Examensarbete

HÖGSKOLAN I SKÖVDE

EMULERING AV EN PRODUKTIONCELL MED VISIONGUIDNING

Virtuell idrifttagning

EMULATION OF A PRODUCTIONCELL INCLUDING ROBOT VISION

Virtual Commissioning

Examensarbete inom huvudområdet automation
Grundnivå 30 Högskolepoäng
Vårtermin År 2017

Johan Einevik
John Kurri

Handledare: Mikel Ayani
Examinator: Amos Ng

Äkthetsintyg

Denna examensrapport är inlämnad av Johan Einevik och John Kurri till Högskolan i Skövde för examen vid institutionen för Ingenjörsvetenskap. Härmed intygas att allt material i denna rapport är vårt eget. Tydliga referenser ges till material som hämtats från annat håll.

……
Ort och datum

……
Johan Einevik

……
John Kurri

Förord

Vi skulle vilja tacka de personer som varit involverade i projektet och bidragit med sin expertis inom de områden som krävts. Ett stort tack till elektroautomatik som bidragit med allt ifrån kompetens till programvaror och hårdvaror vilket underlättat projektet avsevärt. Tack till vår handledare Samuel på Elektroautomatik som har tillgodosett alla de önskemål vi lyft fram och bidragit till en välkomnande atmosfär samt inspirerande intresse för projektet.

En person som vi vill lyfta fram lite extra och tacka är vår handledare Mikel, för både engagemang och tillmötesgående av de ändringar vi behövt i emuleringsmjukvaran. Vi vill också tacka vår examinator Amos för värdefull återkoppling till rapporten. Vi vill också tacka för deras återkoppling på det akademiska arbetet i rapporten samt svar på de frågor som uppstått under projektets arbetsgång.

Vi skulle också vilja tacka kunden som gett värdefull återkoppling i de tester som utförts och hur de tror att emulering kan tänkas utnyttjas i framtiden samt att de avvarade tid och visade stort intresse av arbetet som utförts.

Skövde, juni 2017

Johan Einevik och John Kurri

Abstract

Using a virtual twin of a production cell, makes it possible for programming and different functional testing of panels to be performed in early stages of development. A virtual twin contributes to a simpler debugging and to identify problems and minimize cost in commissioning of the production cell. The aim for the project is to investigate how well an emulated cell will perform compared to the real production cell in a factory acceptance test. Another objective is to investigate how you can use real CAD models in the emulation and what type of criteria the models should meet. The project had a lot of challenges and one of them was the difficulty to emulate the safety systems. This was solved by bypassing the safety in the PLC program. One important thing about emulation is communication between the different software used in the system. In this project, it proved successful to distribute the software on three computers to ease the workload of the programs used in the emulation. To use the emulated model instead of the real system is still in the research phase but in this project a lot of useful applications could be identified that could change commissioning in the future.

Sammanfattning

Genom att använda sig utav en virtuell kopia utav en produktionscell kan programmering och funktionstester av olika paneler testas i ett tidigt stadie. En virtuell kopia bidrar också till enklare felsökning och minskning av kostnader vid idrifttagning. Tanken med projektet är att undersöka i vilken utsträckning som emuleringsmodellen kan ersätta den riktiga cellen vid ett funktionstest för leverantören. Det som också undersöks är i vilken utsträckning riktiga CAD-ritningar kan användas och vilka krav som ställs på ritningarna för att underlätta emulering. Projektet hade flera utmaningar och en av dem som uppkom under projektets gång var problemet med att det inte gick att emulera säkerhetssystemen. Detta löstes genom att bygla alla säkerhetskretsar i PLC-programmet. En viktig del i emulering är kommunikation mellan de olika programvarorna i systemet. I projektets visade det sig fördelaktigt att dela upp programmen i emuleringssystemet för att fördela resurserna över tre datorer. Att använda sig utav en emuleringsmodell istället för en riktig produktionscell är fortfarande i forskningsstadiet men genom projektet har många användningsområden identifierats och skulle kunna förändra idrifttagning i framtiden.

Keywords

Digital twin, Emulation, Emulation of Robot Vision, Factory Acceptance Test, Virtual Commissioning

Innehållsförteckning

Äkthetsintyg I
Förord II
Abstract III
Sammanfattning III
Keywords III
Figurförteckning VII
Förkortningar VIII
1 Introduktion 1
1.1 Elektroautomatik 1
1.2 Bakgrund till studien 1
1.3 Syfte 1
1.4 Mål 1
1.5 Avgränsningar 1
1.6 Produktionscellen 1
1.7 Hållbar utveckling 2
1.8 Metod 3
1.8.1 Metod Rapport 3
1.8.2 Metod för byggande av emuleringsmodell 4
1.8.3 Verifiering 6
1.9 Rapportens disposition 6
2 Teoretisk referensram 7
2.1 Sensorer 7
2.2 Vision – Maskinens ögon 7
2.2.1 Digitalfotografering och vision 8
2.2.2 Användning av Vision 11
2.3 Emulering och Simulering 11
2.3.1 Verifiering och validering 12
2.4 Digital twin 12
2.4.1 Virtuell idrifttagning 13
2.5 PLC 13
2.6 Robot 14
2.7 Offline-programmering av robot 15
2.8 Ingångar och utgångar 16

2.9 Programvara ... 16
2.9.1 Cognex In-Sight Explorer ... 16
2.9.2 ABB RobotStudio ... 16
2.9.3 Siemens TIA-Portal ... 16
2.9.4 Simmumatik3D ... 16
2.10 Hållbar utveckling ... 16
3 Litteraturstudie ... 18
3.1 Virtuell idrifttagning ... 18
3.2 Digital Twin ... 19
3.3 Slutsatser litteraturstudie ... 20
4 Funktionsbeskrivning av system ... 20
5 CAD-modeller ... 24
6 Virtuell miljö ... 26
6.1 Programkod PLC ... 26
6.2 Kommunikation ... 27
6.3 Förenklingar ... 28
6.4 Emuleringsmodellen i Simumatik3D ... 29
6.5 Emulering av roboten ... 29
6.6 Emulering av Vision ... 30
7 Test och verifiering av virtuell modell ... 32
7.1 Test av Robotverktyget ... 33
7.2 Test av automatikcykel ... 34
7.3 Verifiering ... 34
8 Validering ... 35
8.1 Upprättande av VFAT-Protokoll ... 35
8.2 VFAT med kund ... 35
9 Diskussion ... 37
9.1 Målen ... 37
9.2 Metoden ... 37
9.3 Vision ... 38
9.4 Förenklingar och förändringar ... 38
9.5 Hållbar utveckling ... 39
9.6 Tidsplan ... 39
10 Slutsats ... 41

10.1 Användbarhet av virtuell idrifttagning 41
10.2 CAD-modeller 41
10.3 Virtuellt FAT-test-protokoll 41
10.4 Utfört VFAT med kund 41
10.5 Vision 42
10.6 Framtida arbete 42
11 Referenser 44
Bilaga A 46
Bilaga B 47

Figurförteckning

Figur 1 visar hela produktionscellen i Robotstudio som den är konstruerad av i 3D-CAD 2
Figur 2. flödesschema för metod 5
Figur 3. i bilden A belyses motivet med en liten ljuskälla långt ifrån och bilden B med en stor ljuskälla nära motivet. Notera skillnaden på skuggorna. 8
Figur 4. Kromatisk aberration syns i bilden som lila och gröna linjer där fält av hög kontrast möts. 9
Figur 5. Bilden A är tagen med en Nikon D600 på ett högt ISO-värde (25600) och bilden B på ett lågt ISO-värde (100). 10
Figur 6. Jämför binär A, monokrom B och färg C. 10
Figur 7. Högupplöst bild A och lågupplöst B 11
Figur 8 beskriver förhållandet mellan Ekologisk socialt och ekonomisk hållbar utveckling för att få hållbar utveckling 17
Figur 9. Flödesdiagram över processen. 21
Figur 10 visar en bild ur kamera i den riktiga produktionscellen när den tar en bild uppdraget att lokalisera hörn när roboten håller upp ett mellanlägg över en ljusplatta. 22
Figur 11 visar en bild ur kameran i den riktiga produktionscellen när den identifierar och lokaliserar plockpunkt för roboten 23
Figur 12. rullbanan A är innan förenkling av modell i MeshLab och rullbanan B är efter förenkling ... 25
Figur 13. En säkerhetskrets som är byglad med AlwaysTRUE (A). 26
Figur 14. Systemöversikt PLC. 27
Figur 15 visualiserar hur kommunikationen är uppbyggd mellan de olika mjukvarorna och att de är fördelade på dator 1 - 3. 27
Figur 16. Jämför robotens sammanställda I/Os i de olika programmen 30
Figur 17 visar hur rutinen CameraTrig ser ut med de extra kommandona i 31
Figur 18. Vision-programmet identifierar bild tagen i Simumatik3D med produktboxar med foton på från den verkliga cellen. 32
Figur 19. A visar produkten innan uppdatering av grafik och ljussättning och B är produkten efter uppdateringen. 33
Figur 20 visar en jämförelse med en bild tagen ur den verkliga cellen A och en bild tagen ur emuleringsmodellen till B. 33

Förkortningar

ADC – Analog to Digital Converter
DAC – Digital to Analog Converter
CAD – Computer Aided Design.
CAR – Computer Aided Robotics.
FAT – Factory Acceptance Testing.
VFAT – Virtual Factory Acceptance Testing.
PLC – Programmable Logic Controller.
ISO – International Standard Organization.
HMI – Human Machine Interface.
I/O – Inputs/Outputs.
RGB – Röd, Grön, Blå.
OPC – Open Platform Communication eller OLE for Process Control
OPCUA - Open Platform Communication Unified Architechture
LD – Ladder Logic
IL - Instruction List
ST – Structured Text
FBD – Function Block Diagram
SFC – Sequential Function Chart

1 Introduktion

Vid idrifttagning går mycket resurser åt till att testa och verifiera maskinutrustning före installation i fabrik. Studien grundar sig i ett behov av att effektivisera idrifttagning av maskinutrustning genom att ge möjlighet att testa program och funktioner utan tillgång till fysisk maskinutrustning.

1.1 Elektroautomatik

Uppdragsgivaren för detta examensarbete är Elektroautomatik. Företaget grundades av Åke Sjöberg år 1956 i Skövde och var då en liten konsultfirma inom el och styrteknik. Idag har Elektroautomatik växt till ett företag med verksamheter i både Göteborg och Skövde. Konsultdelen lever fortfarande kvar men huvudfokus ligger nu på att leverera kompletta lösningar inom industriautomation.

1.2 Bakgrund till studien

Elektroautomatik upplever ofta att deras tid vid nyinstallation av utrustning är kort och kräver stor arbetsinsats. En stor del av denna tid innebär provkörning av utrustningen för att testa kundens samtliga krav på funktioner. Detta innebär att mycket tid går åt att lyfta omkring provkörningsmaterial och preparera för olika tester. Därför vill Elektroautomatik testa att provköra utrustningens funktioner, samt tillsammans med kunden genomföra deras funktionstester, virtuellt.

1.3 Syfte

Syftet med studien är att ta fram ett underlag för framtida behov för vad som kommer krävas för att få maximal nytta av virtuell idrifttagning för Elektroautomatik. För att nå detta syfte kommer stort focus läggas på att inkludera visionguidning i emuleringsmodellen och undersöka i vilken utsträckning visionsystemet kan testas i en virtuell miljö.

1.4 Mål

För att nå syftet med arbetet har ett antal mål identifierats:

- Genomför kalibrering och guidning av visionsystem tillsammans med robot virtuellt.
- Ta fram ett virtuellt FAT- protokoll (VFAT).
- Utföra VFAT med kund.
- Använda samma CAD modeller som vid konstruktion av fysiska cellen.

1.5 Avgränsningar

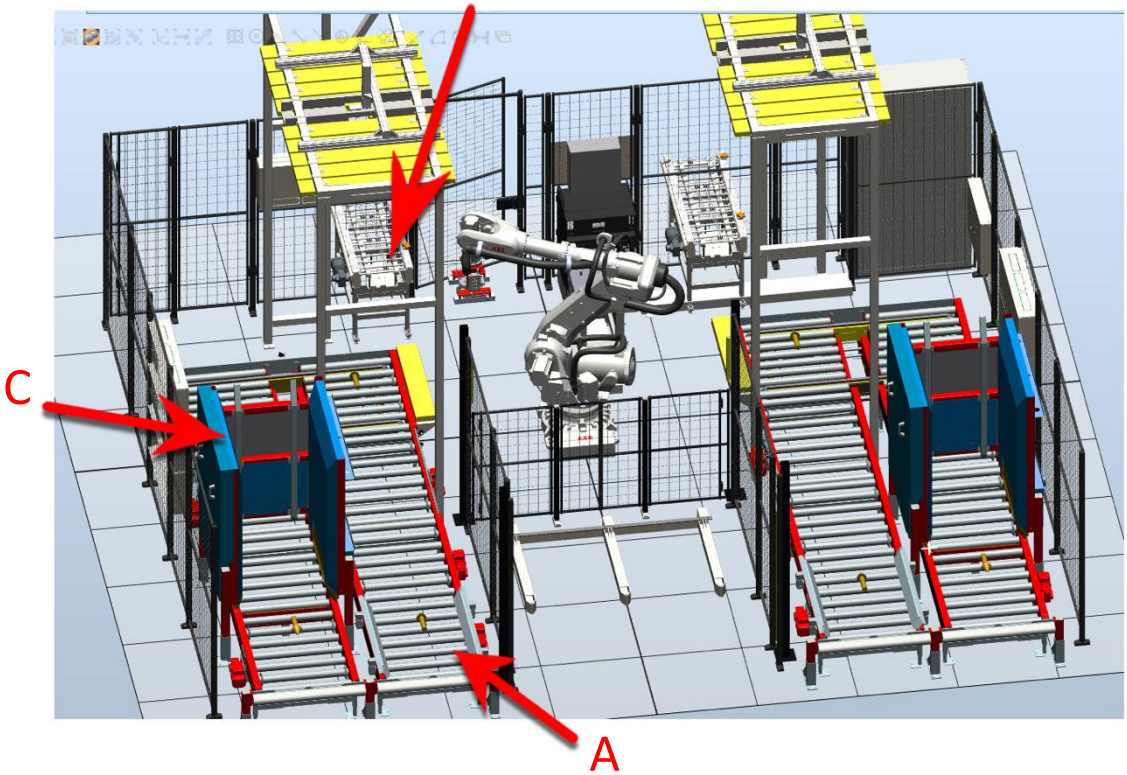
De avgränsningar som satts på arbetet är nödvändiga för att hålla arbetets omfattning till en översiktlig nivå och för att hjälpa till att styra tydligare mot målen:

- Fallstudien kommer att helt begränsa sig till produktionscellen ”upplock mellandel”.
- EA står för programmering av PLC, robot och vision.
- EA tillhandahåller alla CAD-ritningar
- Använda Simumatik3D som emuleringsmjukvara

1.6 Produktionscellen

Produktionscellen som skall digitaliseras är en uppackning av produkter från pall till bearbetning. Produkterna lastas av med en robot som lägger produkterna på en rullbana där de skickas vidare till bearbetning. Produktionscellen består av en robot som servar två stycken parallella rullbanesystem där varje rullbanesystem består av en inbana (A i Figur 1) för pallar staplade med produkter och en

utbana (B i Figur 1) där roboten lämnar produkterna. Produkter kommer staplade på pallar med ett mellanlägg mellan varje lager av produkter och ett trälock ovanpå. Tompallarna staplas sedan av en automatisk pallstaplare (C i Figur 1) vid respektive rullbana som sedan hämtas manuellt av en truck. En vision-kamera vid respektive inbana identifierar; lock, mellanlägg, produkt och tompall samt guidar roboten.



Figur 1 visar hela produktionscellen i Robotstudio som den är konstruerad av i 3D-CAD

1.7 Hållbar utveckling

Under arbetets gång kommer projektets inverkan på en hållbar utveckling att belysas. Både ur ekologisk, socialt och ekonomisk synvinkel. Att alltid ha med hållbarhet i projekt leder till att de blir mer långsiktiga och lönsamma inte bara räknat i pengar utan även för miljön och samhället. Hur projektet påverkar eller inte påverkar hållbar utveckling kommer att diskuteras i slutet av rapporten.

1.8 Metod

Det är flera steg i projektet som behöver angripas med olika arbetsmetoder, en metod för att driva själva projektet och vissa delmoment kräver egna metoder för delmomentens genomförande. De steg som identifierats är:

- Metod för projektet
  - Metod för rapportskrivning
  - Metod för att bygga emuleringsmodellen
    - Bygga
    - Verifiering
    - Validering
  - Metod för VFAT
    - Protokoll
    - Genomförande

1.8.1 Metod Rapport

Den metod vi kommer att använda oss av genom rapportens uppbyggnad är uppstaplad i en punktlista där de viktiga delarna kommer att bearbetas löpande genom projekten och vissa delar kan bockas av och lämnas till genomgång av rapportens struktur samt innehåll. Punktlistan är hämtad och omformulerad för att passa rapportens syfte, mål och arbetsgång. Punktlista: (Andersen & Schwencke, 2010)

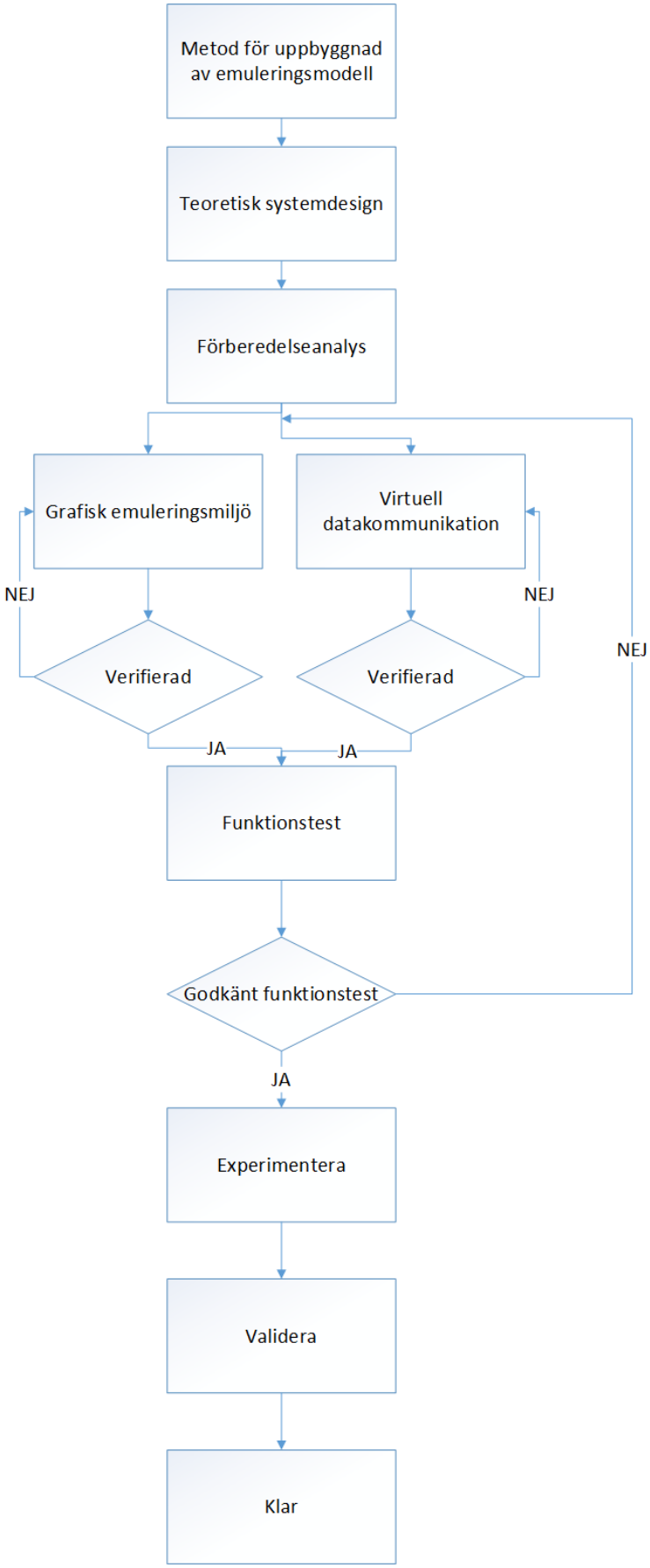
- Starten av examensarbetet går ut på att formulera en omfattande projektbeskrivning med en resursplan och avgränsningar inom projektet.
- När målet med projektet är identifierat och fastställt utformas en tidsplan i form utav ett Gant-schema.
- En struktur på hur rapporten ska vara utformad uträttas samt identifiera rimliga litteraturområden som är väsentliga för rapporten.
- Fastställa litteraturområden och dela upp arbetet i gruppen.
- Hitta relevanta artiklar som styrker och bidrar till rapportens fackliga substans.
- Sammanställa litteratur inom de väsentliga områdena.
- Utforma metoder för hur projektet ska fortskrida.
- Styrka metoderna med relevant litteratur.
- Mid-term rapport där ett första utkast ska vara framställt.
- Utveckla och förbättra språket i rapporten samt komplettera de områden som behöver kompletteras.
- En muntlig presentation genomförs på Mid-term, fokus på presentationsteknik och problemformulering.
- Slutsats och resultat ska skrivas, omfattande förklaring om hur projektet har lyckats eller ej, samt dess reflektioner för framtida arbetet inom området.
- Färdigställa rapporten och dess innehåll.
- Peer-Review på andra rapporter samt åtgärda de kommentarer granskningarna kommer fram till.
- Slutgiltig inlämning.
- Slutgiltig presentation.

