Jon Liiman

Älykäs automaattivarasto

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

27.11.2024

Tiivistelmä

Tekijä:	Jon Liiman
Otsikko:	Älykäs automaattivarasto
Sivumäärä:	19 sivua
Aika:	27.11.2024

Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Matti Välikylä, Lehtori

Insinöörityö toteutettiin Metropolia Ammattikorkeakoululle. Insinöörityön aiheena on Boschin opetuskäyttöön tarkoitetun laitteiston osittainen käyttöönotto ja sen toiminnallisuuden kehittäminen varastointitarkoitukseen. Insinöörityön tavoitteena oli suunnitella ja toteuttaa älykäs automaattivarasto, joka hyödyntää TwinCAT-ohjelmointiympäristöä ja Structured Text (ST) -ohjelmointikieltä.

Kehittämistyön tuloksena syntyi varastointiohjelma, joka mahdollistaa saumattoman yhteistyön nostimen, kuljettimen ja varastopaikkojen hallinnan välillä. Ohjelmaan toteutettiin toimintoja jotka parantavat järjestelmän tarkkuutta ja luotettavuutta. Lisäksi kehitettiin helppokäyttöinen käyttöliittymä, jonka avulla varastoa voidaan hallita intuitiivisesti.

Järjestelmä on modulaarinen ja dynaaminen, mikä mahdollistaa järjestelmän jatkokehityksen. Tulevia kehitysalueita ovat esimerkiksi konenäön integrointi objektien tunnistamista varten ja varastokapasiteetin kasvattaminen. Insinöörityö palvelee automaatiotekniikan opetustarkoituksia ja pienimuotoisia logistiikkaratkaisuja tarjoten samalla pohjan älykkäiden automaatiojärjestelmien kehitykselle.

Avainsanat:	automaatio, varastointi, Beckhoff, TwinCAT, Structured text, automaatiotekniikka

---

Abstract

Author:	Jon Liiman
Title:	Smart automated storaging unit
Number of Pages:	19 pages
Date:	27 November 2024
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Matti Välikylä, Lecturer

This Bachelor’s thesis was conducted for Metropolia University of Applied Sciences. The subject of the thesis was the partial commissioning and functional development of a Bosch educational device for storage purposes. The goal of the thesis was to design and implement an intelligent automated storage system utilizing the TwinCAT programming environment and Structured Text (ST) programming language.

The outcome of the project was a storage program that enables seamless collaboration between the lift, conveyor, and storage slot management. Functions were implemented to improve the system's precision and reliability. Additionally, a user-friendly interface was developed to allow intuitive control of the storage system.

The system is modular and dynamic, allowing for further development. Potential areas for future development include integrating machine vision for object recognition and increasing storage capacity. The system serves educational purposes and small-scale logistics solutions while providing a foundation for the development of intelligent automation systems.

Keywords: automation, storage, Beckhoff, TwinCAT, Structured Text, automation technology

Sisällys

Lyhenteet

HMI: Human Machine Interface. Ihmisen ja koneen välinen käyttöliittymä, jonka avulla käyttäjä voi olla vuorovaikutuksessa koneen tai prosessin kanssa.

I/O: Input/output. Tietojenkäsittelyssä I/O tarkoittaa logiikan ja logiikan ulkopuolisten asioiden kuten oheislaitteiden tai ihmisen välistä viestintää.

PLC: Programmable Logic Controller. Logiikka, pieni tietokone jota käytetään reaaliaikaisten automaatioprosessien ohjauksessa.

XAR: eXtended Automation Runtime. Reaaliaikainen ohjelmaympäristö, mistä voi suorittaa ohjelmakoodia, jolla ohjata laitteistoa.

1 Johdanto

Automaatio ja robotiikka ovat keskeisiä 2020-luvun teollisuudessa ja niiden rooli logistiikassa ja varastonhallinnassa on kasvanut merkittävästi viime vuosikymmeninä. Älykäs automaattivarasto, joka pystyy optimoimaan varastointiprosessit ja hakutoiminnot, tarjoaa yrityksille mahdollisuuden parantaa tehokkuutta, vähentää virheitä ja optimoida tilankäyttöä. Tämän insinöörityön tavoitteena on suunnitella ja toteuttaa älykäs automaattivarasto, joka hyödyntää TwinCAT-ohjelmointiympäristöä ja Structured Text (ST) -ohjelmointikieltä. Työn päätavoitteena on luoda järjestelmä, joka mahdollistaa kevyiden objektien varastoinnin ja varastosta hakemisen tehokkaasti käyttöliittymän kautta. Keskeisiä alitavoitteina ovat:

1. Ohjelmiston kehittäminen, joka mahdollistaa varastoinnin ja varastosta hakemisen käyttöliittymän avulla.
2. Nostimen ja muiden laitteiden integrointi toimivaksi kokonaisuudeksi.
3. Varaston käyttövirheiden estäminen ja ohjelmiston virheenkestävyyden varmistaminen.
4. Laitteiston jatkokehityksen mahdollistaminen tulevia käyttäjiä varten.

Näiden tavoitteiden saavuttamiseksi työssä hyödynnetään modernia automaatioteknologiaa, ja valitut työkalut, kuten TwinCAT ja ST-kieli, mahdollistavat laitteiston monimutkaisten toiminnallisuuksien tehokkaan hallinnan. Työ palvelee opetustarkoituksia ja tarjoaa pohjan laajemmalle automaatiosovellusten kehitykselle.

Työ on jaettu neljään pääosaan: ensin esitellään TwinCAT-ohjelmointiympäristö ja ST-kieli, sen jälkeen käydään läpi varaston toimilaitteet ja tehdyt muutokset, seuraavaksi tarkastellaan varastointiohjelman toteutusta, ja lopuksi tarkastellaan lopputulosta ja pohditaan jatkokehitysmahdollisuuksia.

2 Työssä hyödynnetty suunnitteluohjelmisto

Tässä luvussa esitellään työssä käytettyä suunnitteluohjelmistoa ja ohjelmointikieltä ja käsitellään miksi työ toteutettiin näitä käyttäen.

2.1 TwinCAT

TwinCAT on lyhenne sanoista ”The Windows Control and Automation Technology”. TwinCAT 3 on Beckhoffin kehittämä PC-pohjainen automaatioteknologian ohjelmistoalusta, joka mahdollistaa reaaliaikaisen automaatio-ohjauksen Microsoft Windows-käyttöjärjestelmissä. [1.]

Jos laitteella ei entuudestaan ole Visual Studiota TwinCATin asennuksen yhteydessä, myös Visual Studio ladataan tietokoneelle. TwinCAT toimii integroidusti Visual Studion kanssa ja mahdollistaa kehitystyön tutussa ja laajasti käytetyssä kehitysympäristössä.

TwinCAT mahdollistaa ohjelman kehittämisen ja testaamisen kehitysympäristössä maksutta viikoittain uusittavalla lisenssillä [2]. Jos haluaa käyttää kehitysympäristössä luotua ohjelmaa esimerkiksi teollisuuskäytössä tauotta, tulee lisenssistä tällöin maksaa.

