CODEN:LUTEDX/(TEIE-3116)/1-42(2022)

Industrial Electrical Engineering and Automation

Utveckling av digital tvilling
- I robotmiljö

Jonatan Claesson
Oliver Hedetoft

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Utveckling av digital tvilling

- I robotmiljö

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University at Campus Helsingborg

Jonatan Claesson
Oliver Hedetoft

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2022

Sammanfattning

I detta examensarbete är målet att utveckla en digital tvilling av en robot tillsammans med tre rullband. Målet är att paket ska transporteras via rullbanden till roboten som ska flytta det till en ny plats. Detta uppnås genom att programmera en PLC och ett HMI, vilket görs i TIA som är en utvecklingsmiljö för PLC:s från Siemens. Simulering av PLC:n görs i PLCSIM Advanced och SIMIT och sedan ska simuleringen av den digitala tvillingen ske i RobotStudio. Det ska gå att simulera fel och göra tester på den digitala tvillingen för att se hur den reagerar och om de åtgärder som har programmerats i PLC-koden exekveras. Denna arbetsmetod heter virtuell driftsättning som innebär att skapa en virtuell miljö och arbeta i den istället för med det fysiska objektet. I simuleringen finns det två lägen som den digitala tvillingen kan användas i, det finns manuellt läge där allting styrs av användaren med hjälp av ett simulerat HMI och autoläge där enbart PLC:n styr simuleringen.

Det hela resulterade med en digital tvilling som utvecklats genom att använda fyra olika program som har en öppen kommunikation via PROFINET och ett gemensammt minne mellan programmen. Genom denna kommunikation går det att överföra signaler och data så att PLC:n i ena änden styr simuleringen i RobotStudio i den andra änden.

Nyckelord: Digital tvilling, virtuell driftsättning, PLC, simulering, PROFINET

Abstract

The goal of this thesis is to develop a digital twin of a robot and three conveyor belts. Packages are transported on the conveyors before the robot picks them up and delivers them to a new destination. The conveyors and the robot are controlled by a PLC and an HMI, coded and created in a program called TIA. The PLC is simulated in two programs, PLCSIM Advanced and SIMIT, and the complete digital twin is simulated in the program RobotStudio. In the simulation, it should be able to perform tests and failures on the digital twin to see how it reacts and if the PLC executes the right measures for a failure. This method is called virtual commissioning and works in a virtual environment instead of working on a real physical object. The digital twin should work in two modes in a simulation, either manually when the user is controlling it using the HMI or the auto mode where the PLC controls everything in the simulation.

The project resulted in a digital twin where it uses four programs to have open communication with PROFINET and a shared memory between the programs. Through this communication, it is possible to send signals from the PLC in TIA to the simulation in RobotStudio.

Keywords: Digital twin, virtual commissioning, PLC, simulation, Profinet.

Förord

Examensarbetet beskrivet i denna rapport har varit en avslutande del i högskoleutbildningen för oss studenter inom elektroteknik med inriktning på automation.

Vi vill tacka Automationsteknik AB för assistans och möjligheten att utföra detta arbete hos dem. Vi vill tacka Håkan Marke, Olle Lundberg, Johannes Herbst och Eric Karlsson samt alla från Automationsteknik AB lite extra för hjälp och handledning under projektets gång. Ett extra stort tack till Ola Fischer från Automationsteknik AB som varit handledare från företaget och hjälpt oss mycket under arbetets gång.

Vi vill även tacka ABB och Siemens för att de tillhandahåller licenser till de programmen som har använts under examensarbetet och för det extra stöd de har gett.

Tack till både vår handledare Morten Hemmingsson för stöd och hjälp under projektets gång, samt Mats Lilja för att ha tagit rollen som vår examinator.