1.8.2 Metod för byggande av emuleringsmodell

Vid projektets start genomfördes en brainstorming för vilka steg som kan komma att behövas för att bygga en emuleringsmodell med de förutsättningar som tilldelats. Därefter sammanställdes en önskvärd arbetsgång:

1.	Systemförståelse med ingående detaljer och dess funktioner.	Kapitel 4
2.	Enklare flödesdiagram över systemet.	Figur 4
3.	Förbereda CAD-ritningar.	Kapitel 5
4.	Programkodsinlärning, vad gör vad (I/O).	Kapitel 6.1
5.	Programkommunikation.	Kapitel 6.2
6.	Bygga emuleringsmodell.	Kapitel 6
7.	Skapa mekanismer.	Kapitel 6.4
8.	Felsöka.	Kapitel 7
9.	Verifiera.	Kapitel 7.4
10.	Testa/experimentera.	Kapitel 7
11.	Validera resultatet.	Kapitel 8

Ett flödesdiagram (Figur 2) ritades för de moment som identifierats och i vilken ordning dessa bör genomföras. Även vilka aktiviteter som är beroende av varandra för att kunna genomföras identifierades för att ge en tydligare bild av i vilken ordning aktiviteter behöver ske och vilka aktiviteter som kan ske parallellt. Som underlag för detta arbete användes den metod som Banks (2005) presenterar i boken Discrete-Event System Simulation. En annan källa vilken gav inspiration till arbetsmetod för uppbyggnad av emuleringsmodell var rapporten ”Virtual Commissioning with Process Simulation (Tecnomatix)” av Guerrero et.al. (2014).



Figur 2. flödesschema för metod

1.8.3 Verifiering

Punktlista hämtad ifrån (Banks, et al., 2005)

1. Kontrollera modellen och dess funktioner med en expert inom området, beter sig systemet som det verkliga systemet.
2. Skapa flödesdiagram över hur systemet och dess kommunikation fungerar. Kontrollera att systemet och koden beter sig som det är tänkt utifrån flödesdiagrammet.
3. Kontrollera gentemot det verkliga systemet. Representerar modellen det verkliga systemet? Kontrollera in-parametrar och studera rimligheten i resultaten.
4. Kontrollera parametrarna i systemet, är parametrarna korrekt inskrivna och är modellens mått överensstämmande med det verkliga systemet.
5. Förklara hur systemet fungerar och de ingående detaljerna för att verifiera att modellens uppbyggnad är korrekt.
6. Kontrollera att fysiken fungerar som den bör, att krockar och dylikt är som planerat. Kontrollera också att grafiken representerar systemet på ett verklighetstroget sätt.
7. En felsöknings-panel som förklarar eventuella fel som uppkommer i programmet, att de meddelande som bör förmedlas visas på ett korrekt sätt.

1.9 Rapportens disposition

Kapitel 1 ger bakgrundsinformation som projektet grundas på och beskriver de metoder som används i projektet. Kapitel 2 och kapitel 3 är den teoretiska grund som projektet bygger på, faktagrundad information tydliggörs och tidigare studier inom projektområdet analyseras. Kapitel 4 – 8 beskriver genomförandet av den praktiska delen i projektet. Kapitel 9 och kapitel 10 analyserar resultatet från genomförandet och knyter resultatet till den teoretiska grunden.

Kapitel	***Kort beskrivning***	***läsarrekommendationer***
1. Introduktion	*Bakgrundsinformation och metoder för projektet*	*Samtliga läsare*
2 & 3. Teoretisk grund	*Relevant fakta lyfts fram och tidigare studier analyseras*	*Läsare som önskar mer kunskap av komponenter och olika studier inom projektets ämnesområde*
4 – 8. Praktiskt genomförande	*Beskriver hur emuleringsmodellen byggts upp och utvärderats*	*Läsare med tidigare kunskap inom området som intresseras av projektets genomförande*
9 – 10. Projektavslut	*Diskussion och slutsats*	*Samtliga läsare*

2 Teoretisk referensram

En kort beskrivning av olika begrepp inom projektets ämnesområde för att öka förståelsen för de komponenter projektet berör.

2.1 Sensorer

Människor har flera olika sensorer som kategoriseras i våra fem sinnen; syn, hörsel, känsel, lukt och smak. Dessa använder människan för att kunna läsa av sin omgivning och använda denna information som beslutsunderlag till allt hon gör. Alla dessa sinnen är något sensorerna ska kunna ersätta och samla data ifrån. Den första moderna sensorn skapades av Edwin Hall i slutet av 1800-talet, teorin gick ut på att snurra en magnet på en platta för att detektera förändringarna i magnetfältet (Ramsden, 2006). Sedan dess har sensorerna utvecklats och blivit allt mer sofistikerade och komplexa.

En sensor är en omvandlare som konverterar en fysisk variabel i någon form till en annan form som är användbar i den givna applikationen den är ämnad för (Groover, 2014). En sensor konverterar en fysisk signal t.ex. temperatur eller tryck med avseende att förenkla signalen till t.ex. en elektrisk spänning för att mäta det efterfrågade (Groover, 2014). De två största grupperna inom sensorer är analoga och diskreta sensorer, den analoga sensorn indikerar förändringar konstant kontra den diskreta sensorn som mäter när läget förändras (Groover, 2014). De diskreta sensorerna går att dela in i två grupper binär som endast har av- eller på- läge och digitala som skickar bitar av kod (Groover, 2014). Inom området sensorer är det viktigt att veta om att det finns två sätt att konvertera signalerna, ADC (Analog till Digital) och DAC (Digital till Analog). Den mest använda är ADC (Groover, 2014).

De sensorer som används i projektet beskrivs nu mer i detalj och är; optiska sensorer, mekaniska sensorer och robotsensorer:

- En optisk sensor är en sensor som detekterar olika fysiska eller kemiska parametrar genom att analysera optiska egenskaper (Korotchenkov, 2011). Ett exempel på en enkel optisk sensor är en pH indikator sticka, den ändrar färg med vid olika pH-nivåer (Korotchenkov, 2011). De sensorer som används inom industrier är oftast elektriska och fungerar alla på ungefär samma sätt. En optisk sensor skickar ut ljus och sedan mäter indata, det kan antingen vara en sändare och mottagare där ena sensorn skickar ljus som upptäcks av mottagaren ifall ljuset bryts så indikerar det att någonting är i vägen. En annan enkel optisk sensor är en sändare och en reflektor, den fungerar så att ljuset sänds till en reflektor och sändaren detekterar själv ifall ljusstrålen bryts. (Megyesi, 2015)
- En mekanisk sensor är i enkelhet en knapp som har ett på och ett av läge. Signalen som sänds är analog och behöver omvandlas till digitalt för att förenkla användningen. (Groover, 2014)
- Sensorer inbyggda i roboten kallas för pulsgivare och finns för att kalkylera vart och i vilken rotation verktyget finns i förhållande till robotens bas och dess utgångspunkt.

2.2 Vision – Maskinens ögon

Ett av människans sinnen är synen och hon tar in otroliga mängder information genom synen som hjärnan behöver bearbeta och göra begriplig. Detta system som människan har biologiskt har utvecklats under årmiljonerna genom evolutionen och när hon använder synen känns det enkelt och helt naturligt. (Davies, 2012)

I modern tid har människan konstruerat maskiner att utföra vissa arbetsmoment och för att klara av mer komplicerade moment har maskinerna försetts med utrustning som motsvarar det mänskliga

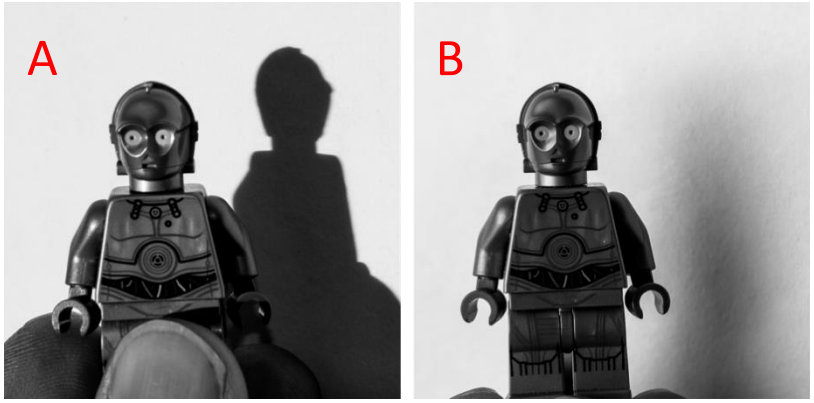
sinnet syn. Det som tagit evolutionen miljoner år att utveckla, utvecklar nu människan åt maskiner på några årtionden. (Davies, 2012)

Ett vision-system består av en digitalkamera som tar en bild av det önskade motivet, en bildbehandlingsprocess som förbereder bilden visuellt och slutligen tolkas bilden utifrån angivna parametrar. Ett vision-system är en integrerad del i ett system där förmågan att se hjälper till att lösa de önskade uppgifterna. Beroende på vad som skall utföras krävs det olika egenskaper av systemet och de olika stegen. (Davies, 2012)

2.2.1 Digitalfotografering och vision

Beroende på vilka uppgifter vision-systemet är utvecklat för att hantera kan det vara byggt för att ta olika typer av bilder. Typiskt är det en kamera med någon form av optik anpassat för uppgiften, i vissa fall med filter som endast släpper igenom önskade våglängder. Optiken formar och fokuserar ljuset på en digital bildsensor vars uppgift är att fånga upp informationen i ljuset och göra om det till digitala rådata. Denna rådata omvandlas med hjälp av en förprogrammerad algoritm i kamerans elektronik till en digital bildfil. Den digitala bilden som skapas i kameran är en representation av det motiv som kameran fotograferar och hur den representationen ser ut påverkas av en mängd olika faktorer. (Davies, 2012)

Det tillgängliga ljusets på motivet är mycket betydelsefullt när det kommer till hur den slutgiltiga bilden kommer att se ut. Antalet ljuskällor och hur de är placerade i förhållande till motivet med avseende på infallsvinkel och avstånd påverkar hur ljuset reflekteras. Styrkan på ljuskällorna och förhållandet i styrka mellan ljuskällorna påverkar bland annat skuggbilden på motivet. Storleken på ljuskällan har betydelse för om det blir mjuka eller hårda skuggor vilket kraftigt inverkar utseendet på bilden (se Figur 3). (Davies, 2012)



Figur 3. i bilden A belyses motivet med en liten ljuskälla långt ifrån och bilden B med en stor ljuskälla nära motivet. Notera skillnaden på skuggorna.

Optiken har till uppgift att styra ljuset som reflektera från motivet och samla det till en bild på sensorn. Beroende på utformning och kvalitet på optiken påverkar det bilden på olika sätt. Optikens brännvidd bestämmer bildvinkeln och kan också beskrivas som förstorningsgraden. Längre brännvidden på optiken ger mer förstorningen av motivet på sensorn. Bländaröppningen sitter ofta i optiken och styr mängden ljus som släpps in. En annan effekt av stor bländaröppning är att skärpedjupet blir kortare desto mer bländaren öppnas. Skärpedjup är hur långt ifrån fokusavståndet i djupled som bilden är skarp. Optik innehåller alltid optiska fel och två exempel på fel är kromatisk aberration och distorsion. Kromatisk aberration innebär att olika våglängder av ljus bryts olika mycket och syns tydligast som

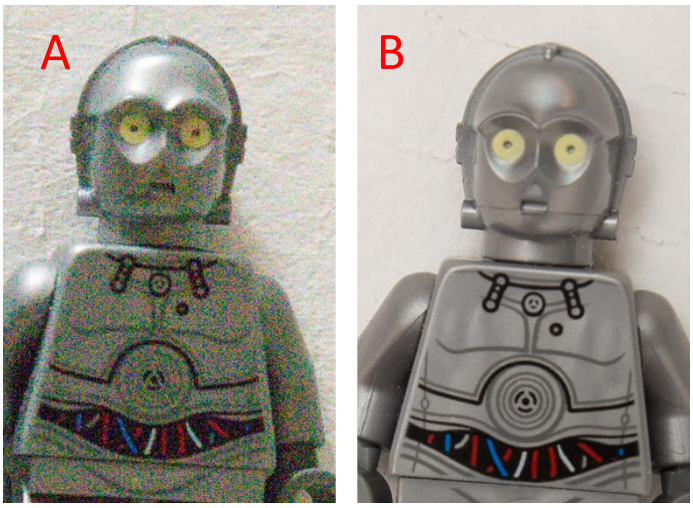
färgskiftningar i skarpa kanter med stor kontrast som exemplet i Figur 4 där en svart konstruktion är fotograferad mot en ljus bakgrund. (Davies, 2012)

Figur 4. Kromatisk aberration syns i bilden som lila och gröna linjer där fält av hög kontrast möts.

Distorsion innebär att bilden blir förvrängd så att det ser ut som att den buktar ut eller in och ger effekten av att raka linjer avbildas som böjda. Beroende på hur optiken är byggd och vilka material som används blir effekten av dessa fel mer eller mindre synbart. Kvaliteten på optiken påverkar även generellt hur skarpt motivet kan avbildas. (Davies, 2012)

Slutartiden påverkar under hur lång tid sensorn är exponerad för det ljus som optiken släpper igenom. En kort slutartid ger en mörkare bild men klarar av mer rörelser i bilden och risken för rörelseoskärpa blir då mindre. Rörelseoskärpan kommer av hur långt ett motiv hinner röra sig under tiden som slutaren är öppen. Sammanfattat ger det att lång slutartid hinner samla upp mer ljus men bilden blir då känsligare för rörelseoskärpa. (Davies, 2012)

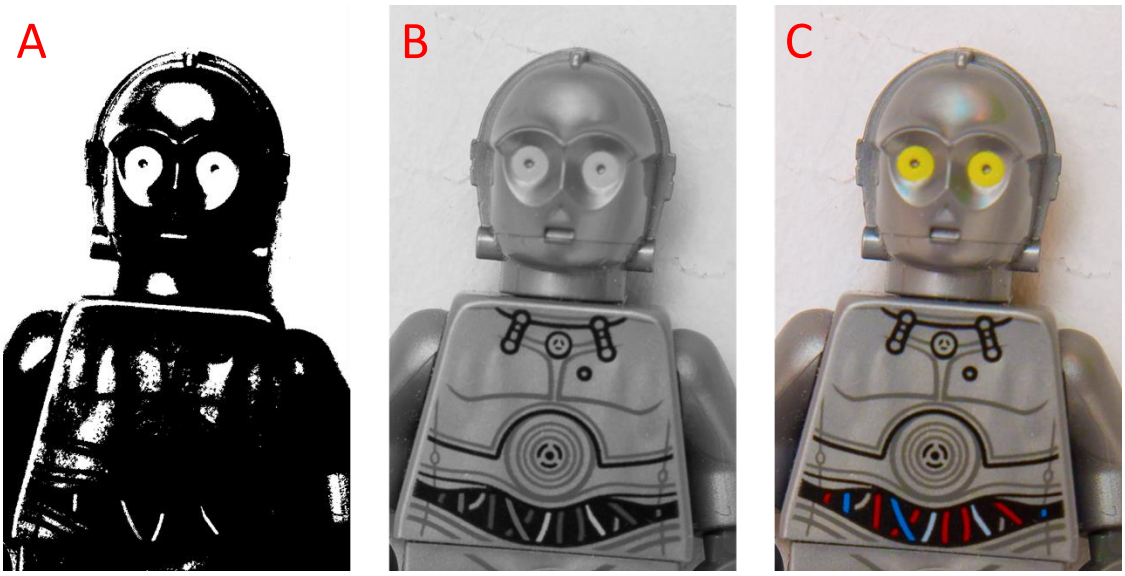
Sensorns utformning påverkar bilden på flera olika sätt. En stor sensor kräver större optik men kan samla in mer ljus. Den slutliga digitala bilden som läses av sensorn får samma proportioner som sensorn och storleken i antal pixlar i den digitala bilden bestäms av antalet pixlar på sensorn. Ljuskänsligheten på sensorn är en variabel som går att ändra och denna variabel kallas ISO. Högre ISO ger en ljuskänsligare sensor och därmed en ljusare bild. Men med högre ljuskänslighet får det bieffekten att processen blir känsligare för elektroniskt brus när ljuset digitaliseras till en bild. Kvaliteten på sensorn avgör bland annat hur mycket brus som tillförs bilden vid olika ISO-tal (se Figur 5). Sensorn har också ett dynamiskt omfång som begränsar hur stor skillnad den kan ta upp mellan det ljusaste och mörkaste i bilden utan att det avbildas som helt vitt eller helt svart. (Davies, 2012)



Figur 5. Bilden A är tagen med en Nikon D600 på ett högt ISO-värde (25600) och bilden B på ett lågt ISO-värde (100)

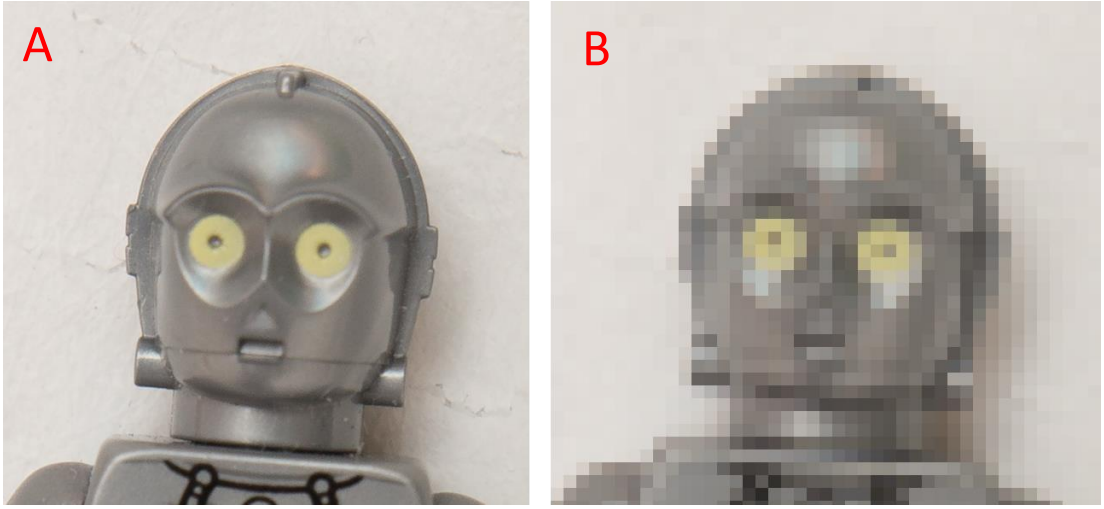
Rådata som samlas upp av sensorn behöver tolkas av en algoritm för att kunna visas som en bild. Denna process kallas för rå-konvertering och beroende på hur denna algoritm ser ut påverka detta den slutliga bilden. När det gäller Vision-system kan typen av bilder delas in i tre kategorier (se Figur 6) (Davies, 2012):

- Binära (Det finns enbart två värden på pixlar svart och vit). A i Figur 6.
- Monokroma (ex. gråskala). B i Figur 6.
- Färgbilder (ex. RGB). C i Figur 6.



Figur 6. Jämför binär A, monokrom B och färg C.

En digital bild kan också ha olika stor pixelupplösning (se Figur 7). En bild med ex. 6000 x 4000 pixlar kan återge med detaljer än en bild med 3000 x 2000 pixlar under i övrigt identiska förhållanden. (Davies, 2012)



Figur 7. Högupplöst bild A och lågupplöst B

Fler pixlar och färger ger också mer komplexa bilder som kräver mer datorkraft för att behandla. Ett vision- system bör därför skapa bilder med en ändamålsenlig komplexitet. Bilderna bör vara tillräcklig komplexa för att lösa uppgiften utan att de blir onödigt krävande. (Davies, 2012)

För att underlätta för algoritmerna att hitta det som söks i en bild kan en del bildbehandling utföras på en bild. Det kan vara i form av olika digitala filter som ökar kontrasten på olika sätt eller förstärker kanter. (Davies, 2012)

2.2.2 Användning av Vision

Ett Vision-system kan ha många olika applikationer och nya användningsområden utvecklas hela tiden. Det kan vara en del i ett system för automatisk guidning av fordon, ansiktsigenkänning för övervakning, fingeravtrycksläsning, automatiska fartkameror och mycket mer. Inom tillverkningsindustrin används Vision-system bland annat för kvalitetskontroller, för att guida robotar vid montering eller materialhantering. Inom industrin är det ofta en relativt kontrollerad miljö och då går ofta omgivningen att optimera för att få ut maximalt av ett Vision-system med hjälp av t.ex. utstuderad belysning och enkla bakgrunder som underlättar mycket för databehandlingsfasen och identifieringen. (Davies, 2012)

2.3 Emulering och Simulering

Simulering betyder egentligen att imitera fysiska genskaper av ett system i en virtuell miljö. Alltså en representation av hur systemet reagerar över tid (Banks, et al., 2005). Vilket betyder att ett system kan simuleras antingen för hand eller med hjälp av en dator. Rapportens fokus ligger på den datorstyrda simuleringen.