Työ suoritettiin TwinCAT-ohjelmistoa käyttäen pääasiassa kahdesta syystä:

- Laitteistossa oli valmiiksi Beckhoffin logiikka, tarvittavat kirjastot ja koodipohja, jolla nostimen ohjaus oli jo mahdollista.

- Työn tekijällä oli aikaisempaa kokemusta TwinCATin käytöstä ja ST-kielestä.

Nämä seikat edesauttoivat laitteiston integroinnissa toimivaksi kokonaisuudeksi tavoitteiden mukaisesti.

2.2 ST-kieli ohjelmointikielenä

Structured Text (ST) on tekstuaalinen ja yksi viidestä IEC 61131-3 -standardin mukaisesta ohjelmointikielestä, jota TwinCAT PLC Control tukee. [3.]

Structured text koostuu sarjasta ohjeita, jotka voidaan suorittaa korkean tason kielissä (”IF..THEN..ELSE…”) tai silmukoissa (WHILE..DO) [3]. Kuvassa 1 esitetään ST-kielen syntaksin perusteet, jotka ovat keskeisiä varastointiohjelman logiikan toteutuksessa.

```
IF value < 7 THEN

    WHILE value < 8 DO

    value := value + 1;

    END_WHILE;

END_IF;
```

Kuva 1: Yksinkertainen esimerkki ST-kielen syntaksista. [3.]

ST on selkeä, helposti luettavissa ja sen rakenne muistuttaa korkean tason ohjelmointikieltä, mikä auttaa ohjelmoijia ymmärtämään koodia nopeasti. ST:n tukemat korkeamman tason ohjausrakenteet mahdollistavat joustavamman ja tehokkaamman ohjelmoinnin verrattuna graafisiin ohjelmointikieliin, kuten tikapuukaavio (LD) tai toimilohkokaavio (FBD)

Työn ohjelman keskeisimpiä toimintoja on CASE-rakenteella luotu ohjelmakulku. CASE-rakenteella voidaan luoda sekvenssi monivalintaratkaisulla, jossa muuttujien arvo määrittää mihin ohjelmalohkoon siirrytään. CASE-rakenne tekee siis koodista selkeämmän kuin useiden sisäkkäisten (”IF..THEN..ELSE...”) rakenteiden käyttö. Usein CASE-rakenteen lohkojen sisällä käytetään kuitenkin (”IF..THEN..ELSE…”) rakenteita.

3 Insinöörityössä käytetty laitteisto

Laitteisto on alun perin Boschin koulutus- ja messukäyttöön tarkoitettu, ja suurin osa komponenteista onkin Boschin omia. Laitteisto on lähes kokonaisuudessaan esillä kuvassa 2. Tässä osiossa käydään läpi laitteiston toimilaitteet ja komponentit, ja lopuksi tarkastellaan laitteistoon tehtyjä muutoksia.

3.1 Toimilaitteet ja komponentit

Laitteiston ohjaus – Beckhoff PLC

Laitteiston logiikkana toimi Beckhoff (Tuotekoodi CX5130) PLC. Logiikkaan on liitetty EtherCAT-teknologialla etä-I/O, johon on liitetty laitteiston muut toimilaitteet. PLC on vastuussa varastointijärjestelmän keskeisten laitteistokomponenttien ohjauksesta ja niiden toiminnan synkronoinnista. Se hallinnoi muun muassa nostimen liikkeitä, kuljettimen toimintaa ja varastopaikkojen hallintaa, mahdollistaen saumattoman yhteistyön laitteiston eri osien välillä.

Nostintoiminnallisuus – Bosch pneumaattinen sylinteri

Kappaleiden nostamisesta vastaa Boschin pneumaattisesti toimiva sylinteri (tuotekoodi 0 822 393 609). Tämä kaksitoiminen magneettinen sylinteri kykenee nostamaan objekteja imukuppia käyttämällä

Kelkkojen kuljetusjärjestelmä – Bosch liukuhihnamoduuli

Kelkkojen ja varastoitavien objektien liikkumista varastojärjestelmässä ohjaavat Boschin AS 1 liukuhihnamoduulit (tuotekoodi 3 842 999 759). Moduuliin integroitu epätahtimoottori pyörittää liukuhihnaa. Laitteistossa on käytössä kaksi moduulia, jotka mahdollistavat sujuvan kelkkojen kuljetuksen käyttäjän ja varaston välillä.

Kelkkojen pysäytys – Bosch erottaja

Boschin pneumaattiset erottajayksiköt (tuotekoodi 3 842 318 826) mahdollistavat kelkkojen pysäyttämisen liukuhihnalla. Kolme erottajaa mahdollistavat sujuvan ja tehokkaan varaston käytön sekä luotettavan kelkkojen kotiutussekvenssin.

Nostimen kuljetus – Beckhoff askelmoottori

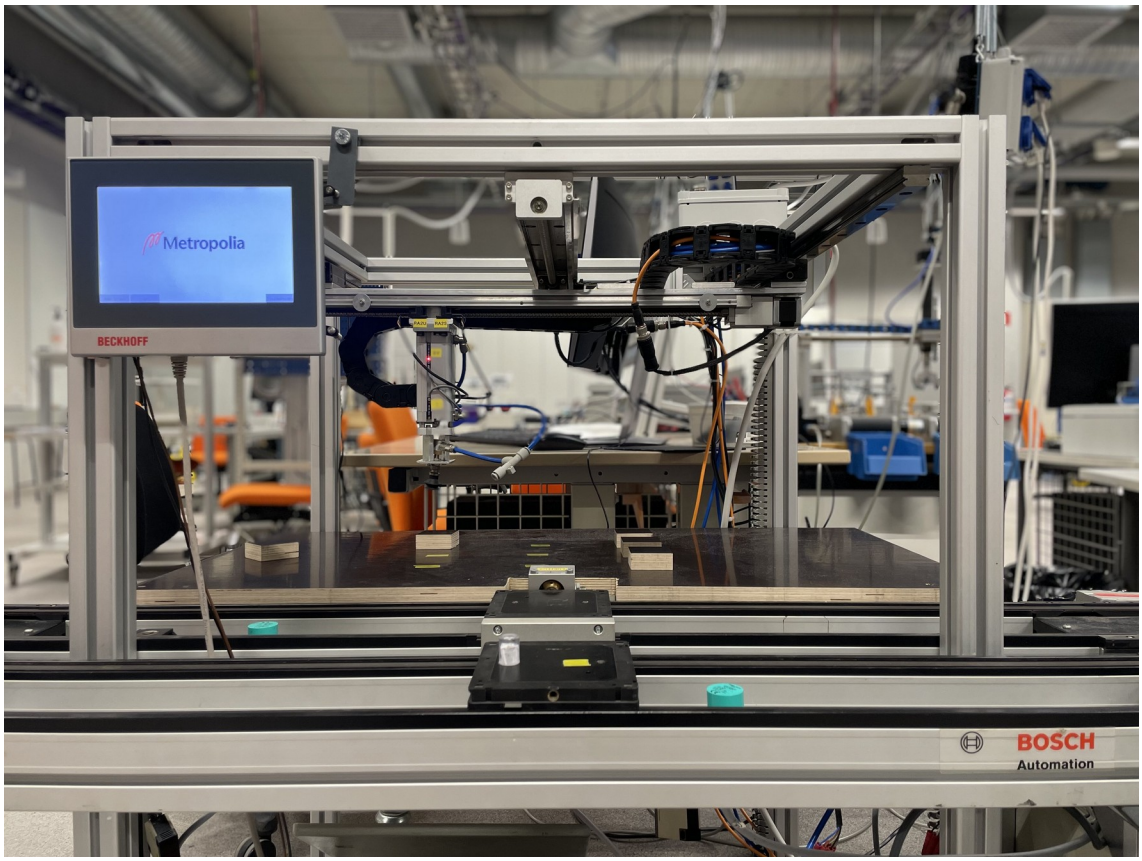
Beckhoffin askelmoottorit (tuotekoodi AS1020-0000) ohjaavat nostinta varaston ja liukuhihnan välillä. Askelmoottoreiden tarkkuus ja hienosäätö takaa sen, että objektit liikkuvat liukuhihnan ja varastopaikkojen välillä saumattomasti ja toistettavasti.

