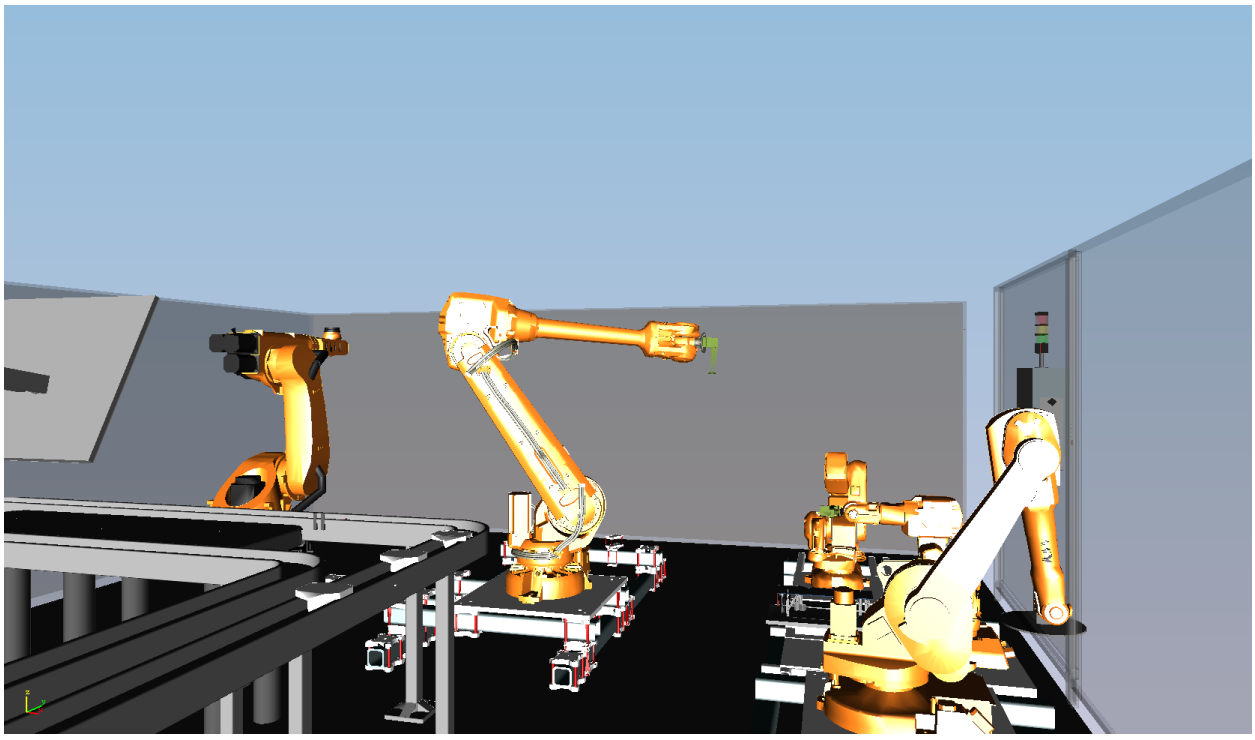
CHALMERS
UNIVERSITY OF TECHNOLOGY



Virtual commissioning med Oculus Rift

Kandidatarbete - SSYX02-16-06

Anton Albo
Johan Eriksson
Hampus Lindvall
Julius Pettersson
Erik Sedelius
John Weman

Institutionen för signaler och system
Chalmers tekniska högskola
Maj 2016, Göteborg, Sverige

Examinator: Knut Åkesson, Docent, institutionen för signaler och system
Handledare: Petter Falkman, Docent, institutionen för signaler och system

Slutrapport
Instutitionen för signaler och system
Chalmers Tekninska Högskola
SE-412 96 Göteborg
Sverige

Framsida: Modell av PSL-laboratoriets produktionslina simulerad i Process Simulate. Visualiseras ur det mänskliga objektet Jacks perspektiv.

Maj 2016, Göteborg, Sverige

Sammanfattning

Virtual commissioning är en metod för att i en virtuell miljö validera PLC-kod, HMI-interaktioner och operatörens säkerhet. I och med att valideringen sker virtuellt stoppas inte den verkliga produktionen och säkerhetsåtgärder kan testas redan innan systemet används fysiskt. Det gör implementationer av förändringar i ett produktionssystem säkrare och sparar både tid och pengar.

Syftet med projektet är att utvärdera vad som kan testas och valideras med en virtual commissioning med Oculus Rift. Oculus Rift är ett verktyg för att återge den virtuella miljön i 3D och har en mängd sensorer som skall användas för att ändra det operatören ser beroende på var denne tittar. Då operatören exempelvis tittar till höger skall det som finns till höger visas upp, det kallas head-tracking.

I projektet är en modell av produktionscellen i PSL (produktionssystemslaboratoriet på Chalmers Tekniska Högskola) konstruerad så att den kan kommunicera med en PLC via en OPC-tunneler. Modellen kommunicerar också med Oculus Rift, vilket möjliggör visualisering av produktionscellen i en 3D-miljö där head-tracking också används.

De slutsatser som kan dras med hjälp av det slutgiltiga konceptet är att; PLC-kod kan valideras, HMI:er kan testas, viss säkerhetsutrustning kan undersökas och ny utrustning kan utvärderas vid en virtual commissioning med Oculus Rift.

Författarnas tack

Ett speciellt tack går till en rad personer, vilka utan deras hjälp projektet ej hade gått att genomföra. Vi vill visa vår tacksamhet till:

Petter Falkman, för handledning och inspiration att föra projektet framåt.
Martin Dahl, för stor tillgänglighet och väldigt god support genom hela projektet.
Adnan Khan, för att vara till förfogande under projektet.
Kristoffer Carlsson, för hjälp med pythonprogrammering.

Anton, Johan, Hampus, Julius, Erik & John
Maj 2016

Ordlista

CAD Computer Aided Design - Datorsystem för att underlätta skapande, modifiering och analys av en design.

Head-tracking Funktion för att spåra användarens huvudrörelser genom att använda Oculus Rifts sensorer.

HMD Head Mounted Display - En skärm som fästes på huvudet.

HMI Human Machine Interface - Visuellt gränssnitt mellan maskin och operatör.

OLP Off-line programming - Kommando i Process Simulate som innehåller interna ordrar för robotar.

OPC OLE for Process Control (OLE = Object Linking and Embedding) - Standardiserat språk och kommunikationsserver mellan olika signalkällor.

PLC Programmable Logic Controller - Robust styrenhet för industriellt bruk.

PSL Production System Laboratory - Produktionssystemslaboratoriet på Chalmers Tekniska Högskola.

TCL Tool Command Language - Ett scriptspråk som i detta fall används för att skapa moduler till Jack 8.2.

TIA Totally Integrated Automation - Samlingsnamn på Siemens egna mjukvaror till PLC.

VR Virtual Reality - Datorteknik som återskapar en verklig miljö i en virtuell värld som en användare kan interagera med.

WinCC Siemens interface software - Mjukvara av Siemens för att programmera visuellt gränssnitt.

Figurer

1.1 Kommunikation mellan projektets delområden. Dubbelpil innebär informationsflöde åt båda håll och de mörka rutorna är mjukvara medan den ljusa är hårdvara. 3

2.1 Simuleringsmodell avbildad från Chalmers PSL i Process Simulate. Siffrorna är kopplade till viktiga komponenter i cellen. 8
2.2 De tre klosstornsvarianter som kan konstrueras i den virtuella produktionscellen. 8
2.3 Virtuell modell respektive verklig produktionscell ur två olika vinklar. 10
2.4 Konvertering av filer för att de skall kunna hanteras av Process Simulate. 10
2.5 **a)** Magnet-gripper till IRB4600. **b)** Rörligt gripdon till IRB140. . . . 11

3.1 Uppkopplingen mellan PLC, dator med TIA-portal och OPC-servers anslutning mot klient-dator med Process Simulate. Ljusa rutor innebär hårdvara och mörkare rutor indikerar mjukvara. 18
3.2 Visuellt gränssnitt i programmet WinCC för styrning av produktionscellen i Process Simulate. 18
3.3 Projektöversikt av PLC-program i Siemens TIA-portal. I högra fönstret listas programmets tillhörande funktionsblock och i vänstra fönstret visas hårdvarans komponenters index. 20
3.4 Instans av funktionsblock till en robot. 20
3.5 Struktur över inre sekvensen i funktionsblocket för roboten. 21
3.6 Exekveringstabell över PLC-programmets kod. 21
3.7 Användande av Telnet i Windows kommandotolk. Här anges serverdators IP-adress samt önskat portnummer. 22
3.8 Lyckad anslutning mellan portar via Telnet-tjänsten. I detta läge är angiven port öppen och redo för signalutbyte. 23
3.9 Siemens OPC-Scout visar signalutbytet i realtid mellan server och klient. 23

4.1 En användare som bär VR-glasögonen Oculus Rift. 25
4.2 Orientering av yaw, pitch och roll i förhållande till användarens riktning vid användning av Oculus Rift. 28

4.3 Filstrukturen hos den framtagna Jack-modulen och pythonprogrammet. Jack-modulen är den del som kommunicerar med Jack medan pythonprogrammet är den del som sköter Oculus Rift och knapptryckningar. Observera alltså att Jack och Oculus Rift inte är några delar av modulen eller programmet utan bara kommunicerar med dem. De mörka rutorna är mjukvara medan den ljusa är hårdvara. 29
4.4 De vyer som går att utvinna från det mänskliga objektet i Process Simulate. Bilden till vänster är det mänskliga objektets vänstra ögas vy som skall visas upp i det vänstra ögat i Oculus Rift. Den högra bilden är vyn från det mänskliga objektets högra öga och den visas upp för höger öga i Oculus Rift. 30

5.1 Kommunikationsschema över hela systemet. Oculus är den modul som beskrivs i avsnitt 4.3. De ljusa rutorna är hårdvara och de mörka rutorna är mjukvara. 33

B.1 Signaler och operationer IV

C.1 Menyruta för Importer.exe V
C.2 Menystruktur för att uppgradera .co till .cojt VI
C.3 Ikon då resource är i läge för modeling VI
C.4 Menystruktur för att skapa Links VII
C.5 Menystruktur för att skapa Joints VII
C.6 Menystruktur för att skapa olika poser VIII
C.7 Menystruktur för att definiera resource som Tool IX

Innehåll

1 Introduktion **1**
1.1 Tidigare arbeten inom virtual commissioning . . . 1
1.2 Syfte . . . 2
1.3 Problem . . . 2
1.4 Projektuppbyggnad . . . 2

2 Virtuell produktionscell **5**
2.1 Bakgrund . . . 5
2.1.1 Val av programvara . . . 5
2.1.2 Beskrivning av Process Simulate . . . 6
2.2 Problem . . . 6
2.2.1 Uppförande av virtuell modell . . . 7
2.2.2 Verktyg för kommunikation med en styrenhet . . . 7
2.2.3 Utvärdering av processäkerhet . . . 7
2.2.4 Utvärdering av personsäkerhet . . . 7
2.2.5 Likhet med verkligheten . . . 7
2.3 Resultat . . . 8
2.3.1 Modelluppbyggnad . . . 9
2.3.1.1 Definiering av gripdon . . . 11
2.3.1.2 Definiering av transportband . . . 11
2.3.2 Operationsuppbyggnad . . . 12
2.3.3 Verktyg för simulering av säkerhet . . . 12
2.3.3.1 Simulering av fel . . . 13
2.3.4 Signaler som kommunikationsverktyg . . . 14

3 Styrning av virtuell produktionscell med en PLC **15**
3.1 Bakgrund . . . 15
3.1.1 Kommunikation och informationshantering mellan enheter . . . 16
3.2 Problem . . . 16
3.2.1 Kommunikation mellan enheter på skilda destinationer . . . 16
3.2.2 Konvertering av kod mellan fysisk och virtuell cell . . . 17
3.2.3 Simulering av säkerhet och säkerhetsutrustning . . . 17
3.3 Resultat . . . 17
3.3.1 Utrustning och komponenter . . . 17
3.3.2 Styrsystemets gränssnitt och funktionalitet . . . 18
3.3.3 Struktur och uppbyggnad av logisk kod . . . 19

3.3.4 Kommunikation via OPC-server 22

4 Visualisering av virtuell produktionscell med Oculus Rift 25
4.1 Bakgrund 25
4.1.1 Aktuella användningsområden för virtual reality 26
4.1.2 Virtual reality som stöd vid forskning och utbildning 26
4.1.3 Oculus Rifts påverkan på människan 26
4.1.4 Programvaran Jack och styrning av mänskliga objekt 27
4.2 Problem 27
4.3 Resultat 27
4.3.1 Importering och användning av JackScript 27
4.3.2 Konstruktion av moduler till Jack i Tcl 27
4.3.3 Spårande av huvudrörelser med Oculus Rift 28
4.3.4 Filstruktur av den färdiga modulen 29
4.3.5 Kommunikation mellan script genom ”socket” 30
4.3.6 Visualisering och rendering till Oculus Rift 30

5 Fullständigt koncept 33

6 Diskussion 35
6.1 Utvärdering av produktionsutrustning 35
6.2 Utvärdering av säkerhet 35
6.3 Utbildningsmöjligheter 36
6.4 Hållbarhetsaspekter av virtual commissioning 37
6.5 Validering av PLC-kod 38
6.6 Alternativa verktyg 38
6.6.1 Alternativa simuleringsprogram 38
6.6.2 Alternativa kommunikationsverktyg mellan PLC och Process Simulate 39
6.6.3 Alternativa HMD:er för visualisering 39
6.7 Vidareutveckling av det framtagna konceptet 39
6.7.1 Interaktion mellan mänskliga objekt och modellens processer . 39
6.7.2 Simulering av fysisk utrustning och automatisk kodgenerering till PLC 40
6.7.3 Förbättring av den visuella presentationen 40
6.7.4 Förstärkning av den virtuella upplevelsen 41

7 Slutsatser 43

Referenser 45

A Produktlista I

B Signaler och operationer III

C Process Simulate V
C.1 Konvertera filer V
C.2 Skapa kinematik för gripper VI

1
Introduktion

Inom tillverkningsindustrin sker idag en upptrappning av både mångfald av produkter samt mängden produkter som en fabrik skall kunna producera, enligt Erik Lindskog (föreläsning, 2015-november-03). Det har medfört ett generellt skifte hos producenterna, från att fokusera på kostnadsreducering till tidseffektivisering vid omställning av produktion [1]. Vidare säger [1] att många företag löser detta genom virtual commissioning där produktion, processtyrning, HMI-interaktion samt operatörssäkerhet simuleras virtuellt och på det sättet sparar tid inom framför allt inom utvecklings-, implementerings- och valideringsfaserna.

Virtual commissioning ger alltså, enligt [2], ett verktyg för att exempelvis validera att PLC-koden ger ett önskat beteende och på det sättet sparas resurser, i form av både arbetstimmar och material. Tillverkare som utfört virtual commissioning har sett en reducering i arbetstimmar med upp till 30 % [3]. I dagsläget krävs det dock mycket tid, pengar och en omfattande projektstruktur för att genomföra virtual commissioning, vilket kan leda till att det enbart blir ett dyrt experiment om det inte går som planerat [3]. På Chalmers Tekniska Högskola bedrivs i skrivande stund ett forskningsprojekt som heter ”Virtual commissioning of manufacturing stations including PLC logics, VirtCom”. Forskningsprojektet har till syfte att undersöka vad för mjukvara som finns för att genomföra virtual commisioning samt avgöra vad som är möjligt att validera och testa [4]. Inblandade parter i detta projekt är enligt [4] bland annat Volvo Cars, AB Volvo, Scania, GKN aerospace, NEVS, KUKA och ABB. I forskningsprojektet ingår detta kandidatarbete och dessa resultat kommer att bidra till forskningen.

1.1 Tidigare arbeten inom virtual commissioning

1999 introducerades en metod för att styra simuleringsmodeller med kontrollsystem [5]. Denna tidiga version av virtual commissioning kallas av [5] för ”soft-commissioning” och beskrivs som en bra metod för att öka produktionssystems flexibilitet.

Under 2000-talet har flertalet virtual commissioning-projekt genomförts på mekatroniska produktionssystem [6, 7, 8]. Bland annat har virtual commissioning genomförts av en monteringscell innehållande robotar där en virtuell modell styrs av en PLC [9]. Resultatet beskrivs av [9] som lyckat där den virtuella modellen gav en noggrann representation av den verkliga monteringscellen i termer av layout, resurser, utrustning och funktionalitet. Cykeltiden i den virtuella modellen blev på sekundnivå lika lång som den verkliga cellens cykeltid. Virtual commissioning har

även applicerats inom sjukvården där den har använts för att planera doser av laserbehandling vid angripande av ögoncancer [10].

Ett kandidatarbete inom virtual commissioning har också utförts där VR-glasögonen Oculus Rift har utnyttjats som verktyg för att förstärka den visuella upplevelsen [11]. Detta för att ge användaren ett bättre helhetsintryck och en ökad interaktion med simuleringen. I [11] åstadkom gruppen en 3D-representation av en produktionscell med hjälp av Oculus Rift, dock lyckades gruppen inte implementera head-tracking. De lyckades inte heller implementera en hållbar kommunikation mellan PLC och Process Simulate.

1.2 Syfte

Syftet med projektet är att med hjälp av virtual commissioning med Oculus Rift säkerställa funktionen hos PLC-logik, HMI:er och säkerhetsmoduler innan dessa används i en verklig miljö. Vidare skall det undersökas vad som är möjligt att testa och validera med virtual commissioning med Oculus Rift.

1.3 Problem

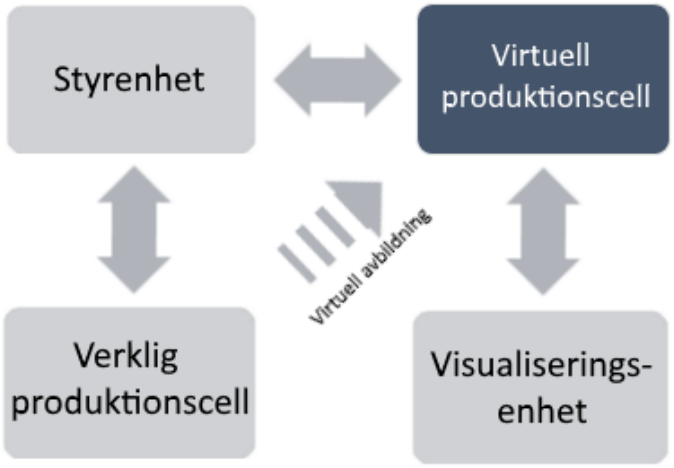
Projektets primära uppgift är att genomföra en virtual commissioning med Oculus Rift av en produktionscell som finns i PSL (Production Systems Laboratory) på Chalmers Tekniska Högskola. Virtual commissioning innefattar en sammankoppling mellan den verkliga och virtuella världen, där följande tre delområden är grundläggande att arbeta med för att kunna genomföra projektet:

- Uppförande av virtuell produktionscell.
- Styrning av den virtuella produktionscellen.
- Visualisering av den virtuella produktionscellen.

Projektet omfattar inte framtagning av kod för styrning av den verkliga cellen utan fokus kommer ligga på att styra den virtuella modellen. Vidare kommer inte heller cellen optimeras utan bara avbildas.

1.4 Projektuppbyggnad

De tre delområdena i avsnitt 1.3 fyller var för sig en funktion för att genomföra virtual commissioning men slutresultatet bygger i grunden på kommunikationen dessa emellan. Följaktligen är arbetet med sammankoppling viktigt, med en struktur på kommunikationsupplägg enligt Figur 1.1.