För att studera hur ett system reagerar i olika situationer behövs en simuleringsmodell som kan simulera tid och de olika scenarier som kan uppkomma över tid. De olika beteendena som utvecklas över tid mellan objekt, enheter och system studeras matematiskt och logiskt för de antaganden som systemet är uppbyggt på i syfte att de kan bli förståeligt för intressenten. Simuleringsmodeller har som uppgift att undersöka frågor i ett system som annars kan vara svåra, komplexa och kostsamma i verkligheten att undersöka som. Simuleringsmodeller kan användas för att undersöka eventuella förändringar som kan förbättra systemet eller för att studera funktioner inom ett system, men också ifall systemet uppfyller de krav som ställs. Enkla system kan simuleras genom matematiska och

sannolikhetsuträkningar men när det gäller komplexa system med många ingående variabler krävs ett datorstyrt simuleringsprogram. Simuleringsprogram kan undersöka olika uppsättningar ekvationer och förhållanden för att kunna representera verkligheten på ett förståeligt sätt. (Banks, et al., 2005)

Emulering liknar simulering i många aspekter då båda kan representera en miljö i tre dimensioner. Simulering och emulering kan representera data på ett noggrant sätt, det som definieras som noggrant i detta sammanhang är att resultatet är statistiskt betydelsefullt. Med denna definition av noggrannhet kan det skiljas från modell till modell. En modell ska också skapas utan störande element, d.v.s. att som det riktiga systemet ska det emulerade eller simulerade systemet agera verklighetstroget och inte förvåna observatören. Emulering och simulering fungerar på liknande sätt men fokuserar på olika delar. Emulering fokuserar på rörelser och fysiska egenskaper av ett system där modellerna används främst till att beskriva funktioner i en riskfri miljö och demonstrera hur systemet agerar i verkligheten. Emuleringsmodeller är skapade för att representera verkligheten mer noggrant och visa hur systemet agerar för att kunna testa fler scenarier än vad som skulle vara praktiskt i verkligheten på kortare tid. En stor skillnad mellan emulering och simulering är att simulering är byggd för att testa så många olika möjliga scenarier som möjligt. Emuleringsmodeller är skapta för att i realtid representera verkligheten för att identifiera hur systemet agerar och ge en mer realistisk bild över hur kommunikationen fungerar mellan olika system. (McGregor, 2002)

2.3.1 Verifiering och validering

Verifiering är en process för att skapa en modell som överensstämmer med verkligheten. Alla de viktigaste komponenterna och funktionerna bör överensstämma så långt som möjligt med det verkliga systemet för att kunna säga att modellen är validerad. En validerad modell är en subjektiv bedömning över hur nära verkligheten ett system agerar, det ger också en trohetsaspekt i modellen. En modell utan validering är ganska meningslös då den inte har någon substans i hur systemet faktiskt agerar och kan därför vara missvisande med de resultat som framställs. Verifiering är en process för att bekräfta att modellen är en trovärdig representation av systemet som modellen bygger på. Processen för att verifiera modellen bygger på att jämföra det verkliga systemet med den virtuella och sedan granska de olika egenskaperna samt funktionerna mellan systemen. Processen för verifiering upprepas tills den virtuella modellen överensstämmer med verkligheten och blir accepterad som en trovärdig representation. (Banks, et al., 2005)

Validering är en jämförelse mellan hur väl den virtuella modellens resultat överensstämmer med det verkliga systemet. Svarar modellen verklighetstroget på den fråga som modellen är byggd för, är det ett certifikat på att modellen representerar och beter sig som det verkliga systemet. Det som ska besvaras i valideringen av en modell är om modellen är uppbyggd och implementerad på ett korrekt sätt i simuleringsverktygen och om alla parametrar är representerade på ett korrekt sätt. (Banks, et al., 2005)

2.4 Digital twin

Enligt Shafto et. Al. (2012) är en Digital Twin en digital kopia av en maskin. Den innehåller en mängd data från maskinen som fysiska egenskaper, form, programmerade funktioner och speglar alla förändringar samt händelser som sker hos den fysiska motsvarigheten under dess livstid. Den digitala tvillingen lagrar alla sensorhändelser och är hela tiden uppkopplad mot sin fysiska motsvarighet för att logga alla händelser. En Digital Twin kan användas för att hitta lösningar på uppkomna fel på den fysiska versionen. En Digital Twin kan också användas för att simulera förändringar som då kan prövas

virtuellt innan de införs på den fysiska tvillingen. Olika tänkbara felscenarier som kan uppkomma på utrustningen kan simuleras och motverkas innan de uppstår. (Shafto, et al., 2012)

När det gäller flygindustrin beskriver José Ríos et. al. (2015) Vad som definierar en Digital Twin. En Digital Twin skall följa med en produkts hela livscykel. Från början (konstruktion och tillverkning), under användning (Användning och underhåll) och i slutet (återvinna, skrota). En Digital Twin av ett system skapas när systemet designas och uppdateras under hela användningen. Författaren gör också skillnad på om systemet är smart eller inte. Beroende på hur smart ett system är kan det hålla kontakt med sin Digital Twin och uppdatera den med aktuell status. Alla underhåll och reparationer kan loggas i den digitala tvillingen och alla modifikationer som utförs under livscykeln synkas också. Ett system kan ha statisk och dynamisk information där dynamisk information är sådan som uppdateras under användning som t.ex. drifttimmar och underhållsintervall. (RÍOS, et al., 2015)

2.4.1 Virtuell idrifttagning

Vid konventionell idrifttagning kan inte program och funktioner testas fullt ut förrän maskinen är uppmonterad. Efter montering kvarstår ofta fel i programkod och konstruktion som behöver felsökas och rättas till, vilket ofta är en tidskrävande och dyr process. Virtuell idrifttagning är att skapa en Digital Twin under konstruktionsfasen som programmerare och konstruktörer kan testa sina funktioner i. Detta underlättar avsevärt när program och funktioner kan prövas innan den faktiska maskinutrustningen monteras. Detta gör att t.ex. kommunikationsprotokollen mellan robot och PLC-program kan kontrolleras samt sensorers placering prövas. Därefter ska de programmen som testats virtuellt gå att överföra med minimal eller ingen modifiering till den fysiska maskinutrustningen. Det blir alltså en virtuell idrifttagning av den digitala tvillingen först för att kunna korta ner idrifttagningen av den fysiska utrustningen. (Hoffmann, et al., 2010)

2.5 PLC

PLC står för Programmable Logic Controller och översatt programmerbar logisk kontroll, i rapporten används benämningen PLC som inom området är en vanlig term. PLC är ett styrsystem som bygger på logiskt tänkande att värden eller signaler kan antingen vara på eller av. Vid programmering av PLC kan programmeraren använda sig av olika språk se (IEC standard 61131-3), dessa språk är egentligen som mänskliga språket att samma sak kan sägas på olika sätt. PLC bygger oftast på ett system som styr ingångar och utgångar och fungerar i stort som en vanlig dator men är optimerade för styrning istället för beräkningar. (Bolton, 2015)

De fem vanligaste språken enligt Dag H. Hanssen (2015) som PLC programmeras i (IEC Standard 61131-3) är:

- LD (Ladder Diagram), villkoret ligger till vänster, själva funktionen i mitten och vad som ska sändas vidare är till höger. Alla olika funktioner ligger uppifrån och ner som i en trappa med varje logik som ett trappsteg och där av namnet Ladder (trappa). LD är ett språk som används av många och är i många fall kompatibel med andra språk vilket gör att det i många fall integreras i andra språk för att göra komplexa program. Språket baseras på relä diagram vilket gör språket enkelt att förstå och ger en överskådlig bild över vad som händer i programkoden.
- IL (Instruction List), fungerar likt maskinkod där varje rad är en funktion eller kommando. Liknar ST men är mer som ett kommando där språket säger vad som ska hända och göras i varje rad. Språket ligger på en låg nivå d.v.s. nära maskinens egna styr-kod och den största fördelen är att språket inte kräver någon avsevärd datorkraft.

- ST (Structured Text), det språk som är mest likt vanlig programmering där koden skrivs som en löpande text med instruktioner och funktioner inbakade i texten för att styra systemet. Detta språk ligger på en hög nivå i programmeringen och kan därför utföra många kommandon i en och samma textrad.
- FBD (Function Block Diagram), Liknar LD men skiljer sig åt genom att funktionerna är i block och varje enskilt block kan kallas med olika variabler som sedan utför en ekvation och resultatet skickas vidare till systemet. FBD använder sig utav ”och”, ”eller” och ”inte” vilket gör FBD till ett enkelt och överskådligt programmeringsspråk då det bygger på boolesk algebra. FBD klassas som ett högnivå språk vilket gör att fler instruktioner kan utföras med en funktion.
- SFC (Sequential Function Chart), Detta språk är en blandning mellan FBD och LD som skrivs vertikalt istället för horisontellt. Blocken läses uppifrån och ner med villkor mellan varje block. I blocken ligger händelser eller funktioner som utförs först när villkoret ovanför blocket är sant. Det går att argumentera att SFC inte är ett traditionellt programmeringsspråk men är ett väldigt bra verktyg för att strukturera kod och blanda de olika språken för en överskådlig och strukturerad bild över vad som händer.

2.6 Robot

Teknikutvecklingen drivs av människors vilja att uppnå det hon drömmer och fantiserar om. Inte ovanligt beskrivs dessa drömmar och fantasier om tekniska framsteg först i skönlitteraturen och samma sak är det med robotar. Nya tekniska framsteg sporrar nya idéer hos författare och den verkliga teknikutvecklingen låter sig också inspireras av skönlitteraturen. När det gäller litterära verk med robotar har dessa många gånger inkluderat en domedagshistoria där robotar tagit över människans plats på jorden. Isaac Asimov skrev de tre robotlagarna som sedan blev fyra när lag noll tillkom. (Blomsjö, 2006)

”§0 En robot får inte skada mänskligheten eller genom passivitet tillåta att mänskligheten skadas. Denna lag står över de övriga lagarna.

§1 En robot får inte skada en mänsklig varelse eller underlåta att hjälpa en mänsklig varelse som kommit till skada.

§2 En robot måste alltid lyda en mänsklig varelse såvida det inte är i konflikt med första lagen.

§3 En robot måste skydda sig själv från skada såvida det inte är i konflikt med första eller andra lagen.” (Blomsjö, 2006)

När det kommer till dagens industrirobotar går det att se att första delen i lag ett och två är implementerade till en viss del genom att robotar och dess omgivande system är konstruerat för att minimera risken för skada på människor och utrustning. Att dagens industrirobotar lyder människor enligt lag två, gör de genom att följa programmen människor skrivit i roboten. Däremot har inte industrirobotar ett medvetande och kan därför inte reflektera över andra delen i §1 och §2. Då dessa lagar låter bra ur en moralisk synvinkel kan de vara svåra att följa i praktiken. (Blomsjö, 2006)

George Devol och Joseph Engelberg lät sig inspireras av Isaac Asimov historier och utvecklade 1959 den första roboten för industriellt bruk. Roboten som fick namnet Unimate, rörde sig med hjälp av hydraulik och programminnet var på en magnettrumma och hade en noggrannhet på en tiotusendels tum. (IFR, 2012)

I industrin införskaffas robotar för att utföra arbetsuppgifter som en robot klarar bättre och effektivare än människor eller till uppgifter som är skadliga för människor eller utförs i en för människor skadlig miljö (Blomsjö, 2006). Till en början var det mest materialhantering men efter att robotsystemen blev mer och mer avancerade ökade även användningsområdena inom industrin (Blomsjö, 2006). Nu används industrirobotar även inom processbearbetning som t.ex. sprutlackering, svetsning och monteringsapplikationer (Blomsjö, 2006). En industrirobot består generellt av en programmerbar styrenhet med gränssnitt för kommunikation med människor och andra maskiner samt en manipulator med minst tre leder (ISO, 2011). I änden på manipulatorn kan ett verktyg monteras för att roboten skall ha möjlighet att påverka omgivningen på önskvärt vis (Blomsjö, 2006)

Definitionen av en industrirobot enligt ISO 10218–1:2011 är:

> *"Automatically controlled, reprogrammable multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automations."*

Fritt översatt blir det: Automatiskt styrd, omprogrammerbar universell manipulator, programerabar I tre eller fler leder, vilken kan vara antingen fast monterad eller mobil för att användas i industriell automatisering. (ISO, 2011)

2.7 Offline-programmering av robot

För att prata om Offline-programmering behöver först begreppet Online-programmering beskrivas. När programmering av robot sker online är programmeraren hela tiden på plats vid roboten och behöver ha tillgång till roboten, hela arbetscellen där roboten manövrerar. Detta gör att programmeraren får direkt feedback om vad som fungerar och kan validera programmet hela tiden under programmeringen. Roboten är upptagen med programmeraren under hela tiden och kan inte användas i produktion under tiden den programmeras. (Blomsjö, 2006)

Den stora skillnaden med Offline-programmering är att programmeraren inte behöver tillgång till själva roboten fysiskt utan kan först skapa programmet för att sedan ladda in det färdiga programmet i roboten. Detta gör att tiden som roboten är upptagen med programmeraren blir betydligt kortare än vid online-programmering. Detta gör även att en robot kan programmeras innan robotcellen är monterad och detta minskar ledtiden för leverans av robotcellen då programmeringen kan ske parallellt med övrigt konstruktions och monteringsarbete. (Blomsjö, 2006)

Att gå från Online-programmering till Offline-programmering kan liknas med att göra om ett inre ställ på en maskin till ett yttre ställ. Istället för att stoppa maskinen för att sedan göra en förändring eller verktygsbyte kan förändringen förberedas utanför maskinen under drift för att sedan snabbt implementeras under ett kort stopp. (Rahimi & Karwowski, 1992)

När ett program är skapat Offline finns risken att programmet är dåligt anpassat för den verkliga cellen då programmeraren saknat tillgång till den riktiga cellen och inte kunnat kontrollera programmet. Detta gör att en hel del justeringar behöver göras när programmet skall köras i det verkliga systemet. Ett hjälpmedel för programmeraren är att använda sig av Compuder-Aided Robotics (CAR) Vilket innebär att en miljö av CAD-filer (Computer-Aided Design) byggs upp i en virtuell miljö och en modell av roboten styrs av en virtuell robotcontroller. Detta gör att robotprogrammet kan testas i en virtuell miljö som kan efterlikna den verkliga i mycket stor grad. Denna möjlighet att testa programmet virtuellt

minskar eller helt eliminerar behovet av att justera programmet när det laddats in och körs i det fysiska systemet. (Rahimi & Karwowski, 1992)

2.8 Ingångar och utgångar

Inputs och outputs är så PLC:n kommunicerar med andra komponenter, varje input och output har en unik adress i PLC:n som kan användas i programmeringen för att kommunicera med rätt komponent vid rätt tidpunkt. Inputsignaler som är analoga måste först konverteras till binära värden vilka sedan kan delas upp i olika storlekar. Digitala inputsignaler är signaler som antingen kan vara på eller av vilket används i bl.a. knappar och brytare. De signaler vi kommer beröra är de med bitar mellan 1 och 32, dessa signaler delas in i olika klasser som visar hur många bitar en signal kan transportera mellan PLC:n och komponenten den talar med. 1 bit kallas bara för en bit signal eller en bool 1 bit kan varieras mellan på eller av alltså en etta eller nolla. Nästa steg är en Byte vilket innefattar 8 stycken bitar och kan då skicka ett värde mellan 0 och 255 alternativt få 8 signaler med ett och noll. Går vi till en storlek större på signalen kallas de Words (ord) och har 16 bitar vilket gör att värdet kan variera mellan 0 och 65535. Nästa steg är ett double Word (dubbel ord) som då har 32 bitar och genast blir enormt mycket större med plats att skicka 32 enskilda signaler med samma variabel. Genom att använda sig av hopbakade signaler kan information sändas mellan objekt snabbare då ett kommando kan aktivera flera signaler till olika mottagare. (Hanssen, 2015)

2.9 Programvara

Nedan följer en kort presentation om vilka programvaror som används i projektet och dess användningsområde.

2.9.1 Cognex In-Sight Explorer

Används för att programmera kameror från tillverkaren Cognex. Programmet har även en emulator inbyggd där det går att lägga in bilder och skicka kommandon till Vision-programmet så det beter sig som den riktiga kameran. (Cognex corporation, 2017)

2.9.2 ABB RobotStudio

Används för offline-programmering av ABB robotsystem. Robotstudio har ett grafiskt 3D-gränssnitt där rörelser kan kontrolleras och en miljö av 3D-objekt kan byggas upp för att representera den verkliga arbetsstationen vilket roboten programmeras för. (ABB, 2017)

2.9.3 Siemens TIA-Portal

Används för att programmera och simulera PLC-program för Siemens produkter. (Siemens AG, 2017)

2.9.4 Simmumatik3D

Används för att emulera produktionsutrustning i syfte att testa fysik, rörelser och programkod i en virtuell miljö. (Ayani, 2017)

2.10 Hållbar utveckling

Hållbar utveckling är mer än att bara ha ett statiskt hållbart samhälle. Det innebär att samhället utvecklas hållbart. Att kunna utveckla samhället så de som lever i svält eller andra humanitära svårigheter ständigt får det bättre och samtidigt hållbart utnyttja jordens resurser. En populär definition ger Brundtlandskommissionen i FN-dokumentet från 1987, Vår gemensamma framtid: (Gulliksson & Holmgren, 2015)

”En hållbar utveckling är en utveckling som tillfredsställer dagens behov utan att äventyra kommande generationers möjlighet till att tillfredsställa sina behov.”

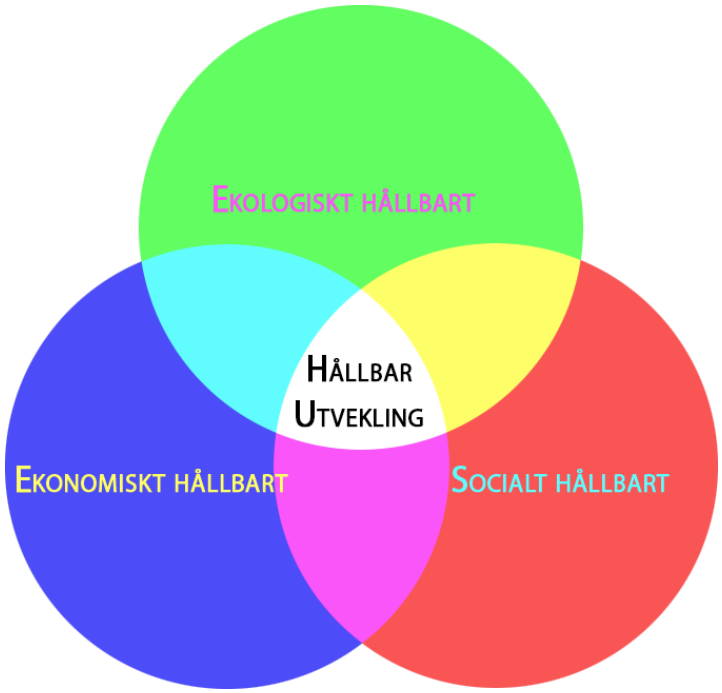
Idag utnyttjar människor jordens resurser på ett sådant sätt att de inte hinner återskapas i den takt de förbrukas. Förutom att förbruka resurser förgiftas och förorenas planeten hela tiden vilket inte är hållbart. För att kunna leda samhället mot en hållbar utveckling måste varje person själv inse och reflektera över vad hen kan göra för att påverka utvecklingen. En bra inställning är att vara hoppfull positiv realist. Hållbar utveckling består av tre stycken huvudfaktorer; ekologisk, socialt, ekonomiskt hållbart och överlappningsytan där de tre möts är hållbar utveckling (se Figur 8). (Gulliksson & Holmgren, 2015)

Ekologiskt hållbart är när samhället inte förbrukar mer av naturen än vad som naturen kan återskapa. Samhället behöver leva av räntan på det som planetens ekosystem producerar. Ekosystemet behöver vara i balans. (Gulliksson & Holmgren, 2015)

Socialt hållbart är när människor känner sig trygga och själva har kontroll över sina liv. Raka motsatsen till socialt hållbart är krig, andra väpnade konflikter och allt lidande som följer med det. Även svält och analfabetism är långt ifrån socialt hållbart. Det finns många faktorer som spelar in på den sociala hållbarheten och en definition av social hållbarhet är enligt Gulliksson & Holmgren (2015):

”Ett socialt hållbart samhälle är rättvist, jämlikt, inkluderande och demokratiskt. Det garanterar en rimlig livskvalitet för nuvarande och kommande generationer.”
(Gulliksson & Holmgren, 2015)

Ekonomiskt hållbart är det om fördelningen och hushållningen av resurser möter behovet för alla utan att äventyra framtidens möjlighet att tillfredsställa sitt behov. (Gulliksson & Holmgren, 2015)



Figur 8 beskriver förhållandet mellan Ekologisk socialt och ekonomisk hållbar utveckling för att få hållbar utveckling

3 Litteraturstudie

Litteraturstudien tar upp två ämnen som anses särskilt användbara att studera. Fyra tidigare arbeten om virtuell idrifttagning studeras samt två stycken arbeten om Digital Twin.