Kuljettimen tehon hallinta – ABB taajuusmuuttaja

ABB taajuusmuuttajat (tuotekoodi ACS50-01E-01A4-2) vastaavat kuljettimen moottorien ohjaamisesta. Kahta taajuusmuuttajaa käytetään molemmille liukuhihnamoduulien moottoreille. Liukuhihnan nopeus on asetettu vakioksi, sillä hihnan nopeutta ei ole tarpeen vaihtaa varastointitarkoituksessa.

Tunnistaminen ja sensoriikka – Pepperl+Fuchs kapasitiivinen anturi

Pepperl+Fuchs (tuotekoodi CJ10-30GM-E2) kapasitiiviset anturit tarjoavat äärimmäisen tarkan tunnistuksen liukuhihnalla kulkevista kelkoista. Laitteiston kolme sensoria varmistavat, että kelkkojen kuljetusjärjestelmä toimii luotettavasti.



Kuva 2: Kuva laitteistosta, jossa on esillä lueteltujen toimilaitteiden ja komponenttien lisäksi myös liukuhihnalla kulkevat kelkat, joilla objekteja kuljetetaan.

3.2 Laitteistoon tehdyt muutokset

Keskeisin muutos laitteistossa oli kelkkojen havaitsemiseen tarkoitettujen anturien vaihto optisista kapasitiivisiin. Optisen anturin käyttäminen oli laitteistossa ongelmallista kelkkojen ollessa mustia ja nostimen kuljettamisessa käytetyn kiskon ollessa heijastavaa metallia, sillä optinen anturi on hyvä havaitsemaan hyvin heijastavia pintoja ja ei niin hyvä havaitsemaan heikosti heijastavia pintoja [4]. Optisten anturien käyttö johti siihen, että anturit tunnistivat kelkkojen yläpuolella liikkuvan kiskon, jos niiden sensitiivisyys oli säädetty niin, että ne kykenivät tunnistamaan mustat kelkat. Toinen optisiin antureihin liittyvä ongelma laitteistossa on se, että ne tunnistavat herkästi myös varastoa operoivan henkilön kädet, joka voi johtaa varaston virheelliseen toimintaan.

Kapasitiivisia antureita käytetään tuotantolinjoissa objektien havaitsemiseen, ja laskemiseen. Kapasitiivisella anturilla on myös lyhyempi havaintokantama. [5.] Lyhyemmän havaintokantaman ansiosta kapasitiivinen anturi on mainio kelkkojen havaitsemiseen niin, että se ei havaitsisi kelkkojen päällä liikkuvaa metallista kiskoa eikä niin herkästi varastoa operoivan käyttäjän käsiä.

Kuljettimen käyttöönottoa varten kuljettimen taajuusmuuttajat tarvitsivat vain käskyn PLC:n I/O:lta liikkuakseen, sillä kuljettimen moottorit olivat jo valmiiksi yhdistetty taajuusmuuttajiin. Kuljetinhihna liikkui heti kelkkojen liikuttamiseen sopivalla nopeudella ja oikeaan suuntaan.

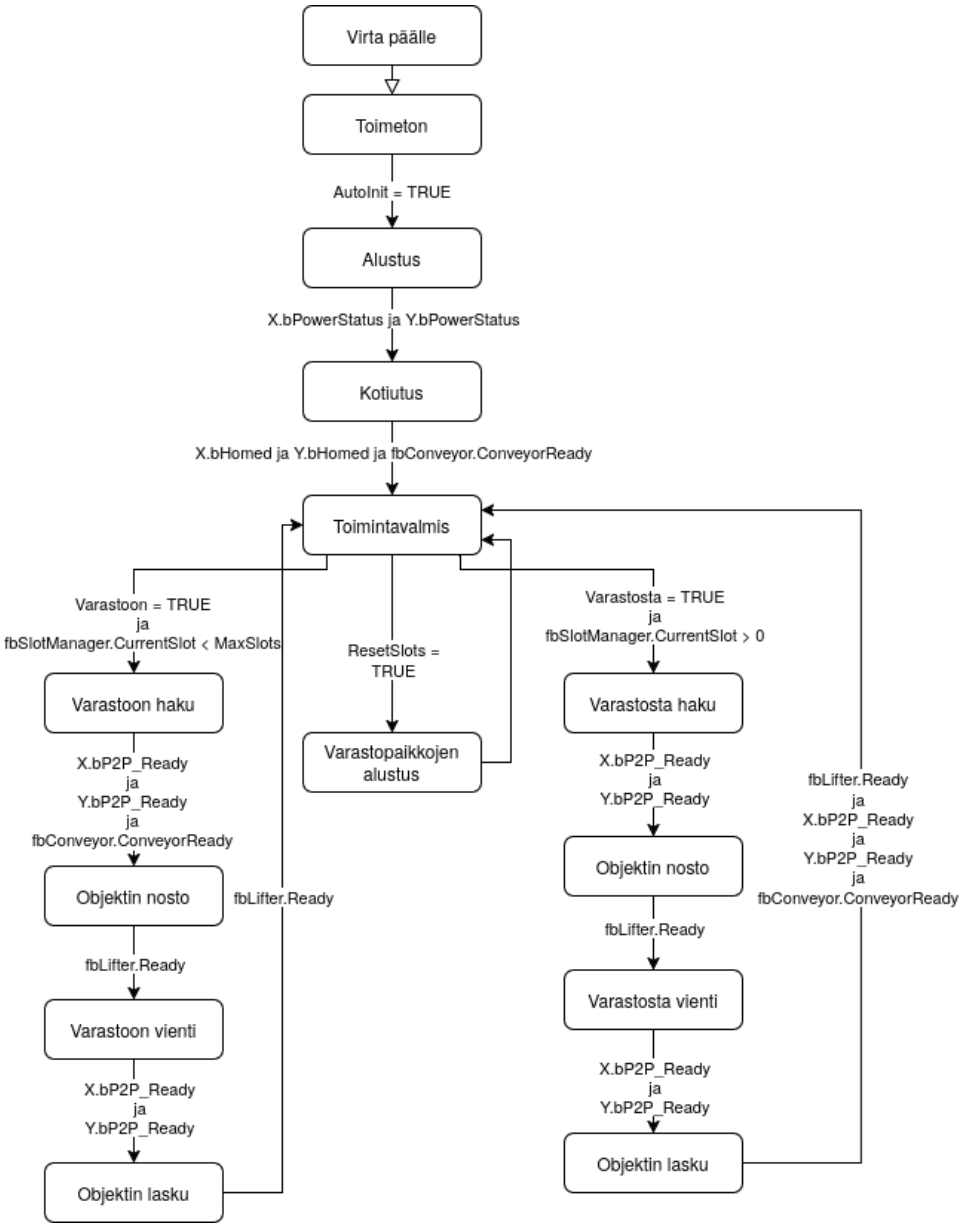
Nämä laitteistoon tehdyt muutokset mahdollistivat nostimen ja muiden laitteiden integroinnin toimivaksi kokonaisuudeksi tekemällä kelkkojen havaitsemisesta luotettavampaa ja toistettavampaa, ja edesauttoivat käyttövirheiden estämisessä tekemällä varastoa operoivan henkilön käsien tunnistamisesta epätodennäköisempää.