Figur 1.1: Kommunikation mellan projektets delområden. Dubbelpil innebär informationsflöde åt båda håll och de mörka rutorna är mjukvara medan den ljusa är hårdvara.

Målet är att en operatör skall kunna använda ett verktyg för visualisering av den virtuella modellen. Modellen skall kunna simulera de operationer som används i den verkliga produktionscellen och operatören skall kunna studera dessa som om denne befann sig i cellen fysiskt. Genom någon typ av HMI skall det vidare vara möjligt att interagera med den virtuella modellen och styra över cellens arbete.

Den hårdvara och mjukvara som används i projektet återfinns i Bilaga A.

2
Virtuell produktionscell

Detta kapitel beskriver kort det simuleringsprogram som används vid genomförandet av detta projekt. Vidare behandlas de problem som skall lösas inom modelleringsområdet samt de resultat som uppnåtts inom detta område.

2.1 Bakgrund

En av de mest använda modelleringstyperna är eventbaserad simulering och har utvecklats kontinuerligt sedan sent 1950-tal [12]. I [12] beskrivs hur utvecklingen hela tiden tagit stora steg men att det i början var dyrt och fanns krav på hög kompetens inom datorvetenskap för att använda programmen. Under 1980-talet blev simuleringar allt vanligare inom organisationer då programmen blev mer användarvänliga. Det var först i början av 1990-talet som simuleringsprogram började användas av allmänheten, detta på grund av mer kraftfulla datorer, sjunkande pris på PC:n, användandet av Windows och inte minst uppkomsten av internet. I och med dessa faktorer har utvecklingen av simuleringar tagit stora steg som lett till de program som idag finns tillgängliga på marknaden, i vilka man kan optimera processer och återge resultatet som en virtuell värld.

2.1.1 Val av programvara

Technomatix-paketet från Siemens består bland annat av Process Simulate och Process Designer, där Process Simulate är ett eventbaserat robotsimuleringsprogram som används för att hjälpa företag att optimera sin produktion samtidigt som kostnaderna hålls nere. I programmet kan en 3D-modell av ett automatiserat produktionssystem byggas upp så att felsökning och validering kan ske virtuellt.

Det finns flera anledningar till att dessa delar ur Technomatix-paketet används i det aktuella kandidatprojektet. Flertalet andra programvaror klarar att simulera produktionsprocesser men dessa är inte skapade för simulerings- och visualiseringsändamål såsom Process Simulate är. En välkänd programvara för robotsimulering är RobotStudio som utvecklats av ABB. Detta program fungerar utmärkt för offlineprogrammering, men är inte skapat för visualisering vilket gör det olämpligt att använda då en virtual commissioning skall genomföras [13]. Det finns även program som fokuserar på simulering för att förbättra processeffektivitet, ett exempel på ett sådant program är AutoMod [14].

Det finns olika program som går att använda då en virtual commissioning skall genomföras. Förutom Process Simulate kan Delmia, utvecklat av Dassault Systems,

användas för skapande av simuleringsmodell och styrning via PLC [9]. Det mest förekommande alternativet är dock kollaboration mellan WinMOD och INVISION, där WinMOD möjliggör kommunikation mellan styrenhet och virtuell modell medan INVISION står för visualisering av simuleringsobjektet [6]. Till skillnad från dessa program kan Process Simulate interagera med Jack som är ett program för mänsklig simulering [15]. Det är via Jack som kopplingen mellan Oculus Rift och Process Simulate kan möjliggöras, vilket gör att Jack-funktionen är fundamental för genomförandet av en virtual commissioning med Oculus Rift.

Sammanfattningsvis kan Process Simulate hantera de tre grundläggande funktioner som krävs för att genomföra detta projekt; skapa simuleringsmodell, styra modellen med PLC och koppla ihop modellen med Oculus Rift.

2.1.2 Beskrivning av Process Simulate

För att bygga upp en fungerande simuleringsmodell i Process Simulate krävs även stödjande arbete i Process Designer. Process Designer är det verktyg i vilket modellens komponenter importeras och placeras i olika bibliotek [16]. Exempel på komponenter är transportband, robotar och ingående material. I Process Designer importeras alltså alla CAD-filer som skall användas i produktionssimuleringen. Verktyg, såsom robotar och fixturer, placeras i ett så kallat ”Engineering Library” och det material som skall bearbetas har sin plats i ett ”Resource Library”. Det är alltså i Process Designer som grundstrukturen för 3D-simuleringen skapas. När projektet sedan öppnas i Process Simulate kommer alla importerade delar finnas tillgängliga.

Process Simulate är det verktyg där själva simuleringen utförs. I detta program placeras resurserna ut i 3D-miljön för att sedan operationer skall kunna definieras för till exempel robotar och transportband [15]. En styrka i Process Simulate är möjligheten att skapa Smart Components, vilket sammanfattat innebär att logikbaserade funktioner kan appliceras på varje enskild komponent i modellen. Dessa kan användas för att erhålla rörelser, avläsning av materialflöde eller andra händelseförlopp i modellen genom initiering från in- och utsignaler [17]. Två stora fördelar med Smart Components är att denna logik kan sparas ned och användas återkommande i andra projekt, samt bredden av möjligheter till att skapa egna skräddarsydda beteenden till de olika komponenterna.

Vidare finns det möjligheter att både göra sekvensstyrda simuleringar internt i Process Simulate, men det går också att genomföra signalstyrda simuleringar. Dessa signaler kan hanteras både internt i Process Simulate och externt via en PLC.

2.2 Problem

Under denna rubrik återfinns problem kopplade till skapandet av den virtuella produktionscellen. De områden som berörs är uppförande av virtuell modell, verktyg för kommunikation, utvärdering av process- och personsäkerhet samt likhet med verkligheten.

2.2.1 Uppförande av virtuell modell

En grundläggande del för att kunna genomföra en virtual commissioning är att det finns en virtuell miljö där simuleringen kan ske. En användbar modell finns i dagsläget inte tillgänglig då PSL ständigt utvecklas och inget liknande arbete skett på den produktionscell som nu är i drift. Färdigmodellerade komponenter finns tillgängliga för de delar som skall återanvändas från föregående produktionscell, vilken simuleras i det tidigare kandidatarbetet [11]. På grund av att virtual commissioning nu skall genomföras på en annan produktionscell ligger däremot uppgiften i att modellera resterande och placera ut alla ingående komponenter för denna.

2.2.2 Verktyg för kommunikation med en styrenhet

För att kunna testa simuleringsmodellen med verklig kod krävs att modellen kan kommunicera med externa parter. Ett problem är därför att ge modellen någon form av verktyg för att kunna kommunicera med en styrenhet.

2.2.3 Utvärdering av processäkerhet

En viktig del vid optimering av en produktionscell är processsäkerhet. För att spara resurser gäller det att göra rätt från början vilket lättast uppnås genom att produktionscellens processer fungerar korrekt, samt att fel i processerna förs upp till ytan direkt så att de kan åtgärdas [18]. Uppgiften ligger således i att säkerställa att processerna fungerar så som de är tänkta att göra. Det är exempelvis ett stort problem om hela tillverkningsprocessen genomförs utan att robotarna försörjs med material då denna outnyttjade produktionstid är dyr sett till både personal- och driftkostnader.

2.2.4 Utvärdering av personsäkerhet

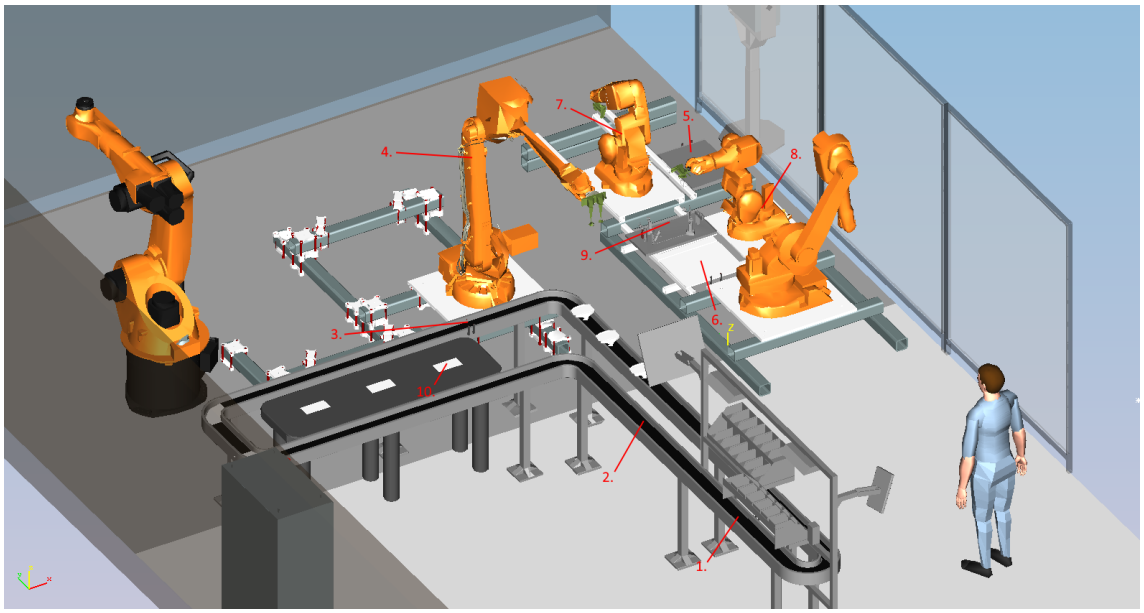
Ytterligare ett problem är avsaknaden av möjligheten för en operatör att kunna testa och utvärdera säkerhetsanordningar utan att själv behöva utsätta sig för de risker som finns kopplade till en fysisk produktionscell. Det finns i dagsläget möjlighet att studera och testa säkerhet i produktionssystem via virtual commissioning, dock utan möjligheten för en operatör att själv befinna sig samt agera i den virtuella miljön. Själva modelleringen ligger som grund i att kunna lösa en signifikant del av detta övergripande problem.

2.2.5 Likhet med verkligheten

Vidare finns det en del komplikationer som kan uppstå då en avbildning görs av verkligheten, vilket i detta fall innebär övergången från den faktiska produktionscellen i PSL till en modell i Process Simulate. En modell är aldrig till 100 % i enighet med verkligheten och innebär att slutsatser dragna från en simulering skulle kunna innebära felaktiga anpassningar vid implementation i en fysik miljö. Problemet att hantera ligger därmed i att kunna skapa en virtuell modell som är representativ för verkligheten och kan användas som grund för dess validering.

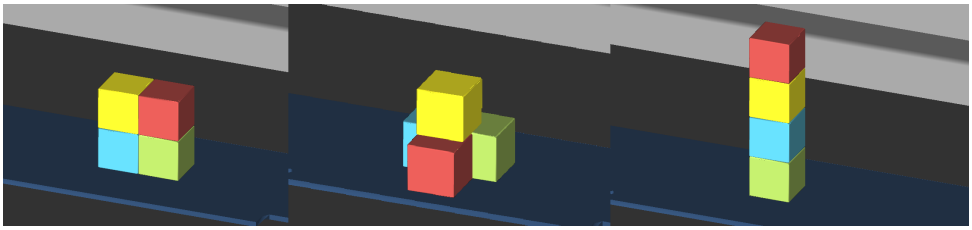
2.3 Resultat

Den slutgiltiga simuleringsmodellen illustreras i Figur 2.1. Designen och de ingående komponenterna är baserade på den verkliga produktionscellen som finns placerad i PSL.



Figur 2.1: Simuleringsmodell avbildad från Chalmers PSL i Process Simulate. Siffrorna är kopplade till viktiga komponenter i cellen.

Tre av den virtuella modellens ABB-robotar (nr. 4, 7 och 8 i Figur 2.1) opererar för att bygga olika typer av klosstorn, vilka baseras på olika kombinationer av det ingående materialet i form av fyra klossar. Utifrån operatörens önskemål anpassas robotarnas operationer för respektive torn, där klossar samt färdiga torn däremot hämtas och lämnas på samma positioner oavsett typ. De tre varianterna av klosstorn som kan tillverkas i den virtuella produktionscellen illustreras i Figur 2.2.



Figur 2.2: De tre klosstornsvarianter som kan konstrueras i den virtuella produktionscellen.

Processen för hur de olika tornen byggs skiljer sig åt men har en genomgående likhet i operationernas struktur. Siffrorna i listan nedan motsvaras av siffrorna i Figur 2.1 och arbetsgången sker enligt:

I. Fyra klossar placeras ut på en platta för transport vid operatörens arbetsstation (1). Transportbandet (2) kör sedan fram plattan med klossarna till

angiven position för upphämtning (3). Upp till tre stycken olika plattor med klossar kan skickas in i processen samtidigt och läggs då in i ett kösystem på transportbandet. Följaktligen kan flera olika klosstorn initieras samtidigt men konstrueras i sekvens efter varandra utan ytterligare ingripanden från en operatör.

II. Den större ABB-roboten IRB4600 (4) hämtar två klossar och initierar därmed hanteringen av material från robotarna. Dessa levereras på avlastningsyta(5) för den första av de två mindre ABB-robotarna IRB140, varvid operationen sedan upprepas för resterande två klossar till avlastningsyta för nästa robot (6).

III. De två mindre robotarna (7,8) konstruerar klosstornet på närliggande platta med arbetsyta(9) enligt specifikation för respektive variant av torn. Detta sker simultant där de båda robotarna arbetar samtidigt och därmed samsas om samma svängrum och arbetsyta.

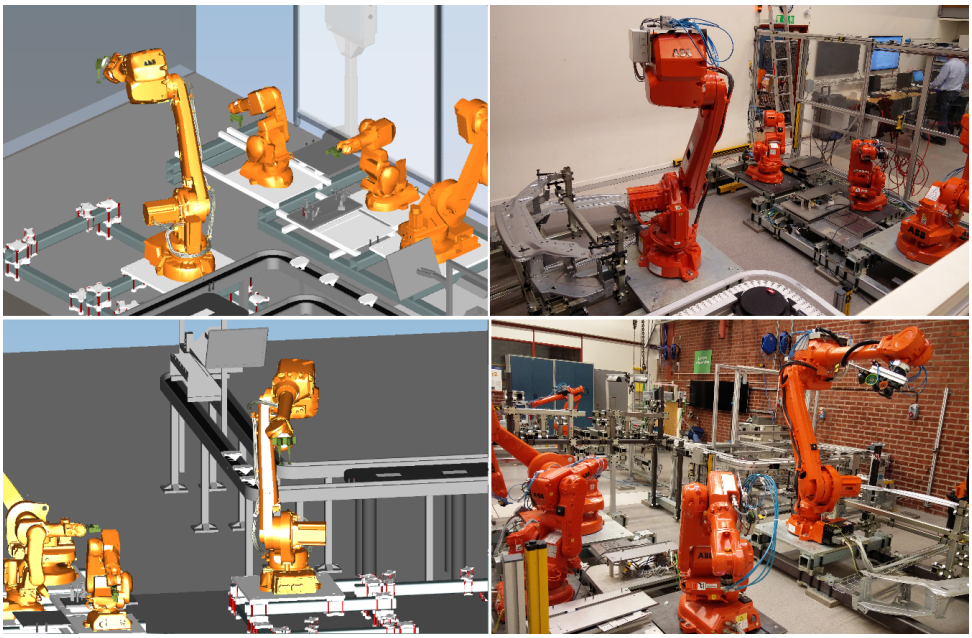
IV. Den större roboten (3) hämtar upp plattan (9) och levererar det färdiga tornet på angiven plats vid transportbandet (10). Tillräcklig avlämningsyta finns för att möjligöra att upp till tre olika torn kan lagras innan de plockas vidare av operatör.

2.3.1 Modelluppbyggnad

Den virtuella produktionscellen är uppbyggd av Smart Components som beskrivs i 2.1.2. De robotar som opererar i produktionscellen finns redan skapade som Smart Components och hämtas från ABB:s hemsida [19]. Även varningslamporna är sedan tidigare färdigdefinierade Smart Components, men resten av komponenterna måste skapas och göras smarta. Nedan listas de komponenter som används för att bygga upp modellen:

- 4 st brioklossar i olika färger
- 1 st ABB-robot IRB4600
- 1 st magnet-gripper för IRB4600
- 2 st ABB-robot IRB140
- 2 st rörliga gripdon, ett för varje IRB140
- 1 st ABB-robot IRB1600 (Används ej i processen)
- 1 st KUKA-robot KR30 (Används ej i processen)
- 4 st robotpedestaler
- Transportband uppbyggt av flera olika sektioner
- 3 st skids/transportplattor för klossar
- 1 st platta för förflyttning av klosstorn
- 17 st olika typer av sensorer (se avsnitt 2.3.3)
- Bord för avlämning av klossar
- 1 st ljusfyr
- Transportbandsställning
- Glasvägg
- Glasgrind
- Väggar
- Golv

Alla ingående komponenter är utplacerade för att efterlikna den verkliga produktionscellen. Placeringen illustreras i Figur 2.3.



Figur 2.3: Virtuell modell respektive verklig produktionscell ur två olika vinklar.

Komponenter som transportbandsställning och robotpedestaler finns att tillgå som CATpart-filer från en server som hanteras av institutionen för produktutveckling på Chalmers Tekniska Högskola. Övriga komponenter skapas med hjälp av programmet Catia V5 där 3D-modeller av komponenterna modelleras upp. För att kunna använda modeller från Catia V5 i Process Simulate måste dessa genomgå en konvertering av filformat enligt Figur 2.4.

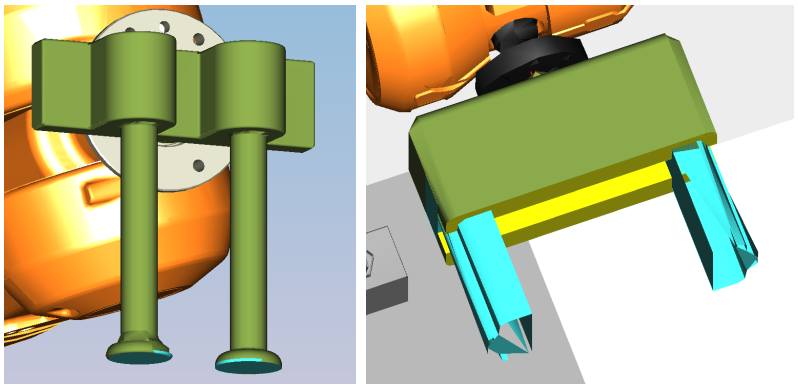
Figur 2.4: Konvertering av filer för att de skall kunna hanteras av Process Simulate.

Det är viktigt att CATpart-filer sparas om som igs-filer för att det skall vara möjligt att importera komponenterna till Process Simulate. Process Simulate hanterar endast cojt-filer, men i Technomatix-paketet ingår en filkonverterare vid namn ”Importer”, som kan konvertera .igs till .co. Dessa co-filer måste sedan uppdateras inne i Process Simulate med verktyget ”Upgrade co to version” så att de blir .cojt-filer. 3D-modeller importeras via Process Designer enligt 2.1.2 och finns sedan tillgängliga i Process Simulate. Djupare beskrivning av hur konvertering genomförs återfinns i Bilaga C.1.

Komponenter som inte behöver kunna röra på sig, såsom väggar, golv och robotpedestaler behöver inte göras till Smart Components eftersom de fyller sin funktion genom att vara statiska 3D-modeller. Däremot kräver gripdon till robotarna och transportbandet definierad kinematik och signalstyrning för att fylla sin funktion.