3.1 Virtuell idrifttagning

Binnberg & Johansson (2016) utförde ett arbete där uppdraget var att undersöka möjligheten att korta ner idrifttagningar och stopptiden vid modifieringar av en produktionscell. En robotcell valdes ut för experimentet och metoden som användes var att bygga en digital emulering av robotcellen. Genom att använda en emuleringsmodell av robotcellen menar författarna att mycket tid kan sparas in genom att testa PLC- och robotprogram i en virtuell miljö innan den körs på den tänkta hårdvaran. För att kunna emulera med de verkliga programkoderna kopplades ABB RobotStudio, WinCC och PLCSIM till emuleringsmjukvaran Simumatik3D. (Binnberg & Johansson, 2016)

Liknande slutsatser drar även Makris, Michalos, och Chryssolouris (2012). De menar att de kunde minska tiden det tar att introducera nya arbetsmoment i robotcellen och de kunde också säkerställa att alla funktioner som krävdes kunde testas innan förändringen implementerades i den fysiska cellen. Makris, Michalos, och Chryssolouris (2012) kunde också bevisa att de minskade de totala kostnaderna med 15 % samt att de ökade framtida möjligheter att konfigurera systemet när nya förändringar sker vilket ökade livslängden på maskinerna då de kunde konfigureras mot de nya kraven. (Makris, et al., 2012)

Johansson och Nilsson (2015) och Guerrero, et al. (2014) undersökte möjligheten att verifiera PLC program i en virtuell miljö. Johansson och Nilsson (2015) jämförde Simuleringsprogrammet Plant Simulation och emuleringsprogrammet Simumatik3D för att visualisera den hypotetiska maskinutrustningen som skulle styras av en PLC. PLC-programmet programmerades i Siemens Step 7. Författarna diskuterar mycket om skillnaderna mellan Simumatik3D och Plant Simulations möjligheter att utvärdera PLC-program. Författarna kommer fram till att Simumatik3D ger en mer realistisk påverkan av PLC programmet tack vare sin fysikmotor men har svårt att hantera stora komplexa system. Plant Simulation är bättre för större system men ger inte riktigt samma realistiska återkoppling. Resultatet av jämförelsen ger Simumatik3D fördel och det ses som programvaran att föredra att använda för utvärderingen av PLC-program. (Johansson & Nilsson, 2015)

Författarna Johansson och Nilsson (2015) drar slutsatsen att det är fullt möjligt att verifiera PLC-program i en virtuell miljö. Dock kräver arbetsmetoder och programvara mer utveckling för att det skall kunna implementeras i det dagliga arbetet vid driftsättning. Ett förslag författarna ger är att bygga upp standarder i arbetssättet mellan PLC-programmerare och simuleringsansvariga. (Johansson & Nilsson, 2015)

Guerrero, et al. (2014) byggde upp sin modell i Siemens Technomatix och kopplade det till en riktig PLC som körde programmet. Studien utfördes på teknologiska institutionen på universiteten i Monterrey, Mexiko. Systemet som användes var utrustning från ett mekatroniklabb på universitetet som används för att bygga automatiserad produktionslina i laborationsmiljö för utbildning. Den första stationen i linan, en Pick- and Place station användes för denna rapport. En metod på fem steg användes för att skapa den virtuella produktionscellen. (Guerrero, et al., 2014)

1. Förstå systemet, hur det fungerar och hur kommunikationen går mellan de olika systemens in och utgångar.
2. CAD- Skapa och/ eller förbered alla CAD-filer för att skapa 3D-grafiken till den virtuella miljön. Dessa delades in i två kategorier, statiska och rörliga. Alla statiska delar sammanfördes till en modell då de inte skall röra sig i förhållande till varandra. Alla rörliga delar blev alla unika objekt i den virtuella miljön för att kunna röra sig i förhållande till varandra.
3. Skapa den virtuella miljön med kinematikregler om hur de olika lederna och länkar kan röra sig i förhållande till varandra och med vilka hastigheter de verkar. I detta steg kopplas även PLC-logiken ihop med den virtuella miljön genom att logiken kördes i en soft-PLC.
4. PLC-programmet prövas och verifieras med hjälp av den virtuella miljön.
5. PLC-programmet laddas in och körs på den fysiska utrustningen. Den virtuella maskinen kan fortfarande vara sammankopplad med PLC:n när den körs på den fysiska utrustningen och kan då användas som ett övervakningssystem. (Guerrero, et al., 2014)

Några av de utmaningar som de tidigare studierna visa på är bl.a. Att mjukvaran är dåligt anpassad att prata med varandra och det krävs mycket jobb med att manuellt koppla I/O signaler mellan de olika programvarorna. Systemen, var för sig, är väl utvecklade för att simulera en PLC, robot controller eller bygga upp en 3D-miljö men det saknas ett system för att skapa en helhet. (Markis, et al. 2012)

Enligt Binnberg & Johansson (2016) var en annan utmaning att konvertera de CAD-modellerna som var tilldelade till ett format som Simumatik3D kunde hantera. Det var också nödvändigt att minska upplösningen på CAD-modellerna för att minska påfrestningen på systemet då mindre komplexa filer kräver mindre av datorns prestanda att hantera. En annan utmaning var också att hela projektet behövde köras på en och samma dator vilket gjorde att alla programmen kopplade till emuleringen fick dela på en enda dators resurser. (Binnberg & Johansson, 2016)

3.2 Digital Twin

Rosen, Von Wichert, Lo, & Bettenhausen, (2015) beskriver i en artikel hur Digital Twin och Autonom produktion hör ihop och att de kommer vara viktigt för framtidens produktion. Författarna menar att med en högre grad av digitalisering av produktionssystem kommer möjligheten till att kraftigt öka produktiviteten. Genom att bygga upp ett modulärt och autonomt produktionssystem fås ett system som automatisk kan anpassa sig till oförutsedda händelser utan kravet på ingrepp av personal. Förutsättningen för detta menar författarna är att all maskinutrustning har tillgång till en Digital Twin av sig själva. Rosen et. al. (2015) beskriver att Digital Twin-system är nästa stora steg i simulering och optimeringsteknik. (Rosen, et al., 2015)

Debroy, et al. (2016) ville ta reda på hur långt utvecklingen har kommit för att kunna skapa en Digital Twin för utrustning för additiv tillverkning av metall, 3D-skrivare i detta fall. Syftet är att kunna emulera själva processen och resultatet av en utskrift för att snabbare och billigare få ett korrekt och önskat resultat när det kommer till den verkliga utskriften. (Debroy, et al., 2016).

Författarna konstaterar att mycket forskning finns kvar att göra för att få en pålitlig Digital Twin för additiv tillverkning men att det är inom räckhåll att kunna konstruera en av första generationen. För att gå vidare menar författarna att det kommer krävas sammanställda databaser med information om olika legeringars egenskaper i olika temperaturer och hur de beter sig under smältning och stelning. (Debroy, et al., 2016)

3.3 Slutsatser litteraturstudie

Fyra artiklar om virtuell idrifttagning och två artiklar om Digital Twin har analyserats. Det kan konstateras att både dessa ämnen är relativt nya inom produktionsteknisk forskning (Debroy, et al., 2016). När det kommer till digital Twin definierades begreppet först 2010 och 2012 av NASA i deras tekniska vägledning (Rosen, et al., 2015); (Shafto, et al., 2010); (Shafto, et al., 2012). Konceptet virtuell idrifttagning är beroende av att det finns någon form av Digital Twin inblandad i processen (Makris, et al., 2012); (Guerrero, et al., 2014). Författarna till de olika studierna hittar olika utmaningar för att lösa sina uppgifter och gemensamt är att mjukvaran ännu inte är särskilt långt utvecklad. Andra utmaningar de stöter på vid virtuell idrifttagning är att få kommunikationen att fungera på ett smidigt sätt mellan simuleringsprogram och de olika programmen som kör programkoden från t.ex. robot och PLC. Detta är en utmaning som kommer att beaktas även i detta arbete. Hanteringen av 3D-CAD-ritningar är också en utmaning som behöver beaktas. Binnberg & Johansson (2016) har beskrivit metoder för konvertering och förenkling av CAD-modeller som avses användas i detta arbete.

Det kan också konstateras att anledningen till att försök med virtuell idrifttagning görs av författarna med förhoppningen att få en kortare och effektivare väg från konstruktion till drift vilket visades av (Makris, et al., 2012); (Debroy, et al., 2016); (Binnberg & Johansson, 2016)

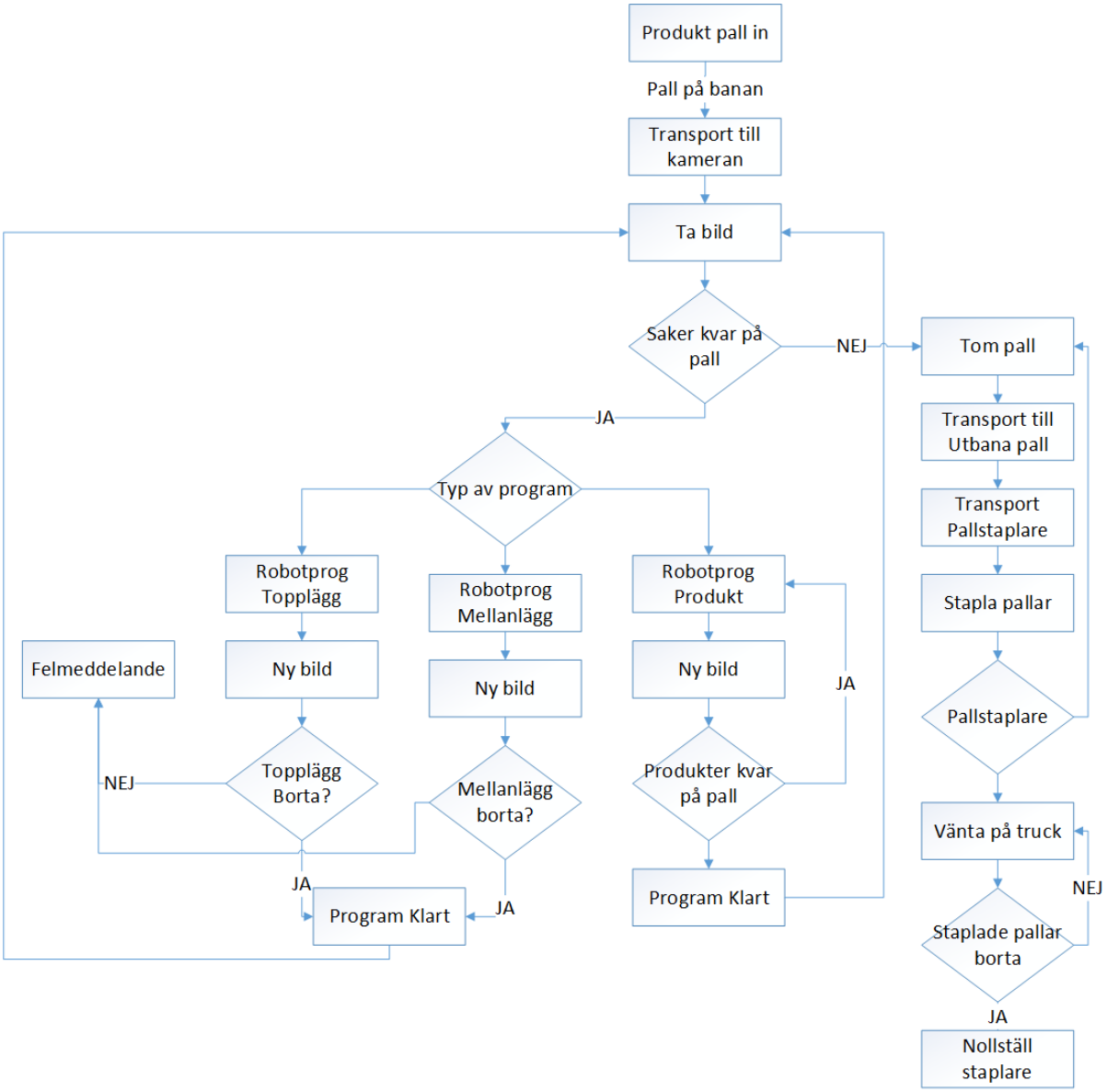
Konceptet för Digital Twin går långt utanför det om Virtuell idrifttagning. En Digital Twin skall kunna användas under hela utrustningens livscykel för att kunna övervaka och förenkla bland annat övervakning och underhållsaktiviteter (Rosen, et al., 2015). Virtuell idrifttagning är bara en liten del av vad en Digital Twin kan användas till.

Vidare konstateras att inget tidigare arbete hittats som försökt att inkludera ett system med visionguidning med en emuleringsmodell. Då visionguidning förekommer i flera av de robotceller som Elektroautomatik bygger finns önskemål om att inkludera det i studien. Enligt vad litteraturstudien visar kommer detta arbete vara helt unikt när det kommer till att inkludera visionguidning i en emuleringsmodell.

4 Funktionsbeskrivning av system

Kapitlet beskriver funktionen av det verkliga systemet som ligger till grund för utformningen av emuleringsmodellen.

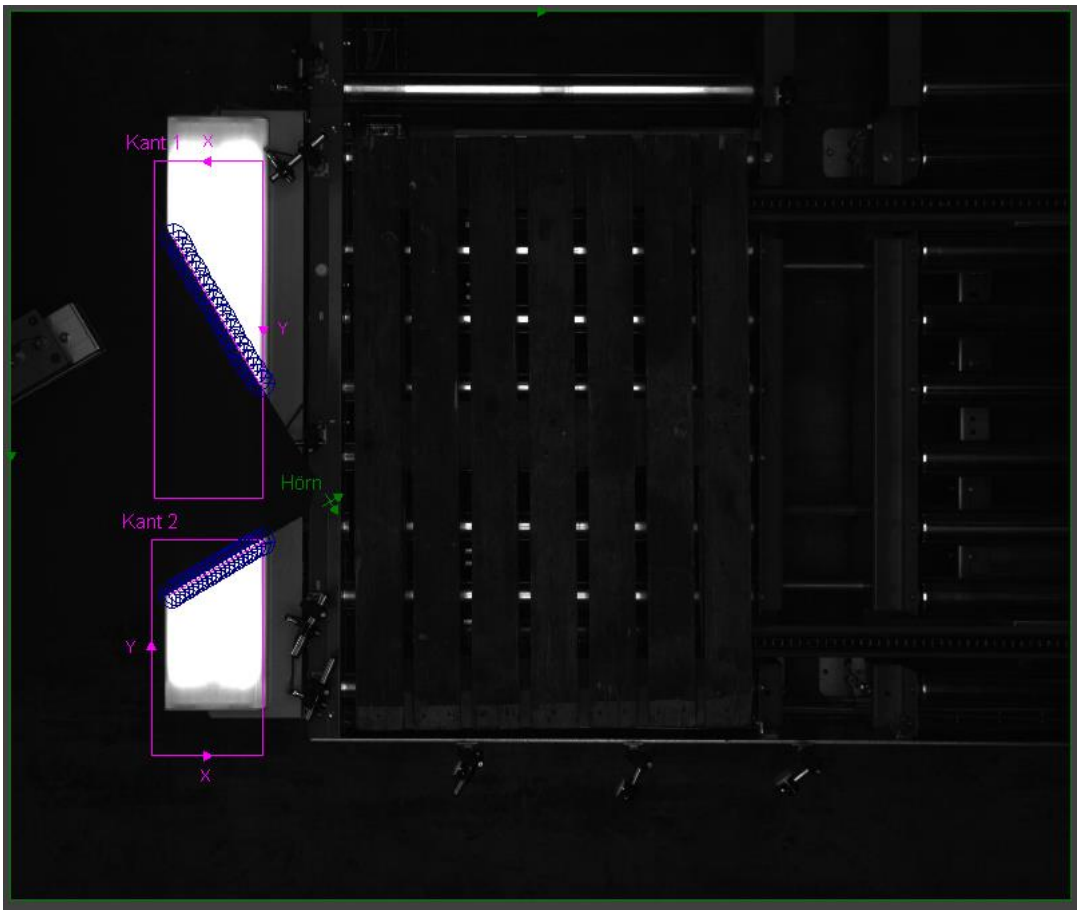
Produktionscellen som studerades var en sorteringscell bestående av två parallella rullbanesystem där layouten är spegelbilden av varandra, inbana 1 och inbana 2. Rullbanesystemen servas av en robot av modellen IRB 6700. Guidningen av roboten sker med en Cognex In-Sight 5605 som skickar koordinater till roboten och vilket objekt den identifierat. PLC:n styr sedan vilket program roboten ska utföra. Produkterna levereras till fabriken på pallar med fyra produkter per plan och mellanlägg av plast mellan varje plan, på varje pall ligger upp till sex lager med produkter och överst ligger trälock. Pallen med produkter lämnas av en truck på rullbanan och transporteras fram till robotens arbetsområde. När pallen är framme tar kameran en bild och sänder vilken artikel det är; lock/mellanlägg, produkt eller tom pall. När PLC:n identifierat vilket program roboten ska utföra hämtar roboten koordinaterna ifrån kameran och programmet utförs. Roboten är utrustad med ett dubbelverkande verktyg, den har en gripare för produkterna och två vakuum kuddar för lock och mellanlägg. För flödesdiagram över produktionscellen se figur 9.



Figur 9. Flödesdiagram över processen

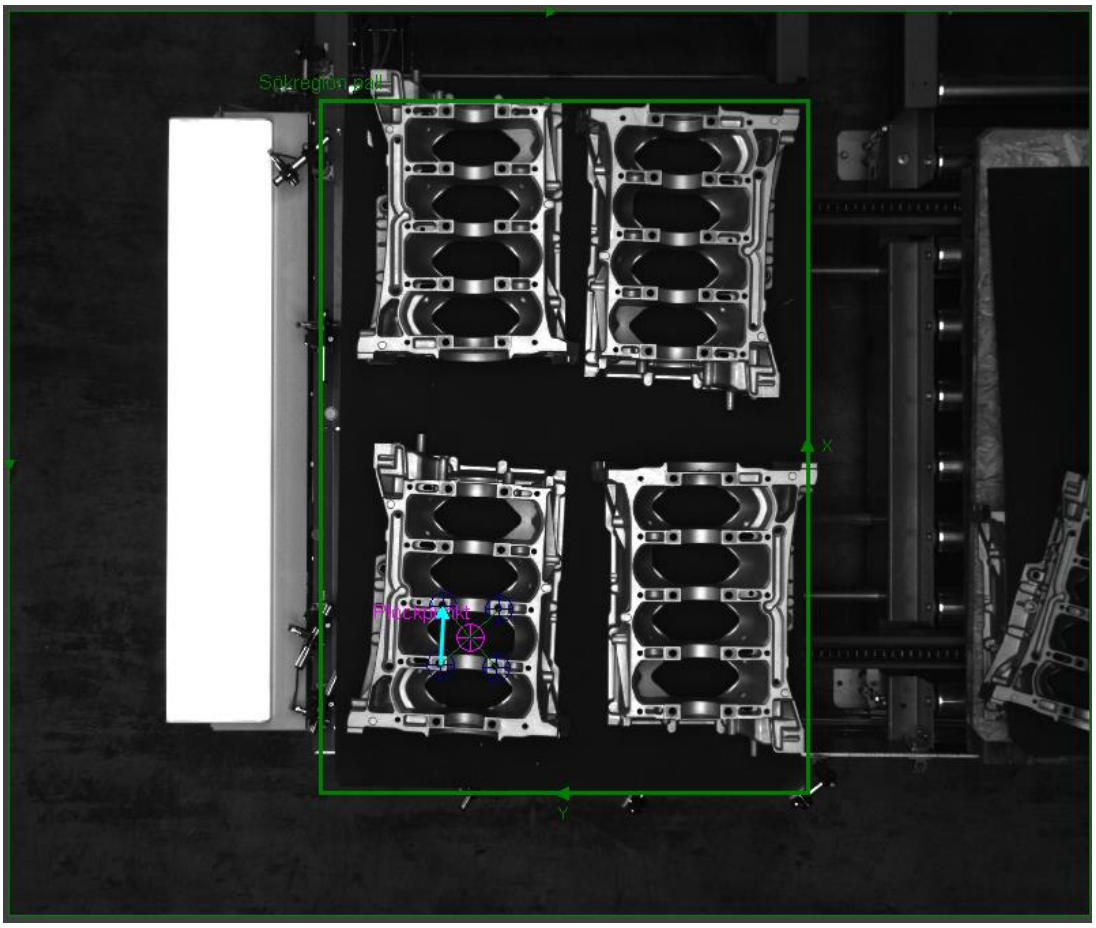
Lock: Första gången kameran tar en bild av ett lock/mellanlägg skickar PLC:n till roboten att det är ett lock den ska hämta. Med vakuumverktyget trycker roboten mot locket och registrerar vilken höjd locket ligger på och sparar den i programmet. Roboten aktiverar sedan vakuumet och lyfter locket och för det med ena hörnet emot en ljusplatta (Figur 10) där kameran tar en bild och skickar koordinaterna med lockets positionering. Med koordinaterna kan roboten räkna ut vart den har locket i förhållande till sitt verktyg för att sedan kunna stapla dessa fint på avsedd plats.

