epätasaisuuksista, jotka vaikuttavat purkausprosessin sujuvuuteen ja tehokkuuteen. Uusi kuorma vaatii aina purkureunan säätöä ja automatisoinnilla saataisiin ylimääräinen säätötyö pois käyttäjältä.

Purkureunojen automaatio-ohjauksen kehityshankkeessa on mukana kumppani, joka on erikoistunut teollisuuden mittajärjestelmiin ja ohjelmointiin. Tämä yhtiö vastaa kamerajärjestelmän ohjauksesta ja ohjelmoinnista, kun taas Jartekin vastuulla on purkureunojen automaattiohjaus. Nykyisellään purkureunat toimivat manuaalisesti, joten automaatio-ohjauksen käyttöönotto parantaa purkausprosessin suorituskykyä ja tehokkuutta.

Kehityshankkeen yhteisissä kokouksissa käydään läpi tavoitteet, aikataulut, tarvittavat viestinnät laitteiden välillä ja ohjelmakomponenttien suunnittelu ja toteutus. Tavoitteena on saavuttaa korkea automaatiotaso purkausprosessissa, jolloin häiriötilanteiden määrä minimoidaan ja purkauksesta tulee hallitumpi.

Yhtiöiden välistä kommunikoinnin parantamiseksi osapuolet ovat päätyneet käyttämään Slack-viestintäpalvelua. Tämä palvelu tarjoaa keskusteluryhmiä projektikohtaisesti, mikä helpottaa yhteydenpitoa kehityshankkeen aikana. Slack-viestintäpalvelun avulla voidaan nopeasti ja helposti viestiä tarvittavista asioista ja jakaa tietoa yhteistyökumppaneiden kanssa.

Kuva 21. Purkauslaite ja purkureunat

6.2 Ohjelmointi

6.2.1 Automatisoitu ohjaus

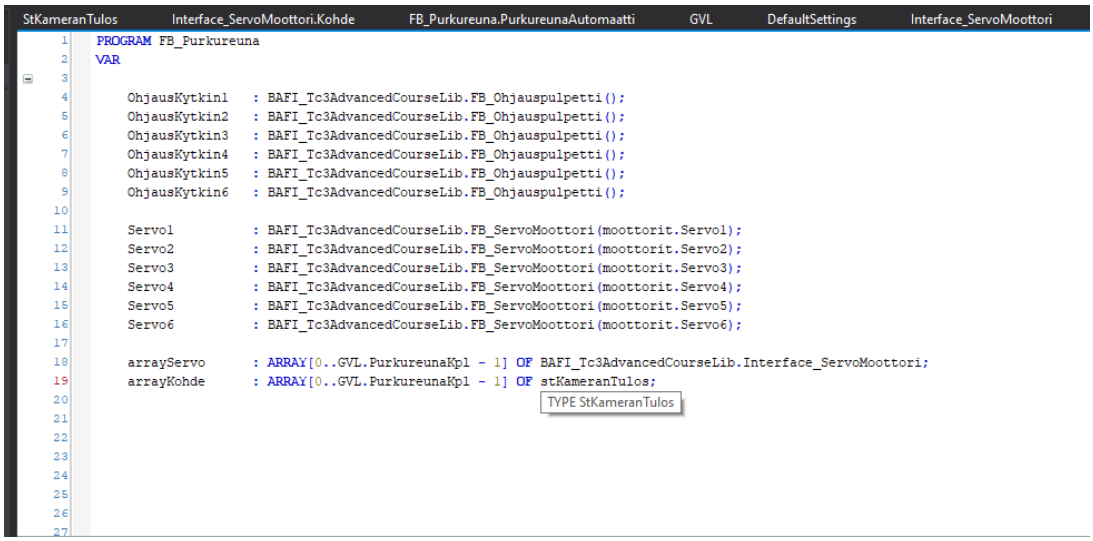
Osion kuvat eivät ole peräisin varsinaisesta ohjelmasta, vaan ne on luotu esimerkkien avulla, jotta voitaisiin saada parempi käsitys opinnäytetyön ohjelmasta. Työssä painotetaan enemmän tehtyjä muutoksia ohjelmaan kuin olemassa olevia komponentteja.

Automaation ohjelmointi toteutetaan TwinCAT-ohjelmalla, mikä on integroitu Microsoft Visual Studio kehitysalustalle. Ohjelmistossa hyödynnetään tehokasta Olio-ohjelmointi menetelmää, joka on nopeuttanut ohjelman tekoa merkittävästi. Ohjelmassa on myös hyödynnetty valmiita luokkia servo-ohjaukseen ja servomoottorien määrittämiseen, mikä on tehnyt ohjelmoinnista entistä sujuvampaa.