2.3.1.1 Definiering av gripdon

De gripdon som används i den virtuella produktionscellen är en magnet-gripper för IRB4600 och rörliga gripdon för båda IRB140-robotarna vilka illustreras i Figur 2.5.



Figur 2.5: a) Magnet-gripper till IRB4600. **b)** Rörligt gripdon till IRB140.

Gripdonen skapas genom att definiera kinematik för de 3D-modeller som importerats. Magnet-gripperns tänkta gripyta definieras i Process Simulate som en yta som vid kontakt med en part (material som förflyttas/bearbetas) griper tag i den. Ytan illustreras av det turkosa området på de platta delarna i botten av gripdonet i Figur 2.5a. Vid givna signaler kan då grippern bära med sig samt släppa parts. Även de rörliga gripdonen bär med sig parts enligt samma princip, där gripdonens kontaktytor med parts görs till gripytor, men dessa gripdon rör också på sig i en gripande rörelse. Dessa rörelser fås genom att definiera en öppen-position och en stängd-position för gripdonen. Det som sker är att valda sidor av 3D-modellen förflyttar sig från en position till en annan då signal ges. De ytor som är valda till att kunna förflyttas illustreras som turkosa i Figur 2.5b. Djupare beskrivning av hur kinematik definieras återfinns i Bilaga C.2.

2.3.1.2 Definiering av transportband

Det transportband som finns i den virtuella produktionscellen är också en Smart Component. Transportbandet är uppdelat i olika sektioner för att kunna köra vissa delar av bandet vid olika tillfällen. Bandet är definierat som en ”skid conveyor” och kan i och med det bära med sig ”skids”, också kallat transportplattor. För att parts skall kunna åka med ovanpå transportplattorna krävs att partsen definieras som ”conveyable parts”, där en eller flera ytor av parten utses för att, vid kontakt med ett transportband, föra med sig hela parten. Det fungerar ungefär på samma sätt som när gripytor definieras för gripdon, vilket beskrivs i avsnitt 2.3.1.1 . För att transportbandet skall kunna röra sig skapas utgångar i ett så kallat ”logic block”. När en signal kommer från en styrenhet utför transportbandet en aktion, så som start eller stop. Det är alltså med hjälp av dessa utgångar som det bestäms när transportbandet rullar och när det står still.

2.3.2 Operationsuppbyggnad

Arbetsgången för konstruerandet av klosstorn, som listats upp med fyra punkter tidigare i detta avsnitt, är uppbyggd av ett antal olika typer av operationer beroende på komponent. Själva förflyttningen av klossar i cellen sker genom så kallade ”Pick-and-place”-operationer, vilka är en av flera grundtyper av operationer som finns tillgängliga i robotarnas fördefinierade logik. Dessa utgår i grunden från en pick-position där material skall hämtas upp och en motsvarande place-position där materialet lämnas. Roboten utnyttjar de tidigare nämnda smarta egenskaperna som finns kopplade till gripdonen och möjliggör därmed förflyttningen av klossar. Denna förflyttning sker via ett antal olika punkter i rummet som anpassats till den omgivning roboten arbetar i för att inte störa eller kollidera med tillverkningscellens andra komponenter. Även hastighet och rörelsetyp är justerade vid kritiska punkter av processen, som då IRB4600-roboten rör sig mellan de två IRB140-robotarna för att leverera klossar.

I och med att gripdonen är definierade som Smart Components sker direkt förflyttning av klossar enkelt, detta är dock inte applicerbart på plattan där tornen byggs. För att kunna flytta plattan tillsammans med klossarna då tornen är färdigkonstruerade finns logik programmerad till slutförandet av den Pick-and-place-operation hos IRB4600-roboten som hämtar klosstornet. Genom OLP(Off-line programming)-kommandot ”#Attach” i operationen fästs klossar och platta i varandra vid den position de placeras, vilket gör att allting rör sig som en enhet då plattan sedan flyttas till sin avlämningsposition.

Operationerna för andra, enklare, rörelser utan någon koppling till materialflödet är istället så kallade ”Object flow”-operationer. Som namnet antyder flyttar dessa övriga komponenter mellan positioner i cellen, vilket exempelvis utnyttjas för att öppna och stänga grinden in till tillverkningscellen. Transportbandet och dess transportplattor styrs uteslutande via signaler och de logic blocks som finns definierade, vilket gör att det inte finns några operationer kopplade till dessa. En sammanställning över alla de operationer som utnyttjas i processen återfinns i anslutning till signalerna i Bilaga B.

För att klossar skall färdas genom samtliga operationer krävs att materialflödet definieras i ”Material Flow Viewer”. Detta gör att materialet existerar i modellen tills sista operationen som är definierad i materialflödet utförts. Eftersom klosstorn skall stå kvar på bordet när de är färdigbyggda används en extra operation, efter att materialflödesprocessen är klar, som inte utför något arbete. Klosstornen kan därmed endast raderas ur flödet om denna operation körs, görs inte detta kommer klosstornen stå kvar på bordet. Operationen används alltså som en hjälp för att flera klosstorn skall kunna existera i processen samtidigt.

2.3.3 Verktyg för simulering av säkerhet

För att säkra att både processer fungerar korrekt och att personer är säkra under processförloppet innehåller den virtuella modellen flertalet säkerhetsåtgärder. De flesta av dessa säkerhetsåtgärder finns främst i förebyggande syfte men implementerat finns även sätt att undvika skador då ett fel inträffat.

Kontroll av materialflöde sker genom sensorer som är placerade på olika ställen i

produktionscellen längs med materialflödeskanalen. Dessa sensorer har bland annat som uppgift att tala om för PLC:n då material finns på plats Det finns också de sensorer som har som uppgift att kontrollera att grindar är stängda och att personer inte går för nära robotarna då processer körs. Listade nedan finns samtliga sensorer i den virtuella produktionscellen med tillhörande förklaringar av deras funktioner. Siffrorna i listan illustreras i Figur 2.1.

- Sensorer på transportband vid startposition (1) som kontrollerar om material finns på plats på transportplatta.
- Sensor på transportband vid startposition (1) som kontrollerar om transportplatta är på plats.
- Sensorer på transportband vid upphämtningsposition (3) som kontrollerar om material finns på plats.
- Sensor på transportband vid upphämtningsposition (3) som kontrollerar om transportplatta är på plats.
- Sensorer som kontrollerar om material finns på plats (6) för Robot 1 att bygga med.
- Sensorer som kontrollerar om material finns på plats (5) för Robot 2 att bygga med.
- Sensor som kontrollerar om plattan där klosstorn byggs finns på plats (9).
- Sensorer som kontrollerar om material finns på avlämningsplatser (10), en sensor per avlämningsplats.
- Sensor som kontrollerar om grind är stängd.
- Sensor mellan vägg och transportband kontrollerar om en person passerar in i robotarnas arbetsområde.

För att göra produktionscellen lämplig för mänsklig interaktion finns ett varningssystem som visar olika färger beroende på vad som sker i cellen. Grön innebär att inga processer är igång och att cellen är redo att köras. Gul innebär att produktionscellen arbetar, det vill säga att processer är igång. Röd innebär att något är fel i produktionscellen.

2.3.3.1 Simulering av fel

När arbete sker i ett verkligt produktionssystem kan vissa typer av fel uppstå på grund av yttre omständigheter, så som utdaterade komponenter eller elfel. Vid simulering i en virtuell miljö uppstår detta däremot inte naturligt och modellen har anpassats till att kunna simulera ett beteende som skulle motsvara fel i verkligheten. Via fördefinierade felaktiga operationer kan manuell styrning av den virtuella produktionscellen testas genom användandet av ett nödstopp, vilket beskrivs mer i avsnitt 3.3.1. De felaktiga operationer som den virtuella modellen kan genomföra är:

- Ett elfel har uppstått och den stora roboten, IRB4600, rör sig okontrollerat.
- IRB4600 placerar material på fel ställe.

När dessa felaktiga operationer utförs är tanken att operatören skall förstå att något gått fel och därmed stoppa processen. Hur interaktionen sker beskrivs i avsnitt 3.3.2

2.3.4 Signaler som kommunikationsverktyg

Kopplat till alla de operationer, sensorer och säkerhetsutrustning som behandlats i tidigare avsnitt finns genererade signaler, vilka är en grund för genomförandet av en virtual commissioning. Signaler genereras för varje operation som skall genomföras i produktionscellen. Till exempel definieras signaler för att gripdon skall kunna gripa parts, för att transportband skall rulla och stoppas samt för att robotar skall kunna köras och pausas. Dessa signaler är utsignaler och aktiveras av att den styrande enheten skickar information om vad som skall utföras. För att tillhandahålla information från de sensorer som finns utplacerade i den virtuella modellen krävs även där signaler, men dessa är istället insignaler eftersom de skickar information till styrenheten vid tillståndsförändring. Samtliga signaler finns listade i Bilaga B.

För att förenkla styrning och efterlikna hur produktionssystem styrs ute i tillverkningsindustrin är operationer ihopbuntade till så kallade robotprogram. I varje robot finns ett robotprogram inläst och varje operation i robotprogrammen motsvarar en unik siffra som PLC:n kan skicka för att initiera att just den operationen skall köras. Att robotarna agerar efter dessa robotprogram möjliggör funktioner som att pausa och återuppta simuleringen i den virtuella modellen. Vilka rörelser som aktiveras av vilken siffra beskrivs i Bilaga B.

Ovanstående är fundamentalt att uppfylla för kommunikation med en PLC och därigenom den externa styrbarheten av modellens arbete, vilket är en av grundpelarna i genomförandet av en virtual commissioning. Vidare har också namn på signaler synkroniserats med de signaler som ligger på en OPC-server för att möjliggöra kommunikation mellan Process Simulate och PLC. Djupare beskrivning av kopplingen mellan Process Simulate och PLC återfinns i avsnitt 3.3.4.

I Figur 2.3 visas den verkliga produktionscellen i PSL bredvid den virtuella modellen av samma produktionscell. Flera komponenter som inte används vid byggande av klosstorn är utelämnade i och med att fokus ligger på att göra en virtual commissioning av just klosstorns-processen. Modellens olika komponenter, såsom robotar och transportband, är utplacerade med sådan verklig likhet att allt material kan förflyttas mellan de platser som definieras ovan i avsnitt 2.3.

3

Styrning av virtuell produktionscell med en PLC

Detta kapitel sammanfattar hur en PLC arbetar och vad den bidrar med inom produktionsstyrning, problematik som uppstår vid kommunikation över nätverk och simulering av säkerhet i en virtuell arbetscell. En resulterande uppkoppling med OPC-server och visuella gränssnitt beskrivs mer ingående samt hur strukturen över logisk kod är framtagen.

3.1 Bakgrund

PLC (Programmable Logic Controller) är en styrenhet likt en dator och är anpassad för industriellt bruk samt vanligt förekommande i produktionssammanhang. En PLC tar emot sensordata som används i kod för att generera styrsignaler för maskiner och processer. Styrenhetens CPU tolkar logiska villkor baserat på boolesk algebra och använder standardiserade programmeringsspråk som LAD-grindar, GRAPH-sekvenser och strukturerad text.

Innan 1969, då den första PLC:n togs fram [20], användes olika typer av reläsystem för att styra industriella processer. De var skrymmande och oflexibla system där ändringar exempelvis innebar att fysiska elkablar var tvungna att kopplas om mellan befintliga kontaktorer och relän. Enligt [20] finns det ett flertal stora skillnader mellan PLC-styrning och reläsystem. Förutom prestandaförbättringar i form av högre hastighet och minskat underhåll så är även en PLC lättare att programmera om, då ändringar utförs i mjukvaran istället för omfattande omkopplingar i hårdvaran. PLC är både stabilare och pålitligare än ett reläsystem då det senare består utav en stor mängd mekaniska delar vilket vid slitage kan försämra funktionen och därmed medföra dålig respons eller felaktiga signaler.

Eftersom en PLC är en typ av dator har den många likheter med en PC men den skiljer sig också på ett par viktiga punkter. En PLC är exempelvis byggd för att klara av en industrimiljö där det är vanligt med störningar i form av elektromagnetiska fält, vibrationer, höga temperaturer och hög luftfuktighet [20]. PLC:n är utformad med ett mer lättanvänt gränssnitt för att hantera stora mängder in- och utsignaler och ställer krav på robusthet då driftsäkerhet är viktigt ur ett produktionsperspektiv. [20] påpekar däremot att en PC är mer lämpad än en PLC vid krävande analyser, komplexa beräkningar och för lagring av data över en längre tid.

Idag är PLC en grundläggande komponent i den industriella automationen [21] och därmed en viktig del i arbetet mot vad som kallas för den fjärde industriella

revolutionen, industri 4.0. Industri 4.0 är en vision för utvecklingen av industriverksamheten där industriella processer kopplas samman med informationssystem och där kommunikation och informationsutbyte sker över något som brukar kallas ”Internet of things” [22].

3.1.1 Kommunikation och informationshantering mellan enheter

OPC (OLE for Process Control) är ett standardiserat verktyg för att kommunicera mellan en PLC och en PC över ett nätverk [23]. OPC-tekniken uppkom då det fanns ett problem med att olika leverantörer av PLC-hårdvara hade olika sätt att lösa kommunikation med deras hårdvara [24]. Detta medför enligt [24] en rad problem:

I. Orsak till mycket dubbelarbete då alla användare måste skriva specifika drivrutiner till alla olika leverantörers hårdvara.
II. Inkonsekvens mellan olika leverantörers drivrutiner, då det finns många hårdvarufunktioner som stöds av en leverantör men inte av de andra.
III. Känslighet för förändringar i hårdvarans funktionalitet då förändringar kan få drivrutinerna att sluta fungera.
IV. Åtkomstproblem uppstår också när två olika datapaket inte kan komma åt samma enhet samtidigt ty de innehåller olika drivrutiner.

OPC-standarden sparar tid, minimerar kostnader och minskar komplexiteten då specialbyggda kommunikationsverktyg inte behöver utvecklas till alla de hundratals enheter som kan finnas i ett produktionssystem.

Användandet av OPC-tekniken har utvidgat användningsområdena för PLC då det standardiserade språket möjliggör kommunikation med simuleringsprogram som stödjer OPC-standarden. Det finna olika typer av överföringstyper inom OPC-standarden, bland annat OPC DA (Data Access) vilken, till skillnad från OPC UA (Unified Architecture), har en bättre realtidsuppdatering för att förhindra signalfenomen som jitter, långsam sampling och tidsfördröjning av olika slag [25]. En alternativ standard för kommunikation har tagits fram av företaget Object Management Group kallad Data Acquisition from Industrial Systems (DAIS) [26], vilket är baserat på COBRA [27] men är till skillnad från OPC mer anpassad för realtidsapplikationer mellan datorer.

3.2 Problem

I detta avsnitt beskrivs de problem som uppkommer vid kommunikation och signalutbyte på distans genom IP-protokoll, skillnaden i kodgenereringen mellan verkliga och virtuella celler samt säkerhetsaspekter i virtuell utrustning.

3.2.1 Kommunikation mellan enheter på skilda destinationer

Om olika komponenter är på geografiskt olika platser är det väldigt viktigt att kunna kommunicera utan att kommunikationen brister. Problem som uppstår då två eller

flera datorer måste kommunicera över nätverk kan vara svårigheter att upprätthålla datautbyte genom datorernas integrerade brandvägg som då blockerar det datorn anser vara riskfullt [25]. Dessa inställningar är deklarerade i ett COM/DCOM-protokoll som är ett överordnat system i Windows-datorer vilket koordinerar datorns olika portar och ställer krav på att hitta en säker anslutning.

3.2.2 Konvertering av kod mellan fysisk och virtuell cell

Det förekommer skillnader i utrustningarnas beteende mellan verklig och virtuell modell. Då komplexa modeller kan tänkas styras av flera olika kommunicerande styrsystem som bland annat robotar och specifika maskinutrustningar, blir en utmaning att simulera från ett och samma styrsystem mot en virtuell cell. Precision vid rörelser och mätningar kan också vara svåra att efterlikna då mekaniken och materialegenskaperna inte uppfattas på samma sätt som vid verklig produktion.

3.2.3 Simulering av säkerhet och säkerhetsutrustning

Även om säkerhetssystem kan testas på en virtuell cell finns det en begränsning mellan verklig och simulerad interaktion. Problematiken ligger främst i att mänskliga objekt i simuleringen inte interagerar på samma naturliga sätt som en människa. Produktionscellen påverkas inte på samma sätt som med en verklig operatör som kan trycka på knappar, påverka nödstopp och ljusbommar, men även känna av värme och andra indikationer som i dagsläge är svåra att imitera verklighetstroget gentemot en användare av virtuell simulering. Det mänskliga objektet kan påverka specifika givare i simuleringen som kan liknas med ljusbommar men för att påverka nödstopp-knappar och annat måste dessa finnas fysiskt och vara tillgängliga för operatören. Begränsningar förekommer även i hur väl förekomsten av vanliga tekniska fel på utrustningen lyckas simuleras. Det är svårt att förvänta sig verkliga fel i elektrisk utrustning, som att en motor exempelvis skulle börja brinna på grund av en kortslutning eller att en mekanisk arm blir utsliten med tiden. Operatören kan enbart uppmärksammas av de visuella avvikelserna och är då blind för värmeutveckling vilket i en verklig miljö skulle indikera fara.

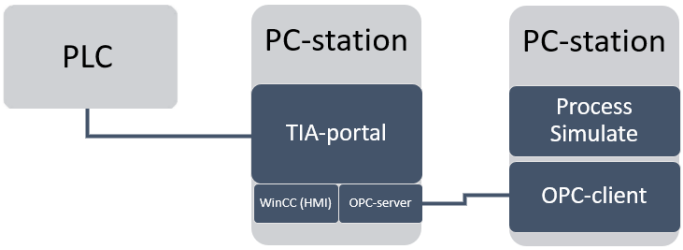
3.3 Resultat

Följande avsnitt beskriver ingående komponenter för utförandet av en virtuell styrning med PLC. Funktionaliteten med ett visuellt gränsnitt beskrivs tillsammans med uppbyggnad av PLC-kodens struktur och konfiguration med en OPC-server för lyckad kommunikation mellan samtliga enheter.

3.3.1 Utrustning och komponenter

Den centrala styrenheten för detta projekt är en Siemens SIMATIC S7-1500 [28] som tillsammans med en PC är anslutna till samma nätverksswitch med ethernet-kabel. Datorn använder Siemens TIA-portal [29] som mjukvara för att kunna konfigurera

och programmera PLC:n. I Figur 3.1 visas en övergripande illustration av samtliga ingående komponenter.

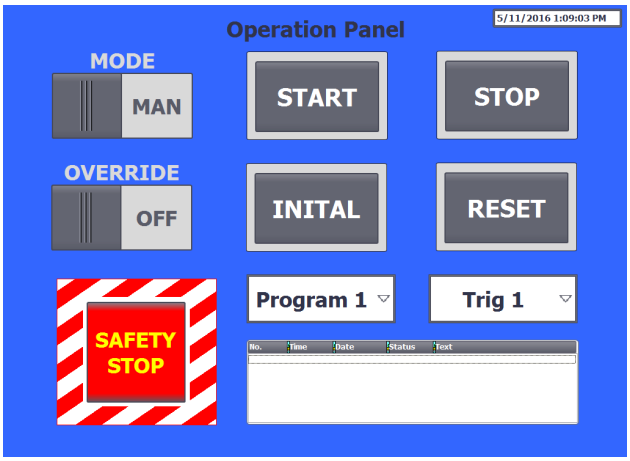


Figur 3.1: Uppkopplingen mellan PLC, dator med TIA-portal och OPC-servers anslutning mot klient-dator med Process Simulate. Ljusa rutor innebär hårdvara och mörkare rutor indikerar mjukvara.