Jonatan Claesson & Oliver Hedetoft

Innehållsförteckning

1 Inledning

I det inledande kapitlet så kommer bakgrunden, syftet, målformuleringen, problemformuleringen och avgränsningarna för detta examensarbete att presenteras. Det kommer även ske presentationer av företaget som examensarbetet har gjorts hos och två andra företag som har hjälpt till under examensarbetets gång. Det kommer även att förklaras vad en digital tvilling är och vad företaget kommer använda den till.

1.1 Bakgrund

Examensarbetet har utförts i samarbete med företaget Automationsteknik AB. Automationsteknik är ett företag som fokuserar sig på bland annat automations- och robotteknik. Företaget arbetar med allt inom tekniska konsulttjänster och "turn-key" lösningar inom process-, trä-, verkstads- och livsmedelsindustrin. "Turn-key" lösningar är färdiggjorda lösningar av problem som ska vara helt planerade för att det ska vara enkla att implementera och använda hos ett företag. Detta examensarbete har även gjorts med viss hjälp av ABB och Siemens AG. ABB är ett företag som arbetar med automatisering, digitalisering och elektrifiering. Siemens AG är ett tyskt multiinternationellt företag som jobbar bland annat med områden som industri, energi, hälsovård och infrastruktur.

Automationsteknik AB har en del av en produktionslinje där de bland annat förflyttar och paketerar olika paket med hjälp av en robot. Den del utav produktionslinjen som ingår i detta examensarbetet innehåller följande delar, som går att se i figur 1:

1. 920, det första rullbandet i produktionslinjen som förflyttar paket som har lagts på av en robot. Rullbandet som är utanför staketet till vänster i figur 1.

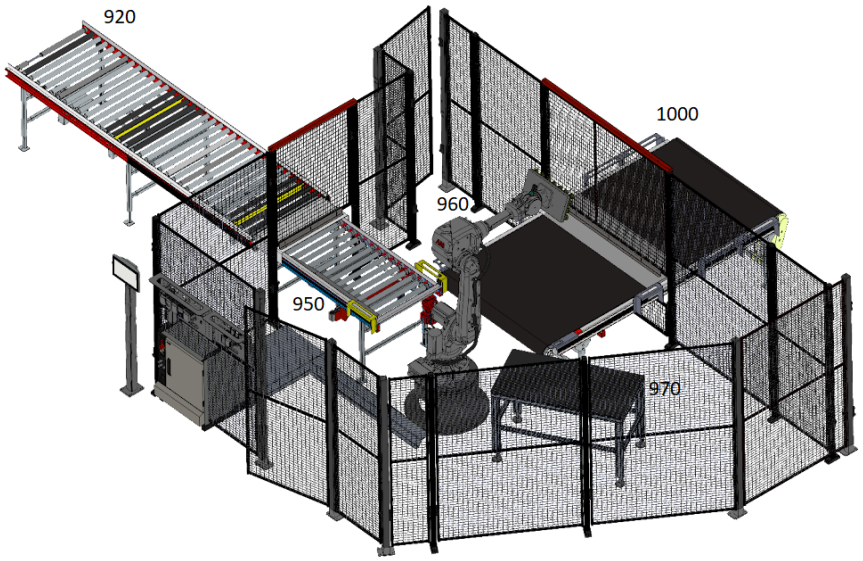
2. 950, det andra rullbandet i produktionslinjen där paketen fortsätter att förflyttas tills de först kommer till en lågfartsgivare som säger till rullbandet att börja röra sig långsammare och sedan till en sensor som är till för att stoppa rullbandet när paketet är vid sin slutposition på 950. Sedan

ska en pusher, som styrs av ett servo, trycka till paketet tills att det kommer till kanten på långsidan av rullbandet. Rullbandet som börjar vid staketet och går innanför mot roboten i figur 1.

3. 960, roboten som flyttar paket från 950 till 970 eller 1000.

4. 970, är ett buffertbord som är till för att lägga paket på. Bordet används om det är fullt på rullband 1000, om det är avstängt eller om roboten som är placerad vid bordet för att hämta paketen är avstängd. Den roboten ingår dock inte i detta examensarbete.