Kuten aiemmin mainittiin, järjestelmässä on käytössä manuaaliohjaus, joka voidaan toteuttaa joko käyttöliittymän kautta tai ohjauspulpetin ohjauskytkimillä. Purkureunojen automaattiseen ohjaamiseen tarvittiin uusia metodeja, jotka lisättiin purkureuna-luokkaan. Luokan uudet metodit ovat EdgesAutoMode, CheckInitation ja PositionTargetOverLimits.

Purkureunan-luokassa määritellään (Kuva 22):

- ohjauskytkimet
- servo-ohjaus
- rajapinnat servo-ohjaukselle
- muuttujat
- hälytykset

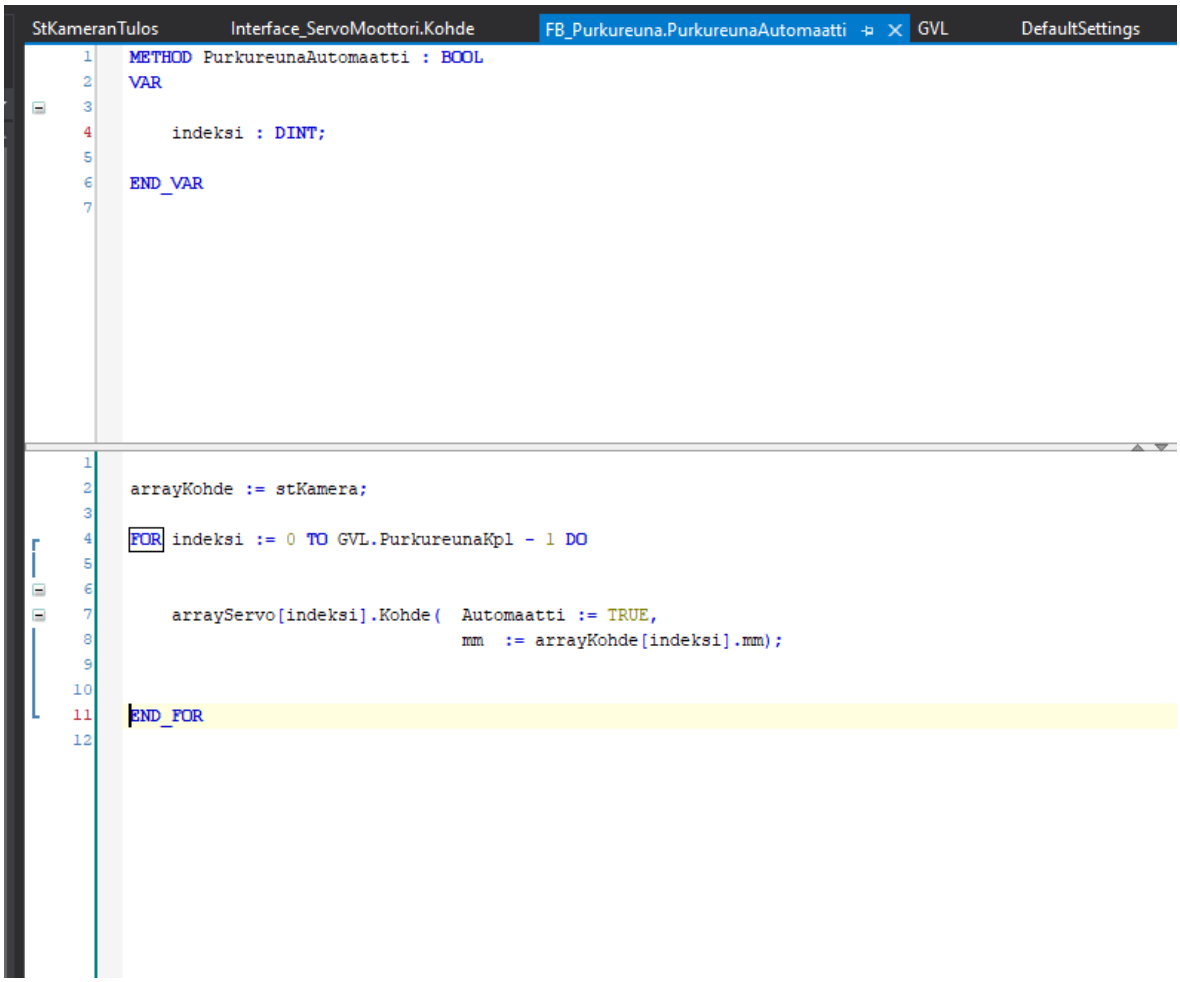


Kuva 22. Esimerkki muuttujien ja ohjelmalohkojen määrittelystä

6.2.2 EdgesAutoMode

EdgesAutoMode-metodi on automaattiohjaukseen tarkoitettu metodi. Metodissa saadaan purkureunan servoille asetusarvot. Mitta-arvo saadaan kamerajärjestelmältä, joka laskee algoritmilla mihin purkureuna pitää kohdistaa. Metodissa ei tarvitse erikseen tehdä jokaiselle servolle omaa ohjausta, koska servomoottorit on määritelty luokan ’alustus’ metodissa. Ohjelmaan on tehty indeksiluku, mikä käy läpi erikseen jokaisen purkureunan servo-ohjauksen FOR- silmukalla (Kuva 23).