Alla komponenter som är anslutna till nätverket via ethernet tilldelas var sin unik IP-adress och kan bestämmas i TIA-portalens hårdvarukonfiguration. Det är nödvändigt att konfigurationen överensstämmer med den verkliga uppkopplingen för att undvika kompileringsfel. Eftersom styrningen huvudsakligen bearbetar inkommande och utgående signaler från Process Simulate är det inte aktuellt att skicka ut signaler till externa IO-noder då dessa är till för fysisk utrustning, exempelvis sensorgivare och motorer. För att operatören ska lyckas interagera med cellen används det visuella gränssnittet och specifika sensorer som modellerats i Processes Simulate att efterlikna ljusbommar för att känna av det mänskliga objektet.

3.3.2 Styrsystemets gränssnitt och funktionalitet

PLC-programmet som styr och kontrollerar rörelserna i Process Simulate är utformat för att på ett pedagogiskt sätt göra det möjligt för operatören att utforska och, till begränsad del, påverka cellen. Till projektets virtuella cell är ett specialanpassat visuellt gränssnitt utvecklat i TIA-portalens instickspogram WinCC [30]. Detta HMI visas i Figur 3.2 och kontrolleras via PC-stationens skärm.



Figur 3.2: Visuellt gränssnitt i programmet WinCC för styrning av produktionscellen i Process Simulate.

En beskrivning av de olika knapparnas funktion listas nedan:

- **MODE** – Försätter cellen i antingen manuellt eller automatiskt läge beroende på vilken typ av styrning som önskas.
- **START** – Startar produktionen när automatik är valt och cellen står i utgångsläget (INITIAL).
- **STOP** – Stannar produktionen vid pågående drift.
- **OVERRIDE** – Denna manöver möjliggör att öppna skyddsdörr under produktion.
- **INITIAL** – Kör samtliga utrustningar till initial position innan uppstart.
- **RESET** – Återställer larmtexter från tidigare aktiva larm som blivit åtgärdade.
- **SAFETYSTOP** – Stannar produktionen omgående och kräver kvittens med reset-knapp innan återställning.
- **Program** – Val av klossprogram, totalt tre stycken olika program.
- **Trig** – Ett förprogrammerat fel tvingas fram, totalt 3 stycken olika fel.
- **Alarm-field** – Denna ruta visar aktuella larm och meddelanden.

Genom att använda de sensorer och förprogrammerade operationer för simulering av fel som beskrivs i avsnitt 2.3.3 kan konsekvenserna av säkerhetsbrister i produktionscellen illustreras. Följaktligen kan kan en operatör testas genom att själv leta efter tänkbara orsaker till tre olika problem som skulle kunna uppstå i verkligheten.

Det första felet försätter en av materialpositionens närvarogivare till första monteringsroboten (IRB140) ur funktion genom att konstant skicka en positiv signal, vilket förhindrar programsekvensen att fortsätta då cellen inte längre vet var materialet är placerat. Detta genererar ett larm i PLC:n som operatören måste återställa med reset.

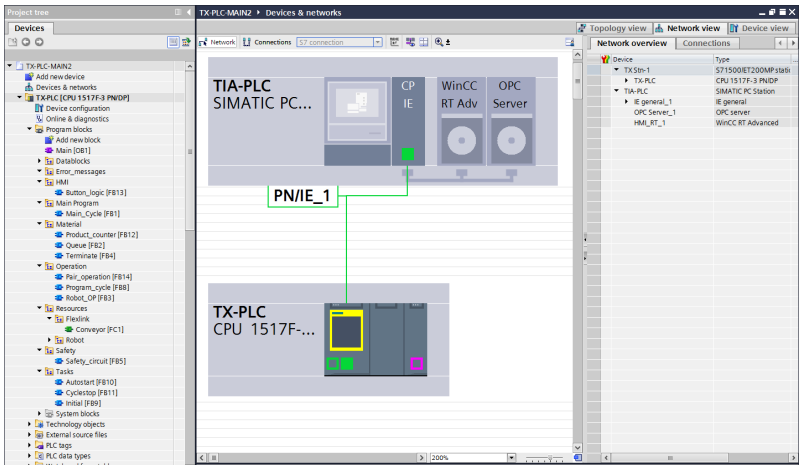
Det andra felet innebär att plockroboten (IRB4600) placerar plattan med klossar på ett annat ställe än planerat, vilket då kommer innebära att första monteringsroboten (IRB140) inte kan utföra sitt jobb då inget material finns att tillgå. Detta larm måste också nollställas via gränssnittet och därefter startas produktionen om på nytt.

Det tredje felet skickar en signal till plockroboten (IRB4600) att utföra en felaktig och farlig rörelse genom att oavbrutet snurra runt med armen utsträckt med risk att slå sönder utrustning eller skada eventuell operatör. Det gäller här för operatören att trycka på nödstopp för att få roboten att sluta snurra. Liknande fel kan tänkas förekomma i verkligheten då en robot utsätts för strömbortfall vilket kan orsaka att felaktiga uppdrag skickas till roboten.

3.3.3 Struktur och uppbyggnad av logisk kod

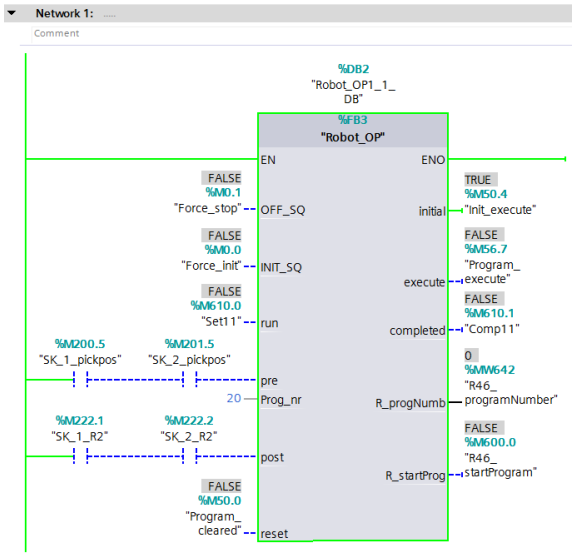
Vid programmering av logisk kod är det att föredra att ha struktur och ett system vid adressering av signaler för att göra det enklare att följa sekvenserna. Som tidigare nämnt i avsnitt 3.3.1 konfigureras hårdvarans komponenter i TIA-portal där all nödvändig information är definierad för att lyckas få en PLC i drift (se Figur 3.3). Vidare skapas ett nätverk av logisk kod i olika funktionsblock. Detta skrivs främst med logiska grindar (LAD) och sekvensblock (GRAPH). Koden, som baseras på

boolesk algebra, skapar operationer för logiska villkor och hanterar styrsystemets in- och utsignaler, interna minnen och databytes i olika former.



Figur 3.3: Projektöversikt av PLC-program i Siemens TIA-portal. I högra fönstret listas programmets tillhörande funktionsblock och i vänstra fönstret visas hårdvarans komponenters index.

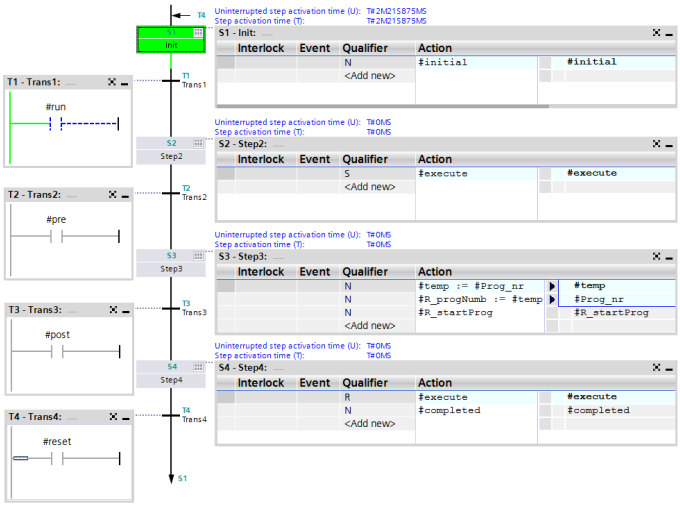
Vid återkommande användning av olika operationer kan instanser av samma funktionsblock skapas för att undvika upprepning av samma kod och på så sätt spara tid. Dessa instanser fungerar som mallar och kan anropas och tilldelas unika pre- och postconditions beroende på applikation. Detta har tillämpats på samtliga robotrörelser för cellen och visas i Figur 3.4. Till höger om blocket definieras ingångssignalerna vilka skickar svaret från den interna funktionen till utsignalerna på vänstra sidan.



Figur 3.4: Instans av funktionsblock till en robot.

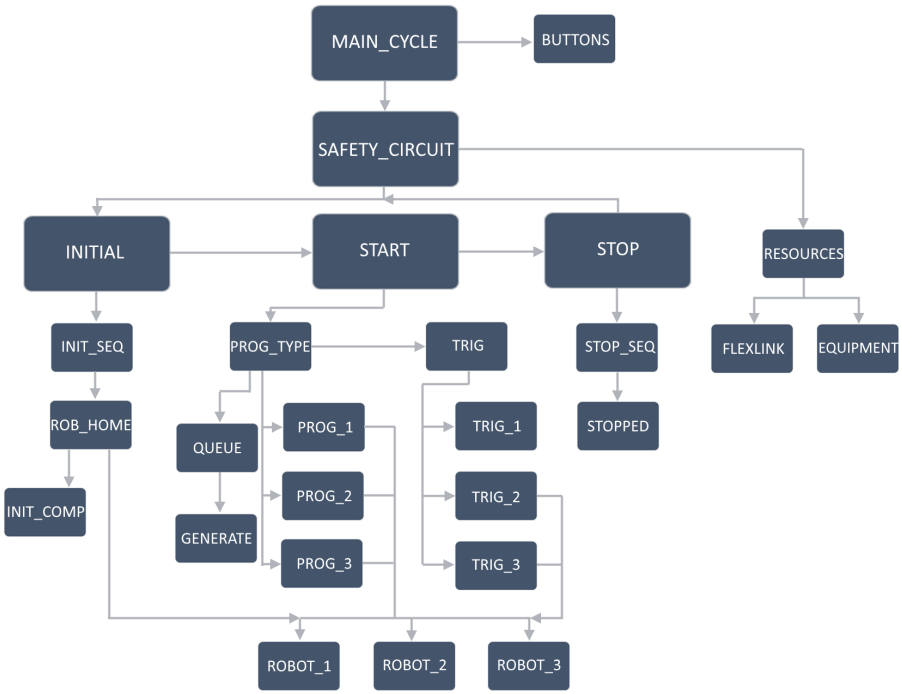
Varje sekvensprogammerat funktionsblock följer en enhetlig standard genom hela projektet. I Figur 3.5 visas det hur den inre sekvensen ”Robot_Op” i Figur 3.4 är

strukturerad. Det är denna sekvens som körs för att avgöra vilka värden utsignalerna i Figur 3.4 skall tilldelas. Grönt block indikerar körande funktion samt vilken exekvering som skickas till utsignalerna och efterföljs av övergångsvillkoret till nästa block. Villkoren är skrivna med LAD-logik och måste få ett boolskt TRUE- eller FALSE-värde för att programmet ska gå vidare.



Figur 3.5: Struktur över inre sekvensen i funktionsblocket för roboten.

Sekvenser och operationer kallas från en av de tre överordnade sekvenserna INITIAL, START eller STOP, och respektive svarar genom att initiera en exekveringssignal som förblir aktiv under tiden funktionsblockets uppdrag genomförs. I Figur 3.6 visas en överskådlig och förenklad exekveringstabell av PLC-programmet.



Figur 3.6: Exekveringstabell över PLC-programmets kod.

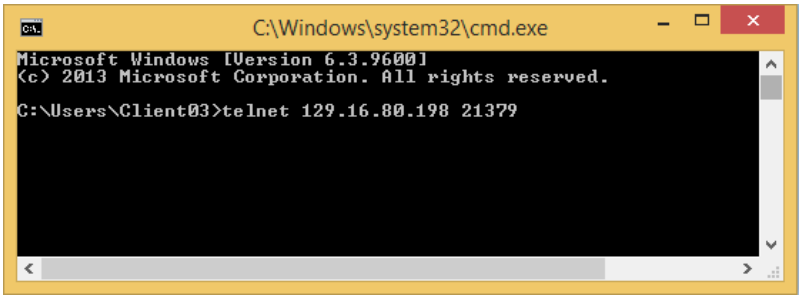
Initialt startas programmet i Main_Cycle, vilket är ett funktionsblock som invänter knapptryckningar från WinCC-panelens manöverknappar. I detta tillstånd måste först automatik väljas med MODE-omkopplaren för att möjliggöra utgångslägessekvensen INITIAL som är en av tre huvudsekvenser som exekveras efter varandra. Denna sekvens återställer samtliga programsekvenser och försätter produktionscellen i utgångsläge. Den andra huvudsekvensen är START och startas först då INITIAL är klar. I START kan programtyp väljas likväl som övriga önskade funktioner. Sekvensen startar flera underprogram för att utföra valt klosstorn och stoppas först då STOP väljs på panelen och avbryter det pågående programmet. Nu kan återigen INITIAL startas vilket sluter cykeln.

Om ett nödstopp eller en ljusslinga påverkas under drift spärras samtliga exekveringar omedelbart och programmet stoppas. Operatören måste därefter kvittera den brutna nödstoppskretsen genom att trycka reset för att återgå till STOP-sekvensen.

3.3.4 Kommunikation via OPC-server

I hårdvarukonfigurationen i TIA-portal definieras en OPC-server och arbetar på samma PC-station men eftersom OPC-servern och simuleringsprogrammet Process Simulate befinner sig på olika datorer (se Figur 3.1) förutsätter det att en dubbelriktad kommunikation genom nätverk fungerar tillfredställande. För god realtidsuppdatering används OPC-standarden OPC-DA (Data Access).

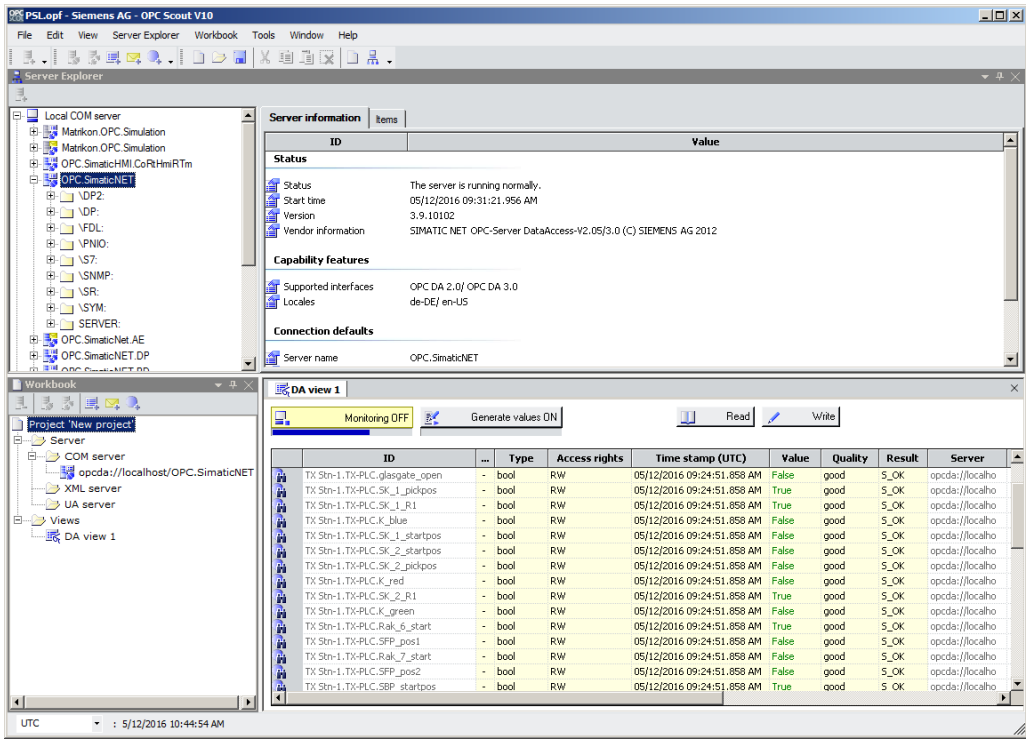
En OPC-tunneller från företaget Matrikon konfigureras på både serverdatorn och klientdatorn med rätt inställningar och förutsättningar för att upprätthålla en stabil och komprimerad överföring av signalutbyten, utan att äventyra brandväggars och TCP/IP-protokolls säkerhet genom COM/DCOM-portar. Dock krävs det att båda datorerna har en så kallad ”Log on as a service”-rättighet på datorns användarkonto vilket går att tilläggas i ”Local Security Policy” på en Windows-dator. Utöver det måste även klientdatorn ha tjänsten ”Telnet-client” eller motsvarande program installerat för att kunna hitta och upprätthålla en kontakt via datorernas olika portar. När verktyget Telnet är tillgängligt kan det köras via kommandotolken enligt Figur 3.7 där serverdatorns IP-adress och portnummer anges. Lyckad anslutning visas i Figur 3.8.



Figur 3.7: Användande av Telnet i Windows kommandotolk. Här anges serverdators IP-adress samt önskat portnummer.

Figur 3.8: Lyckad anslutning mellan portar via Telnet-tjänsten. I detta läge är angiven port öppen och redo för signalutbyte.

TIA-portalen tillhandahåller ett verktyg som heter OPC-Scout och kan användas för att adressera signalerna från PLC:n så att de stämmer överens med Process Simulate. OPC-Scout fungerar enbart som en lyssnare eller tolk, men ifall signalerna mellan de olika programmen redan är identiska räcker det med att Process Simulate stämplar alla signaler med ”TX Stn-1.TX-PLC.” för att adresseringen mot rätt PLC ska stämma. Det är på så vis mer tidsbesparande att adressera på detta sätt vilket projektet också har rättat sig efter. I detta projekt används OPC-Scout endast till att kunna överblicka signalutbytet och status i realtid på samtliga signaler (se Figur 3.9).



Figur 3.9: Siemens OPC-Scout visar signalutbytet i realtid mellan server och klient.

4

Visualisering av virtuell produktionscell med Oculus Rift

I detta kapitel kommer aspekter kring användandet av Oculus Rift och virtual reality (VR) för att visualisera virtuella produktionsceller att hanteras. Problemen som uppstår inom området Oculus Rift när denna koppling skall göras och de resultat som uppnås kommer också att presenteras här.

4.1 Bakgrund

Oculus Rift är en head mounted display (HMD) för att återge VR. Det finns dock ingen vedertagen definition på VR utan de som bedriver forskning och utveckling inom området har alla olika definitioner [31]. I denna rapport definieras VR som datorteknik som återskapar en verklig miljö i en virtuell värld som en användare kan interagera med. VR skapar sinnesupplevelser som syn, hörsel och känsel genom att återge dessa på ett trovärdigt sätt.

Oculus Rift är ett par glasögon som omsluter hela synfältet. I glasögonen visas sedan en bild för höger respektive vänster öga med följd att en visuell 3D-effekt uppnås. Oculus Rift har även en mängd sensorer som känner av bland annat rotationer, magnetfält, med mera. Dessa kan sedan användas för att ändra bilden som användaren ser om denne vrider huvudet. Bilden följer alltså med var användaren tittar och en känsla av att ”vara inne i” den virtuella världen erhålls. Oculus Rift visas i Figur 4.1.