Laitteistoon lisättiin myös tietoliikennekytkin, joka mahdollistaa ohjelmiston kehityksen ilman laitteiston muokkaamista CX5130 PLC:ssä on vain kaksi tietoliikenneporttia [6]. PLC:n kahdesta portista yksi olisi varattu HMI:lle ja toinen etä-I/O:lle, jonka kautta on kytketty laitteiston I/O. Kytkimeen on kytketty PLC ja HMI, ja ohjelmistoa kehitettäessä siihen voidaan liittää tietokone. Kytkin mahdollistaa, että seuraavat käyttäjät voivat jatkaa kehitystyötä ilman, että laitteistoa tarvitsee muokata.

4 Varastointiohjelman kehitys

Varastointiohjelman kehitys on suoritettu TwinCAT 3 -ohjelmiston TwinCAT eXtended Automation Engineering (XAE) Shell -versiolla. Kehitystyö aloitettiin osittain jo valmiilla PLC-ohjelmalla, jossa nostimen ohjaus oli toiminnallisesti mahdollista. Varastointiohjelman kulku on helppo visualisoida kuvasta 3, jossa ohjelma etenee käynnistämisestä toimintavalmiuteen, josta voidaan valita haluttu etenemisvaihtoehto varaston toiminnalle.

Kehitystyön versiohallintaa varten työssä hyödynnettiin Git-versionhallintajärjestelmää GitHub-alustalla. Versiohallinta mahdollistaa koodin muokkausten seuraamisen ja paluun aikaisempiin versioihin. GitHub on alusta, jonka avulla kehittäjät voivat tehdä yhteistyötä samassa projektissa ja tallentaa koodinsa pilvipalveluun. [7.] Git on julkaistu avoimen lähdekoodin GNU General Public License -lisenssin version 2.0 mukaisesti [8].



Kuva 3: Varastointiohjelman vuokaavio. Ohjelma etenee vaiheesta toiseen, kun nuolien välissä määriteltyjen muuttujien arvot muuttuvat todeksi tai vastaavat nuolien yhteydessä esitettyjä ehtoja.

4.1 Kehitystyön vaiheet

4.1.1 Älykäs varastointiohjelma

Kun laitteistossa ei ole virtaa, esimerkiksi sähkökatkon vuoksi, katoaa nostimen askelmoottoreiden paikkatiedot. Nostin voisi siis viimeisen käyttökerran jälkeen olla keskellä varastoa, kun ohjelmakoodissa nostimen koordinaatit viittaisivat nostimen sijaitsevan jossain kulmassa. Ohjelmaa ajettaessa väärillä koordinaateilla nostin siis mitä todennäköisimmin ei toimi halutulla tavalla. Tätä varten nostimelle on kehitetty kotiutussekvenssi, joka suorittaa käynnistymisen yhteydessä ohjelmaketjun joka huolehtii, että askelmoottoreilla on sähkökatkon jälkeen oikeat paikkatiedot ohjelman toimivuuden takaamiseksi. Näin ohjelman toiminnasta saadaan mahdollisimman toistettavaa.

Askelmoottorit ovat liikkeissään erittäin tarkkoja, kun liikkeet ja pysähdykset ovat hallittuja. Anturien toiminta on myös toistettavampaa, kun havaittava objekti saapuu anturille hallitusti. Kuljetinta ei siis kannata ohjelmoida suorittamaan kotiutusta vauhdikkaasti. Nostin voitaisiin ohjelmoida liikkumaan antureita kohti mahdollisimman hitaasti, jotta kotiutus toimisi mahdollisimman tarkasti ja toistettavasti. Tämä on hidasta, sillä jos sähkövirta palautuu ja jos nostin sijaitsee esimerkiksi aivan päinvastaisessa kulmassa, johon kuljetin on tarkoitus kotiuttaa, niin nostimella kestää kauan liikkua antureiden luo. Kotiutus on siis parasta toteuttaa kaksivaiheisesti kuljettamalla nostin molemmilla akseleilla itsenäisesti ensin vauhdikkaammin antureita kohti, ja nostimen saapuessa antureiden kohdalle liikuttaa sitä hieman takaisinpäin ja palata antureille mahdollisimman hitaasti. Tällä tavalla kotiutus on mahdollisimman tehokas ja tarkka. Kuva 4 esittää kotiutussekvenssin koodin pääkohdat, joissa nostin tuodaan nopeasti antureiden lähelle ja kotiutuksen hienosäätö suoritetaan hitaammilla liikkeillä.

```
CASE eHomeState OF
    E_HomeState.Inactive:
         IF bHomeCmd THEN
             eHomeState := E_HomeState.Jog;
         END_IF

    E_HomeState.Jog:
        bJogBackward:=TRUE;
        IF bHomeCalibrationCam THEN
            bJogBackward:=FALSE;
            eHomeState := E_HomeState.Delay;
        END_IF

    E_HomeState.Delay:
        IF tonHomeDelay.Q THEN
            eHomeState := E_HomeState.SecondPhase;
        END_IF

    E_HomeState.SecondPhase:
        IF bHomeCalibrationCam THEN
            rJogVelocity := 15;
            bJogForward:= TRUE;
            eHomeState := E_HomeState.SecondDelay;
        END_IF

    E_HomeState.SecondDelay:
        IF tonSecondDelay.Q THEN
            bJogForward:= FALSE;
            eHomeState := E_HomeState.FinalHoming;
        END_IF

    E_HomeState.FinalHoming:
        rJogVelocity := 1;
        bJogBackward:= TRUE;
        IF bHomeCalibrationCam THEN
            bJogBackward:= FALSE;
            rJogVelocity := 10;
            eHomeState:= E_HomeState.FinalDelay;
```

Kuva 4: Case-rakenteella luotu sekvenssi nostimen kotiuttamiseksi.