5. 1000, det sista rullbandet där roboten placerar ett visst antal paket i ett förvalt mönster innan det ska förflytta paketen till nästa robot som ska ta hand om dem. Det bredare rullbandet visas i figur 1 nedan.



Figur 1. CAD-modell av den digitala tvillingen.

Om något går fel i produktionen, som att roboten tappar ett paket eller om något av rullbanden slutar fungera, så kan det ta lång tid att åtgärda de problemen. Om något i produktionslinjen går sönder så kan det även leda till stora kostnader. För att undvika såna här problem och för att vara förberedd på hur man ska lösa dem samt för att spara tid och pengar så vill Automationsteknik

AB göra en digital tvilling av denna station av i sin produktionslinje. I detta examensarbete tillhandahåller företaget följande till studenterna:

- Färdiggjord PLC-kod för roboten och rullbanden som används till den verkliga produktionslinjen
- CAD-modeller av produktionslinjen
- Licenser för mjukvaruprogrammen från ABB och Siemens

En digital tvilling är en virtuell modell av en process eller ett objekt, till exempel en robot, en hel produktionslinje eller en stad. Det kan ses som en brygga mellan den digitala världen och den riktiga, då tvillingen tar indata och signaler från det verkliga objektet eller processen. Den digitala tvillingen kommer dock inte skicka tillbaka signaler eller data till den riktiga roboten utan allt sker i en virtuell miljö. Detta kallas offline-programmering då man programmerar på ett objekt rent virtuellt enbart och att det inte har någon påverkan på objektet i verkligheten. Kommunikationen är enkelriktad då det enbart skickas data och information från den verkliga världen till den digitala, och inte åt andra hållet. Med den digitala tvillingen ska man kunna utföra simuleringar som ska motsvara hur det verkliga objektet eller processen beter sig i verkligheten.

1.2 Syfte

Syftet med examensarbetet är att få fram en digital tvilling av produktionslinjen i figur 1 som ska användas för att göra tester, simuleringar och visualiseringar av roboten och produktionslinjens uppgifter. Detta ska göras genom att använda CAD-modeller av den fysiska produktionslinjen och sedan genom att använda sensorer, givare, PLC och HMI för att få in- och utdata. Diverse mjukvaruprogram kommer att användas för att programmera, simulera och visualisera roboten, PLC:n och till slut hela systemet. PLC:n och roboten ska kunna kommunicera med varandra genom att koppla in- och utsignaler från roboten i RobotStudio till PLC:n i TIA och tvärtemot. In- och utsignaler simuleras av den digitala tvillingen och kommer från PLC-programmet respektive från simuleringen i RobotStudio. Givare och sensorer som finns i verkligheten simuleras i RobotStudio som ger ut givarsvar som skickas till TIA som får PLC:n att utföra en sekvens beroende på vad signalen har för värde. Det förväntade resultatet är att kunna använda den

digitala tvillingen genom simuleringar för att optimera produktionslinjen och för att kunna åtgärda problem som kan uppstå i verkligheten. Det är därför viktigt att PLC-koden ej modifieras.

1.3 Målformulering

Målet med examensarbetet är att få fram en fungerande digital tvilling för produktionslinjen där simuleringar och tester ska kunna genomföras på den. Den digitala tvillingen ska motsvara den fysiska produktionslinjen i en virtuell miljö och efterlikna det som finns i verkligheten så mycket som möjligt. Den ska även kunna simuleras fel i anläggningen, till exempel fel i återkopplingen från givarsignalerna.

1.4 Problemformulering

1. Hur tillverkas en digital tvilling?
2. Hur simuleras servoaxlarna som styr pushern och det ena rullbandet?
3. Hur ska alla mjukvaruprogram kommunicera med varandra?
4. Vad för slags fel går att simulera med den digitala tvillingen?