Ohjelma ottaa huomioon kameran mittausvirheet. Mikäli mittauksen tulos ylittää purkureunan ylä- tai alarajan, käyttöliittymään tulee virheilmoitus ja vanha mitta-arvo säilyy ohjauksessa. Ohjelmaa ei kuitenkaan ole vielä kunnolla testattu, joten on epävarmaa, olisiko parempi jättää edellinen mittatieto vai kohdistaa kameran ylä- tai alarajan lähelle. Tämä kysymys voidaan ratkaista vasta testaamalla ohjelmaa erilaisissa olosuhteissa ja määrittämällä, mikä koodi toimii parhaiten. On myös tärkeää huomioida, että eri tilanteissa saattaa olla erilaisia käyttötarkoituksia, joissa kohdistus halutaan joko rajan läheisyyteen tai edelliseen mittatietoon. Tämän vuoksi on tärkeää suorittaa laajat testit ja arvioida eri tilanteita ennen lopullisen ohjelman määrittämistä.

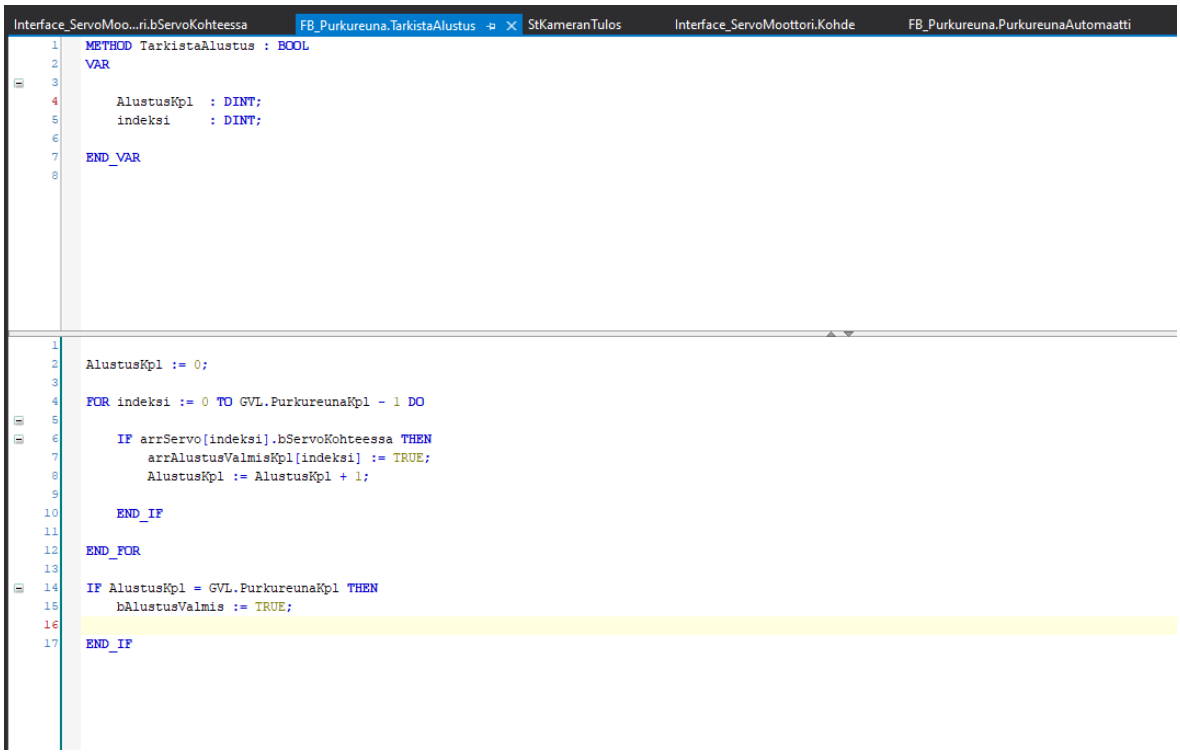


Kuva 23. Esimerkki servo-ohjauksesta

6.2.3 CheckInitation

CheckInitiation-metodi tarkistaa, onko purkureunojen aloitusasema hyväksytty. Jos aloitusasemaa ei ole vahvistettu, automaattinen purkureunan toiminto ei ole mahdollinen ja alustus on suoritettava uudelleen. Purkureunojen alustus tapahtuu joko käyttöliittymän painikkeella tai automaattitoiminnolla purkauslaitteen ollessa valmis ja kuorman purettua loppuun.