Kuljetin integroitiin toimimaan nostimen kanssa yhteistyössä luomalla ohjelmaan toimilohko ”FB_Conveyor”, jonka tarkoitus on ohjelman käynnistyessä kotiuttaa liukuhihnalle asetetut kelkat varaston käytön mahdollistaville paikoille ja kuljettaa pääohjelman käskystä kelkkoja varaston ja varaston käyttäjän välillä. Kuva 5 esittää FB_Conveyor-toimilohkon koodin, jossa käyttäjä valitsee käyttöliittymän kautta, halutaanko varastoida objekti vai hakea varastosta objekti.

```
E_ConveyorState.ToiminnanValinta:
    ConveyorReady := TRUE;
    IF VarastostaHMI = TRUE THEN
        bP1 := FALSE;
        bConveyor := TRUE;
        ConveyorReady := FALSE;
        bP2 := TRUE;
        IF bA2Sensor = TRUE THEN
            bConveyor := FALSE;
            ConveyorReady := TRUE;
            ConveyorState := E_ConveyorState.ToimintaVarastosta;
            VarastostaHMI := FALSE;
        END_IF
    END_IF
    IF VarastoonHMI = TRUE THEN
        bP1 := FALSE;
        bP3 := FALSE;
        bConveyor := TRUE;
        ConveyorReady := FALSE;
        IF bA1Sensor = TRUE THEN
            ConveyorState := E_ConveyorState.ToimintaVarastoon;
            VarastoonHMI := FALSE;
        END_IF
    END_IF
```

Kuva 5: FB_Conveyor-toimilohkon koodia. Toimilohko on toteutettu case-rakenteella, ja valinnasta riippuen siirrytään sekvenssin seuraavaan osaan.

Varaston organisointia varten ohjelmaan luotiin toimilohko ”FB_SlotManager”, jonka tarkoitus on huolehtia varaston täyttämisestä ja tyhjentämisestä. Arkkitehtuurisesti FB_SlotManager-toimilohkossa on kolme eri metodia, mitä voidaan pääohjelmassa kutsua ja toimilohko palauttaa pääohjelmalle vain sen varastopaikan numeron, mihin objekti varastoidaan tai mistä varastopaikasta objekti noudetaan varastosta. Varastopaikkojen numeroille on määritelty koordinaatit. Toimilohkon metodit ovat objektin varastoimiseen, objektin noutamiseen ja varaston alustamiseen. Kuva 6 esittää toimilohkon Stack-metodin, joka palauttaa seuraavan käytettävän varastopaikan numeron pääohjelmalle.

Toimilohkon varastointitoiminto on suunniteltu siten, että se voi varastoida kolme objektia päällekkäin yhteen varastopaikkaan. Tämä mahdollistaa tilan säästämisen ja varastokapasiteetin tehokkaan käytön. Varastopaikkoja on ohjelmoitu varastoon 9 kappaletta, mutta tilaa varastossa olisi moninkertaiselle määrälle, jos objekteja olisi tarve varastoida enemmän.

Varastosta hakeminen tapahtuu samalla periaatteella kuin varastointi: objekti otetaan pois varastosta samassa järjestyksessä, kuin se on sinne asetettu. Varastohakutoiminto on optimoitu, jotta hakuprosessi olisi mahdollisimman nopeasti toistettavissa, ja se pystyy palauttamaan objektit ilman virheitä.

Varaston käyttövirheiden estämiseksi ohjelmaan on kehitetty tapa estää laitteistoa hakemasta tyhjästä varastosta objektia, sekä täyttämästä varastoa yli sen kapasiteetin. Näissä tapauksissa käyttöliittymä ilmoittaa käyttäjälle ongelmatilanteesta. Varastoa voi kuitenkin käyttää useammilla tavoilla väärin kuin on helposti ennustettavissa. Niitä tapauksia varten, joissa käyttäjän virheellinen käyttö tai ohjelmointivirhe ohjelmakoodissa aiheuttaa varaston organisoinnissa ongelmia, on kuitenkin kehitetty toiminto varaston alustamiselle. Varaston alustaminen tyhjentää laitteiston muistin varastoiduista objekteista.

```
// Method is repeated if current slot is full
WHILE NOT ConditionMet DO
// Check if there is space, when MAX_SLOTS represents the total amount of slots in the storaging system.
IF (CurrentSlot <= MAX_SLOTS) THEN
    // coords[CurrentSlot].n_max represents the maximum amount of objects that can be stored in the specific(CurrentSlot) slot. n_max is 3 for all storage slots and 0 for the sled's slot.
    IF (coords[CurrentSlot].n < coords[CurrentSlot].n_max) THEN
        coords[CurrentSlot].n := coords[CurrentSlot].n +1;
        ReturnCoordinates := CONCAT('Stored in slot ', INT_TO_STRING(CurrentSlot));
        ConditionMet := TRUE;
    ELSE
        // Move to the next slot if the current one is full
        CurrentSlot := CurrentSlot + 1;
    END_IF
ELSE
    ReturnCoordinates := 'All slots are full';
    ConditionMet := TRUE;
END_IF
END_WHILE
```

Kuva 6: Esimerkki "Stack" metodista FB_SlotManager toimilohkossa. Metodi varastosta hakemiseen toimii samalla periaatteella.