1.5 Motivering av examensarbete

Målet med examensarbete var att hitta ett projekt som fångade upp ett intresse som passade båda studenterna och fokuserar på kunskaper inom elektroteknik. Eftersom att utbildningen har fokuserat mycket på automationsteknik, styr- och reglerteknik och programmering så valdes ett projekt som motsvarade deras intressen och kunskaper på en bra nivå.

Projektet hos Automationsteknik AB valdes på grund av att studenterna kände att kunskaperna som behövdes passade deras bild av vad de var intresserade av och vad de har lärt sig under sin tid på utbildningen. Projektet väckte även ett stort intresse hos studenterna på grund av att det är

ett brett kompetensområde som arbetet täcker och för att det är ett stort arbetsområde och ett för framtiden.

1.6 Avgränsningar

Den digitala tvillingen ska inte motsvara Automationsteknik AB:s hela arbetslinje utan är nerskalad till en mindre del, som går att se i figur 1. Detta bestämdes av företaget när de hade börjat planera projektet och idén lades fram till studenterna.

En digital tvilling ska vara motsvarigheten till ett verkligt objekt i en virtuell miljö, det har företaget redan ordnat då de har CAD-modeller som används i den virtuella miljön och PLC-kod sedan tidigare. Det innebär att det är inget som studenterna behöver göra i detta examensarbete. Detta examensarbete är avgränsat till att använda TIA, SIMIT, PLCSIM Advanced, WinCC Runtime och RobotStudio för att bygga en digital tvilling med hjälp av företagets CAD-modeller samt PLC- och RAPID-kod.

2 Teknisk bakgrund

I detta kapitel beskrivs metoder och kunskaper som är viktiga för detta examensarbete och vilka program som används. Funktioner och nödvändig information om programmen beskrivs i denna del av rapporten.

2.1 Virtuell Driftsättning

För att konkurera med företag inom branschen som Automationsteknik AB är med i så krävs det att de ständigt utvecklas och jobbar mot att effektivisera deras system och produkter. I en bransch som blir tuffare för varje år då nya teknologier utvecklas och dagens samhälle styrs mot att bli mer automatiserat så kommer företag inom automation att ha en nyckelroll. För att jobba mot att effektivisera hela företaget och deras produktion så går alltmer företag mot att jobba med virtuell driftsättning.

Virtuell driftsättning är ett arbetssätt för att skapa en virtuell miljö, t.ex. av en produktionslinje som finns hos Automationsteknik AB, där de har 3D-modeller och kopior av den fysiska linjen. Kopiorna i den virtuella miljön ska efterlikna den verkliga miljön till en viss grad som bestäms av företaget beroende på vad för tester och simuleringar de vill utföra, och vad de har för krav. Virtuell driftsättning är ett väldigt sparsamt arbetssätt, både tids- och ekonomimässigt. [1]

Att göra felsökningar och ändringar i en fysisk miljö kan ta väldigt lång tid och kan även resultera i stora kostnader. För att undvika detta så jobbar man i en virtuell miljö istället för att identifiera problem så tidigt som möjligt och åtgärda dem. Genom detta så kan man både spara pengar och tid för att se till så att arbetet i den fysiska miljön ska vara så kort som möjligt. [1]