For -silmukalla käydään läpi, ovatko purkureunat saavuttaneet kohdeaseman ja sen jälkeen lisätään luku muuttujaan ’AlustuKpl’. Jos kaikki servot ovat valmiita, niin ’bAlustusValmis’ muuttuu TRUE tilaan ja purkureunojen automaattitoiminto on valmis käytettäväksi (Kuva 24). Jos jokin servoista ei saavuta kohdeasemaa tai menee häiriöön, niin käyttöliittymään tulee hälytys.



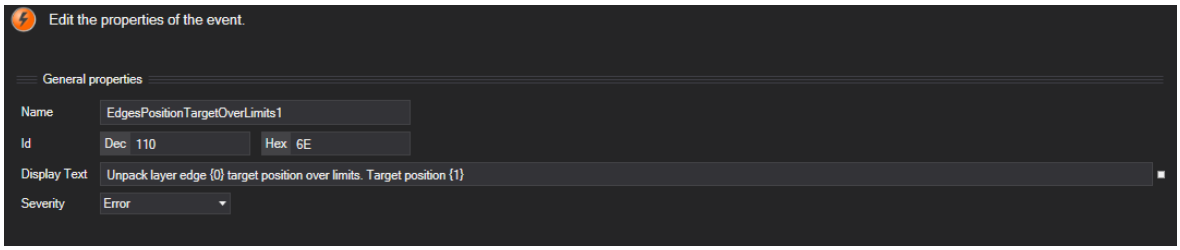
Kuva 24. Alustuksen tarkistus

6.2.4 PositionTargetOverLimits

PositionTargetOverLimits on metodi, joka asettaa häiriön aktiiviseksi, jos mittauksen tulos ylittää ennalta määrätyn raja-arvon. Tämä hälytyskäsittelyjärjestelmä on suunniteltu valvomaan mittauslaitteiden toimintaa ja tunnistamaan poikkeavuuksia, jotka voivat viitata mahdollisiin ongelmiin.

Ohjelma käy FOR-silmukalla läpi paikannukset ja vertailee parametrien arvoja. Lisäksi ohjelma tarkistaa, ettei hälytys ole jo aktivoitu, jotta vältetään tilanne, jossa hälytys laukeaa jokaisella ohjelmakierroksella uudelleen. Tämä on tärkeää, jotta ohjelma suorittaa toimintonsa tehokkaasti ja vältetään tarpeettomia häiriötilanteita. (Kuva 26.)

Mikäli mittauksen tulos ylittää asetetun raja-arvon, aloitetaan ensin hälytyksen nollaaminen. Tämän jälkeen ilmoitetaan, missä servossa havaittiin mittausvirhe ja näytetään samalla kyseisen mittauksen tulos, jotta virhemarginaali voidaan tarkistaa ja arvioida virheen laajuutta. Lopuksi hälytys aktivoidaan uudelleen ja se näkyy käyttöliittymässä.



Kuva 25. Hälytyksien käsittely

```
FOR indeksi := 0 TO GVL.PurkureunaKpl - 1 DO

    IF arrKohde[indeksi].mm > parametrit.Ylaraja OR
        arrKohde[indeksi].mm < parametrit.Alaraja AND NOT
        alarm[indeksi].Raised
    THEN

        alarm[indeksi].Clear();
        alarm[indeksi].AddUdint(indeksi + 1);
        alarm[indeksi].AddUdint(arrKohde[indeksi].mm);
        alarm[indeksi].Raise();

    END_IF

END_FOR
```

Kuva 26. Esimerkki hälytyksen käsittelystä

6.2.5 Kamerajärjestelmä

Kamerajärjestelmän ja ohjelman välillä käytetään kommunikointiin UDP-protokollaa ja JSON-muotoista rajapintaa järjestelmien väliselle kommunikoinnille. JSON-tiedostot sisältävät kamerajärjestelmän havaintoja, jotka lähetetään UDP-protokollaa käyttäen. Ohjelmoinnissa on pystytty hyödyntämään valmiita JSON -ja UDP-kirjastoja (Kuva 27).