Mellanlägg: Roboten gör samma procedur som med topplocket men den lägger mellanläggen i en separat hög intill locken. Locken och mellanläggen staplas på pallar precis utanför cellens staket och är lättåtkomliga för att hämtas med truck.



Figur 10 visar en bild ur kamera i den riktiga produktionscellen när den tar en bild uppdraget att lokalisera hörn när roboten håller upp ett mellanlägg över en ljusplatta.

Produkt: När kameran identifierar att det ligger produkter överst, identifierar kameran vilken koordinat varje produkt ligger på samt dess orientering. Kameran räknar sedan ut vart roboten ska ha sitt TCP (Tool Center Point) och skickar dessa x- och y-koordinater till roboten (Figur 11). Roboten som redan har plockat av ett lock eller mellanlägg vet då på vilken höjd dessa produkter ligger och vilka koordinater de har på arbetsområdet. Varje produkt plockas upp och skannas av en datamatrix läsare, denna information kollas med ett övergripande system som identifierar om detta är rätt artikel. När roboten har fått okej att produkten överensstämmer lägger roboten produkterna på ett rullbanesystem vidare mot produktion.



Figur 11 visar en bild ur kameran i den riktiga produktionscellen när den identifierar och lokaliserar plockpunkt för roboten

När alla produkterna och locken är bortplockade skickar kameran att det ligger en tom pall på rullbanan, när PLC:n fått denna information skickar den pallen vidare på transportbanan och vidare till en pallstaplare. Pallstaplaren tar varje pall och staplar dessa till en stapel som är upp till tolv pallar hög, när pallstaplaren är full skickas stapeln ut för en truck att hämta.

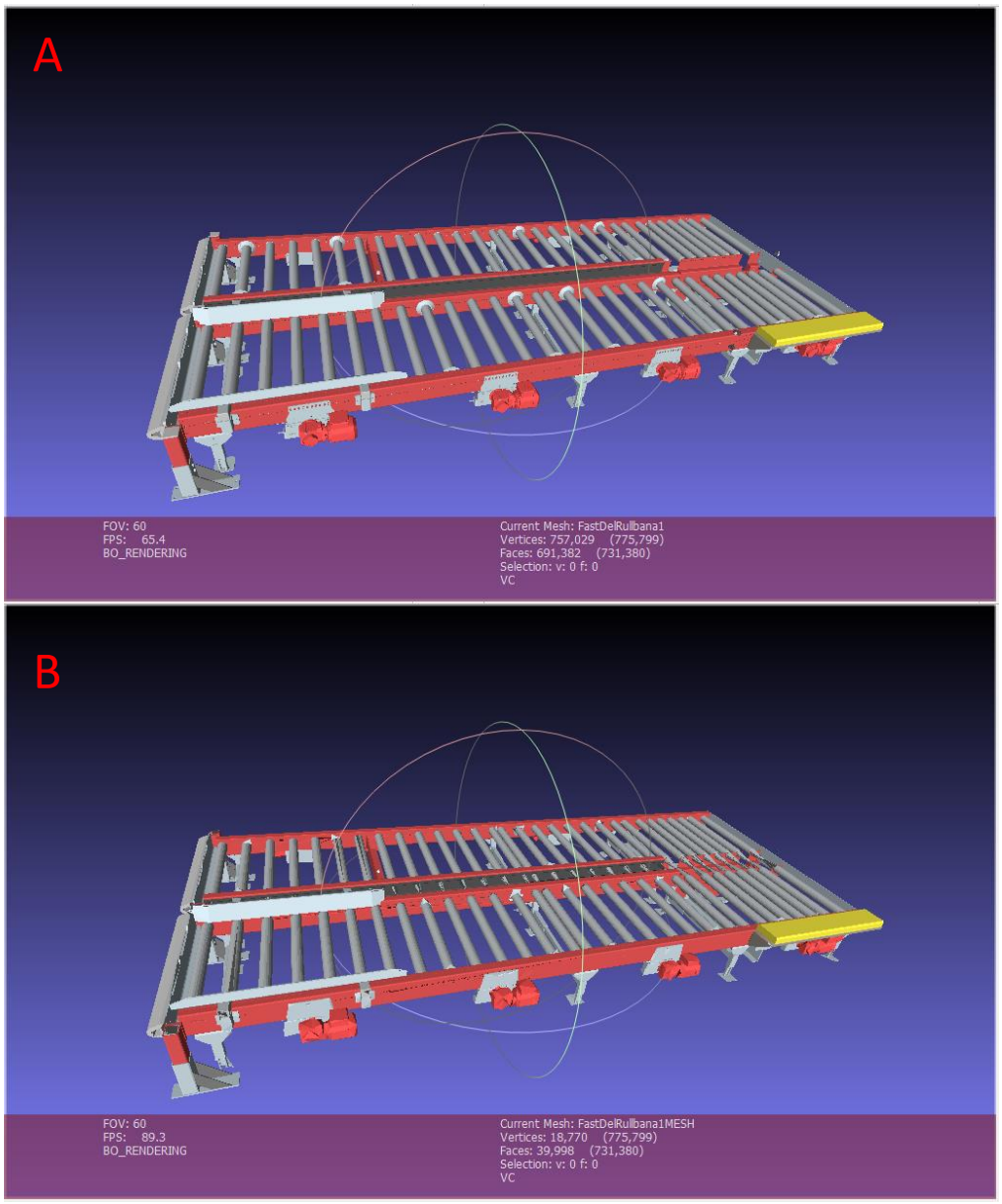
5 CAD-modeller

I kapitlet beskrivs arbetsgången för bearbetning av 3D-CAD-ritningar använda i emuleringsmodellen.

3D-modellerna som används i emuleringsmodellen är i grunden samma som elektroautomatik skapat när de konstruerat Produktionscellen i sin 3DCAD-programvara Inventor. 3D-modellerna levererades i fil-formatet Jupiter Tesselation (JT) och importerades till RobotStudio. Hela produktionscellen var i detta läge uppbyggd efter den layout vilken den konstruerats av EA. De ingående 3D-modellerna är mycket detaljerade och denna höga detaljrikedom är inte nödvändig för att användas i emuleringsmjukvaran Simumatik3D. När CAD-modellerna används i emuleringsmodellen med fysik och rörelser är det mycket krävande för grafikkortet. I originalskick skulle modellerna dra onödigt mycket resurser utan att den höga detaljrikedomen utnyttjades, därför behövde modellerna förenklas något för att bli mer lättanvända.

Från RobotStudio exporterades 3D-modellerna utifrån hur de skulle användas i Emuleringen. Detaljer som skruvar och invändiga mekanismer togs bort för att förenkla modellerna. Det viktigaste i uppdelningen av modellerna var att alla delar i emuleringen, som skall kunna röra sig i förhållande till varandra blev separata 3D-modeller. Formatet som valdes för exporten ur Simumatik3D var VRML (Virtual Reality Modeling Language) då detta var kompatibelt med den senare importen till Simumatik3D.

Programvaran som valdes för att förenkla 3D-modellerna ytterligare var MeshLab. I MeshLab minskades antalet ytor till ca: en fjärdedel genom att använda funktionen Quadratic Edge Collapes. En mycket lättare fil erhölls utan att kompromissa för mycket på kvaliteten. Efter förenklingen av modellen användes ett filter smoothing face vertices som rätar ut plana ytor eftersom förenklingen reducerar antalet vinklar och det skapar ojämna plan i modellen. Ett exempel på en före- och efterbild på ett rullbanesystem visas i Figur 12 där antalet ytor minskats från 691380 till 39998. När 3D-modellerna förenklades i MeshLab tappades informationen om viken färg de olika ytorna i originalfilen hade. Detta ledde till att alla förenklade 3D modeller som importerades i Simumatik3D bara kunde vara enfärgade.



Figur 12. rullbanan A är innan förenkling av modell i MeshLab och rullbanan B är efter förenkling

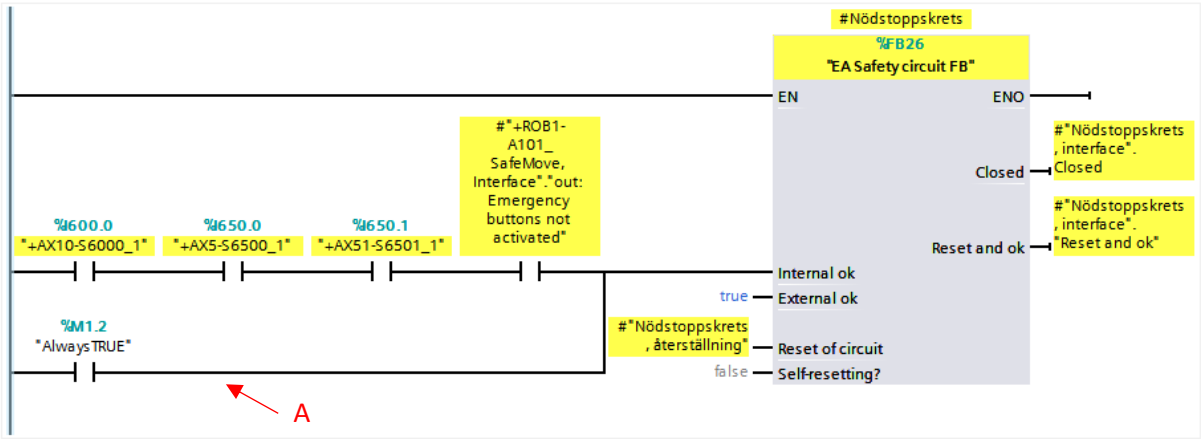
3D-Modellerna som används i emuleringsmodellen är enbart för att ge en visuell representation av produktionscellen och har ingen egen fysik. Fysikstrukturer läggs in där det behövs och en 3D-modell kan sedan följa denna fysik för att ge en korrekt visuell bild av hur cellen ser ut i verkligheten.

6 Virtuell miljö

I detta kapitel beskrivs hur den virtuella miljön är uppbyggd i de olika programmen och hur de kommunicerar med varandra.

6.1 Programkod PLC

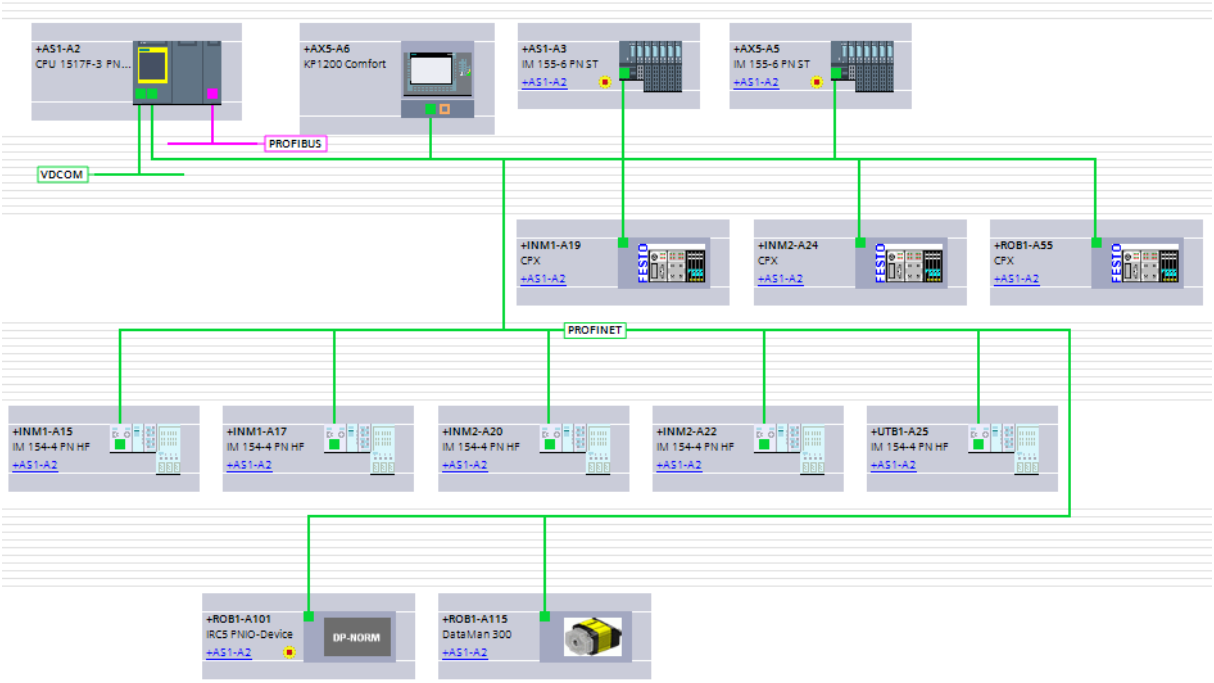
Programvaran som användes i emuleringen för PLC-programmering var Tia portal V14 SP1. Programmet som användes i emuleringen var programmerad sedan tidigare av Eektroautomatik för den verkliga robotcellen vilket i detta fall betyder att säkerheten redan finns med i programmet. Eftersom emuleringsmjukvaran som användes inte klarar av att emulera nödstopsknappar och säkerhetsfeedback behövdes säkerheten i programmet kringgås. Genom att söka igenom programmet och göra en alternativ gren förbi säkerhetsknapparna kunde knapparna kopplas förbi men bibehålla programmets uppbyggnad. Ett exempel på detta visas i Figur 13.



Figur 13. En säkerhetskrets som är byglad med AlwaysTRUE (A).

Feedbacklooparna ifrån knapparna togs bort för att inte få felmeddelande i programmets HMI. När nödstoppen och feedbacken var förbikopplade uppstod felmeddelande i programmet och PLC:n stannade, detta löstes genom att programmera bort de felmeddelande som stoppade programmet. Genom att skriva in en variabel, ”AlwaysFALSE” som stoppar upp programkoden innan felmeddelandet skickas, kunde stoppet i programkoden undvikas. Tia laddar upp PLC-programmet på PLCSIM Advanced som för TIA fungerar och beter sig som en riktig PLC CPU. Genom att använda protokollet OPC-UA kunde Simumatik3D kommunicera med den virtuella PLC:n. Med hjälp utav WinCC Runtime Proffesional kunde HMI-panelen emuleras i en dator och kommunicera med den virtuella PLC:n. Ändringen som gjordes i HMI-panelen var på vilken IP-adress som kommunikationen skulle ske över. Genom att ändra IP-adressen ifrån subnät 192.168.0.x till 192.168.1.x kunde också kommunikationen ske via en router vilket underlättade när flera datorer kopplades samman.

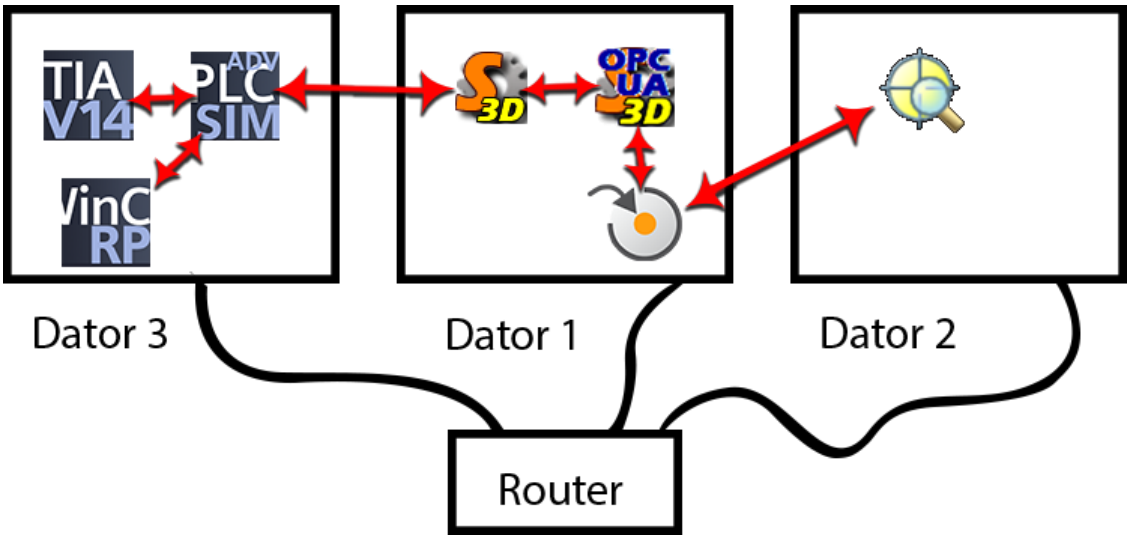
Kommunikationen mellan robot och PLC:n är en viktig del och kräver snabb överföring av signaler, det verkliga systemet har en uppdateringsfrekvens någonstans vid 20ms och PLCSIM-Advanced uppdateras med en frekvens av 100ms, genom denna fördröjning behövde signalerna grupperas och sändas som Words och Bytes för att snabba upp kommunikationen. Genom att använda sig av de befintliga I/O-modulerna i programmet kunde signalerna grupperas med modulnamnet (Se Figur 14). Då programmet skickar signaler i Words och Bytes kan signalerna grupperas som 16- respektive 8-bitar och sändas som en signal mellan programmen och minska kommunikationen ifrån 495 signaler till 80.



Figur 14. Systemöversikt PLC.

6.2 Kommunikation

Det systemet som byggdes upp består av tre stycken datorer uppkopplade med Ethernet via en router. Routern användes för att kringgå begränsningarna av rättigheter att sätta statiska IP-adresser på de olika datorerna, via routern kunde IP-adresser tilldelas till de olika datorerna. Figur 15 visualiserar hur kommunikationen sker mellan programvarorna på de olika datorerna. En kommunikationsstandard som användes är OPCUA och enligt OPC Foundation (2017) står OPCUA för Open Platform Communication Unified Architecture, vilket är en standard för industrikommunikation mellan maskiner.



Figur 15 visualiserar hur kommunikationen är uppbyggd mellan de olika mjukvarorna och att de är fördelade på dator 1 - 3.

Dator 1 Kör emuleringsprogrammet Simumatik3D och Robotstudio. Robot studio och Simumatik3D kommunicerar via OPCUA för att synkronisera robotens axelrotation och för att läsa och skriva I/O:s i roboten. Via Telnet-protokoll skickar robotprogrammet i Robotstudio kommandon och läser resultat i dator 2 från Vision-emulatorn i programmet Cognex In-Sight Explorer. Telnetkommunikationen från roboten sker med Socket kommunikation och kräver PC interface aktivt som option i robot controllern.

Dator 2 kör Cognex In-Sight Explorer med vision-kamerans emulering. Bilder läggs manuellt in i kameraemulatorn från Simumatik3Ds kamera. Beroende på vilket uppdrag roboten har gett Vision-programmet tolkas bilden och resultatet skrivs i celler i Vision-programmets kalkylark. Robotprogrammet får värden för vad kameran identifierar samt dess koordinater genom att läsa i de aktuella cellerna. Denna kommunikation från robotprogrammet sker över Telnet-länk med något Cognex kallar ”Native Mode Commands” vilket är en lista av kommandon som kan skrivas till Vision-programmet.

Dator 3 exekverar PLC programmet via PLCSIM Advanced. PLCSIM Advanced har en egen OPCUA server som gör att Simumatik3D kan läsa och skriva I/Os direkt i PLC-programmet. Dator 3 har också WinCC Runtime Professional som emulerar HMI-panelen till PLC:n.

6.3 Förenklingar

De förenklingar som gjordes i projektet har ändrat vissa aspekter av programmet men målet var att förändra så lite som möjligt av funktionen av programmet genom dessa förenklingar. Det första som identifierades och krävde förenkling var säkerhetens påverkan av programmet. Genom att låta de säkerhets block som byglades vara kvar men ändå koppla bort knapparna kunde programmet och avbrott som kodats in vara kvar. Eftersom funktionsblock som stoppar produktionen vid eventuella felkoder är kvar stoppas robot och produktionscell ifall kriterierna för stopp uppfylls även utan säkerhetsknappar. Genom att använda samma variabel genom hela programmet och ändrat minimalt i programmet kan programmet med lite modifikation laddas in i den riktiga robotcellen och fungera som det ska. Något annat som förenklades i modellen var Feedbacken ifrån säkerhetsknapparna vilket gav felkoder då programmet förväntade sig att få signalen tillbaka, detta löstes genom att sätta in en variabel som blockerar felkoden att sändas. För att programmet är uppbyggt på ett redundant programmeringssätt sätts programmet i stopp ifall felkoder ifrån knapparna uppstår, men eftersom felkoderna aldrig sänds undviks programstopp. Robotens verktyg krävde också förenkling då emuleringsprogrammet hade begränsningar i hur ett robotverktyg kunde modelleras vilket skapade några små förändringar i både robotprogram och robotverktyget. En sensor ersatte verktygets kollision med objekt och de rörliga komponenterna i verktyget flyttades ifrån verktyget till basen av roboten för att bibehålla cellens verkliga rörelser men ingen visuell presentation av dem i emuleringsmodellen. Verktygets förändringar påverkade inte programmet avsevärt men det påverkade den visuella representationen av cellen, robotprogrammet ändrades för att detektera signalen ifrån sensorn som en kollision med verktyget. Ändringen ifrån kollisionen till sensorn anses inte påverka funktionen av cellen utan endast funktionen i själva verktyget. För en fullständig lista över de förenklingar som utförts se bilaga A.