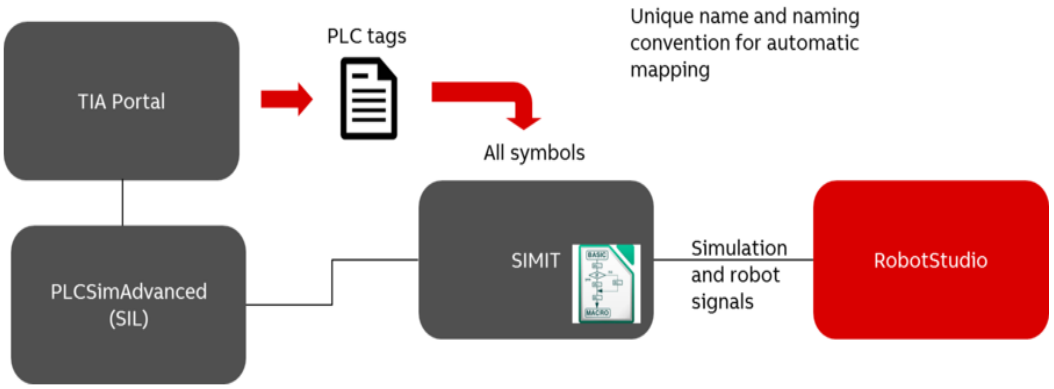
Med virtuell driftsättning är målet att tillverka en digital tvilling där 3D-modeller och I/O-system sammankopplas för att att få en efterliknande modell av det som finns i verkligheten. Med hjälp av diverse program och med programmering av en PLC och ett HMI ska de kunna sammankopplas och kommunicera med varandra.

2.2 Datorprogram

I detta examensarbete har fyra program använts för konstruktionen av den digitala tvillingen. Dessa program är:

- Totally Integrated Automation Portal (TIA), för programmering av PLC och HMI.
- PLCSIM Advanced, för simulering av PLC:n.
- SIMIT Simulation Platform (SIMIT), för simulering och sammankoppling mellan PLCSIM och RobotStudio
- RobotStudio, för visualisering och simulering av roboten och rullbanden.

I figur 2 nedan visas en förenklad förklaring av hur programmen är kopplade till varandra. Det börjar med att PLC:n programmeras i TIA. "PLC tags" är PLC-signaler som adresseras i TIA och överförs till SIMIT genom PLCSIM Advanced koppling (avsnitt 2.2.3.5). I SIMIT startas en simulering av ett program som samtidigt startar upp en instans i PLCSIM Advanced. Den instansen kan sedan PLC-programmet i TIA laddas upp till, vilket gör att signalerna i TIA och SIMIT är sammankopplade. I SIMIT används sedan ett minne som delas med RobotStudio (SHM, avsnitt 2.5.4), för att överföra signaler mellan dessa två program. Minnet delas då man startar en simulering i SIMIT som sedan sammankopplas med RobotStudio med hjälp av smartkomponenter, Genom det kan signaler som är gjorda i TIA överföras till RobotStudio med hjälp av SIMIT som är en brygga mellan programmen.



Figur 2. Visualisering av kommunikationen mellan programmen. [2]

2.3 Totally Integrated Automation Portal (TIA)

TIA är ett mjukvaruprogram utvecklat av Siemens AG, som används inom många olika områden i automationsbranschen på grund av sitt breda användningsområde. Programmet används för att integrera områden och komponenter i ett automationsprojekt som:
Säkerhet, kontroll, HMI, PLC, drivsystem och rörelsekontroll.

TIA används exempelvis till att skriva PLC-kod som kan styra robotar, servoaxlar och mycket annat. I programmet används olika språk för att programmera PLC:n och HMI:n, som förklaras i avsnitt 2.3.1. TIA används för att programmera PLC, HMI och rörelsemotorer vilket gör att det blir ett stort användningsområde för programmet i en virtuell miljö.

I detta program går det även att utföra simuleringar av ett HMI (avsnitt 2.3.3). Företaget har ett HMI som står utanför cellen i sin riktiga produktionslinje som används för att styra rullbanden och pushern i manuellt läge.