Figur 4.1: En användare som bär VR-glasögonen Oculus Rift.

Utöver Oculus Rift finns det några andra liknande verktyg som skulle kunna användas; HTC Vive, Samsung Gear VR och Google Cardboard. De skiljer sig åt vad gäller pris, antal sensortyper och antal sensorer. Det finns även skillnader kring hur utveckling av egen mjukvara går till.

4.1.1 Aktuella användningsområden för virtual reality

VR finns i dagsläget inom ett flertal områden så som flygsimulatorer, produktutveckling, visualisering, rehabilitering, psykoterapi [32], datorspel och utbildning.

I olika processer inom produktutveckling, som exempelvis design, prototypframtagning och tillverkning, i fordonsindustrin har VR visat sig vara mycket fördelaktigt [33]. Vidare visar [33] att VR har medfört ökad kvalité av slutprodukten, reducerad kostnad av hela processen, minskning av tid det tar för produkten att komma ut på marknaden och ökat välmående hos användarna.

4.1.2 Virtual reality som stöd vid forskning och utbildning

Användandet av VR för utbildning inom sjukvården gör att exempelvis en kirurg kan öva på en svår operation flera gånger utan någon risk för att skada en patient [34]. Det möjliggör även en tydlig visualisering av kroppsdelar och skadescenarion utan att det behövs verkliga exempel, vilket framförallt är tidseffektivt och säkerställer möjligheten att alltid kunna visa upp ett visst scenario.

Utbildning av operatörer som skall arbeta inom produktion fungerar lika bra med VR som i verkligheten [35]. [35] visar att de operatörer som tränas i verkligheten har samma kunskapshållning och hade lika lätt att tolka kunskap som de som använder VR.

Oculus Rift har även använts för forskning inom medicin, där programmet Molecular Rift har tagits fram med syftet att underlätta medicinutveckling där man i programmet kan visualisera atomer, molekylbindningar med mera [36]. Visualiseringen gör det möjligt att undersöka proteiner på ett helt nytt sätt vilket kan leda till nya upptäckter inom medicinforskningen.

4.1.3 Oculus Rifts påverkan på människan

Det har även forskats på hurvida förmågan att hålla balansen påverkas vid användandet av Oculus Rift. Denna studie gjordes i Danmark och balansen testades genom att försökspersonerna fick spela ett snowboardspel, med en egenutvecklad snowboard för att styra karaktären i spelet, där bilden visades dels en omgång på en vanlig bildskärm och en gång genom Oculus Rift [37]. Enligt [37] var det svårare att ha kontroll över kroppen när man bar Oculus Rift oavsett vilka fysiska förutsättningar och levnadsvanor försökspersonerna hade innan undersökningen.

En nackdel som VR-verktygen Oculus Rift och andra HMD:er har är att de kan framkalla simulationssjuka hos användaren [38]. Detta kan begränsa hur mycket användaren kan nyttja dessa verktyg och kan influera dennes välmående negativt.

4.1.4 Programvaran Jack och styrning av mänskliga objekt

Jack är en mjukvara som kan användas för att göra analyser av och operationer på mänskliga objekt i Process Simulate [39]. I Jack kan moduler läsas in där modulerna är småprogram som innehåller utökad funktionalitet för programmet. Exempelvis finns en modul för Kinect-kameror och en modul för att simulera när mänskliga objekt går i trappor färdiga i Jack.

I Jack finns en modul för att samarbeta med Process Simulate som heter Jack Collaboration. Den sätter upp ett server-klient-förhållande mellan Jack och Process Simulate och medför att om det mänskliga objektet byter position i Jack byter det mänskliga objektet i Process Simulate sin position till samma som i Jack.

4.2 Problem

Problemet är att det inte finns ett naturligt sätt att använda Oculus Rifts hårdvara tillsammans med den styrutrustning och simuleringsmjukvara som återfinns i industrisammanhang. I dagsläget används enbart Oculus Rift som en extern bildskärm där simuleringen visualiseras oberoende av Oculus Rifts orientering och rörelse.

4.3 Resultat

I detta avsnitt kommer de viktigaste resultaten och slutsatserna inom visualiseringen av den virtuella produktionscellen med Oculus Rift att framföras. Beskrivningar av de script och filer som bygger upp systemet kommer att redovisas och dess kommunikation kommer att redogöras.

4.3.1 Importering och användning av JackScript

JackScript är ett scriptspråk baserat på Python 2.6 och implementerat i Jack. Detta scriptspråk kan användas för att skicka signaler till Jack och få mänskliga simuleringsobjekt att exempelvis skapas, roteras och förflyttas. Samtliga metoder som kan utföras av JackScript finns dokumenterade i Jack.

”JackLib” är en egen implementering av viss funktionalitet från det importerade JackScript-biblioteket ”js”. Det finns metoder och klasser implementerade i ”JackLib” för att skapa mänskliga objekt samt rotera och translatera dessa. ”JackLib” är till för att flytta så mycket funktionalitet som möjligt bort från den del av modulen som måste skrivas i scriptspråket Tcl (Tool Command Language) till Python. Detta för att Python är lättare att arbeta i då det har mycket dokumentation, många färdiga bibliotek med användbara funktioner samt en stor användarbas där det finns information att hämta.

4.3.2 Konstruktion av moduler till Jack i Tcl

Det krävs huvudsakligen två Tcl-filer för att skapa en modul till Jack. En huvudfil för modulen, som lämpligtvis döps efter namnet på modulen, samt en index-fil vid namn

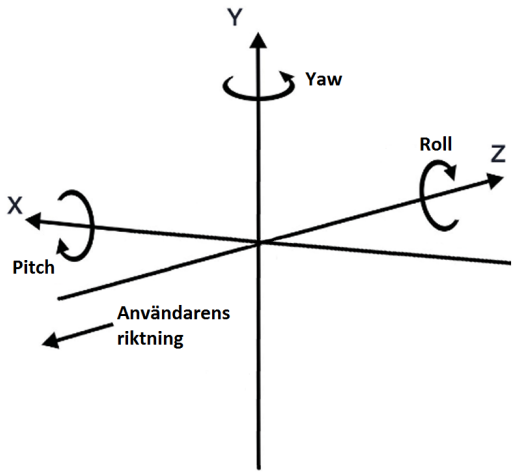
”pkgIndex.tcl”. Den senare av de två skall innehålla information kring alla filer som anropas från huvudfilen och var programmet kan hitta dem. Huvudfilen för modulen kan innehålla funktionalitet som menyuppdelning, anrop och importering av andra filer samt eventhantering.

Vid konstruktion av moduler till Jack är det viktigt att ha följande i åtanke:

- Alla filer som finns med i ”pkgIndex.tcl” kommer att köras en gång av Jack när modulen startas. Därför bör det inte finnas några anrop av metoder ”löst” i koden då de i sådana fall kommer exefieras innan modulen har startats helt.
- Undvik användandet av loopar (framförallt långa till oändliga loopar) i modulen och alla eventuella underliggande script och funktioner som anropas i huvudfilen. Annars är risken stor att det uppstår låsningar mellan olika moduler samt mellan Jack och den aktuella modulen.
- För funktionalitet som skall upprepas kontinuerligt bör modulen göras eventbaserad för att undvika låsningar både i modulen och mellan andra moduler om flera skall användas.

4.3.3 Spårande av huvudrörelser med Oculus Rift

För att implementera head-tracking måste sensordata läsas från Oculus Rift och tolkas på ett sätt som gör att Jack ändrar på det mänskliga objektets orientering. Detta utförs utifrån de fördefinierade rotationerna yaw, pitch eller roll. Dessa rotationer visas i Figur 4.2. Anledningen till att koordinatsystemet i detta projekt är definierat enligt Figur 4.2 är på grund av att det är på det sättet som Oculus Rift har definierat sitt koordinatsystem.



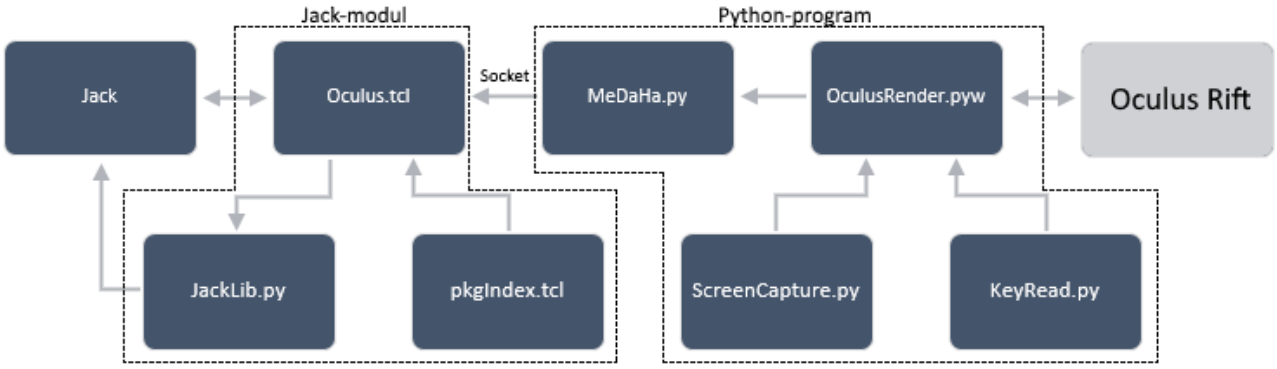
Figur 4.2: Orientering av yaw, pitch och roll i förhållande till användarens riktning vid användning av Oculus Rift.

Oculus Rifts sensordata hämtas av scriptet ”OculusRenderer.pyw” med hjälp av ett importerat bibliotek som heter ”pyovr” vilket innehåller diverse funktioner för att upprätta en koppling, hämta sensordata från Oculus Rift samt olika funktioner för rendering till Oculus Rift. Sensordatan beskriver de olika rotationerna yaw, pitch och roll. Datan kan avläsas med ett givet intervall som definieras i koden och ju mindre intervall desto mer data flödar i programmen. För att undvika att hela tiden skicka

samma data, det vill säga om Oculus Rifts orientering ej har ändrats, implementeras ett tröskelvärde. Detta innebär att bara när Oculus Rift har vridits en viss mängd skickas sensordatan. Takten som programmen läser av sensordatan kan därför ökas utan att systemet översvämmas av redan känd information.

4.3.4 Filstruktur av den färdiga modulen

Den färdiga modulen i Jack består av flertalet filer som kommunicerar med varandra. I Figur 4.3 visas alla filer i systemet och även vilka som kommunicerar med vilka.



Figur 4.3: Filstrukturen hos den framtagna Jack-modulen och pythonprogrammet. Jack-modulen är den del som kommunicerar med Jack medan pythonprogrammet är den del som sköter Oculus Rift och knapptryckningar. Observera alltså att Jack och Oculus Rift inte är några delar av modulen eller programmet utan bara kommunicerar med dem. De mörka rutorna är mjukvara medan den ljusa är hårdvara.

De framtagna scriptens syften och funktioner är:

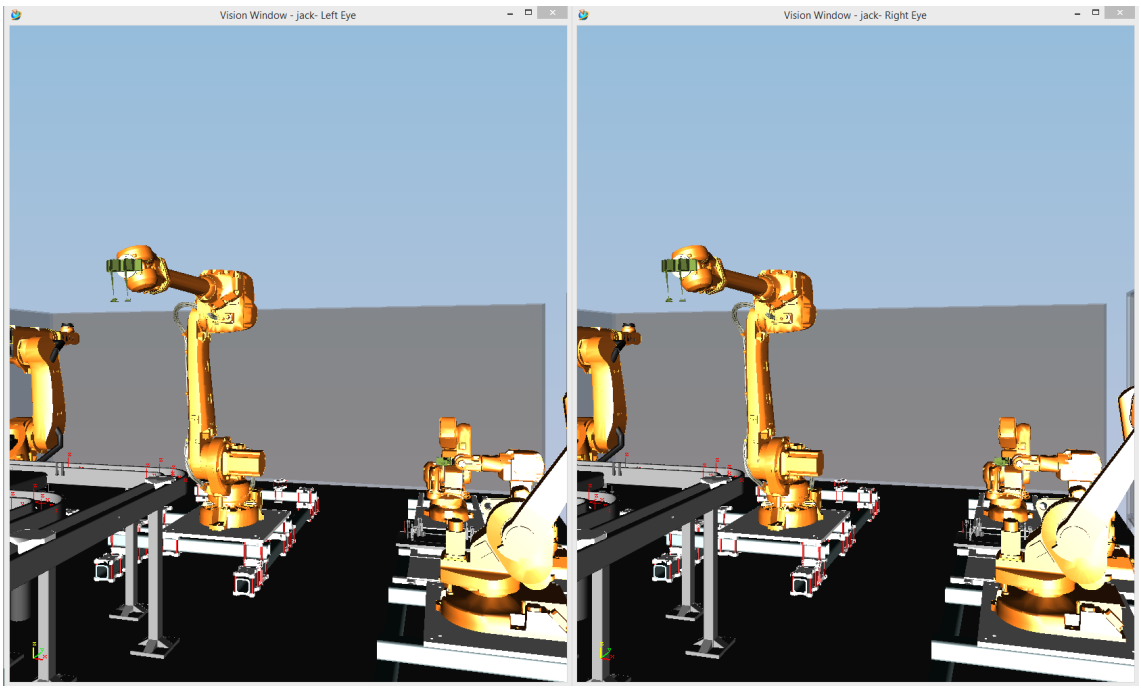
- **KeyRead.py** - Läser kontinuerligt knapptryckningar från tangentbordet.
- **ScreenCapture.py** - Läser av bilden som visas på datorns skärm och presenterar denna som en sträng av pixeldata.
- **MeDaHa.py** - (Message Data Handler) Skickar kontinuerligt meddelanden via socket till ”Oculus.tcl”.
- **OculusRenderer.pyw** - Det program som sammanfogar funktionaliteten från de ovanstående scripten samt hanterar renderingen till Oculus Rift och avläsningen av dess sensorer. Det är detta pythonprogram som körs utanför Jack.
- **Oculus.tcl** - Det script som kan läsas in som en modul i Jack. Hanterar när ett nytt event uppkommer i form av ny data från antingen Oculus Rift eller tangentbordet. Den kopplar sedan dessa event till metoder i ”JackLib” som skall utföras av Jack.
- **JackLib.py** - Ett bibliotek som kan skapa mänskliga objekt och få den att utföra förflyttningar, rotationer med mera. Dessa metoder blir anropade av ”Oculus.tcl” när nya event har uppkommit.
- **pkgIndex.tcl** - Innehåller information kring alla filer som anropas från ”Oculus.tcl” och var programmet kan hitta dem.

4.3.5 Kommunikation mellan script genom ”socket”

Eftersom Jack är låst till Python 2.6 och ”pyovr” måste köras med Python 2.7 eller nyare måste ”OculusRenderer.pyw” köras fristående från Jack vilket medför att en kommunikation mellan dem är nödvändig. Detta möjliggörs genom användandet av pythonbiblioteket ”socket”. Via ”socket”-kopplingen skickas sensordata samt kommandon om förflyttning och rotation från ”OculusRenderer.pyw” till ”Oculus.tcl”.

4.3.6 Visualisering och rendering till Oculus Rift

Från ett mänskligt objekt i Process Simulate kan två vyer för det mänskliga objektet erhållas, en för varje öga (se bild 4.4). Dessa två bilder renderas sedan som en bild till Oculus Rift och en 3D-presentation av produktionscellen uppnås.



Figur 4.4: De vyer som går att utvinna från det mänskliga objektet i Process Simulate. Bilden till vänster är det mänskliga objektets vänstra ögas vy som skall visas upp i det vänstra ögat i Oculus Rift. Den högra bilden är vyn från det mänskliga objektets högra öga och den visas upp för höger öga i Oculus Rift.

För att åstadkomma en 3D-representation av produktionscellen måste bilden på datorns skärm avbildas och sparas. Detta görs i scriptet ”ScreenCapture.py” som använder sig av biblioteket ”WxPython”. I ”WxPython” finns metoder för att spara ner skärmen som en bitmap. Denna bitmap konverteras till en sträng med pixeldata som skickas till ”OpenGL” (Open Graphics Library) som finns i ”OculusRenderer.pyw”.

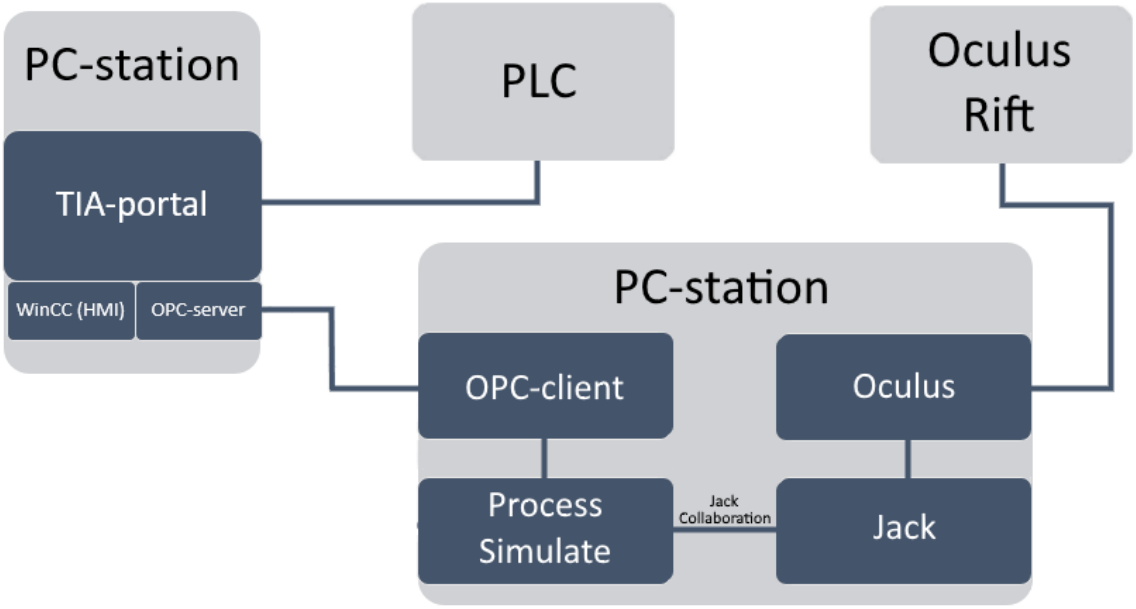
”OpenGL” är en specifikation som finns implementerad i ”pyovr” som där används för att rendera till Oculus Rift. Pixeldatan från ”ScreenCapture.py” tas in i ”OculusRenderer.pyw” och visas upp i Oculus Rift genom ”OpenGL”. Vänstra vyn från Process Simulate visas upp för vänstra ögat och den högra vyn visas upp för det högra ögat. Observera att dessa två fönster måste vara uppe vid användadet av

denna modul då scripten hela tiden avbildar det som visas på skärmen. Ändras det som visas på skärmen sker samma förändring för det som skickas till Oculus Rift.

”OpenGL” och ”WxPython” har olika referenssystem för var man skall börja rita ut- och läsa av pixlar. ”WxPython” läser av skärmen från övre högra hörnet medan ”OpenGL” ritar ut dessa pixlar från nedre vänstra hörnet. Detta medför att bilden i Oculus Rift blir både uppochned- och spegelvänd. Därför måste bitmapen roteras 180° och spegelvändas i ”ScreenCapture.py” innan den skickas till ”OculusRenderer.pyw”.