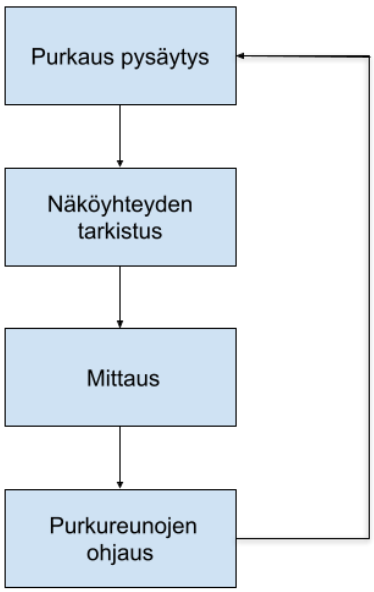
Kun ohjelma vastaanottaa JSON-tiedoston, se muokataan mittauksiksi, aikaleimaksi ja mittauksen numeroksi. Ohjelma ilmoittaa myös, että tulos on saavutettu ja viesti on vastaanotettu. Mittausdata on struktuurimuodossa, jotta sitä on helpompi käsitellä ohjelmassa.

Ennen mittauksen suorittamista kamerajärjestelmälle lähetetään viesti, jossa kerrotaan mittauksen aloittamisesta. Viestissä kerrotaan myös, onko purkauslaite automaattitilassa ja aktivoidaan mittauksen aloitus. Tämä varmistaa sen, että mittaus tehdään oikeaan aikaan ja järjestelmä on valmis vastaanottamaan mittausdatan.

Laitteen mittaukset suoritetaan vain silloin, kun sen purkausohjelma on käynnissä. Kun käytetään komentoa "purkauslaitteen pysäytä", muuttuja nimeltä "UnpackLayerMeasurePermission" asetetaan päälle. Tämä tarkoittaa, että laite on valmis lähettämään mittausviestin järjestelmälle. On myös tärkeää ottaa huomioon, ettei mittauskohteen näkyvyys ole estetty. Kun kerros on purkautumassa, se on purkureunojen edessä ja mittaus epäonnistuu. Tämän vuoksi on tehty metodi, joka käyttää valokennoja tarkistaakseen, ettei kerroksen edessä ole esteitä mittauksen onnistumiselle. Viestin lähettäminen sallitaan vain silloin, kun purkaus on pysähdyksissä ja mittauskohteessa ei ole näköesteitä.

```
// For testing purposes..
IF bSendStatus THEN
    bSendStatus := FALSE;
    Measure.Timestamp := dtTimestamp;
    Measure.MakeMeasure := BOOL_TO_DINT(MakeMeasure);
    // NOTE: UDP.SendLocalDebug(...) will sen messages to sLocalHost:nLocalPort !!
    // Use UDP.Send(...) to send sRemoteHost:nRemotePort
    UDP.SendLocalDebug(fbJSON.Stringify(ADR(Measure), SIZEOF(Measure)));
END_IF
```

Kuva 27. Esimerkki viestin lähettämisestä loopback-osoitteeseen



Kuva 28. Purkureuna-automatiikan kaavio

6.3 Käyttöönotto

6.3.1 Ohjelmasimulaatio

Ohjelmien simuloiminen ennen varsinaista käyttöönottoa on erittäin tärkeä vaihe, joka auttaa välttämään mahdollisia ongelmia. Simulointi mahdollistaa ohjelman testaamisen erilaisissa olosuhteissa ja tilanteissa ennen kuin se otetaan käyttöön varsinaisessa tuotantojärjestelmässä. Tämä auttaa tunnistamaan mahdollisia ongelmia ja virheitä, jotka voivat johtaa ohjelman kaatumiseen tai muihin toimintahäiriöihin.

Tuotekehityksen tullessa jo valmiiksi toimivaan laitokseen on erittäin tärkeää, että kehitysprosessi ei häiritse laitoksen tuotantoa. Ohjelman simuloiminen ennen käyttöönottoa voi auttaa tunnistamaan mahdollisia häiriöitä ja estämään niiden syntymistä. On myös tärkeää varmistaa, että ohjelma on yhteensopiva laitoksen muiden järjestelmien ja laitteiden kanssa ennen sen käyttöönottoa.