6.4 Emuleringsmodellen i Simumatik3D

Emuleringsmodellen byggdes upp i Simumatik3D där fokus ligger på att systemet skall vara verklighetstroget gällande ingående komponenter och hur dessa är kopplade. Ett exempel är att en rullbana behöver en motor för att röra sig. En motor behöver en motorkontaktor som styr motorn och motorkontaktorn behöver spänning från en strömkälla och styrsignaler. Allt detta emuleras i Emuleringsmjukvaran. För att spara tid beslutades att enbart bygga upp ena sidan av produktionscellen, inbana ett, då inbana två är uppbyggd som en spegelbild av den första. PLC-programmet krävde dock att vissa signaler skulle vara till även från inbana två för att tillåta drift. Detta löstes genom att koppla dessa signaler via en brytare som när den är opåverkad är till.

För att skapa en representativ virtuell miljö användes 3D-CAD-modellerna och dess layout som utgångsläge. Genom att överföra delarna ifrån robotstudio kunde det interna koordinatsystemet användas för att föra in modellerna på rätt plats, Robotstudio använder sig utav millimeter och origo vid basen av roboten och Simumatik3D använder sig av meter men origo på samma plats. Genom att dra nytta av likheterna mellan programmen kunde modellerna enkelt placeras ut på rätt koordinat. Efter att modellerna var på plats lades de mekanismer och fysiska objekt in som behövdes för att ge en tillräcklig representativ miljö för att klara emuleringsmålen. En virtuell strömkälla lades till för att mata el till alla förbrukare. En av dessa förbrukare av el är tryckluftskompressorn vilken försörjer samtliga pneumatiska komponenter med tryckluft, andra är motorkontaktorerna, PLC och robot.

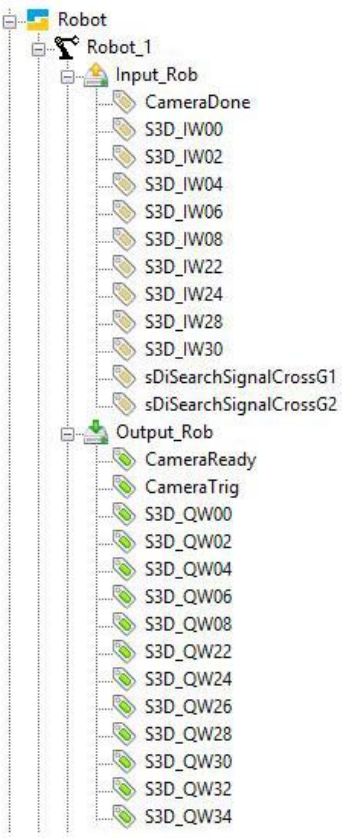
6.5 Emulering av roboten

För att köra robotprogrammet används ABB RobotStudio vilket kommunicerar med Simumatik3D över OPCUA. Alla in- och utgångar i roboten som skall kommunicera med PLC:n grupperades i gruppsignaler om 16 bitar, Words, som underlättar kommunikationen då det endast skickas ett tal som sedan görs om i PLC:n till respektive bit. Detta gav att det blev färre unika I/Os (inputs och output signals) att skriva i OPCUA och Simumatik3D och liknande manöver genomfördes i Siemens Tiaportal för PLC.

Roboten är underordnad PLC:n i produktionscellen och alla uppgifter som roboten utför initieras från PLC:n och sedan rapporterar roboten till PLC:n när den är redo för nästa instruktion. All kommunikation med kameraemulatorn sker via roboten men det är PLC:n som ger uppdraget till roboten att ta bild och resultatet från kameran vidarebefordras av roboten till PLC:n.

Roboten har många olika signaler i både BOOL, och Integer som skall kommuniceras med PLC:n. Detta ger att 21 stycken variabler synkroniseras i stället för 137 som antalet blir om alla I/Os skulle synkroniseras var för sig. Eftersom den aktuella versionen av PLC använder en CPU som tillämpar Byte Swap löstes detta för roboten genom att göra det när de grupperade in och utgångssignalerna adresserades (se Figur 16). Genom att skriva t.ex. ”8–15,0–7” istället för ”0–15” kommer första och andra Byten att byta plats och då stämmer det med hur PLC:ns CPU behandlar adresseringen i Words (se under Robotstudio i bild 16).

Simumatik3D



OPCUA

Name	Data Type	Access
CameraDone	Boolean	Write
CameraReady	Boolean	Read
CameraTrig	Boolean	Read
S3D_IW00	Word	Write
S3D_IW02	Word	Write
S3D_IW04	Word	Write
S3D_IW06	Word	Write
S3D_IW08	Word	Write
S3D_IW22	Word	Write
S3D_IW24	Word	Write
S3D_IW28	Word	Write
S3D_IW30	Word	Write
S3D_QW00	Word	Read
S3D_QW02	Word	Read
S3D_QW04	Word	Read
S3D_QW06	Word	Read
S3D_QW08	Word	Read
S3D_QW22	Word	Read
S3D_QW24	Word	Read
S3D_QW26	Word	Read
S3D_QW28	Word	Read
S3D_QW30	Word	Read
S3D_QW32	Word	Read
S3D_QW34	Word	Read

RobotStudio

I/O System

Name	Type	Device Mapping
S3D_IW00	GI	8-15,0-7
S3D_IW02	GI	24-31,16-23
S3D_IW04	GI	40-47,32-39
S3D_IW06	GI	56-63,48-55
S3D_IW08	GI	72-79,64-71
S3D_IW22	GI	184-191,176-183
S3D_IW24	GI	200-207,192-199
S3D_IW28	GI	232-239,224-231
S3D_IW30	GI	248-255,240-247
S3D_QW00	GO	8-15,0-7
S3D_QW02	GO	24-31,16-23
S3D_QW04	GO	40-47,32-39
S3D_QW06	GO	56-63,48-55
S3D_QW08	GO	72-79,64-71
S3D_QW22	GO	184-191,176-183
S3D_QW24	GO	200-207,192-199
S3D_QW26	GO	216-223,208-215
S3D_QW28	GO	232-239,224-231
S3D_QW30	GO	248-255,240-247
S3D_QW32	GO	264-271,256-263
S3D_QW34	GO	280-287,272-279

Figur 16. Jämför robotens sammanställda I/Os i de olika programmen

6.6 Emulering av Vision

Några förändringar i robotprogrammet genomfördes på grund av att kameraemulatorn inte kunde arbeta helt automatiskt. När PLC:n kommenderar roboten att ta en bild med kameran kommer den så småningom till en rutin som heter ”VisionTrig” där själva kommandot till kameran skickas via Telnet. När roboten skickar SW8 till kameraemulatorn ska den ta en bild och vänta på utvärdering. Men eftersom den inte är någon fysisk kamera kommer den istället för att ta en bild bara generera ett grått fält. För att få in en bild för utvärdering i kameran modifierades rutinen VisionTrig något i RAPID-kod enligt Figur 17. Efter att roboten skickat SW8 till kameraemulatorn sätter roboten utgången TrigCamera och där efter utgången CameraReady. Utgången CameraReady sätter knappen CameraDone till att lysa upp i Simumatik3D vilket är en indikering att en bild är tagen. Utgången TrigCamera Skickas till Simumatik3D och triggar där den kameran att ta en bild i 3D-miljön. Simumatik3Ds kamera har en inbyggd FTP-klient som har konfigurerats till att skicka en bild till In-Sight-emulatorn som har en FTP-server i dator två. Bilden hamnar då i en mapp i dator två och kan manuellt dras från mappen och in i In-Sights kameraemulator för utvärdering. WaitDI CameraDone gör att programmet väntar där tills ingången är satt. CameraDone sätts genom att trycka på knappen i Simumatik3D och då går robotprogrammet vidare och nollar utgångarna den just satt.

```
!***********************************************************************
!  PROCEDUR: VisionTrig.
!   Beskrivning: Triggar kameran att ta en bild och utvärdera
!***********************************************************************
PROC VisionTrig(VAR socketdev ComSocket,VAR errnum escapeError)
    VAR num nVisionTOKomm:=60;
    VAR string stReceived;

    IF NOT SocketGetStatus(ComSocket) = SOCKET_CONNECTED RAISE escapeError;
    SocketSend ComSocket\Str:="SW8" + "\0D\0A";
    SetDO CameraTrig ,1;
    SetDO CameraReady ,1;
    WaitDI CameraDone, 1;
    setDO CameraReady, 0;
    SetDO CameraTrig ,0;
    SocketReceive ComSocket \Str:=stReceived\Time:=nVisionTOKomm;
    IF StrPart(stReceived,1,1)<>"1" Raise escapeError;

    ERROR
    RAISE escapeError;
ENDPROC
```

Figur 17 visar hur rutinen CameraTrig ser ut med de extra kommandona i

Kameraemulatorn har Vision-programmet nerladdat i sig och roboten kan skriva och läsa ur Vision-programmet via Telnet med Cognex Native mode commands. När en bild laddas in i emulatorn kan den utvärderas beroende på de parametrar som är givna och roboten kan sedan läsa av resultatet av denna utvärdering.

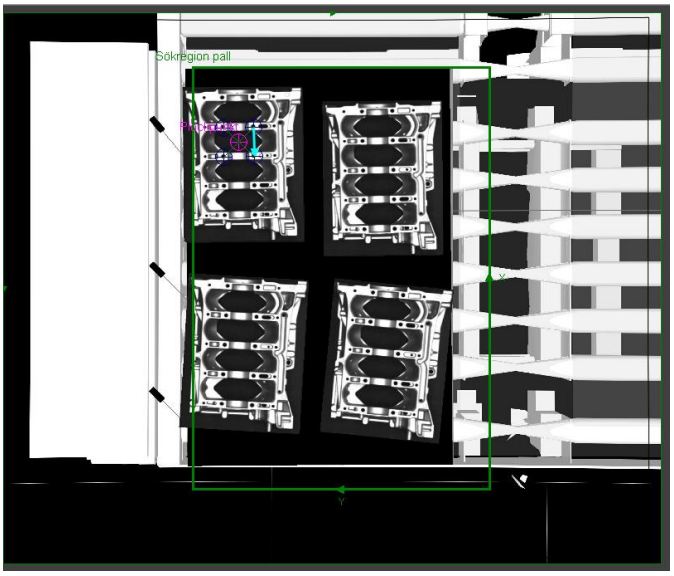
7 Test och verifiering av virtuell modell

I kapitlet beskrivs hur tester och verifiering utförts för att säkerställa rätt funktion på emuleringsmodellen.

Testningen av emuleringsmodellen har utförts löpande under hela uppbyggnaden och testningen av cellen. Vid uppstarten av kommunikationen har I/O-tester utförts för att identifiera vilka ingångar som behövde vara till när cykeln startade samt i vilka lägen alla komponenter ska vara i för att startvillkoren i PLC:n skulle vara uppfyllda. Användningen av den emulerade HMI-panelen har varit en stor hjälp när I/Os skulle felsöka då eventuella fel gav ett felmeddelande att just den signalen borde vara på eller av vid uppstart. Genom HMI-panelen har också larm kunnat identifieras om vart roboten står och vart den bör vara. Genom att använda samma variabelnamn i emuleringen som i grupperingslistan i PLC:n har det underlättat kommunikationen och samla ihop signalerna på ett enkelt sätt vid felsökningen av I/Os. Listan som skapades har alla signaler grupperade i Word (16 bitar) eller Byte (8 bitar) istället för att varje enskild I/O skickas för sig. På så vis behövde felsökningen endast ske i den grupp om 16 eller 8 bitar där felkoden uppstod och inte i hela I/O listan. Det som ställde till med problem i felsökningen av I/Os var att i Siemens använde Byte Swap, vilket innebar att första Byten och sista Byten byter plats.

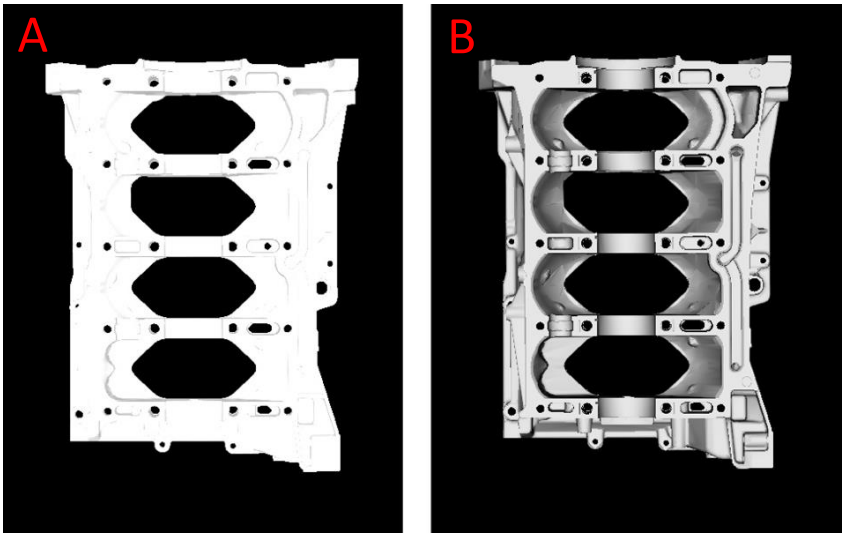
Vid modellerandet av fysiken i Simumatik3D användes ett enklare program för att säkerställa att fysiken fungerade på ett bra sätt med förbehåll för eventuella ändringar då riktiga programmet laddades in. Vid skarp körning av det riktiga PLC-programmet visade det sig att givarna låg på fel ställen och platserna justerades in. Det framkom också att sensorerna som monterats på robotverktyget inte fungerade som tänkt och fick förenklas genom att lägga till en sensor för att detektera kollision. När dessa justeringar var gjorda kunde en halv cykel köras i Automatik.

För att ställa in vision-kameran i Simumatik3D jämfördes bilden i Simumatik3D med en bild från den riktiga kameran monterad i den riktiga cellen. De första testerna med identifiering och lokalisering av produkter skedde med rektangulära boxar i samma storlek som produkterna och på dessa boxar lades en bild av produkten tagen med den verkliga kameran. Boxarna lades på olika höjder under kameran i emuleringen och kunde identifieras och lokaliseras av Vision-programmet. Ett exempel visas i figur 18.

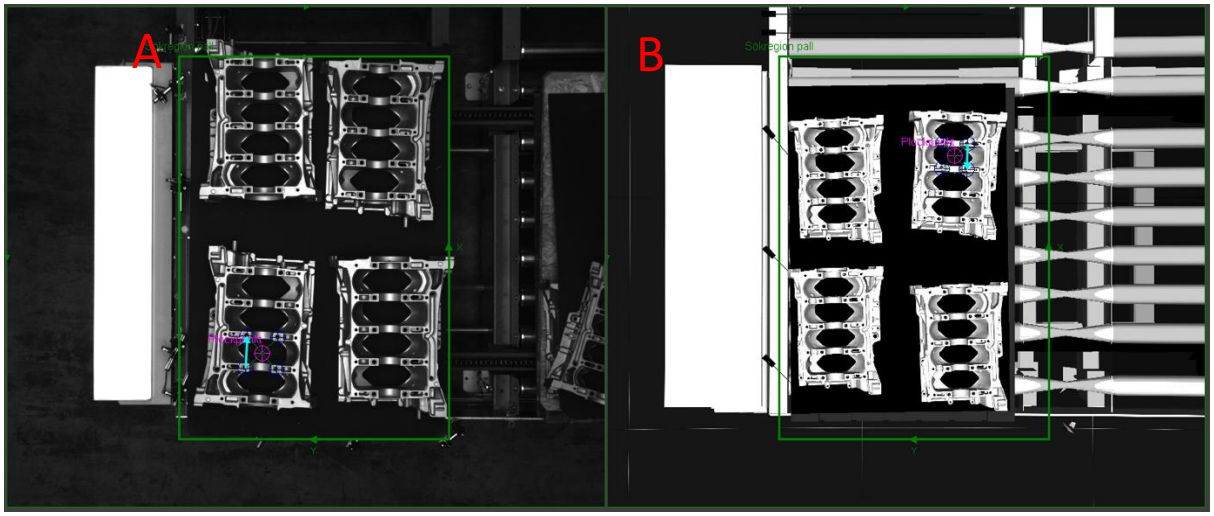


Figur 18. Vision-programmet identifierar bild tagen i Simumatik3D med produktboxar med foton på från den verkliga cellen.

Nästa mål var att klara av att identifiera produkten som CAD-modell. Till att börja med gick det inte att identifiera CAD-modeller men efter en uppdatering av grafikmotorn och möjligheten att ställa in ljuset i Simumatik3D kunde olika ljusuppsättningar testas tills en hittades som fungerade. Det vinnande konceptet var att det omgivande ljuset (Ambient) ställdes ner till nästan svart och ett oändligt ljus rakt ovanifrån belyste modellerna (se Figur 19). För att verifiera Vision-systemet jämfördes bilder tagna i den verkliga cellen och jämfördes med bilder ur emuleringsmodellen och ett exempel på detta visas i Figur 20.



Figur 19. A visar produkten innan uppdatering av grafik och ljussättning och B är produkten efter uppdateringen.



Figur 20 visar en jämförelse med en bild tagen ur den verkliga cellen A och en bild tagen ur emuleringsmodellen till B

7.1 Test av Robotverktyget

För att få robotverktyget att fungera krävdes en hel del tester. Till att börja med var ambitionen att få med alla rörelser i verktyget. Då detta blev för avancerat beslutades att enbart ha statiska delar i verktyget. Det som ingår i verktyget i emuleringen är grafik för att representera verktyget visuellt, en ”objekt lock” för vardera gripdon och en höjdsensor. Höjdsensorn skall endas vara till när verktyget kommer ner till den sökta höjden genom att gå emot den fysik som ingår i det objekt som roboten

söker höjd mot. När roboten sedan lyfter upp föremålet får denna inte trigga höjdsensorn då detta generar fel i systemet nästa gång roboten skall söka höjd för att lägga ned föremålet. Ca:10 tester göras med metoden ”Trial and error” för att hitta rätt förhållande på avståndet mellan ”objekt locker” och höjdsensorn. Det riktiga gripdonet har sensorer för när pneumatikcylindrarna är i sina ändlägen. Dessa signaler behöver PLC:n för att hålla koll på verktyget och för att fortfarande kunna få dessa signaler till PLC:n skapades cylindrar med givare som fyller denna funktionen och placerades vis sidan av utan att fylla en funktion i övrigt.

7.2 Test av automatikcykel

När gripdonet fungerade och kameran kunde guida produkt och hörn testades systemet att köras i automatik. En pall med två lager produkter med mellanlägg emellan och ett lock ovanpå kördes i maskinen. Maskinen fungerade som förväntat med undantag från att den manuella hanteringen av Visionsystemet ibland ledde till att PLC:n inte han få det värde som den förväntat sig av kameran och felkoden som skapades var ”Vision kan inte hitta lock/mellanlägg/mellandel”. I detta skede kunde systemet verifieras att bilder togs i det ögonblick de skulle och att roboten och rullbanorna kördes enligt beskrivningen. När roboten plockat av pallen och kameran skulle identifierat tompall upptäcktes att det inte var standardpallar som skulle köras i systemet. När en CAD-modell av en pall av rätt sort skaffats och färger och ljus justerats något kunde även denna identifieras i Vision-programmet och den fördes då vidare till pallstaplaren.

7.3 Verifiering

För att förstå hur robot och PLC-programmen fungerade understödde EA kontinuerligt med expertis från de programmerare som programmerat produktionscellen. Detta har även lett till att emuleringsmodellens funktioner har kunnat verifieras efter hand som de blivit klara. En annan hjälp till att förstå och se systemets funktioner har varit att använda den emulerade HMI-panelen i Siemens WinCC Runtime Proffesional. Genom att köra funktioner och läsa av händelser och givare från HMI-panelen har modellen kunnat verifieras.

8 Validering

I kapitlet beskrivs hur modellen validerats genom att utföra ett VFAT med kunden och jämföra resultatet med ett riktigt FAT utfört på den riktiga cellen.

8.1 Upprättande av VFAT-Protokoll

För att validera modellen utfördes ett VFAT på modellen. Testerna som genomfördes var tagna ur det vekliga FAT-protokollet som kunden hade för att kontrollera produktionscellen innan den godkändes för att levereras till kund. Då den emulerade cellen har en del förenklingar och begränsningar av vad som är möjligt att testa valdes områden ut ifrån det faktiska FAT-protokollet. Urvalet gjordes med en bedömning av vilka punkter modellen klarade av och som ansågs relevanta att genomföra. Av urvalet skapades ett virtuellt FAT-protokoll, VFAT.

Ett av målen som sattes upp i början av projektet var att använda ett VFAT för att utvärdera hur väl robotcellen som modellerades överensstämde med verkliga cellen och dess FAT. VFAT användes som en validering av modellen. Då den riktiga cellen som levereras av Elektroautomatik också genomgått ett FAT kunde protokollen och resultaten jämföras mellan det riktiga testet och det virtuella testet. De områden som testades i VFAT är HMI-panelen där knappar och layout kollas så att det överensstämmer med den verkliga cellen. Det som också testades var om de manuella funktionerna i HMI-panelen fungerade som de ska samt att korrekt feedback ifrån systemet presenterades i panelen vid rätt tidpunkt. Funktionen av den emulerade cellen testades och hur väl den virtuella modellens rörelser och funktioner överensstämde med den riktiga cellen. Det som också togs ut som en punkt att jämföra var felmeddelanden mellan systemen, t.ex. ifall en sensor går sönder i emuleringsmodellen skulle den agera på samma sätt som det verkliga systemet.