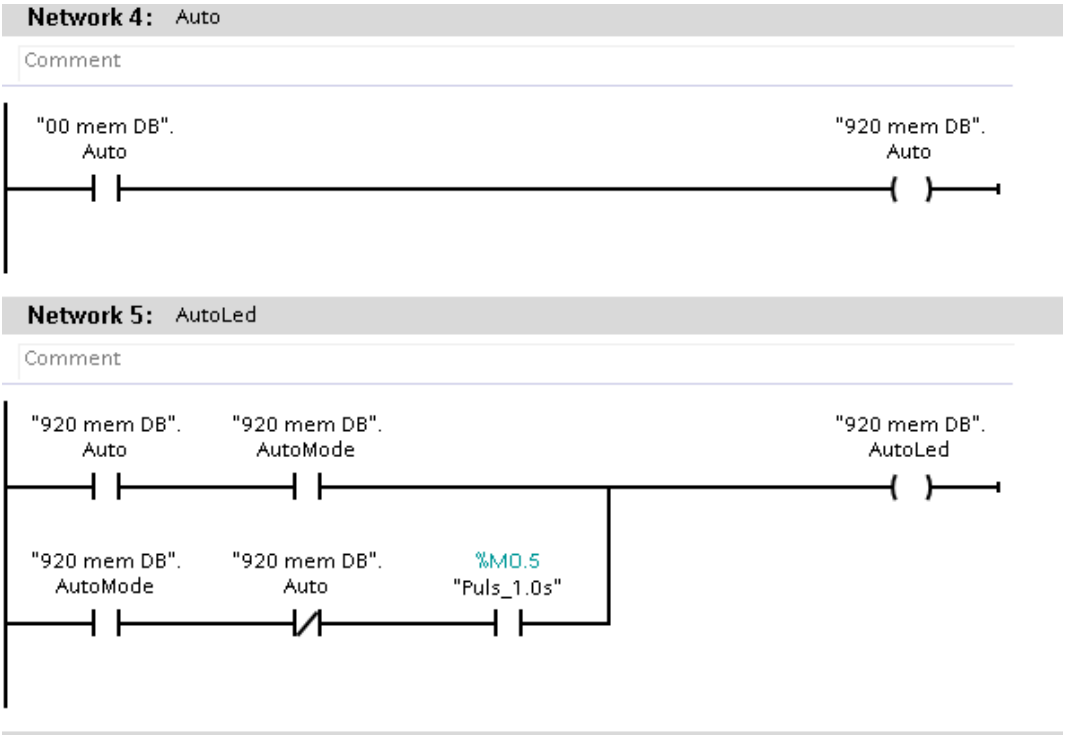
2.3.1 Programmable Logic Controller (PLC)

En PLC är en programmerbar styrenhet som används inom mestadel automationsteknik för att styra maskiner och motorer. För att programmera en PLC kan man använda olika språk, i TIA används följande språk:

- Structured Text (ST)
- Ladder logic Diagram (LD)
- Sequential Function Charts (SFC)
- Function Block Diagram (FBD)
- Instruction List (IL)

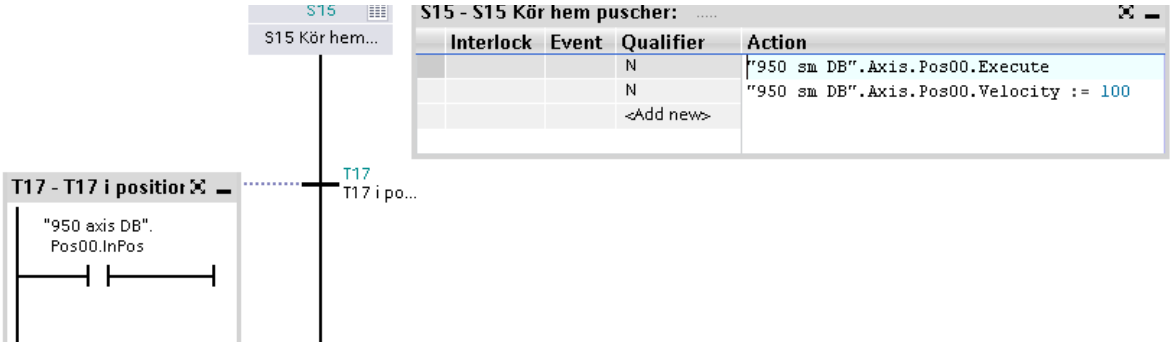
Figur 3 visar ett exempel på hur LD ser ut i TIA. Det baseras på att det finns nätverk (“Network”) som det finns en utsignal till. Till exempel i nätverk 4 (Auto), är det en insignal som man läser in längst till vänster och en utsignal som man skriver till längst till höger. I nätverk fem (AutoLed) finns det två grenar som motsvarar en ELLER(OR)-grind genom att det är två parallella grenar som läses in samtidigt och det räcker med att villkoren för en av grenarna är uppfyllda för att utsignalen ska bli en etta (TRUE). När två insignaler ligger jämte varandra, som de gör i nätverk