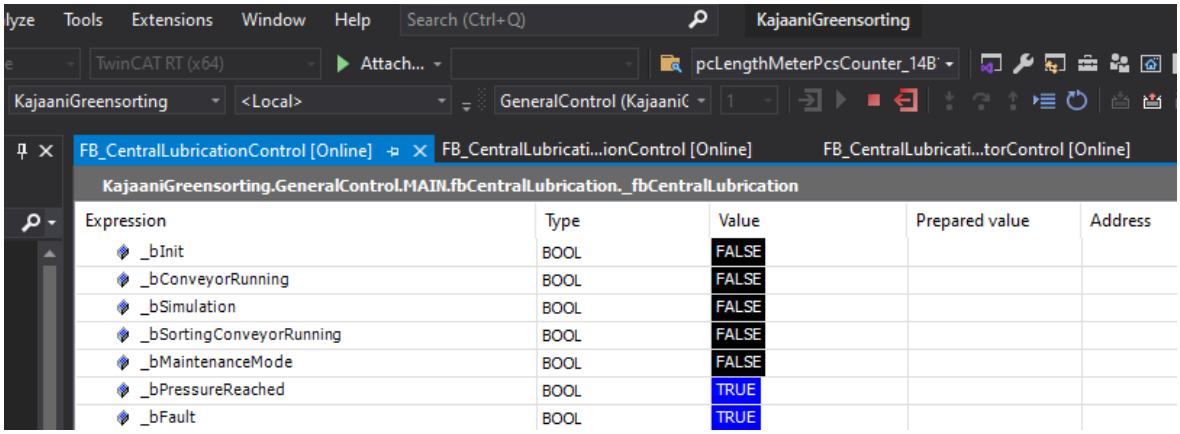
TwinCAT 3:n lokaali toiminta tarkoittaa, että ohjelma suoritetaan suoraan tietokoneessa ilman erillistä ohjelmisto-ohjainta tai erillistä laitteistoa (Kuva 29). TwinCAT 3:n lokaali toiminta mahdollistaa reaaliaikaisen suorituksen, joka on erittäin tärkeää monissa automaatiojärjestelmissä.

Lokaali toiminta TwinCAT 3:ssa toteutetaan käyttämällä Windows-käyttöjärjestelmän ominaisuuksia, kuten ydinkäsittelyyn varattua aikaa (real-time priority) ja alustan ajastintarkkuutta (high-resolution timer). TwinCAT 3:n lokaali toiminta vaatii myös asianmukaista tietokoneen laitteistoa, kuten reaaliaikaiseen suoritukseen soveltuvaa moniydinsuoritinta ja riittävää määrää RAM-muistia.

TwinCAT 3-NC Motion ohjelmisto mahdollistaa NC-liikkeiden ohjelmoinnin ja simulaation TwinCAT 3-ohjelmistoympäristössä. Toiminnolla voidaan simuloida ja testata NC-ohjelmia ennen niiden lähettämistä koneelle. Kun simuloidaan, pitää määrittää servon anturi simulaatioksi.

TwinCAT 3:n lokaali toiminta on tehokas ja nopea tapa toteuttaa automaatiojärjestelmiä, ja se on suosittu ratkaisu monissa teollisuussovelluksissa. TwinCAT 3 tarjoaa myös

mahdollisuuden toteuttaa järjestelmiä hajautetusti, jolloin eri laitteet voivat toimia yhdessä verkossa. (Beckhoff f.)



Kuva 29. Ohjelman pyörittäminen lokaalissa

6.3.2 Käyttöönotto laitoksella

Käyttöönoton suunnittelu on tärkeä vaihe automaatiojärjestelmän käyttöönotossa, ja sen huolellinen toteutus voi säästää aikaa ja vaivaa myöhemmissä vaiheissa. Asiakkaan kanssa on sovittu sopiva päivä käyttöönotolle, ja ennen käyttöönottoa on hankittu tarvittavat VPN-tunnukset, jotta päästään etänä ja paikan päällä ohjelman online-tilaan.

On myös tärkeää varmistaa, että kaikki laitoksen henkilökunta ja operaattorit ovat tietoisia käyttöönoton ajankohdasta ja prosessista. Tämä auttaa varmistamaan, että kaikki osapuolet ovat valmiita ja tietävät, mitä odottaa.

Ohjelman hardwarelle ja kirjastoihin on tehty muutoksia, ja siksi on tarpeen aktivoida konfiguraatio käyttöönoton yhteydessä. Aktivoinnin yhteydessä järjestelmä sammuu ja käynnistetään uudelleen. Tämä tarkoittaa ohjausjännitteen- ja turvapiirin pois tippumista. Ohjausjännite käynnistetään uudelleen ja turvapiiri kuitataan.