5
Fullständigt koncept

De avsnitt som behandlats tidigare krävs för att genomföra en virtual commissioning med Oculus Rift. Systemet vävs samman då alla delar interagerar med varandra. Hela systemet illustreras i Figur 5.1. Djupare beskrivning av de ingående komponenterna finns att hitta i respektive kapitel tidigare i rapporten.



Figur 5.1: Kommunikationsschema över hela systemet. Oculus är den modul som beskrivs i avsnitt 4.3. De ljusa rutorna är hårdvara och de mörka rutorna är mjukvara.

Användaren ges illusionen av att befinna sig i den verkliga produktionscellen då VR-glasögonen placeras på huvudet. Detta förstärks ytterligare genom möjligheten att röra sig runt med hjälp av datorns tangentbord men framför allt genom den head-tracking-funktion som implementerats.

Genom det visuella gränssnittet i WinCC har användare möjlighet att initiera ett önskat klosstorn. När valet är gjort startar processen i Process Simulate styrt av en PLC som skickar signaler, via en OPC-server, till den virtuella modellen. Användargränssnittet är uppbyggt på samma sätt som hos verkliga produktionssystem för att underlätta övergången till den virtuella modellen. Då simuleringen startats sköts styrningen av processerna autonomt via dubbelriktad signalkommunikation grundad i PLC-koden och logiken i det program som byggts upp.

Genom den visuella representationen i Oculus Rift har operatören möjlighet att skapa sig en verklighetsförankrad överblick över vad som sker då robotarna arbetar i produktionscellen. Säkerhetsåtgärder kan testas via de olika typer av fel som kan aktiveras under simuleringen. Antingen aktiveras felen av operatören själv eller någon utomstående för att testa operatörens förmåga att agera som om det skett i en verklig miljö. De framkallade felen initieras via samma användargränssnitt som tidigare och ger möjlighet för en operatör att testa på följande typer av situationer som skulle kunna ske i verkligheten:

- Vid simulering av en trasig givare kan i huvudsak felsökningsförmågan utvärderas, där operatören kan göra en bedömning på vad som bör åtgärdas genom att navigera sig runt i robotarnas område och göra en okulär undersökning då processen stannat upp. Genom displayen med larmtext i användargränssnittet informeras operatören med felmeddelanden, vilket vidare underlättar att kunna göra en bedömning.
- Ett annat scenario då roboten placerar klossar på felaktig position kan hanteras på liknande sätt, där operatören går över till manuell styrning via användargränssnittet och på så sätt kan föra tillbaka robotar till ursprungsläge för omstart.
- Vid simulering av elfelet där roboten rör sig okontrollerat ges vidare möjligheten att testa operatörens stressförmåga, där en önskad intuitiv reaktion är att nödstoppet aktiveras. Detta görs via användargränssnittet som därmed tillför ytterligare en dimension i interaktionen mellan den fysiska och virtuella världen.

För att ytterligare förstärka känslan av att befinna sig i en verklig miljö används ljusbommar och grindar för simulering av säkerhet, det vill säga då operatören navigerar in sitt virtuella jag i robotarnas område eller grindar står öppna så stoppas produktionen.

6
Diskussion

I detta avsnitt diskuteras de resultat som presenterats löpande i rapporten. Aspekterna produktionsutrustning, säkerhet, utbildning, hållbarhet, validering av kod, alternativa verktyg och vidareutveckling kommer behandlas och analyseras utifrån ett helhetsperspektiv.

6.1 Utvärdering av produktionsutrustning

Då en virtual commissioning genomförs skapas en möjlighet att utvärdera den utrustning som används i produktionscellen. I den virtuella modellen kan en användare byta ut eller förflytta produktionsutrustning för att se om detta skulle leda till förbättringar, försämringar eller ligga på samma nivå som tidigare system. Möjligheten till denna virtuella testning medför sannolikt stora fördelar i form av att utrustning kan utvärderas innan den införskaffas eller konstrueras.

Viktigt att ha i åtanke vid virtuell testning av ny utrustning är att modellerad utrustning inte alltid överensstämmer med verklig utrustning. Detta problem skulle emellertid kunna undvikas till stor del genom att tillverkare skapar egna Smart Components till de produkter de tillverkar, gärna efter någon typ av industriell standard. Vid genomförande av detta projekt fanns exempelvis inga färdigkonstruerade virtuella gripdon. Dessa har alltså skapats genom att efterlikna ett par verkliga gripdon så bra som möjligt. Det är dock mycket svårt för en enskild utvecklare att ha koll på ursprungsobjektets fulla funktionalitet vilket kan medföra att varje enskild utvecklare bygger sin Smart Component på olika sätt. Det som skulle krävas för att bygga en Smart Component som är representativ med verkligheten är en CAD-ritning som ger information om form, material och kinematik. Dessutom skulle det behövas data rörande in- och utsignaler som komponenten skall kunna hantera.

6.2 Utvärdering av säkerhet

Att genomföra en virtual commissioning med Oculus Rift av ett produktionssystem medför möjligheten att utvärdera fel i processer och brister i personsäkerheten redan innan ett fel eller en olycka inträffar på riktigt. En utvärdering kan göras via den PLC-styrda virtuella modellen, men med hjälp av Oculus Rift får användaren en chans att vistas i den virtuella miljön och i den söka efter processfel och säkerhetsbrister på ett mer intuitivt sätt. Istället för att befinna sig i den fysiska miljön, där operatören är utsatt för risker, kan produktionssystemet analyseras virtuellt. De processfel och säkerhetsbrister som lokaliseras med hjälp av virtual commissioning

med Oculus Rift kan då åtgärdas tidigare, vilket skapar förutsättningar för bättre processer och en säkrare arbetsplats.

Det finns även möjlighet att testa säkerhetsutrustning, i detta projekt är det utrustning i form av nödstopp, ljusbommar, säkerhetsgrindar och ljusfyrar. Det är vitalt att operatörer skall känna sig säkra på sina arbetsplatser och att ha ett verktyg som förebygger säkerhet skulle sannolikt ge ökat förtroende för ett företag, både inom och utanför organisationen. Viktigt att tänka på är emellertid att genomförande med virtual commissioning med Oculus Rift inte garanterar förebyggandet av verklig säkerhet och att det därför inte är rekommenderat att förlita sig fullständigt på den virtuella utvärderingen utan fortsatt utveckling.

6.3 Utbildningsmöjligheter

VR är ett verktyg som visat sig ha stora fördelar vid utbildning, just för att det ger en möjlighet för nyanställda operatörer att bekanta sig med maskiner och utrustning på ett säkert sätt utan att de behöver befinna sig i den verkliga produktionsmiljön. Något som däremot måste ifrågasättas innan det fullständigt går att lita på VR som operatörsträning är om träning i den virtuella världen ger samma förståelse för arbetet som träning i den verkliga miljön gör.

VR har testats som utbildningsmetod inom flera olika områden med mestadels positiva resultat vilket talar för att virtual commissioning med Oculus Rift skulle vara en utmärkt metod att använda vid utbildning. Samtidigt som operatörer kan träna fritt, utan press på att minsta fel kostar stora summor pengar för företaget, kommer störningar som uppkommer i produktion på grund av utbildning att reduceras. Produktionsflödet kommer således kunna operera mer kontinuerligt eftersom samtliga operatörer kommer behärska sitt arbete från den dag de börjar arbeta i produktion. Virtual commissioning med Oculus Rift skulle dessutom ge möjligheten att utbilda operatörer redan innan den fysiska miljön existerar alternativt om operatören skall utbildas i ett land för att sedan resa iväg och arbeta i ett annat.

Det finns också möjliga nackdelar med att använda VR som utbildningsmetod. Som beskrivs i avsnitt 4.1.3 har Oculus Rift-glasögonen i vissa fall visat sig framkalla simuleringssjuka hos användaren. Detta kan medföra att användaren mår dåligt och helt enkelt inte kan prestera på sin maximala nivå. Det är svårt att idag veta precis hur stora följderna av denna simuleringssjuka skulle kunna bli, men det är absolut en faktor som bör tas i beaktande innan VR som utbildningsmetod börjar användas storskaligt inom olika organisationer.

Oculus Rift försvårar användarens kroppskontroll vilket skulle kunna påverka upplärningen negativt. En faktor som skulle kunna vara den som skapar denna svårighet för kroppskontroll är att det med Oculus Rift på huvudet inte ges samma uppfattning om omgivningen vilket då skulle kunna medföra svårigheter att hålla balansen och problem vid kontroll av kroppsrörelser. Överdriven användning av Oculus Rift skulle också kunna medföra monotona rörelsemönster, inte minst för nacken. I och med att simuleringssjuka och enformiga rörelser kan bidra till psykisk respektive fysisk ohälsa finns risk för förhöjd nivå av sjukskrivningar. Detta skulle sannolikt överbrygga de positiva effekter VR skulle ge inom utbildning.

6.4 Hållbarhetsaspekter av virtual commissioning

Det finns inget direkt mått på hur noggrant en hållbar version av den virtuella produktionscellen måste efterlikna den verkliga produktionscellen. Skall processer provköras kan det räcka med att processens berörda komponenter är inkluderade och placerade någorlunda likt verkligheten. Skall däremot robotars rörelsemönster på detaljnivå eller planering av golvyta analyseras måste även omkringliggande komponenter och andra faktorer som kan tänkas vara i vägen finnas med i den virtuella modellen. Hur lik den virtuella modellen måste vara den verkliga cellen beror alltså på hur, och inom vilka områden, den virtuella modellen skall användas.

I detta projekt är likheten med sådan noggrannhet att produktionscellens funktion uppfylls av den virtuella modellen vilket gör det möjligt för användare att undersöka hur väl produktionscellens processer fungerar och göra eventuella justeringar av dessa. Det går till exempel att upptäcka produktivitetssänkande faktorer såsom flaskhalsar och därför med hjälp av den virtuella modellen optimera processerna i den verkliga produktionscellen.

Ett par avvägningar måste göras för att avgöra huruvida en virtual commissioning är hållbar att genomföra. Dels kan en väldigt komplicerad process behöva en virtuell modell med en noggrannhet på mikrometernivå och då är det möjligt att den tid som måste läggas på att modellera verkligheten inte väger upp för de fördelar modellen bär med sig. Däremot kan en process som inte är så känslig för noggrannhet vara väldigt bra att skapa virtuellt för att utvärdera olika faktorer utan att stanna processer eller till och med innan den skapas fysiskt.

När visualisering med Oculus Rift inkluderas i virtual commissioning behöver även utvecklingskostnaderna kring detta tas i beaktning. En styrka hos den framtagna modulen för Oculus Rift är att den inte är modellspecifik utan kan användas på alla virtuella modeller skapade i Process Simulate. Den behöver alltså inte skapas igen utan kan återanvändas vilket medför att arbete bör fokuseras på att förbättra den existerande modulen och inte skapa en ny.

Genomförande av virtual commissioning medför en fördel i att en fabrik eller produktionscell kan skapas och optimeras innan den byggs upp fysiskt. I och med att fabriken/produktionscellen utvärderas virtuellt kommer inte lika många problem uppstå vid själva byggandet vilket sparar både material och tid. Detta minskar därmed miljöpåverkan eftersom felbyggen och överflödig materialåtgång bidrar mer negativt på miljön än vad energiförbrukningen vid genomförande av en virtual commissioning gör.

Ur ett ekonomiskt perspektiv är det svårare att dra några slutsatser om fördelarna med virtual commissioning. Tillverkare har sett att virtual commissioning bidrar till en reducering av arbetstid med upp till 30%. Det är dock svårt att vara helt säker på att det väger upp för allt arbete och alla kostnader som krävs för genomförandet av en virtual commissioning.

6.5 Validering av PLC-kod

En av styrkorna med att genomföra virtual commissioning är som tidigare nämnt möjligheten att kunna testa verklig PLC-kod i en virtuell miljö. Genom användande av samma styrenhet kan det skapas goda möjligheter att enkelt växla mellan verklig och virtuell miljö för att därmed uppnå en gemensam utveckling och optimering av koden. För att detta skall kunna genomföras effektivt i praktiken krävs att produktionssystemet i modellen kan styras med samma signaler från PLC:n som produktionssystemet i verkligheten.

Projektets PLC-kod är direkt anpassad till hur den virtuella modellen är uppbyggd med operationer och sensorer och alltså inte efter verklig utrustning, vilket leder till att koden inte kan valideras då den inte är är komplett i jämförelse med en verklig cell. I befintliga utrustningar hos företag som vill utföra en virtual commissioning finns oftast ett färdigt PLC-projekt, vilket gör att simuleringen speglas från den existerande koden istället för att skapa ny kod. Fördelen blir att koden blir ett riktmärke för hur simuleringen ska gå till, samt att signalerna kan användas till grund för vilka signaler som måste finnas i simuleringen.

Förutsatt att ovanstående uppfylls kan validering av kod i den virtuella modellen generera flertalet fördelar vid upprättande av produktionssystem. Det går att testa ändringar i styrkod och felsöka processer virtuellt, vilket leder till reducerad risk för fysiska haverier men framförallt möjliggör utvärdering av ett produktionssystem utan att behöva störa inblandade processer.

6.6 Alternativa verktyg

Under projektets gång har ett stort antal programvaror och hårdvaror använts där vissa av dessa har varit låsta för detta projektet, andra har vidareutvecklats från tidigare projekt och vissa själva har valts. Under projektets gång har dessa verktyg testats och deras styrkor och svagheter utvärderats. I detta kapitel kommer alternativ för dessa verktyg ges.

6.6.1 Alternativa simuleringsprogram

Process Simulate är inte det enda program som kan användas vid genomförande av en virtual commissioning. Fördelen i Process Simulate ligger i den enkla uppkopplingen med Jack eftersom båda är program som utvecklas av Siemens. Det finns dock en möjlighet att andra program för mänsklig simulering finns tillgängliga och med fördel då kanske används med andra program än Process Simulate. Om det finns andra program som kan lösa samma problem skulle dessa kunna vara värda att testa eftersom Process Simulate även har negativa sidor. I Process Simulate finns bland annat flertalet buggar som ibland ställer till stora problem. Exempelvis är programmets inbyggda simuleringsfunktion CEE (Cyclic Event Evaluator) inte pålitlig och simulerar stundtals bara vissa delar. När samma modell istället kopplas till en PLC fungerar allt som väntat, vilket tyder på att det är CEE som är bristfällig. I andra fall fungerar det däremot åt motsatt håll där funktioner ger önskat resultat i CEE men fallerar då en PLC kopplas på, trots korrekt styrkod. Eftersom projektet avgränsats

till att använda givna programvaror så är detta något som borde undersökas för att maximera resultatet av en virtual commissioning med Oculus Rift.

Programvaror som orsakar mindre problem under utvecklingsarbetet skulle bidra till värdet av att genomföra virtual commissioning med Oculus Rift eftersom projektet skulle kunna genomföras på kortare tid och därmed öka hållbarheten ur ett ekonomiskt perspektiv. Viktigt att tänka på är dock att modulen för Oculus Rift inte är byggd för andra simuleringsprogram men att det sannolikt skulle gå att återanvända delar av Oculus-koden.

6.6.2 Alternativa kommunikationsverktyg mellan PLC och Process Simulate

Kommunikationen mellan PLC och Process Simulate kan ske på flera olika sätt men användandet av OPC-standard är väl etablerad lösning av industriellt signalutbyte och fyller detta ändamål inom virtuell simulering. Problemet med OPC DA är vägen genom datorernas brandväggar och säkerhetsprotokoll vilket förhindrar fri kommunikation, men som också är en nödvändighet då datorernas integritet är viktig att hålla skyddad från yttre angrepp och skadliga programvaror.

Vill man undvika att använda en tunneller som lösning på detta problem kan andra OPC-standarder tillämpas, bland annat OPC UA vilket istället kommunicerar via webservrar. Fördelen med OPC DA gentemot denna är realtidsuppdateringen som tidigare nämnts i avsnitt 3.1.1.

6.6.3 Alternativa HMD:er för visualisering

Utöver Oculus Rift DK2 finns det i nuläget ett flertal andra VR-glasögon, dessa är HTC Vive, Samsung Gear VR, Google Cardboard samt konsumentversionen av Oculus Rift. Samtliga av dessa skulle kunna användas för att uppnå ett liknande eller möjligtvis bättre resultat av visualiseringen så länge de klarar av att läsa av användarens huvudrörelser och visa bilden på ett verklighetstroget sätt. Slutligen är det nödvändigt att utvecklare har möjlighet till att läsa sensordata och rendera bild till HMD:n.

6.7 Vidareutveckling av det framtagna konceptet

Eftersom studier visar att utvecklingen av virtual commissioning först tagit ordentlig fart under 2000-talet finns det stora möjligheter till fortsatt utveckling, speciellt inom virtual reality-området där användaren kan befinna sig i den virtuella miljön.

6.7.1 Interaktion mellan mänskliga objekt och modellens processer

För att utöka upplevelsen av mänsklig interaktion med den virtuella produktionscellen kan den virtuella modellen utvecklas så att mänskliga objekt kan interagera med processerna. Det mänskliga objektet skulle kunna stoppa processer genom att

trycka på ett virtuellt nödstopp. Eftersom det mänskliga objektets perspektiv återges genom Oculus Rift skulle användaren då få en känsla av hur det fungerar i verkligheten när denne stoppar en process som går fel.

Det skulle även vara intressant att implementera flera mänskliga objekt så att en flyttar ner klossar på transportbandet medan en annan tar hand om de färdiga klosstornen för att kunna analysera hur samarbetet mellan människa och maskin fungerar i produktionscellen. Detta skulle vidare kunna användas för att låta flera operatörer befinna sig och verka i den virtuella modellen samtidigt.

6.7.2 Simulering av fysisk utrustning och automatisk kodgenerering till PLC

Ett stort problem för simulering av befintlig utrustning med färdig PLC-kod är PLC-projektets hårdvarukonfiguration som behöver vara identisk med verklig utrustning. Eftersom hårdvaran inte finns tillgänglig virtuellt medför det att PLC:n inte kan kompilera. PLC:n måste luras att tro att det finns utrustning kopplad till den trots att det inte finns. Det finns utrustning och moduler som kan imitera hårdvara men inget som detta projekt har hunnit beröra.

Ett alternativ att lösa kompileringsproblemet är att konfigurera om hårdvaran och ta bort samtlig utrustning som inte finns tillgänglig fysiskt, samt adressera om den utrustningens signaler till antingen PLC:n eller hårdvara som finns tillgänglig för den PLC-uppsättning som ska utföra en virtual commissioning. Problemet blir då att denna konfiguration senare inte kan direkt överföras tillbaka till den verkliga produktionsutrustningen då konfigurationen inte längre stämmer.

För att en användare av den virtuella cellen ska kunna interagera med cellen kan däremot ytterligare utrustning behövas tilläggas. Det är främst nödstopp och säkerhetsutrustning som borde finnas till tillgänglig för en operatör men även teknik som möjliggör att användare kan trycka på virtuella knappar och påverka cellen med händer hade varit väldigt fördelaktigt.

Det är i dagsläget inte anpassat för att en användare med Oculus Rift att själv kunna styra cellen via det visuella gränssnittet då det inte syns i den virtuella världen. En klar fördel hade varit om Process Simulate kunde avspegla ett användargränssnitt inne i den virtuella cellen som skulle kunna ge operatören nödvändig information. Det hade förstärkt användarupplevelsen och därför en väldigt bra sak att vidareutveckla till framtida användning.

Då utvecklingen av PLC-användadet i Industri 4.0 lägger grund för automatisk kodgenerering tack vara standardiserade bibliotek av färdiga funktionsblock och utveckling av överordnade system i kombination med högspråk finns det inget som tyder på att PLC:n inte skulle kunna använda detta för virtuell simulering likaså.