Nostimen, kuljettimen ja varastointiorganisoinnin toimilohkoja kutsutaan ja hallitaan ”Motion”-pääohjelmasta. Tavoitteena oli pitää pääohjelma loogisena, organisoituna ja ymmärrettävänä käyttämällä toimilohkoja, jotta seuraavan kehittäjän olisi mahdollisimman vaivatonta jatkaa laitteiston kehittämistä. Kuva 7 esittelee pääohjelman rakennetta, jossa varastointiohjelman toimilohkot integroidaan yhtenäiseksi prosessiksi.

```
E_Mode.hmi:
    CASE eState OF
        E_State.inactiveHMI:
            IF AutoInit THEN
                AutoInit:= FALSE;
                X.PowerOn();
                Y.PowerOn();
                eState := E_state.initialisingHMI;
            END_IF

        E_State.initialisingHMI:
            IF X.bPowerStatus AND Y.bPowerStatus THEN
                X.FindHome();   //Begins the homing cycle for the lifter
                Y.FindHome();   //Same thing as above but for the other axel.
                fbConveyor.HomeConveyor();  //Begins the homing of the conveyor
                eState := E_state.homingHMI;
            END_IF

        E_State.homingHMI:
            IF X.bHomed AND Y.bHomed AND fbConveyor.ConveyorReady THEN  //When homing sequences are finished, the storaging unit is ready to be used.
                eState := E_state.readyHMI;
            END_IF

        E_State.readyHMI:
            IF Varastoon = TRUE THEN
                Varastoon := FALSE;
                fbSlotManager.Stack();  // Stack method of FB_SlotManager is called which returns "CurrentSlot" variable which is the slot to which the object is to be stored in
                IF fbSlotManager.CurrentSlot < 10 THEN  //If the slot that the fb_slotmanager returns is 10, it means the storage is full and we don't want the storaging sequence to begin.
                    fbConveyor.KplVarastoonHMI();  // FB_Conveyor is called to start the stacking sequence
                    X.P2P(Slots.Coords[SlotSledX].x);   //Tämä on paikka, jonka päälle mennään odottelemaan kelkkaa josta poimia palikka
                    Y.P2P(Slots.Coords[SlotSledY].y);   // This is the place on top of which the lifter goes to wait for the sled with an object
                    eState := E_state.move2pickHMI;
                ELSE
                    eState := E_State.readyHMI;
                END_IF
            END_IF
            IF Varastosta = TRUE THEN
                Varastosta := FALSE;
                fbSlotManager.Retrieve();  // Retrieve method of FB_SlotManager is called which returns "CurrentSlot" variable which is the slot from which the object is retrieved from.
                IF fbSlotManager.CurrentSlot > 0 THEN  //If the slot that the fb_slotmanager returns is 0, the storage is empty and we don't want the retrieving cycle to begin.
                    fbConveyor.KplVarastostaHMI();  // FB_Conveyor is called to start the retrieving sequence
                    X.P2P(Slots.Coords[fbSlotManager.CurrentSlot].x);  // Tämä on paikka, mistä palikka haetaan varastosta
                    Y.P2P(Slots.Coords[fbSlotManager.CurrentSlot].y);  // This is the place in storage from which the object is to be retrieved.
                    eState := E_state.move2pickHMIvarastosta;
                ELSE
                    eState := E_state.readyHMI;
                END_IF
            END_IF
            IF ResetSlots = TRUE THEN
                ResetSlots := FALSE;
                fbSlotManager.Reset();
            END_IF
```

Kuva 7: Osa "Motion"-pääohjelman koodia. Laitteisto saatetaan ensin käyttövalmiuteen, jonka jälkeen ohjelma jää odottamaan käyttäjän käskyjä.

4.1.2 Käyttöliittymän toteutus käyttäen PLC HMI -laajennusta

Yksi insinöörityön päätavoitteista oli saada varastointiohjelma ohjattavaksi käyttöliittymän kautta, ja se onnistui käyttämällä TwinCAT runtime PLC HMI -laajennusta. Käyttöliittymänä työssä käytettiin Beckhoff CP2607-0000 Panel PC:tä, jossa on TwinCAT 3.1 XAR ja PLC HMI -laajennus asennettuna. Paneelissa on asennettuna Windows Compact Embedded -käyttöjärjestelmä: CE lyhennettynä. Paneeli ja PLC on yhdistetty kommunikoimaan RJ45-kaapelilla.

PLC HMI:n kanssa on mahdollista ohjata PLC-ohjelman visualisaatiota käyttöliittymästä tai joltain muulta tietokoneelta [9]. Tätä varten PLC-ohjelmassa täytyy olla luotuna VISU-objekti ja TargetVisualization, joka mahdollistaa PLC HMI:n käytön. VISU-objektiin luodaan käyttöliittymään halutut ruudut, joista ohjelmaa ohjataan.

Kun käyttöliittymä on paneeli tai jokin muu tietokone, jossa on oma käyttöjärjestelmä, täytyy PLC-ohjelman ohjaamiseksi käyttöliittymästä suorittaa muutamia lisävaiheita ja tarkistuksia fyysisen yhdistämisen lisäksi.

- TwinCAT 3 Build 4018.0 ADS tai uudempi tulee olla asennettuna paneelille tai käytettävälle tietokoneelle [9].

- ADS-yhteys on luotu PLC:n ja paneelin tai käytettävän tietokoneen välille. Tämä tehdään TwinCAT icon → Router → Edit Routes kautta [9].

- Tc3PlcHmi kansio tulee kopioida ohjelmointilaitteelta tai PLC:ltä paneelille tai käytettävälle tietokoneelle. Tämä tulee tehdä manuaalisesti. [9.] Käytännössä ”Visu”-kansion kopiointi paneelille tai käytettävälle tietokoneelle riittää.

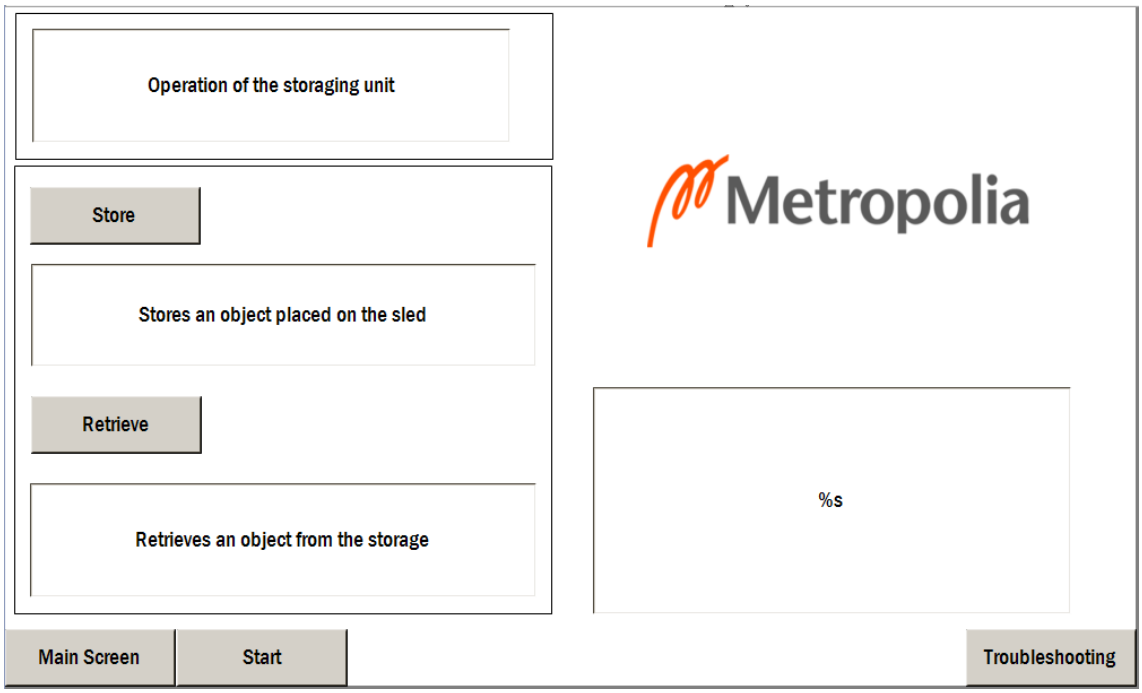
- Tc3PlcHmi.ini-tiedosto pitää mukauttaa paneelilla tai käytettävällä tietokoneella vähintäänkin niin, että siinä ilmoitetaan PLC:n AmsNetID ja sen visualisointiruudun nimi, joka halutaan paneelilla tai käytettävällä tietokoneella käynnistää [9].