8.2 VFAT med kund

Kundens åsikter användes genom att jämföra hur väl den virtuella modellen överensstämde med den riktiga cellen då ett fel åkallades och den virtuella modellens agerande överensstämmande med den riktiga cellen. Ett flertal punkter jämfördes (se bilaga B) för att undersöka hur väl den virtuella modellen överensstämde med verkligheten. En av de punkter som kollades var hur den virtuella HMI-panelen agerade med blinkande ljus och fast sken på de olika knapparna, i detta fall överensstämde modellen väl med de tester som utförts i FAT. En av de punkter som inte överensstämde med FAT var hur utgångslägeskörning skulle fungera, en trasig givare i verktyget skulle sätta cellen i ett felläge och inte flytta roboten. Det som skedde i modellen var att roboten flyttade på sig och sedan försökte utföra uppdraget. Felkoden uppkom först när roboten skulle börja söka höjd mot pallen. Tyvärr undersöktes inte om det var PLC-programmet eller emuleringsmodellen som orsakade detta.

En punkt som kunden ansåg viktig att undersöka i en emuleringsmodell var cykeltiderna. P.g.a. de förenklingar som behövts. Övriga utvärderingspunkter som framkom var:

- Önskad möjlighet att testa kommunikation med överordnat system i emuleringen.
- Att använda en verklig HMI-panel menar kunden skulle ge en bättre känsla för operatören vid ett funktionstest.
- Vid tester av robotcell med risk för kollisioner bör möjligheten finnas att testa robotens rörelseövervakningssystem.
- För att nyttja emuleringsmodeller regelbundet behövs standarder på utformning, arbetssätt och programvaror utarbetas.

- Förutsättningarna för emuleringsmodellen med dess begränsningar klargörs för att säkerställa att det som testas virtuellt är representativt för det verkliga systemet.
- Att kunna testa säkerhetssystemet med nödstopp, säkerhetsbrytare, få rätt felmeddelande och larm.
- Om säkringar bryts skall rätt felkod och larm synas i HMI-panelen.

9 Diskussion

I detta kapitel diskuteras metoden och resultatet, hur det gick och vad som kan förbättras.

9.1 Målen

Målen har nåtts förutom att kalibrering av Vision-systemet inte genomförts. Detta för att det ansågs vara för resurskrävande att genomföra en kalibrering där ca:800 bilder behöver överföras manuellt från emuleringen till In-Sight emulatorn från Simumatik3Ds kamera. Att inte kunna lägga in bilder för utvärdering automatiskt i Cognex programvara är en medveten begränsning Cognex lagt till för att användaren skall vara i behov av att köpa en av deras kameror. En önskan är att Cognex kan lägga in automatisk bildupptagning i In-Sight-emulatorn som en licensfunktion. Ett VFAT-protokoll togs fram med de punkter som kunde utföras i modellen och VFAT genomfördes tillsammans med kunden. Protokollet var användbart och kunden kunde testa några av de punkter som uppkommit som anmärkningar när den fysiska produktionsutrustningen testats. Återkopplingen från kunden under VFAT var mycket värdefull för att utvärdera projektet och flera slutsatser kunde dras. Vision-guidning har kunnat genomföras med mycket goda resultat och vision-programmet kunde användas orört från den verkliga cellen. Enbart 3D-modeller från konstruktionen av den riktiga cellen har använts för att representera alla ingående delar. Dock har modellerna fått förenklas en hel del och färgerna från originalmodellerna kunde inte fås med till emuleringen.

Hade målen varit smalare och mer inriktade hade det antagligen gett mer kvalitet på resultatet. En variant som diskuterats var om fokus hels skulle lagts på att undersöka möjligheten att emulera ett Vision-system. Men detta hade i sin tur varit svårt om det inte funnits ett fullständigt system att utvärdera resultatet i. Målen kändes till en början mycket rimliga när tidsplanen sattes upp men när modellen blev klar och experimenten först kunde påbörjats i vecka 17 istället för i vecka 11 som först var planerat, ledde det till att kvaliteten på måluppfyllnaden sänktes. Även ambitionsnivån sänktes något när modellens färdigställande drog ut på tiden. Exempelvis han inte det senaste robot och PLC-programmet implementeras vilket kan ha lett till vissa skillnader i beteende mellan det verkliga och det emulerade systemet.

Något examinatorn kommenterade när han läste projektbeskrivningen var att det kändes som ett väldigt stort projekt och att enbart fokusera på emulering av vision skulle räcka till ett examensarbete. Detta borde gett författarna en tankeställare och fundera på om det kunde gått att smalna av projektet något för att öka kvaliteten på studien.

9.2 Metoden

Under rubrik 1.5.2 Metod för byggande av emuleringsmodell, hade en metod valts ut för att på ett strukturerat sätt kunna bygga upp emuleringsmodellen med hög kvalitet. I den lista på 11 punkter som beskriver stegen utfördes steg 1 – 4 men på steg 5, programkommunikation, kom de första stora problemen. Kommunikationen mellan Simumatik3D och siemens PLCSIM fungerade inte som väntat med Siemens TIA Portal 14. Då detta är en helt vital del att få simuleringen att fungera lades mycket resurser på felsökning och kontakt med Siemens support. Under tiden påbörjades punkt sex och sju, bygget av modellen och skapandet av mekanismer i Simumatik3D. Ett enklare PLC-program skapades i Codesys för att pröva att alla mekanismer fungerade som det var tänkt. Detta steg var en verifiering att modellen med ingående fysik fungerade som den skulle innan det riktiga PLC-programmet kördes i modellen. Ett extra steg som också genomfördes för att få testa emuleringsmodellen innan

kommunikationen fungerade var att ladda in programmet i en riktig CPU som sedan kopplades upp mot Simumatik3D. Detta var inte heller ett planerat steg men det gjorde att modellen kunde börja testas innan den virtuella CPU:n var igång. En lärdom som tagits av detta är att konstruktionen av emuleringsmodellen kan mycket väl påbörjas som ett separat projekt och kan delvis testas utan att kommunikationen fungerar. En annan lärdom är att i förväg tydligare identifiera vilka aktiviteter som måste ske i bestämd ordning som är beroende av varandra och vilka som kan ske oberoende av varandra.

9.3 Vision

Att kunna emulera Vision-systemet har bjudit på flera utmaningar. De utmaningar som identifierats innan experimentet påbörjades var att få grafiken tillräckligt verklighetstrogen för att Vision-programmet skulle kunna utvärdera bilder tagna i emuleringen på samma sätt som i den verkliga maskinen. Vision-programmet fungerade för att identifiera både produkter och mellanlägg/lock från emuleringen tack vare att programmet var så pass robust och klarade de skillnader som blev mellan verkligheten och emuleringsmiljön i Simumatik3D. För identifiering av produkter användes i första steget block med samma ytterdimensioner som produkten och med en bild av den riktiga produkten på den. Bilden som användes kommer från den riktiga kameran monterad i den riktiga cellen. Detta var som väntat enkelt för vision-programmet att identifiera. Utmaningen var istället att kunna identifiera om produkten skickades in som en 3DCAD-fil av den riktiga produkten. Att använda CAD-produkter var något som ansågs som en av de största utmaningarna vad gällande emulering av vision. Med den grafikmotor Simumatik3D hade i början av projektet gick det inte att få till en tillräckligt verklighetstrogen bild av produkten, då kontrasten var väldigt låg och det saknades skuggor. När Simumatik3D:s grafikmotor sedan uppgraderades och det gavs möjlighet att justera ljus och skuggor mer i detalj, kunde ljuset sättas upp på ett sådant sätt att 3DCAD-modellen kunde representeras på ett tillräckligt verklighetstroget sätt för att kunna identifieras av Vision-programmet. Detta är en stor fördel då det betyder att Vision-programmet skulle kunna utvecklas enbart med hjälp av tillgänglig CAD-data under konstruktionsfasen och inte är beroende av den verkliga maskinen och produkterna.

För att få den virtuella kameran att stämma överens med den vekliga kameran i produktionscellen krävs att ett antal parametrar går att ställa in på den virtuella kameran det är sensorns storlek i pixlar, dimensioner i mm och optikens brännvidd. När parametrarna är inställda ska kameran kunna placeras på samma position som den verkliga kameran och ge en bild från emuleringsmodellen som motsvarar den verkliga kameran. En genväg som skulle vara möjlig är att enbart förhållandet mellan optikens brännvidd och sensorns dimensioner är den samma. Den kameran som användes i Simumatik3D hade inte möjligheten att ställa in parametrar på objektivet men det som var förinställt var tillräckligt nära för att fungera ändå.

9.4 Förenklingar och förändringar

De förändringar som gjordes genom projektet är saker som skulle kunna lösas i programmeringsfasen ifall emuleringsmodellen är färdig i ett tidigare skede. Genom att programmera säkerheten sist i både roboten och PLC:n kan det minimera ändringarna som gjordes i programmet, det skulle också ge programmerare möjligheten att testa programmet under tiden det skrivs. Genom att bygga emuleringsmodellen direkt efter konstruktionsfasen och ge programmerarna möjligheten att testa sina program i en emulerad miljö skulle tiden för att driftsätta maskinen minimeras. Testerna i den emulerade miljön skulle också ge möjligheterna att testa mängder av olika scenarion på ett enkelt sätt då programmen bara behöver startas om för att återgå till utgångsläget. Skillnaden emot den virtuella

miljön är att vid en omstart går alla fysiska delar tillbaka till sin ursprungliga position medan i den verkliga cellen krävs det flytt av alla fysiska delar ifall programmet hamnar i ett driftstopp. Den största fördelen av en emuleringsmodell är när den modelleras innan programmering sker men efter att konstruktionsunderlaget är färdigställt.

Det finns fördelar och nackdelar med att ha program, konstruktionsunderlag och en nästan färdigbyggd cell vid starten av en emulering. Fördelarna är att programmen redan är testade och nästan färdigställda vilket underlättar modellering av funktioner i emuleringen. Med ett förprogrammerat PLC-program kan funktioner i emuleringsmodellen verifieras mot programmet under tiden modellen byggs. Problematiken är att många fördelar med emulering försvinner eftersom emuleringsmodellen kunde ha bidragit till att enkelt identifiera eventuell felprogrammering och funktioner som inte fungerade som tänkt redan i programmeringsfasen. De förenklingar som gjorts kunde också ha undvikits om modellen byggts i konstruktionsfasen och bidragit till utformningen av programmen som användes istället för att behöva modifieras för att anpassas till emuleringen.

9.5 Hållbar utveckling

Det är inte helt självklart hur detta projekt kan påverka en hållbar utveckling. För att först se till den ekonomiska hållbarheten är denna kanske mest uppenbar. Hela syftet med en virtuell idrifttagning är att kunna spara resurser vid ett FAT eftersom flertalet tester genomförs i en virtuell miljö och inte behöver tillgång till den riktiga produktionscellen. Ur en ekologisk synvinkel kan det argumenteras för att det går åt mindre energi att köra tester i en virtuell miljö i en dator istället för i den verkliga maskinen. Således går det åt mindre tid- och energiresurser ju mer av idrifttagningstesterna som kan genomföras virtuellt. Med hänsyn till det sociala perspektivet är det inte lika uppenbart och det är svårt att motivera en påverkan åt det ena eller andra hållet. Möjligen kan det argumenteras för att personal slipper bära runt på produkter vid ett test om dessa moment kan utföras virtuellt. Detta skulle bidra till en bättre arbetsmiljö för de som genomför FAT. Vidare kan det argumenteras för att det går åt mindre personal och att det i förlängningen skulle kunna leda till arbetsbrist vilket då skulle vara dåligt för den sociala hållbarheten.

9.6 Tidsplan

Allt gick enligt tidsplanen ända fram till att PLC-programmet skulle kopplas upp mot Simumatik3D. Först då upptäcktes att den virtuella PLC som körs i Siemens PLCSIM inte kan koppla upp mot OPCUA. Då detta var en förutsättning för att kunna fortsätta med projektet lades mycket tid på att felsöka och kommunicera med Siemens support. Det framkom att PLCSIM Advanced behövde användas istället för PLCSIM som var tänkt från början och det fungerade nästan. När Elektroautomatiks PLC-program skulle laddas in i den virtuella PLC:n i PLCSIM Advanced kom en felkod som uppmanade till att kontakta supporten. Efter kontakt med support framkom att de var medvetna om buggen och ett den skulle vara löst i service pack 1 (SP1). När PLCSIM Advanced installerats och (SP1) till Tia portal hade släppts från Siemens var projektet efter tidsplanen. En åtgärd som genomfördes för att kunna börja experimentera med modellen innan den virtuella PLC:n var tillgänglig var att köra programmet på en verklig CPU. I den verkliga CPU:n kunde programmet laddas in och koppas upp mot Simumatik3D vilket medförde att projektet inte blev lika försenat som hade varit fallet utan användning av den riktiga CPU:n. Hur mycket detta faktiskt har försenat projektet är svårt att avgöra men uppskattningsvis har det uppkommit ca: fyra veckors extra arbete.

Så här i efterhand hade projektet kanske vunnit på att helt släppa virtuell PLC när det uppdagades att PLCSIM inte gick att använda och gå direkt över till en riktig PLC CPU, om en sådan varit tillgänglig hela projektet. En annan åtgärd hade varit att göra bättre efterforskningar för att se om programvaran som tänkts använda faktiskt klarar av det den var avsedd för och därmed hinna parera för kompabilitetsbristerna mellan programvarorna tidigare.

10 Slutsats

I kapitlet beskrivs de slutsatser som tagits av studien och hur konceptet kan vidareutvecklas.

10.1 Användbarhet av virtuell idrifttagning

Både EA och Kunden anser att de kommer kunna nyttja virtuell idrifttagning för att spara tid och resurser framöver. Projektet har fått upp ögonen en del hos kunden för vad som kan vara möjligt inom området och vad som kan förväntas inom en snar framtid. Den största användbarheten av en emuleringsmodell av en maskinutrustning idrifttagning anses, av både Elektroautomatik och kunden, vara för att programmerare skall kunna testa och verifiera sin programkod innan den verkliga modellen är uppbyggd. Detta skulle korta ner tiderna för idrifttagningen väsentligt då onödiga fel i programmering av funktioner kan upptäckas och kontrolleras innan programmet testas i den verkliga maskinen. Liknande slutsatser har även tidigare studier i området kommit fram till (Makris, et al., 2012 och Guerrero, et al., 2014). En fördel som också framkommit är att emuleringsmodellen kan skickas till kunden för att verifiera att cellen fungerar enligt överenskommet innan maskinen monteras.

Efter genomfört VFAT med kunden dras följande slutsats. För att kunna tillämpa virtuell idrifttagning behöver en standard utarbetas med vilka krav kunden ställer på en emuleringsmodell för att den skall vara godkänd för att användas i ett FAT.

10.2 CAD-modeller

En stor fördel med att använda CAD-modeller från konstruktionsunderlaget av produktionscellen var att layouten kunde föras över rakt av till emuleringsmodellen. CAD-modellerna som används för konstruktionen är dock för detaljrika för att kunna användas i emuleringsmodellen och behöver förenklas för att inte kräva för stora resurser av datorn, vilket även framkom från Binnberg & Johansson (2016). En sak som skulle kunna förbättras i underlagen för att bygga modellen är en layout på var alla givare sitter samt deras beteckningar. Injustering av givares positioner är något som skulle kunna göras i emuleringsmodellen och sedan kan detta underlag användas när den fysiska cellen monteras.

Två saker som skulle underlätta mer vore om CAD-modellen grupperas så att det tydligt framgår vilka delar som rör sig i förhållande till varandra och de delar som kommer att ha en statisk position till varandra. Slaglängder på cylindrar och hastigheter på rullband skall tydligt framgå i konstruktionsunderlaget.

10.3 Virtuellt FAT-test-protokoll

Att ha ett testprotokoll specifikt för att testa modellen har varit mycket värdefullt för att kunna validera modellen. Det har gett relevanta resultat att utvärdera som går att jämföra med det verkliga testet och det har varit lätt för kunden att komma med synpunkter baserade på kontrollerna utfört i VFAT. För att integrera det VFAT-protokollet med FAT-protokollet föreslår studien att vissa punkter i FAT-protokollet markeras som möjliga att genomföra virtuellt. Detta kan införas när en färdig standard utarbetats för vilka krav som ställs på en modell för virtuell idrifttagning.

10.4 Utfört VFAT med kund

Genom de tester som utförts tillsammans med kunden har modellen validerats. Det som är svårt med att validera en emuleringsmodell är att det ofta inte finns mätbara parametrar att undersöka. Eftersom det inte gick att undersöka mätbara parametrar såsom cykeltider är det svårt att undersöka hur väl

modellen överensstämmer med verkligheten. Trots detta ser kunden en stor nytta i att kunna undersöka modellen innan den fysiska cellen monteras. Testerna som utförs kan bli mer specifika samt att buggar i systemet eller eventuella funktioner som inte överensstämmer med vad kunden eftersöker kan upptäckas tidigare. En viktig punkt som kunden lyfter fram är att vid eventuell framtida användning av en emuleringsmodell i VFAT, anser de att en stor vikt ligger på hur tydliga förutsättningarna är innan ett test utförs. Genom att beskriva förenklingar och eventuella brister i modellen innan VFAT utförs, anser kunden att det är en stor fördel ifall kanske även om endast 70 – 80 % av testet går att utföra. Att använda en emuleringsmodell kan leda till att stora delar av programkoden och systemets funktioner testas redan innan FAT utförs på den fysiska produktionscellen. FAT på den fysiska cellen kan då koncentreras till att svara på de frågor som uppkommit under tester med emuleringsmodellen och att genomföra tester av de funktioner som inte gått att testa i modellen. Med en emuleringsmodell att testa programkoden i och möjlighet att utföra VFAT med, anses det leda till mindre felprogrammering och effektivisera tiden vid tillfället för FAT.

10.5 Vision

Studien visar att vision-programmet med relativt enkla medel kan användas och utvärderas i en virtuell miljö om det programmerats för den verkliga applikationen.

För att kunna programmera och utvärdera vision-program helt i en virtuell miljö behövs möjlighet att kontrollera flera variabler. Hur bilden ser ut påverkas av en mängd faktorer som beskrivs i kapitel 2.2. Det som identifierats i detta projekt är att kamerans konfiguration som sensorstorlek och objektivets prestanda behöver kunna ställas in. Övriga faktorer som skulle hjälpa mest att kunna ställa in i den virtuella miljön vore; ljuskällors position, styrka och kvalitet och kunna ställa in olika material och dess egenskaper i att reflektera ljus. När det kommer till ljuskällorna är detta viktigt då vision-programmet skulle kunna kontrolleras med olika former av ljuskällor för bästa resultat. Även störande ljus från exempelvis fönster och övrig lokal belysning skulle kunna emuleras för att kontrollera hur de påverkar vision-programmet. Att kunna kontrollera material i modellen är viktigt för att få med hur ljus reflekteras olika i olika material. Om ovan nämnda parametrar kan påverkas skulle ett vision-program kunna Offline-programmeras i den virtuella miljön i emuleringsmodellen utan tillgång till den riktiga utrustningen, för att sedan föras över till det verkliga systemet.

10.6 Framtida arbete

En intressant studie att göra vore att se hur mycket det faktiskt går att spara med att tillämpa virtuell idrifttagning likt Makris, Michalos, och Chryssolouris (2012) fast omfatta flera projekt. Detta skulle kunna genomföras genom att välja ut ett antal testprojekt och lika många referensprojekt. I testprojekten används virtuell idrifttagning och i referensprojekten används traditionella metoder. Där efter kan en grundlig utvärdering ske och effekten av virtuell idrifttagning skulle kunna mätas. Förutsättningen är att testprojekten och referensprojekten är av liknande komplexitet för att en jämförelse skall bli trovärdig.

En tydlig del som framkommit av studien är önskemålet att även kunna emulera säkerhetsfunktioner som idag är beroende av hårdvarusystem. Detta är något som framtida studier skulle kunna fokusera på för att kunna lägga över mer av kontrollerna vid FAT till en virtuell miljö.

Ett mål med digitala kopior av maskiner borde vara att de kan följa med i hela maskinens livscykel från konstruktion till avveckling. Modellen kan då användas som hjälpmedel för att undersöka effekten av förändringar i maskinen innan de implementeras.

11 Referenser

RÍOS, J., HERNÁNDEZ, J. C., OLIVA, M. & MAS, F., 2015. *Product Avatar as Digital Counterpart of a.* u.o., IOS Press, p. 657.

ABB, 2017. *ABB.* [Online]
Available at: http://new.abb.com/products/robotics/robotstudio
[Använd 15 05 2017].

Andersen, E. S. & Schwencke, E., 2010. *Projektarbete - en vägledning för studenter.* 1:15 red. Lund: Studentlitteratur AB.

Ayani, M., 2017. *Simumatik.com.* [Online]
Available at: http://simumatik.com/
[Använd 15 05 2017].

Banks, J., Carson, J. S., Nelson, B. L. & Nicol, D. M., 2005. *Discret-Event System Simulation.* United States: Pearson Education.