6.7.3 Förbättring av den visuella presentationen

I nuläget blir användarupplevelsen negativt påverkad av att bilden som visas i Oculus Rift flyter runt i synfältet när användaren vrider på huvudet. Det beror på att de inbyggda funktionerna för rendering till Oculus Rift försöker kompensera för användarens huvudrörelser trots att det inte behövs då bilden alltid ska renderas

på samma ställe i Oculus Rifts synfält. Ett första steg i vidareutvecklingen kring Jack-modulen för Oculus Rift bör därför vara att åtgärda detta.

6.7.4 Förstärkning av den virtuella upplevelsen

Det är i dagsläget svårt att motivera fördelar med att använda Oculus Rift vid genomförande av en virtual commissioning. Användarupplevelsen blir mer verklig men det går inte att dra några konkreta slutsatser om hur virtual commissioning förbättras. Det krävs vidareutveckling och implementering av nya funktioner för att förbättra den virtuella upplevelsen. Det skulle kunna ske genom att skapa ett rum där kameror och sensormattor installeras för att följa operatören i realtid. Användaren skulle då ges en möjlighet att med sina egna kroppsrörelser förflytta sig i och aktivt påverka den virtuella miljön.

7
Slutsatser

Utifrån den virtual commissioning med Oculus Rift som genomförts kan följande slutsatser dras:

- PLC-kod kan testas och valideras virtuellt i den modellerade produktionscellen.
- HMI:er går att testa genom att styra den virtuella modellen via det utvecklade gränssnittet.
- Det går att testa och validera viss säkerhetsutrustning (nödstopp, ljusbommar, säkerhetsgrindar och ljusfyrar).
- Befintlig eller ny utrustning kan utvärderas virtuellt.

Till följd av att ovanstående punkter går att testa och validera kan också dess funktion säkerställas. Det går dock inte att dra några konkreta slutsatser om annan säkerhetsutrustning än de som nämnts ovan. Detta då enbart nödstopp, ljusbommar, säkerhetsgrindar och ljusfyrar är implementerade i den virtuella modellen.

Litteraturförteckning

[1] G. Reinhart, G. Wünsch. (2007, December). "Economic application of virtual commissioning to mechatronic production systems". *Production Engineering*. 1(4). s. 371-379. [Online]. Åtkomlig: `http://link.springer.com/article/10.1007/s11740-007-0066-0/fulltext.html`

[2] "SSYX02-16-06 Virtual Commissioning med Oculus Rift". *Chalmers*. (2015). [Online]. Åtkomlig: `https://www.chalmers.se/sv/institutioner/s2/Var-Utbildning/kandidatutbildning/Sidor/SSYX02-16-06.aspx`. [Hämtad: 26 jan, 2016].

[3] *KUKA Systems do Brazil: Leading automotive line builder delivers state-of-the-art robotics production lines*. Siemens Product Lifecycle Management Software Inc. (2013). [Online]. Åtkomlig: `http://www.plm.automation.siemens.com/pub/case-studies/32446?resourceId=32446`. [Hämtad: 26 jan, 2016].

[4] Petter Falkman. (2014). Virtual commissioning of manufacturing stations including PLC logics. VirtCom. [Forskningsansökan].

[5] M. Vorderwinkler, T. Eder, R. Steringer, M. Schleicher. "An Architecture for SoftComissioning – Verifying Control Software by Linking Discrete Event Simulators to Real World Control Systems". i *Proc of ESM'99 – 13th European Simulation Multiconference*. Warsaw. Juni 1-4, 1999. ss. 191-198.

[6] M. Fratczak, P. Nowak, T. Klopot, J. Czeczot, S. Bysko, B. Opilski. "Virtual commissioning for the control of the continuous industrial processes — Case study". i *Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on*. (2015, Augusti). [Online]. ss. 1304. Åtkomlig: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7284021`. [Hämtad: 12 maj, 2016].

[7] H. Krause. (2007). "Virtual commissioning of a large LNG plant with the DCS 800XA by ABB". i *6th EUROSIM Congress on Modelling and Simulation*. Ljubljana, Slovenien. [Online]. Åtkomlig: `http://www.secolon.de/P172.pdf`. [Hämtad: 12 maj, 2016].

[8] S. Seidel, U. Donath, J. Haufe. (2012, December). "Towards an integrated simulation and virtual commissioning environment for controls of material handling systems." i *Proceedings of the winter simulation conference*. s. 252. Winter Simulation Conference. [Online]. Åtkomlig: `http://dl.acm.org/citation.cfm?id=2430099`. [Hämtad: 12 maj, 2016].

[9] S. Makris, G. Michalos, G. Chryssolouris. (2012). "Virtual Commissioning of an Assembly Cell with Cooperating Robots". *Advances in Decision Sciences*. vol. 2012. Article ID 428060. [Online]. Åtkomlig: `doi:10.1155/2012/428060`. [Hämtad: 12 maj, 2016].

[10] N. Koch, W. Newhauser. (2005). "VIRTUAL COMMISSIONING OF A TREATMENT PLANNING SYSTEM FOR PROTON THERAPY OF OCULAR CANCERS". *Radiation Protection Dosimetry*. 115(1–4). s. 159–163. [Online]. Åtkomlig: http://rpd.oxfordjournals.org/content/115/1-4/159.full.pdf+html. [Hämtad: 12 maj, 2016].

[11] M. Björklund, M. Engberg, M. Kastebo, A. Lu, T. Ågren, M. Östman. Virtual Commissioning with Oculus Rift. Kandidatuppsats. Chalmers Tekniska Högskola. Instutitionen för Signaler och System. (2015, Juni). [Online]. Åtkomlig: http://studentarbeten.chalmers.se/publication/219266-virtual-commissioning-with-oculus-rift. [Hämtad: 12 maj, 2016].

[12] S. Robinson. (2005, Juni). "Discrete-Event Simulation: From the Pioneers to the Present, What Next?". *The Journal of the Operational Research Society*. 56(6). s. 619-629. [Online]. Åtkomlig: http://www.jstor.org/stable/pdf/4102035.pdf?_=1459495867469

[13] *RobotStudio 5: Industrial Software Products*. (2013). [Online]. Åtkomlig: https://library.e.abb.com/public/b755e615bec21b52c1257c32004fa70f/PR10296EN_R3.pdf. [Hämtad: 30 mars, 2016].

[14] *Applied AutoMod: Simulation and modeling*. Applied Materials, Inc. (2012). [Online]. Åtkomlig: http://www.appliedmaterials.com/files/automation_software_resources/AutoModDatasheet.pdf. [Hämtad: 30 mars, 2016].

[15] *Process Simulate: Manufacturing process verification in powerful 3D environment*. Siemens Industry Software AB. (2011). [Online]. Åtkomlig: https://www.plm.automation.siemens.com/se_se/products/tecnomatix/manufacturing-simulation/robotics/process-simulate.shtml#lightview%26uri=tcm:741-80351%26title=ProcessSimulate-TecnomatixFactSheet-7457%26docType=pdf. [Hämtad: 5 feb, 2016].

[16] *Process Designer: Powerful 3D environment for manufacturing process planning*. Siemens Industry Software AB. (2011). [Online]. Åtkomlig: https://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-planning/process-design/designer.shtml#lightview%26uri=tcm:1023-4941%26title=ProcessDesigner-TecnomatixFactSheet-7456%26docType=pdf. [Hämtad: 5 feb, 2016].

[17] SPEAR Workstation. *Virtual factory tutorial - A step by step development of a digital production cell*. 2015. s. 57.

[18] J.K. Liker, D. Meier. *The Toyota way fieldbook: a practical guide for implementing Toyota's 4Ps*. New York: McGraw-Hill. 2006. s. 9-10.

[19] ABB - *Industrial Robots*. [Online]. Åtkomlig: http://new.abb.com/products/robotics/industrial-robots. [Hämtad 6 Maj 2016].

[20] W. Bolton. *Programmable logic controllers*. Oxford: Newnes. 2015. s. 5-7.

[21] F. Basile, P. Chiacchio, D. Gerbasio. (2013, Oktober). *On the implementation of industrial automation systems based on PLC*. IEEE transactions on automation science and engineering. 10(4). s. 990. [Online]. Åtkomlig: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6381490.

[22] J. Lee, B. Bagheri, H.A Kao. ”Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial Informatics”. i *Conference on Industrial Informatics*. Porto Alegre. (2014). s. 1-2 [Online]. Åtkomlig: https://www.researchgate.net/profile/Behrad_Bagheri/publication/266375284_Recent_Advances_and_Trends_of_Cyber-Physical_Systems_and_Big_Data_Analytics_in_Industrial_Informatics/links/542dc0100cf27e39fa948a7d.pdf.

[23] Y. Debbağ, E. N. Yilmaz. ”Internet based monitoring and control of a wind turbine via PLC”. i *Smart Grid Congress and Fair (ICSG), 2015 3rd International Istanbul*. Istanbul. (2015). s. 1-5. [Online]. Åtkomlig: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7354935&isnumber=7354913.

[24] Y. Shimanuki. ”OLE for process control (OPC) for new industrial automation systems” i i *Systems, Man, and Cybernetics, 1999. IEEE SMC ’99 Conference Proceedings. 1999 IEEE International Conference on*. Tokyo. (1999). s. 1048-1050. [Online]. Åtkomlig: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=816721&isnumber=17619.

[25] H. Carlsson, B. Svensson, F. Danielsson, B. Lennartson. (Maj 2012). *Methods for Reliable Simulation-Based PLC Code Verification*. IEEE Transactions on Industrial Informatics. 8(2). s. 267-278. [Online]. Åtkomlig: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6121945&isnumber=6179486.

[26] Object Management Group (OMG). ”Data acquisition from industrial systems specification”. Ver.1.1, 2005.

[27] V.F. Wolfe, L.C. DiPippo, R. Ginis, M. Squadrito, S. Wohlevera, I. Zykh, and R. Johnston. ”Real-time CORBA”. 1997. s. 148-157.

[28] Siemens AG. (2014, December). *Siemens SIMATIC - S7-1500: CPU 1517F-3 PN/DP (6ES7517-3FP00-0AB0) - Manual*. Nürnberg, Tyskland. [Online]. Åtkomlig: https://support.industry.siemens.com/cs/attachments/102958163/s71500_cpu1517f_3_pn_dp_manual_en-US_en-US.pdf?download=true.

[29] Siemens AG. (2014, Maj). *Siemens SIMATIC - S7-1500: Getting Started ”TIA Portal V13”*. Nürnberg, Tyskland. [Online]. Åtkomlig: http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/documents/EN/software_complete_en.pdf.

[30] *SIMATIC WinCC: Process visualization with Plant Intelligence*. Nürnberg: Siemens AG. (2012). [Online]. Åtkomlig: http://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-wincc_en.pdf. [Hämtad 2016-03-23].

[31] W.R. Sherman, A.B. Craig. ”Understanding Virtual Reality: Interface, Application, and Design”. USA, San Francisco: Morgan Kaufmann Publishers. 2003. s. 6.

[32] M. Mihelj, D. Novak, S. Beguš. (2014). *Virtual Reality Technology and Applications*. s. 8-11. [Online]. Åtkomlig: http://link.springer.com/book/10.1007%2F978-94-007-6910-6.

[33] G. Lawson, D. Salanitri, B. Waterfield. (2016, Mars). ”Future directions for the development of virtual reality within an automotive manufactu-

rer". *Applied Ergonomics*. [Online]. 53(B). s. 323. Åtkomlig: http://ac.els-cdn.com/S0003687015300260/1-s2.0-S0003687015300260-main.pdf?_tid=c80792dc-f71e-11e5-ad50-00000aacb35&acdnat=1459414908_d972b3a80967fe247816c58f48641784.

[34] R. Riener, M. Harders. (2012). *Virtual Reality in Medicine*. s. 3-7. [Online]. Åtkomlig: http://link.springer.com/book/10.1007%2F978-1-4471-4011-5.

[35] M. Gonzalez-Franco, J. Cermeron, K. Li, R. Pizarro, J. Thorn, P. Hannah, W. Hutabarat, A. Tiwari, P. Bermell-Garcia. (2016, Februari). "Immersive Augmented Reality Training for Complex Manufacturing Scenarios". [Online]. Åtkomlig: http://arxiv.org/pdf/1602.01944v1.pdf.

[36] M. Norrby, C. Grebner, J. Eriksson, J. Boström. (2015). Molecular Rift: Virtual Reality for Drug Designers. *Journal of chemical information and modeling*. 55(11). s. 2475-2484. ACS Publications. [Online]. Åtkomlig: http://pubs.acs.org/doi/abs/10.1021/acs.jcim.5b00544.

[37] P. Epure, C. Gheorghe, T. Nissen, L.O. Toader, A.N. Macovei, S.S.M. Nielsen, D.J. Rosengren Christensen. (2014). The Effect of Oculus Rift HMD on postural stability. ICDVRAT. [Online]. Åtkomlig: http://vbn.aau.dk/files/192688714/Comment_The_effect_of_Oculus_Rift_HMD_on_postural_stability.docx.

[38] S.R. Serge, J.D. Moss. (2015, September). Simulator Sickness and the Oculus Rift A First Look. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 59(1). s. 764. SAGE Publications. [Online]. Åtkomlig: http://pro.sagepub.com/content/59/1/761.full.pdf+html.

[39] Jack. (2011). Siemens Industry Software. [Online]. Åtkommlig: https://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/human-ergonomics/jack.shtml#lightview-close. [Hämtad: 6 Maj, 2016].

A

Produktlista

Nedan följer en tabell över de mjukvaror (MV) och hårdvaror (HV) som ingår i projektet.

Komponent	**Version**	**MV/HV**	**Område**
Jack	8.2, 64-bit	MV	Visualisering
Oculus Rift	DK2	HV	Visualisering
Python	2.7.11, 32-bit	MV	Visualisering
pyovr	0.8.0002, 32-bit	MV	Visualisering
pywin32	214, 32-bit	MV	Visualisering
pyHook	1.5.1, 32-bit	MV	Visualisering
wxPython	3.0, 32-bit	MV	Visualisering
OpenGL	3.1.1, 32-bit	MV	Visualisering
OpenGL Accelerate	3.1.1, 32-bit	MV	Visualisering
Oculus Runtime	0.8.0.0	MV	Visualisering
Catia V5	R2013	MV	Modellering
Process Simulate	12.0TR1	MV	Modellering
Process Designer	12.0TR1	MV	Modellering
MatriconOPC Tunneller	5.0	MV	Styrning
WinCC	Runtime Advanced	MV	Styrning
TIA-Portal	V13 SP1	MV	Styrning
Siemens SIMATIC PLC	CPU 1517-3 PN/DP	HV	Styrning

Tabell A.1: De hårdvaror och mjukvaror som ingår i systemet.

B

Signaler och operationer

Nedan finns samtliga signaler och operationer listade.

KOMPONENT	KOMMENTAR	BETECKNING	SIGNAL	PATH	OPERATION/BETYDELSE
Brioklossar		***K***			
Grön		K_green	*K_green*		Flyttar kloss ner till transportplatta
Blå		K_blue	*K_blue*		Flyttar kloss ner till transportplatta
Röd		K_red	*K_red*		Flyttar kloss ner till transportplatta
Gul		K_yellow	*K_yellow*		Flyttar kloss ner till transportplatta
Flyttplatta		***FP***			
Flyttplatta	*Platta där tornen byggs*	FP	*FP_pos*		Placerar FP i rätt läge vid robotar
Robotar	*Operationer som kallas på via robotprogram nedan*	***R***			
IRB4600	*Stor robot*	R46	*R46_to_R1*	*10*	Flyttar två klossar till R1
			R46_to_R2	*20*	Flyttar två klossar till R2
			R46_FP_pos1	*30*	Flyttar FP med färdigt klosstorn till position 1
			R46_FP_pos2	*40*	Flyttar FP med färdigt klosstorn till position 2
			R46_FP_pos3	*50*	Flyttar FP med färdigt klosstorn till position 3
			R46_Elfel	*60*	Simulerar elfel och går bananas
			R46_Felplacering	*70*	Placerar klossar på fel ställe från bandet
			R46_to_HOME	*80*	Kör roboten till hemmaläge.
IRB140(1)	*Liten robot närmast bandet*	R1	*R1_T1_1*	*10*	Använder kloss för att bygga del av klosstorn 1. Operation 1.
			R1_T1_2	*20*	Använder kloss för att bygga del av klosstorn 1. Operation 2.
			R1_T2_1	*30*	Använder kloss för att bygga del av klosstorn 2. Operation 1.
			R1_T2_2	*40*	Använder kloss för att bygga del av klosstorn 2. Operation 2.
			R1_T3_2	*50*	Använder kloss för att bygga del av klosstorn 3. Operation 2(1).
			R1_T3_4	*60*	Använder kloss för att bygga del av klosstorn 3. Operation 4(2).
			R1_to_HOME	*70*	Kör roboten till hemmaläge.
IRB140(2)	*Liten robot närmast väggen*	R2	*R2_T1_1*	*10*	Använder kloss för att bygga del av klosstorn 1. Operation 1.
			R2_T1_2	*20*	Använder kloss för att bygga del av klosstorn 1. Operation 2.
			R2_T2_1	*30*	Använder kloss för att bygga del av klosstorn 2. Operation 1.
			R2_T2_2	*40*	Använder kloss för att bygga del av klosstorn 2. Operation 2.
			R2_T3_1	*50*	Använder kloss för att bygga del av klosstorn 3. Operation 1(1).
			R2_T3_3	*60*	Använder kloss för att bygga del av klosstorn 3. Operation 3(2).
			R2_to_HOME	*70*	Kör roboten till hemmaläge.