On hyvä käyttää aikaa järjestelmän testaamiseen ja tarkastamiseen ennen sen käyttöönottoa. Tämä auttaa varmistamaan, että kaikki järjestelmän komponentit toimivat oikein, ja että järjestelmä toimii halutulla tavalla.

Ensimmäisellä käynnillä saatiin ohjelma ladattua ja yhteys kamerajärjestelmään. Aluksi yhteydet eivät toimineet ja tilanne korjaantui, kun TwinCAT TCP/IP kirjastoon viitattiin. Tämän jälkeen järjestelmien kommunikoinnit testattiin ja saatiin molempiin järjestelmiin viestit lähetettyä ja vastaanotettua.

TCP/IP on yksi TwinCATin kirjastoista, joka mahdollistaa tietokoneiden ja laitteiden välisen kommunikoinnin IP-verkossa. TCP/IP-kirjaston käyttöön tarvitaan lisenssi, joka voidaan hankkia Beckhoffin verkkosivuilta.

Ilman lisenssiä TCP/IP-kirjastoa voi kuitenkin käyttää kokeiluversioina, mutta tällöin käyttöoikeus rajoittuu seitsemään päivään, minkä jälkeen lisenssi on aktivoitava uudelleen. Tämä voi olla hankalaa, jos järjestelmä vaatii jatkuvaa ja luotettavaa kommunikointia.

Toisella käyntikerralla huomattiin, että toisen kameran suunta oli muuttunut todennäköisesti jonkin laudan osuessa siihen. Kameroiden suojaamiseksi kehitetään lisäsuojauksia, jotta vältettäisiin vastaavat häiriöt jatkossa. Samalla käynnillä myös TwinCAT TCP/IP-lisenssi aktivoitiin järjestelmään, jotta sen käyttö ei vaatisi enää kokeiluversiota.

Kamerajärjestelmän kehitystyö on tällä hetkellä vaiheessa, jossa keskitytään hienosäätämään sen algoritmeja ja testaamaan järjestelmää jatkuvasti ilman, että häiritään laitoksen toimintaa. Kehitystyötä voidaan tehdä ”piilossa”, sillä laitoksen toiminta jatkuu normaalisti. Viestit liikkuvat edestakaisin ja mittausdataa tutkitaan kehitystyön edistämiseksi. Tavoitteena on kehittää entistä tarkempia ja luotettavampia algoritmeja, jotka vastaavat laitoksen tarpeita ja vaatimuksia. Jatkuvan testauksen ja kehityksen avulla pyritään varmistamaan järjestelmän optimaalinen toiminta ja luotettavuus.

Seuraava askel tulee olemaan purkureunojen automaattisen ohjauksen käyttöönottaminen. Kun mittausdata, viestien lähetys- ja vastaanottaminen on luotettavaa, voidaan ottaa purkureunojen automatisoitu ohjaus käyttöön ja seurata kuinka ohjaus toimii.

Lautojen epätasaisuudet voivat aiheuttaa haasteita. On tärkeää kehittää ohjelmaa, joka ottaa huomioon nämä epätasaisuudet ja pystyy korjaamaan niitä.

Lisäksi tarvitaan ratkaisuja siihen, miten mittausdataa hyödynnetään ja miten sitä käytetään ohjaamaan purkureunan servoja. Tämä voi vaatia esimerkiksi erilaisten algoritmien kehittämistä, jotta mittausdata saadaan käsiteltyä tehokkaasti ja tarkasti. Lisäksi on tärkeää testata järjestelmän toimintaa erilaisilla skenaarioilla, jotta voidaan varmistaa sen luotettavuus ja tehokkuus kaikissa tilanteissa.

Kaiken kaikkiaan kehitystyö mittausjärjestelmän parissa vaatii jatkuvaa testausta ja säätöä, jotta voidaan varmistaa sen toiminta kaikissa tilanteissa. Lautojen epätasaisuuksien aiheuttamien haasteiden ratkaiseminen on tärkeää järjestelmän toiminnan kannalta, ja se voi edellyttää uusien ratkaisujen kehittämistä ja ohjelman päivityksiä.

7 Yhteenveto

Opinnäytetyön tarkoituksena oli kehittää purkauslaitteen automatiikkaa, joka on vielä kesken ja odottaa lisää testauksia tuotantolaitoksella. Työn aikana on keskitytty erityisesti olio-ohjelmointimenetelmiin ja niiden hyötyihin purkauslaitteen automatiikan kehittämisessä.