- PLC HMI -ohjelma tulee vielä käynnistää joko manuaalisesti tai asettaa se käynnistymään automaattisesti riippuen käyttöjärjestelmästä, joka paneelilla tai käytettävällä tietokoneella on. Työssä käytetty paneeli toimii CE-käyttöjärjestelmällä, joten käyttöjärjestelmän Start Manager -ohjelman kautta PLC HMI voidaan asettaa käynnistymään automaattisesti Shell-komennolla, jossa on PLC HMI -ohjelman polku. [9.]

Kun käyttöliittymä on paneeli jossa ei ole omaa käyttöjärjestelmää, TargetVisualization käynnistää visualisaation paneelilla automaattisesti, kun ohjelmaa ajetaan ja paneeli on yhdistetty PLC:hen [9].

Käyttöliittymään on luotu ”Troubleshooting”-ruutu yleisimpien ongelmien ratkaisuun. Käyttäjä kykenee navigoimaan tälle ruudulle mistä tahansa käytettävästä käyttöliittymän ruudusta kohdatessaan ongelmatilanteen varaston käytössä.

”Start”-ruudulta ohjataan varastoa. Ruudulle päivittyy myös luettava esitys varastoinnin vaiheista varastoa käytettäessä. Kuvassa 8 esitellään pääasiallinen käyttöliittymän ruutu, josta varastoa ohjataan.



Kuva 8: Esimerkki käyttöliittymän ”Start”-ruudusta, josta varastoa ohjataan. Ohjelman käynnistyttyä %s kohtaan päivittyy luettava selitys varastoinnin vaiheista, kun jotain toimintoa käytetään; esimerkiksi "Storing to slot 1".

4.2 Varastointiohjelman merkitys tavoitteiden saavuttamisessa

Varastointiohjelman kehitys onnistui insinöörityön tavoitteiden saavuttamisessa kaikilta osin:

1. Varastointiohjelma mahdollistaa varastoinnin ja varastosta hakemisen käyttöliittymää käyttäen. Käyttäjä voi suorittaa kaikki keskeiset toiminnot intuitiivisesti, mikä parantaa järjestelmän käytettävyyttä.

2. Varastointiohjelma integroi laitteiston yhtenäiseksi kokonaisuudeksi. Pääohjelma integroi nostimen, kuljettimen ja varaston organisoinnin toimilohkot toimimaan yhteistyössä.

3. Varastointiohjelman toteutuksessa on keskitytty käyttövirheiden estämiseen. Ohjelmaan toteutetut varaston tarkistusmekanismit estävät virhetilanteet, kuten varaston ylitäytön tai tyhjien paikkojen hakemisen. Tämä tekee järjestelmästä luotettavan ja vähentää

käyttäjän tekemien virheiden vaikutuksia. Käyttöliittymän ongelmanratkaisuun tarkoitettu ruutu helpottaa ongelmien ratkaisemisessa.

4. Varastointiohjelman toteutuksessa on pyritty mahdollistamaan sujuva ohjelman jatkokehitys tulevia käyttäjiä varten. Ohjelman modulaarinen rakenne tukee järjestelmän skaalautuvuutta. Ohjelman toimilohkot on suunniteltu siten, että niiden laajentaminen ja päivittäminen on suoraviivaista, mikä tekee järjestelmästä pitkäikäisen ja helposti muokattavan tulevia tarpeita varten.

Alkuperäisten tavoitteiden lisäksi varastoon luotiin varastopaikka-rakenne, joka mahdollistaa useiden objektien varastoimisen samoihin varastopaikkoihin ja varastosta objektien noutamisen järjestelmällisesti.

5 Lopputulos ja pohdinta

5.1 Insinöörityön saavutukset ja kokonaisarviointi

Tässä insinöörityössä toteutettiin sille asetetut tavoitteet ja työ tuotti toimivan älykkään automaattivaraston, joka hyödyntää TwinCAT-ohjelmointiympäristöä ja ST-ohjelmointikieltä. Järjestelmä mahdollistaa sujuvan varastoinnin ja varastosta haun. Järjestelmä on suunniteltu kestämään käyttövirheiden aiheuttamia virhetilanteita.

Tämän insinöörityön kehittämisprosessin aikana luotu kotiutussekvenssi ja varaston tarkistusmekanismien toteuttaminen paransivat järjestelmän luotettavuutta. Käyttöliittymästä hallittavat toiminnot tekevät järjestelmän käytöstä intuitiivista, mikä vähentää käyttäjien koulutuksen tarvetta. Lisäksi järjestelmä on modulaarinen, mikä helpottaa sen jatkokehitystä ja mukauttamista tuleviin tarpeisiin. Järjestelmän modulaarisuus ilmenee sen rakenteessa, jossa toiminnallisuudet on jaettu selkeisiin toimilohkoihin, kuten esimerkiksi ”FB_Conveyor” ja ”FB_SlotManager”. Tämä mahdollistaa yksittäisten osien päivittämisen tai laajentamisen ilman vaikutuksia muihin järjestelmän osiin.

Kokonaisarvioinnissa järjestelmä täytti sille asetetut tekniset ja toiminnalliset vaatimukset. Ohjelmisto on luotettava. Laitteiston käytettävyys tukee sen käyttöä opetustarkoituksissa tai pienimuotoisissa logistiikkaratkaisuissa. Insinöörityön onnistumisen taustalla oli perusteellinen suunnittelu- ja testausvaihe, joka varmisti, että kaikki keskeiset komponentit toimivat toimivat sujuvasti ja tarkoituksenmukaisesti.

5.2 Jatkokehityksen mahdollisuudet

Järjestelmässä on useita jatkokehitysmahdollisuuksia, jotka voisivat laajentaa sen toiminnallisuuksia ja käyttömahdollisuuksia:

1. Konenäön integrointi

Yksi mahdollinen jatkokehityksen suunta on konenäön liittäminen järjestelmään. Konenäkö mahdollistaisi objektien tunnistamisen niiden ominaisuuksien, kuten värin tai koon, perusteella. Tämä laajentaisi järjestelmän käyttötarkoituksia merkittävästi. Esimerkiksi järjestelmä voisi järjestellä objektit automaattisesti varastopaikkoihin niiden värin perusteella tai noutaa halutun värisen objektin myös tilanteissa, joissa se on muiden objektien alla. Tämä edellyttäisi laitteiston lisäpäivityksiä, kuten kameroiden ja niiden hallintaan tarvittavan ohjelmiston lisäämistä.

2. Varastointikapasiteetin lisääminen

Nykyinen järjestelmä mahdollistaa yhdeksän varastopaikkan käytön, mutta varastopaikat on ohjelmoitu väljästi siten, että kapasiteettia voidaan helposti halutessa laajentaa. Ohjelmaan tulisi lisätä varastopaikkoja ja varaston organisoinnin toimilohkoa voisi optimoida suurempien kokonaisuuksien käsittelyyn.