Robotprogram	*Används för att kontrollera robotar med signaler*				
	Program för R46	R46	*R46_startProgram*		Startar valt program från *R_programNumber*
			R46_programNumber		Sätter vilket program som skall köras. Skicka in Path för att välja
			R46_emergencyStop		Stoppar roboten direkt. För att starta igen så används startProgram och roboten
					går då till utgånsläge först och kör sedan om programmet från start.
			R46_programPause		Stoppar roboten direkt. För att starta igen så skickas samma signal igen.
			R46_programEnded		Skickar signal då programmet är klart.
			R46_at_HOME1		Skickar signal då roboten är i hemmaläge
	Program för R1	R1	*R1_startProgram*		
			R1_programNumber		
			R1_emergencyStop		
			R1_programPause		
			R1_programEnded		
			R1_at_HOME1		
	Program för R2	R2	*R2_startProgram*		
			R2_programNumber		
			R2_emergencyStop		
			R2_programPause		
			R2_programEnded		
			R2_at_HOME1		
Transportband					
Raksträckor	*Alla raka delar på bandet. Där nummer 1 motsvaras*	Rak	*Rak_1_start*		Startar bandet så att skids rör sig framåt
	av den första efter att klossar läggs på.		*Rak_1_stop*		Stoppar bandet så att skids stannar
			Rak_2_start		
			Rak_2_stop		
			Rak_3_start		
			Rak_3_stop		
	Raksträcka som slutar vid upphämtning av klossar		*Rak_4_start*		
			Rak_4_stop		
			Rak_5_start		
			Rak_5_stop		
			Rak_6_start		
			Rak_6_stop		
			Rak_7_start		
	Raksträcka som slutar vid utplacering av klossar		*Rak_7_stop*		
Kurvor	*Alla böjda delar på bandet. Där nummer 1 mostvaras*	Kurva	*Kurva_1_start*		
	av den första efter att klossar läggs på.		*Kurva_1_stop*		
			Kurva_2_start		
			Kurva_2_stop		
			Kurva_3_start		
			Kurva_3_stop		
			Kurva_4_start		
			Kurva_4_stop		
			Kurva_5_start		
			Kurva_5_stop		
Utrustning					
Glasdörr	*Glasdörren in till cellen*	glasgate	*glasgate_open*		Öppnar dörr
			glasgate_close		Stänger dörr
Ljussignal	*Lampor som visar status på cellen*	panelview	*panelview_to_RUN*		Lampa blir grön
			panelview_to_FAULT		Lampa blir röd
			panelview_to_READY		Lampa blir gul
Sensorer		***S***			***Alla sensorer ger 1 för triggning och 0 annars***
SK_1_startpos	*Sensor för klossar vid startpos*	SK_1_startpos	*SK_1_startpos*		
SK_2_startpos	*Sensor för klossar vid startpos*	SK_2_startpos	*SK_2_startpos*		
SK_1_pickpos	*Sensor för klossar vid upphämtningspos*	SK_1_pickpos	*SK_1_pickpos*		
SK_2_pickpos	*Sensor för klossar vid upphämtningspos*	SK_2_pickpos	*SK_2_pickpos*		
SK_1_R1	*Sensor för klossar vid R1*	SK_1_R1	*SK_1_R1*		
SK_2_R1	*Sensor för klossar vid R1*	SK_2_R1	*SK_2_R1*		
SK_1_R2	*Sensor för klossar vid R2*	SK_1_R2	*SK_1_R2*		
SK_2_R2	*Sensor för klossar vid R2*	SK_2_R2	*SK_2_R2*		
SBP_startpos	*Sensor för tranposrtplatta vid startpos*	SBP_startpos	*SBP_startpos*		
SBP_pickpos	*Sensor för transportplatta vid upphämtningspos*	SBP_pickpos	*SBP_pickpos*		
SFP_byggpos	*Sensor för flyttplatta vid byggpos*	SFP_byggpos	*SFP_byggpos*		
SFP_pos1	*Sensor för flyttplatta vid position 1 för avlämning*	SFP_pos1	*SFP_pos1*		
SFP_pos2	*Sensor för flyttplatta vid position 2 för avlämning*	SFP_pos2	*SFP_pos2*		
SFP_pos3	*Sensor för flyttplatta vid position 3 för avlämning*	SFP_pos3	*SFP_pos3*		
S_gate	*Sensor vid glasgrind*	S_gate	*S_gate*		
S_jack	*Sensor med funktion som förenklad ljusbom*	S_jack	*S_jack*		
	Sensor med funktion som förenklad ljusbom	S_jack_2	*S_jack_2*		

Figur B.1: Signaler och operationer

C

Process Simulate

C.1 Konvertera filer

I. Gå till Startmenyn

II. Öppna programmappen Tecnomatix

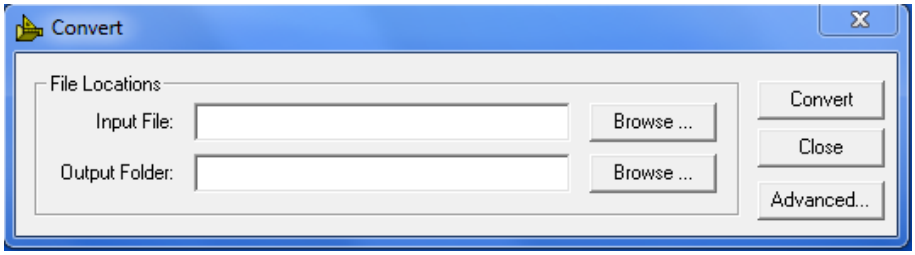
III. Välj eMPower

IV. Klicka på Importer.exe

V. I Input File, anger man önskad CAD-fil.

VI. I Output Folder väljer du den mapp där de konverterade filerna ska ligga. Se Figur C.1

    Filerna måste ligga under systemroot-mappen för att komma åt dem i ett senare skede. Det format som är garanterat att fungera är igs. De konverterade filerna blir i formatet .co (Component).



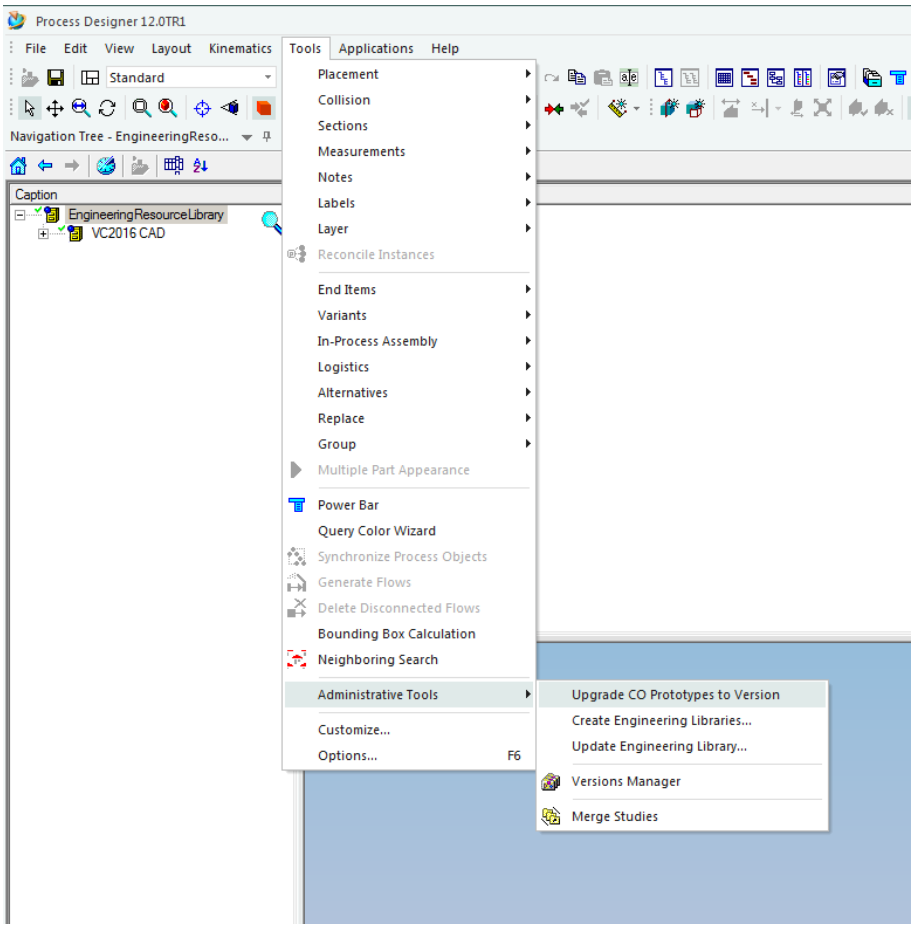
Figur C.1: Menyruta för Importer.exe

VII. Lägg in .co fil i Engineering Library.

VIII. Markera den fil som skall konverteras alternativt ett helt directory med flera filer som skall konverteras.

IX. Gå till Tools > Administrative Tools > Upgrade CO Prototypes to version. Se figur C.2

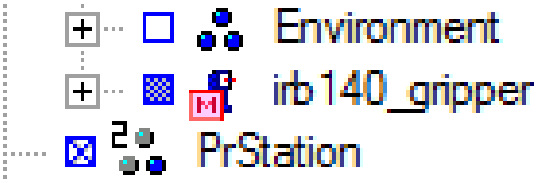
X. Kryssa i Upgrade to .cojt.



Figur C.2: Menystruktur för att uppgradera .co till .cojt

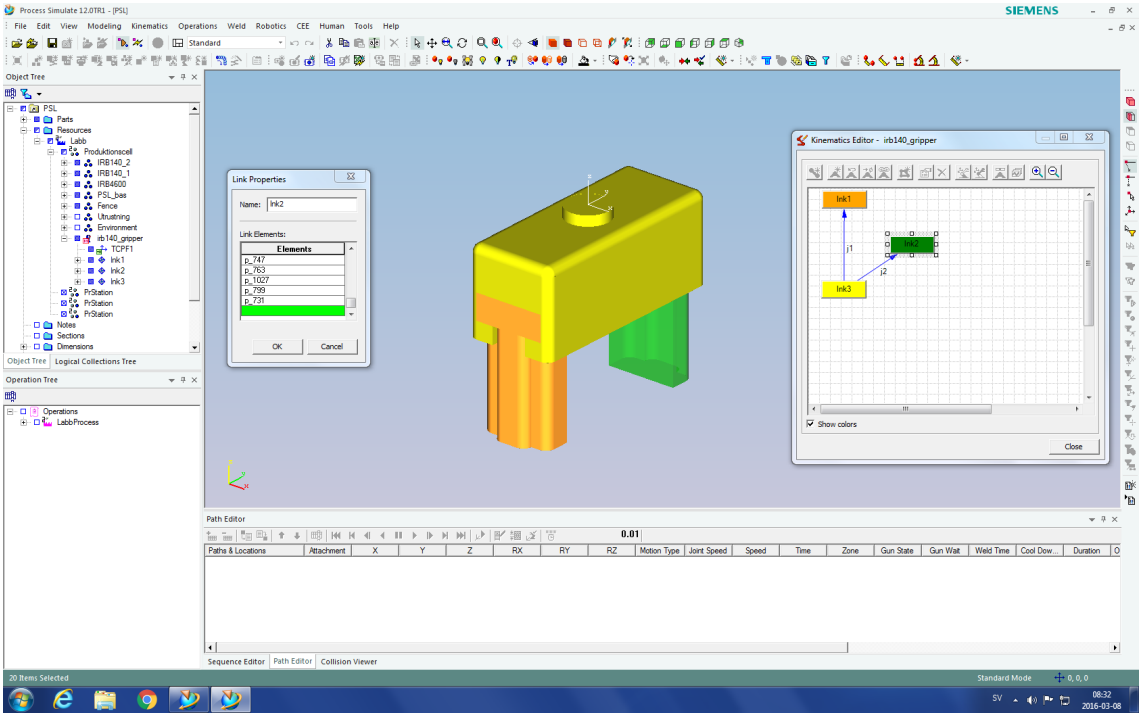
C.2 Skapa kinematik för gripper

I. Skapa en CAD-modell och konvertera enligt C.1.
II. Lägg in filen som en fixture i Process Designer.
III. Öppna Process Simulate och markera delen i Object Tree.
IV. Gå in under Modeling > Set modelling scope och tryck OK. Då dyker det upp ett m över delen i Object Tree som i figur C.3.



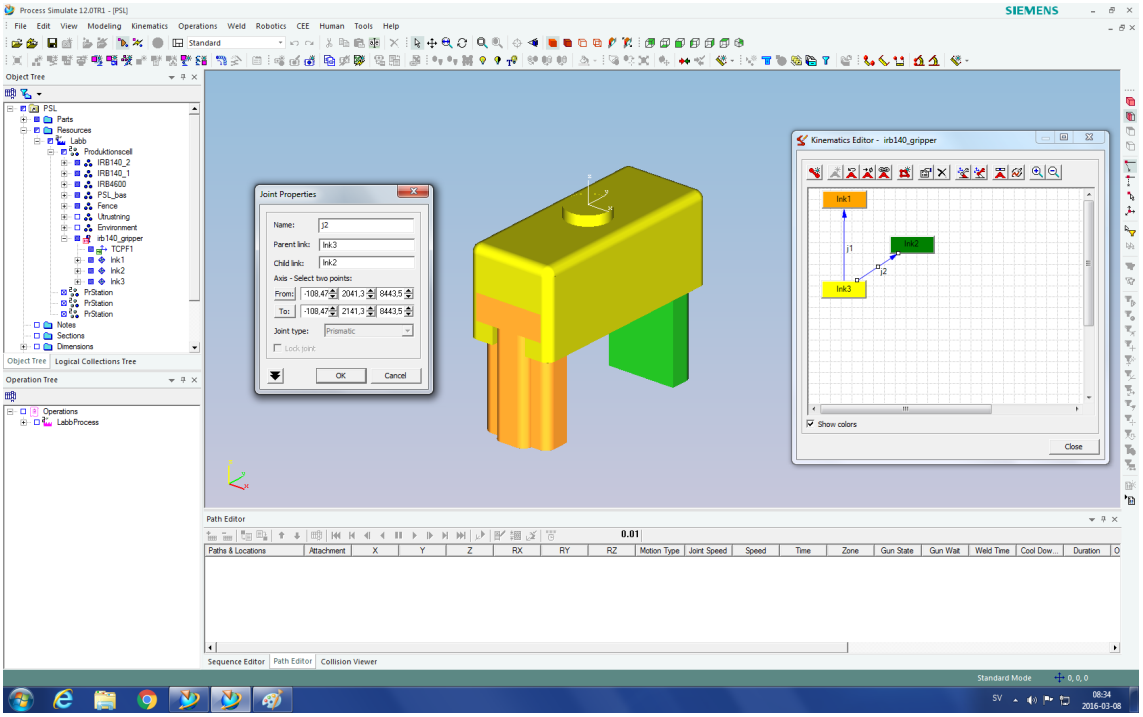
Figur C.3: Ikon då resource är i läge för modeling

V. Markera delen och gå till Kinematics > Kinematics Editor. Klicka på Create Link och välj sidor på den del som är en enhet. Gör detta med samtliga enheter på samma sätt som i figur C.4.



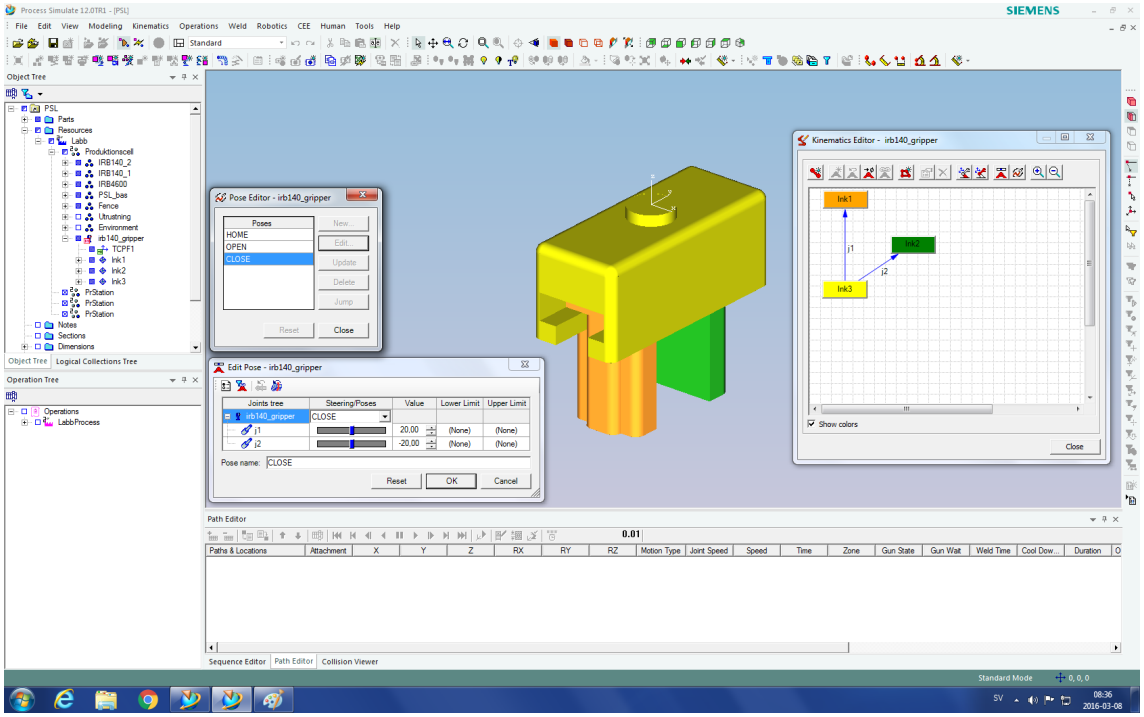
Figur C.4: Menystruktur för att skapa Links

VI. Klicka på en enhet i Kinematics Editor(Heter troligtvis Ink1) och dra till den enhet koppling ska ske med, då skapas joints(j-pilar mellan enheterna). Välj prismatic för linjär rörelse och repetera för att få liknande struktur som i figur C.5.



Figur C.5: Menystruktur för att skapa Joints

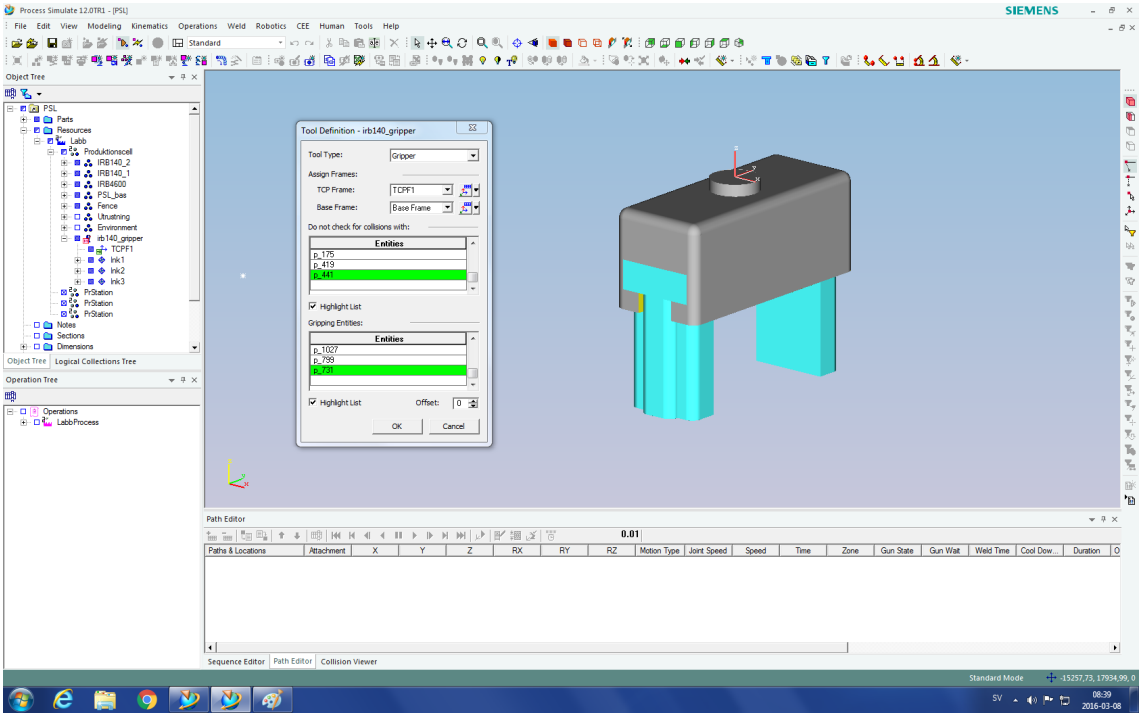
VII. Gå till Open Pose Editor och skapa en ny position genom att ändra värdet på "Valueför varje joint. Positionen kan även ändras i Open Joint Jog för att sedan skapas i Open Pose Editor, detta för att kunna dra delarna till en plats istället för att skriva in "Value". Se figur C.6.



Figur C.6: Menystruktur för att skapa olika poser

VIII. Gå till Kinematics > Tool Definition och välj koordinatsystem för din nya Gripper så att den vet vilken sida som skall fästas mot roboten.

IX. Välj entities. Gripping entities är de sidor som skall gripa tag i något. Collision entities är de sidor som kollision inte skall kontrolleras på. Efter detta borde det likna fönstret i figur C.7 nedan. När allt är klart så gå lämna läget för modeling.



Figur C.7: Menystruktur för att definiera resource som Tool