Binnberg, D. & Johansson, V., 2016. *VIRTUAL COMMISSIONING Emulation of a production cell,* Skövde: Univercity of Skövde.

Blomsjö, G., 2006. *Industriell robotteknik.* Lund: Studentlitteratur.

Bolton, W., 2015. *Programmable Logic Controllers.* 6th red. Nederländerna: Newnes.

Cognex corporation, 2017. *In-Sight Vision Systems.* [Online]
Available at: http://www.cognex.com/productstemplate.aspx?id=13937&rdr=true&LangType=2057
[Använd 15 05 2017].

Davies, E. R., 2012. *Computer and machine Vision.* 4 red. Waltham, Mass: Elsevier.

Debroy, T., Zhang, W., Turner, J. & Babu, S. S., 2016. Building digital twins of 3D printing machines ☆.

Groover, M. P., 2014. *Automation, Production Systems and Computer-Integrated Manufacturing.* London: Pearson education limited.

Guerrero, L. V., López, V. V. & Mejía, J. E., 2014. Virtual Commissioning with Process Simulation (Tecnomatix). *Computer-Aided Design and Applications.*

Gulliksson, H. & Holmgren, U., 2015. *Hållbar utveckling.* 2 red. Lund: Studentliteratur.

Hanssen, D. H., 2015. *Programmable Logic Controllers - A practical approach to IEC 61131-3 Using Codesys.* United kingdom: John Wiley & Sons.

Hoffmann, P., Maksoud, T., Schumann, R. & Premier, G., 2010. VIRTUAL COMMISSIONING OF MANUFACTURING SYSTEMS A REVIEW AND NEW APPROACHES FOR SIMPLIFICATION. *24th European Conference on Modelling and Simulation,* pp. 175 - 181.

IFR, 2012. *History of Industrial Robots..* [Online]
Available at:

http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf [Använd 06 03 2017].

ISO, 2011. *ISO 10218-1:2011.* u.o.:u.n.

Johansson, M. & Nilsson, J., 2015. *VIRTUELL DRIFTSÄTTNING Verifiering av PLC logik mot simuleringsprogram,* Skövde: Univercity of Skövde.

Korotchenkov, G. S., 2011. *Chemical Sensors : Comprehensive Sensor Technologies Volume 5: Electrochemical and Optical Sensors..* New York: Momentum Press.

Makris, S., Michalos, G. & Chryssolouris, G., 2012. Virtual commissioning of an assembly cell with cooperating robots. *Advances in Decision Sciences.*

McGregor, I., 2002. The relationship between simulation and emulation. *Proceedings of the Winter Simulation Conference,* Volym 2, pp. 1683-1688.

Megyesi, B., 2015. *DEBOK.* [Online]
Available at: http://www.debok.net/vetenskap/2015/01/Vad-ar-en-optisk-sensor.html

OPC Foundation, 2017. *OPC Foundation.* [Online]
Available at: Https://OPCFoundation.org
[Använd 27 04 2017].

Rahimi, M. & Karwowski, W., 1992. *Human-Robot Interaction.* London: Taylor & Francis.

Ramsden, E., 2006. *Hall-Effect Sensors : Theory and Application.* Amsterdam: Newnes.

Rosen, R., Von Wichert, G., Lo, G. & Bettenhausen, K. D., 2015. *About the importance of autonomy and digital twins for the future of manufacturing.* u.o., u.n.

Shafto, M. o.a., 2010. Modeling, Simulation, Information. Technoligy & Processing Roadmap.

Shafto, M. o.a., 2012. Modeling, Simulation, Information. Technoligy & Processing Roadmap.

Siemens AG, 2017. *Totally Integrated Automation Portal.* [Online]
Available at: https://www.siemens.com/global/en/home/products/automation/industry-software/automation-software/tia-portal.html
[Använd 15 05 2017].

Bilaga A

Punktlista på samtliga förenklingar:

- Säkerhetskretsen byglades i PLC-Programmet då den inte går att emulera med mjukvara.
- Ready for operation för alla enheter byglades i PLC-Programmet. Det är en signal från varje enhet att den är redo att ta emot och utföra uppgifter som måste vara till och liksom säkerhetskretsen inte gick att simulera.
- Endast in- och utbana ett byggdes upp i emuleringsmodellen eftersom inbana två är en spegelbild av inbana ett
- Manuell överföring av bild från virtuell kamera i Simumatik3D till Cognex In-Sight kameraemulator.
- Hörnmätning toleranser borttagna vilket ger att systemet aldrig ger fel för att mellandel eller lock ligger för fel i robotverktyget vid inmätning av hörn med vision-kameran.
- 3D-Modeller har förenklats för att minska belastningen för datorn. Antalet ytor på 3D-modellerna minskades från en halv ner till en åttondel i vissa fall
- DMC-Läsare alltid ok och förbikopplad. I cellen ingår att läsa av en data matrix kod som sitter klistrad på varje produkt. Hela detta moment kopplades bort. För att roboten skulle gå vidare efter denna åtgärd kopplades en knapp i Simunatik3D till en variabel som sickade programmet vidare till att lämna produkten till utbanan
- Utbanan endast enkel rullbana med borttagning av produkter. Endast själva rullbandet används på utbanan och all övrig funktionalitet användes inte.
- Emulerade Vision-systemet kördes med kalibrering från den riktiga Vision-kameran. Efter som alla bilder överförs manuellt beslutades att det inte var värt att kalibrera visionsystemed få det kräver en sekvens på ca: 800 bilder.
- Omgjord funktion för sökning av höjd. Robotverktyget har två stycken gripare, en för att greppa produkter och en med vakuumkuddar för att lyfta lock och mellanlägg. När ett verktyg skall användas skjuts det fram med hjälp av tryckluftscylindrar. I dessa cylindrar sitter givare som känner av när de är i sina ändlägen. När roboten söker höjden till arbetsobjektet går den långsamt emot arbetsobjektet det tills tryckluftscylindern trycks ihop så att ändlägesgivaren slås ifrån. Detta ger då att sökt höjd är nådd. Istället för att använda givarna i verktygets cylindrar, användes en separat givare på robotverktyget i Simumatik3D som ger signal när verktyget går emot en fysik i emuleringen.
- Inga rörelser i robotens verktyg.
- Inga rörelser i Pallstaplarens gripare utan endast rörelserna upp och ner emulerades.

Bilaga B

VFAT-Protokoll

Introduction

Start the test with checking the clock, setting all counters, alarm logs, etc. to zero.

I HMI

I-1. Hard keys and permanent indications, Machine panel

(Functional description item 6.2.2)

Check the presence and designation of each key and each indication lamp on the operating panel and sub-panels in accordance to the functional description.

Select power on with the pushbutton “MASKIN TILL”.

The lamp “MASKIN TILL” is on.

Drive the machine to initial position by pushbutton “UTGÅNGSLÄGE”

When the machine moves to initial position the lamp “UTGÅNGSLÄGE” is flashing, when initial position is reached the lamp is on with fixed light.

Select automatic mode.

The lamp “START CYKEL” is flashing.

Select manual mode.

The lamp “START CYKEL” is off.

Force an alarm

The lamp “ÅTERSTÄLL LARM” is flashing

Try to acknowledge the alarm by pressing pushbutton “ÅTERSTÄLL LARM”

The lamp “ÅTERSTÄLL LARM” is on with fixed light.

Force a new alarm.

Check that the lamp "ÅTERSTÄLL LARM" starts to flash again.

Restore the emergency stop and press “ÅTERSTÄLL LARM”.

The lamp “ÅTERSTÄLL LARM” goes off.

I-2. Operator panel - HMI screens

(Functional description items 6.1)

- Actual alarms and warnings/messages

Check that they are present in all screens.

- Change between the different operating modes.

Check that it is possible to change operating mode independent of selected HMI-screen.

Check that the field in the header is updated.

- Select different screens.

Check that the time for changing screens doesn't exceed 2 seconds.

I-2.1. Operator guidance

(Functional description item 6.1.1)

Enter a wrong value in any field, or activate a not allowed manual function etc.

Check that there is a guidance that tells the reason for wrong entry or wrong selection.

I-2.2. Lamp test

(Functional description item 6.1.4)

Select the lamp test function.

All lamps are on including the traffic lights. The lamp test is off-delayed for about 20 seconds.

Check with the programmer that the flashing frequency of all lamps is set to 1 Hz.

I-2.3. Menu structure

(Functional description item 6.2.1)

Check that the menu structure is correct and match to what is agreed.

I-2.4. Overview screen

(Functional description item 6.2.1.1)

Check that the graphic illustration corresponds to the reality and that the orientation is correct.

Check following, see Functional description for specification.

- Machine/unit status
- Door status
- Initial position
- Emergency stop
- Workpiece status

- Previous cycle time
- Actual cycle time
- Number of workpieces
- Sequence steps (when necessary)

Check that there is a symbol legend present.

I-2.5. Manual movements
(Functional description item 6.2.1.2)

Check that all movements and functions are present in the screens for manual movements.

Compare with the electrical drawing that all movements and functions are present.

Ensure that it is possible to switch from comment to symbolic in all manual screens.

I-2.6. Sub systems
(Functional description item 6.2.1.6)

Parameters and data for the subsystems in the machine shall be possible to view, setup, diagnose in one or more screens. E.g DMC-camera, Vision Guiding-Camera

I-2.7. Cycle time
(Functional description item 6.2.1.8)

Check that the cycle time is presented with one decimal and that the reference time is possible to set in HMI.

Following shall be present.

1. The total cycle time and the cycle time for each Flow (1 and 2) shall be viewed.
2. The ten latest cycle times shall be saved and viewed.

I-2.8. Sensor layout
(Functional description item 6.2.1.9)

Check that sensor layouts consists of a distinct graphic illustration of the equipment.

The functionality and content will be checked in test V-2 Pair monitoring sensors.

I-2.9. Interface layout
(Functional description item 6.2.1.10)

Check that the signals in the interface layout correspond to the reality and that they are updated correct when the machine is running.

Eg, Photocells interface, DMC-Camera, Vision Guiding-camera

I-2.10. Diagnostic screens

(Functional description item 6.2.1.11)

Check the presence of the diagnostic screens. The correctness will be checked during test V Diagnostics.

Larm,Meddelande,Export larm, Bussdiagnostik,

I-2.11. Service

(Functional description item 6.2.1.12)

Check that the screen for controlling and viewing PLC-variables works correct.

I-2.12. Traffic light

(Functional description item 6.4)

Check that the traffic light works correct during the entire test.

Machine lightstack

Colour	Flashing, 1 Hz	Fixed light	Remark
Red	Unacknowledged alarm	Acknowledged alarm	
Yellow	Andon	Warning	Andon has priority.
Blue	Output place full	Input place empty	Off by alarm
Green		Cycle on	Off by alarm

Yellow:

- The yellow lamp shall go off with automatic as fast as the reason is taken care off.
- When a warning changes to an alarm only the red lamp shall light. The warning text and yellow lamp shall go off. Yellow and red lamp must not

light simultaneously for the same problem.

- A message which not requires any action from the operator, doesn't light up the yellow lamp.
- High temperature in electrical cabinets shall emit a warning and after 2 hours turn over to an alarm.
- Excluded part function doesn't light up the yellow lamp

Blue:

- Blue lamp is only turned on when the cycle is on.
- Blue lamp is not turned on when emptying of the machine is selected.
- Blue lamp is not turned on during the load/unload sequence, since this is a part of the machine time.
- Input place empty: The machine is ready to begin a new work cycle, but the input place is empty. Turned on first when the ordinary cycle time is exceeded.
- Output place full: The machine is ready to end its work cycle, but the output place is occupied. Turned on first when the ordinary cycle time is exceeded..
- Input place empty/ Output place full is just measured in the main flow. Magazines etc. with ingoing articles belongs to warning.

General functions

I-3. Manual mode

(Functional description item 3.1.1)

I-3.1. Manual with interlock

Select manual mode.

Run each movement and sub cycles to its end position.

Check information in HMI regarding 'movement possible to execute' and 'movement active'.

Try to execute an interlocked movement.

Check information about missing conditions in the HMI.

Repeat the test with interlocked movements a couple of times (different movements).

I-3.2. Manual without interlock

Activate the key-switch “MAN UTAN FÖRREGLING”.

> *Check that the key-switch is a type with spring return.*
>
> *Check also that all manual movements are possible to execute.*

Run the normally interlocked movement. **Hazardous mode, make sure that the machine not will be damaged!**

Release the key-switch.

Check that there is an alarm each time you activate the key-switch.

I-4. Automatic cycle (Functional description item 3.2.1.2)

Select "Automatic cycle"

Press “START CYKEL”.

> *The machine repeats complete cycles, the lamp "START CYKEL" is on with fixed light. The lamp is flashing when the cycle can be started.*
>
> Select manual mode during the automatic cycle.
>
> *Check that a change to manual mode is done and that the cycle is interrupted immediately.*

Select "Automatic cycle"

The lamp "*Start cykel*" *starts to flash.*

> Press “START CYKEL”.
>
> *Machine continuous auto cycle*

I-5. Cycle types

(Functional description item 3.2)

Activate any of the cycle types that are present in the machine.

Press ‘Maskin från’ and then ‘Maskin till’.

Check that all cycle types are turned off and that just normal auto is active.

I-5.1. Manually emptying
(Functional description item 3.2.1.1)

Select automatic mode and start the cycle.

Activate the functions 'emptying of machine total', “emptying of machine flow 1” and “emptying of machine flow 2”.

Check that started pallet is completed

> *Check that emptying is possible to activate/deactivate when the cycle is started.*
>
> *The machine completes the workpieces that are present and runs empty. Workpieces on the input place are not released into the machine.*
>
> *The machine and stops after completed cycle.*

I-6. Initial position
(Functional description items 3.3)

I-6.1. Initial position run in manual mode.

Run manually some movements from their initial position.

Press “UTGÅNGSLÄGE”.

> *All movements moves to initial position in manual mode. The movements will stop immediately if the button is released.*
>
> *The lamp "UTGÅNGSLÄGE" is flashing during movement and is on with fixed light when initial position is reached.*

I-6.2. Initial position run in automatic mode.

Select automatic mode and start the cycle.

Press “HÅLL CYKEL” during the first movement.

> *The machine stops immediately.*

Press “UTGÅNGSLÄGE”.

> *The machine moves to initial position in automatic mode. The lamp "UTGÅNGSLÄGE" is flashing during movement and is on with fixed light when initial position is reached.*

Start the cycle again. Press “HÅLL CYKEL” during the second movement.

Press “UTGÅNGSLÄGE”.

Continue as described above until initial position run is ordered from all positions.

Run manually some movements from their initial position.

Select automatic mode and press "UTGÅNGSLÄGE".

The machine moves to initial position in automatic mode. The lamp "UTGÅNGSLÄGE" is flashing during movement and is on with fixed light when initial position is reached.

I-7. Operational stop

(Functional description item 3.4)

I-7.1. Stop after completed cycle

(Functional description item 3.4.1)

Select automatic mode and start the cycle.

Press “STOPP CYKEL”.

The lamp “STOPP CYKEL” starts to flash while the automatic cycle is running. When the cycle is finished the lamp lights with fixed light. The lamp "START CYKEL" remains on as long as the cycle is in progress.

Simulate a fault so that “STOPP CYKEL” not can be reached.

Check that an alarm occurs.

I-7.2. Immediately stop

(Functional description item 3.4.2)

Select automatic mode and start the cycle.

Press “HÅLL CYKEL”.

The machine stops immediately. Nutrunners and marking equipment is finishing their ongoing mission.

The lamp "START CYKEL" goes off and starts to flash as an indication that the cycle is possible to start.

Press “START CYKEL”.

The machine continues the interrupted cycle from the position where it was stopped.

I-8. Restart functions

(Functional description items 3.5)

I-8.1. Restart after power failure

Select automatic mode and start the cycle.

Disconnect the main switch during the automatic cycle.

Restore the main switch.

Reset possible alarm and press “MASKIN TILL”. Run to initial position, press “UTGÅNGSLÄGE”.

Check that information is correct regarding workpiece present/status.

Press “START CYKEL”.

Continue as described above until power failure is simulated in different positions

I-9. Start-up procedure of the machine

(Functional description items 3.7)

The machine is started up according to the following procedure.

1. MASKIN TILL
2. UTGÅNGSLÄGE
3. START CYKEL

Eventually it might be necessary to reset the safety/ emergency stop.

Check that all media such as air, hydraulic, safety test is started/performed automatically.

Check that the Vision Guiding is delayed until it´s illumination is up to full effect

I-10. Energy saving

(Functional description item 3.8)

Select automatic mode and start the cycle.

Simulate input place empty or output place full.

Check that the machine automatically is set in energy saving mode. Motors, pumps, fans, coolant water etc. shall be turned off when the time of inactivity has been longer than the pre-set time.

The time shall be individually adjustable in the HMI for each group that is shut off.

Feed in workpieces to the input place or take away workpieces from the output place.

Restart shall take place automatically.

Check that the Vision Guiding is delayed until it´s illumination is up to full effect

I-11. Service positions

(Functional description item 4.1.4)

Run the machine to initial position.

Select and run the machine to all service positions.

The function is selected and started as a manual function.

The machine moves to the defined service position.

Note! Check that no unnecessary conditions are required to run to service position e.g. complete machine empty.

Check that one of the positions is the reference position for the equipment.

For a robot there shall also be Super initial position, sync. Position and SafeMove sync. Position.

Robot to Initial position

Synk position axis 1-6,

SafeMove: Brake test

SafeMove: Soft Ware Sync

Oil level control axis 1-4

Oil level control axis 5

Oil level control axis 6

Service position control gripper

II Diagnostics

(Functional description item 6)

Check that the presentation of the alarms is correct: +AS1-A2 | Robotcell: Larm FB [FB17] | Robotcell: Säkring utlöst: F802. Matning märkare utbana 2 +UTB1-A210 | Network 14

Check that there is a graphical sign for the function group that has an alarm.

Check that follow-up alarm caused by an in time preceding malfunction is prevented if possible.

Check that alarm and alarm texts are reset with push-button "Återställ larm".

Check that alarm occur regardless of selected operating mode.

When testing the different alarms, check that correct stop category is used.

- Immediate stop
- Time out for a movement
- Stop after complete cycle
- Warning – don't stop the machine immediately
- Messages to the operator – don't stop the machine

II-1. Fuses and Overload relays etc. (Tripp all and check error)

Trip fuses and overload relays, one at a time.

Note! Don't trip the fuse for PLC/operator panel.

Check that an alarm occur with correct text.

Please note that follow-up alarm if possible not occur!

II-2. Pair monitoring sensors

Actuate all sensor pair in both directions, one pair at the time.

Check that an alarm occur with correct text.

At the same time check all sensors in the sensor layout screens.

Verify that all sensors (digital inputs) are present and that correct status is displayed. Compare with the electrical drawing, for missing sensors.

> *Check that all sensors are described with text comment and reference designation.*

II-3. Movement not acknowledged

Try to simulate timeout alarms for all movements.

Disconnect the sensor for the movement's forward position.

Select automatic mode and start the cycle.

> *When the movement has reached the forward position an alarm appears after a certain*

delay. The alarm text shall contain sensor/valve designation.

Disconnect the valve connector (tube or cable), the movement don't start.

When the movement has reached the forward position an alarm appears after a certain delay. The alarm text shall contain sensor/valve designation.

II-4. System alarm

II-4.1. Sub-system

Create a system alarm in each sub-system one at a time e.g. robot, marking unit, nutrunner, camera etc.

Check that an alarm and alarm text occur in HMI for each sub-system.

II-4.2. ProDiag

Simulate different faults during automatic cycle and also in manual mode.

Check the functionality in the ProDiag screens.

II-5. Warnings

Activate a function that creates a warning e.g. low level central lubrication.

Check that the yellow lamp in the traffic light is on with fixed light and that warning text occur. When the reason for the warning is taken care of, the yellow lamp goes off and the text disappears automatically. The machine doesn't stop.

Check that no warnings are saved in the history log for alarms.

The level will be low if no measures are made and the warning will turn over to an alarm. The yellow lamp goes off and the red lamp starts to flash in the traffic light. The warning text has switched to an alarm text.

Some examples are various levels, temperatures, pressures etc.

II-6. Messages

Activate a function that creates a message e.g. robot in manual mode.

There is just a message text. The machine and traffic light are not affected. Check that the message disappear when the reason is gone.

Check that no messages are saved in the history log for alarms.

Some examples are various operator guidance, cycle time to high.

II-7. Alarm/warning/message

(Functional description item 5.1.1)

Actual alarms, warnings and messages are presented in one and the same screen or in separate screens depending on HMI system.

Check that actual alarms, warnings and messages are time and date stamped and presented with the oldest on top.

II-8. Alarm history log (Simulation can only go back 1 hour)

(Functional description item 5.1.2)

Check that the history log for alarms is presented with the newest alarm on top.

Check that only alarms are saved in the history log.

*Check that the log stores **3000** posts.*

Check that there is a function in HMI to export the alarm log to an external media e.g. USB-memory.

II-9. Top ten alarm log (Simulation can only go back 1 hour)

(Functional description item 5.1.3)

Check that there is a log that evaluates the history log and presents the ten most frequent alarms regarding time and number.