3. Ulkoisten järjestelmien integrointi

Tämänkaltaisen järjestelmän voisi tulevaisuudessa liittää yrityksen muihin tietojärjestelmiin, kuten ERP-järjestelmiin. Tämä mahdollistaisi varastointitiedon reaaliaikaisen siirron logistiikkajärjestelmiin, mikä helpottaisi resurssien hallintaa ja optimointia suuremmissa kokonaisuuksissa.

4. Käyttöliittymän laajennukset

Käyttöliittymää voidaan kehittää lisäämällä esimerkiksi reaaliaikainen varastokartta, joka näyttää varastopaikkojen tilanteen visuaalisesti. Tämä voisi parantaa käyttäjän kokemusta ja tehdä järjestelmän käytöstä entistä sujuvampaa.

5.3 Yhdistäminen laajempaan jatkumoon

Tämä insinöörityö tarjoaa paitsi käytännön ratkaisun varastointiin myös hyvän pohjan opetustarkoituksiin ja automaatiotekniikan opiskeluun. Järjestelmän avulla voidaan havainnollistaa automaation keskeisiä periaatteita, kuten ohjelmiston ja laitteiston välistä vuorovaikutusta, käyttöliittymän merkitystä ja virheenkestävyyden tärkeyttä.

Laajemmassa kontekstissa järjestelmän toteutus liittyy nykyaikaisiin logistiikkaratkaisuihin, joissa pyritään tehostamaan varastointia ja vähentämään manuaalista työvoimaa. Järjestelmä voisi soveltua pieniin ja keskisuuriin logistiikkakeskuksiin, joissa tarvitaan yksinkertainen ja kustannustehokas varastointiratkaisu.

Työ osoittaa myös, että TwinCAT-ympäristö ja ST-kieli ovat tehokkaita työkaluja teollisuuden automaatiojärjestelmien kehittämisessä. Näitä teknologioita hyödyntämällä voidaan luoda joustavia ja modulaarisia ratkaisuja, jotka mukautuvat helposti muuttuviin tarpeisiin.

5.4 Huomioitavaa seuraavalle kehittäjälle

Seuraavan kehittäjän tulisi kiinnittää huomiota seuraaviin seikkoihin:

- Ohjelmiston modulaarisuuden ylläpito: Kaikki lisäykset ja muutokset tulisi suunnitella olemassa olevien toimilohkojen rakenteen mukaisesti, jotta järjestelmän skaalautuvuus ja yksinkertaisuus säilyy.

- Konenäön integrointi: Jos järjestelmään lisätään konenäkö, kehittäjän tulee huolehtia sen integroinnista ohjelman toimilohkojen kanssa ja sen synkronoinnista laitteiston toiminnan kanssa. Sopivia työkaluja voivat olla esimerkiksi OpenCV-kirjasto tai TwinCAT Vision.

- Testauksen merkitys: Järjestelmään tehtävät lisäykset on testattava huolellisesti todellisessa käytössä, jotta varmistetaan niiden yhteensopivuus nykyisten toimintojen kanssa.

6 Yhteenveto

Tässä insinöörityössä suunniteltiin ja toteutettiin älykäs automaatiovarasto, joka hyödyntää TwinCAT-ohjelmointiympäristöä ja Structured Text (ST) -ohjelmointikieltä. Insinöörityön

tavoitteena oli luoda järjestelmä, joka mahdollistaa varastoinnin ja varastosta hakemisen käyttöliittymän kautta. Tavoitteet saavutettiin, ja kehittämisprosessin aikana tuotettiin moderni järjestelmä, joka tarjoaa luotettavia ja tehokkaita varastointiratkaisuja.

Keskeisiä insinöörityön tuloksia olivat:

- Toiminnallisen varastointiohjelman kehittäminen, joka integroi laitteiston komponentit, kuten nostimen, kuljettimen ja varaston hallintatoiminnot, toimivaksi kokonaisuudeksi.

- Kotiutussekvenssin ja virheenkäsittelymekanismien toteuttaminen, jotka paransivat järjestelmän tarkkuutta ja virheenkestävyyttä.

- Helppokäyttöisen käyttöliittymän suunnittelu ja implementointi, joka tekee järjestelmän käytöstä intuitiivista.

Insinöörityö tarjoaa myös hyvät mahdollisuudet laitteiston jatkokehitykselle. Laitteistoa voidaan laajentaa esimerkiksi lisäämällä konenäköominaisuuksia, jotka mahdollistavat objektien lajittelun ja hakemisen niiden ominaisuuksien perusteella. Lisäksi varastokapasiteettia voidaan kasvattaa, ja järjestelmä voidaan integroida ulkoisiin tietojärjestelmiin, kuten ERP-järjestelmiin.

Insinöörityö palvelee erinomaisesti opetustarkoituksia ja tarjoaa käytännön ratkaisun varastonhallinnan ja automaation tarpeisiin. Varastointijärjestelmän suunnittelussa käytetyt teknologiat, kuten TwinCAT ja ST-kieli, osoittautuivat tehokkaiksi työkaluiksi teollisuuden automaatioratkaisujen toteuttamisessa. Kokonaisuutena insinöörityö edistää älykkäiden logistiikkaratkaisujen kehitystä ja tarjoaa pohjan tuleville projekteille automaatioalalla.

Lähteet

1 TwinCAT Automation software. Verkkoaineisto. Beckhoff. <https://www.beckhoff.com/en-en/products/automation/twincat/>. Luettu 01.10.2024

2 TwinCAT Licensing. Verkkoaineisto. Beckhoff. <https://www.beckhoff.com/en-en/products/automation/twincat/twincat-3-licensing/>. Luettu 07.10.2024

3 TwinCAT Documentation. Verkkoaineisto. Beckhoff. <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/925248523.html&id=>. Luettu 07.10.2024

4 Optical Proximity Sensors: How They Work and Their Applications. Verkkoaineisto. <https://www.electricity-magnetism.org/optical-proximity-sensor/>. Luettu 26.11.2024

5 Capacitive Proximity Sensors: Understanding the Basics and Applications. Verkkoaineisto. <https://www.electricity-magnetism.org/capacitive-proximity-sensor/>. Luettu 26.11.2024

6 Products. Verkkoaineisto. Beckhoff. <https://www.beckhoff.com/en-us/products/ipc/embedded-pcs/cx5100-intel-atom-r/cx5130.html>. Luettu 26.11.2024

7 What is Git? Verkkoaineisto. GitHub. <https://github.blog/developer-skills/programming-languages-and-frameworks/what-is-git-our-beginners-guide-to-version-control/>. Luettu 14.11.2024

8 About – Free and Open Source. Verkkoaineisto. Git. <https://git-scm.com/about/free-and-open-source>. Luettu 14.11.2024

9 TwinCAT 3 | PLC. Verkkoaineisto. Beckhoff. <https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/136113291.html&id=>. Luettu 12.11.2024