Opinnäytetyön kautta on päästy syventymään sahateollisuuden automatiikkaan ja ymmärtämään sen toimintaperiaatteita paremmin. Työssä on käsitelty TwinCAT3-ohjelmointia ja sen hyötyjä automatiikan kehityksessä. Opinnäytetyö on myös auttanut ymmärtämään olio-ohjelmoinnin hyötyjä purkauslaitteen automatiikan kehittämisessä. Olio-ohjelmoinnin avulla voidaan suunnitella ja toteuttaa joustavaa ja modulaarista ohjelmistoa.

Yhteenvetona voidaan todeta, että opinnäytetyö on tärkeä panos purkauslaitteen automaation kehittämiseen ja sen avulla pyritään parantamaan laitteen tehokkuutta. Työn aikana on opittu paljon olio-ohjelmointimenetelmistä ja niiden hyödyistä.

Lähteet

Beckhoff a. Yritys. Viitattu 6.2.2023. Saatavissa https://www.beckhoff.com/fi-fi/company/

Beckhoff b. Scalable Industrial PC solutions. Viitattu 10.2.2023. Saatavissa https://www.beckhoff.com/fi-fi/products/ipc/pcs/c60xx-ultra-compact-industrial-pcs/

Beckhoff c. Object-oriented programming. Viitattu 5.3.2023. Saatavissa https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/2527303819.html&id=

Beckhoff e. C6030 Ultra-compact Industrial PC. Viitattu 12.3.2023. Saatavissa https://www.beckhoff.com/fi-fi/products/ipc/pcs/c60xx-ultra-compact-industrial-pcs/c6030.html

Beckhoff f. TwinCAT automation software. Viitattu 12.3.2023. Saatavissa https://www.beckhoff.com/en-en/products/automation/twincat/

Candtsolution. 2022. The Differences Between Industrial PCs and PLCs. Viitattu 12.2.2023. Saatavissa https://www.candtsolution.com/news_events-detail/differences-between-industrial-pc-and-plc/

EtherCAT Technology Group a. Why use EtherCAT?. Viitattu 10.3.2023. Saatavilla https://www.ethercat.org/en/why_use_ethercat.htm

EtherCAT Technology Group b. EtherCAT - the Ethernet Fieldbus. Viitattu 10.3.2023. Saatavilla https://www.ethercat.org/en/why_use_ethercat.htm

Graham, L. 2019. Modern Programming: Object Oriented Programming and Best Practices. 1. Painos. Birmingham: Packt Publishing

Jartek a. Yritys. Viitattu 2.2.2023. Saatavissa https://www.jartek.fi/yritys

Jartek b. Intranet. Viitattu 2.2.2023. Saatavissa rajoitetusti https://jartek2.sharepoint.com/SitePages/Yritysesittely.aspx

Kollmorgen Experts. 2020. How servo motors work. Viitattu 14.3.2023. Saatavissa https://www.kollmorgen.com/en-us/blogs/how-servo-motors-work/

Patolinna, O. 2023. TwinCAT 3 Extended. Kurssi Beckhoff Suomen pääkonttorissa 20.1.2023

Sahateollisuus. 2022. Pohjois-Savon päättäjät kokoontuivat sankoin joukoin Iisvedelle. Viitattu 5.2.2023. Saatavissa https://sahateollisuus.com/2022/10/25/pohjois-savon-paattajat-kokoontuivat-sankoin-joukoin-iisvedelle/

Sahateollisuuskirja a. 2023. Debarking. Viitattu 5.2.2023. Saativssa https://sahateollisuuskirja.fi/en/sahatavaran-valmistus/tukkien-kuorinta/

Sahateollisuuskirja b. 2023. Sawn timber manufacturing. Viitattu 5.2.2023. Saatavissa https://sahateollisuuskirja.fi/en/sahatavaran-valmistus/

Sahateollisuuskirja c. 2023. Grading plant. Viitattu 19.4.2023. Saatavissa https://sahateollisuuskirja.fi/en/sahatavaran-valmistus/sahatavaran-lajittelu-kuivauksen-jalkeen-ta-saamo/lajittelulaitosratkaisut/

Sheldon, R. 2023. method (in object-oriented programming). Viitattu 8.3.2023. Saatavissa https://www.techtarget.com/whatis/definition/method

Suhonen, J 2020. "Etsi kuvasta viisi virhettä", Konenäkö prosessinohjauksessa. Viitattu 8.3.2023. Saatavissa https://www.linkedin.com/pulse/etsi-kuvasta-viisi-virhett%C3%A4-konen%C3%A4k%C3%B6-janne-suhonen

Varis, R. 2017. Sahateollisuus. 2. painos. Helsinki: Kustannuspalvelut Kirjakaari Oy/Suomen Sahateollisuusmiesten Yhdistys ry