5, motsvarar det en OCH(AND)-grind som innebär att båda insignalerna (“920 mem DB”. Auto och “920 mem DB”. Automode) måste vara en etta för att utsignalen (“920 mem DB”. AutoLed) ska bli en etta. Insignalen, “920 mem DB”.Auto, har ett streck diagonalt över sig vilket innebär att det är en NOT-grind. Det betyder att den inverterar signalen, en etta blir en nolla och en nolla blir en etta.



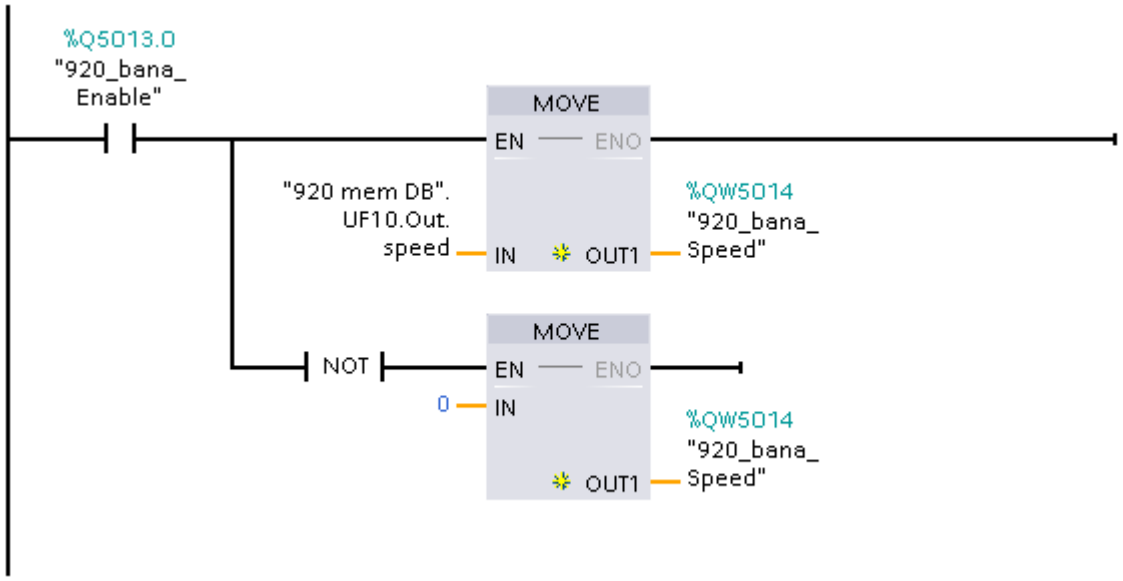
Figur 3. Exempel på LD.

I figur 4 visas hur en SFC ser ut i TIA. Det första som händer i SFC:n är att programmet börjar i ett tillstånd (State) som säger åt PLC:n vad den ska göra. I tillståndet kan den bland annat sätta en insignal till en etta, om den är en BOOL, eller ge den ett reellt värde om signalen är en INT. När PLC:n är klar med tillståndet går den vidare till ett så kallat övergångsvillkor (transition condition) som är ett villkor som måste uppfyllas, t.ex att en insignal som läses in måste vara en etta, innan den kan gå vidare till nästa tillstånd. I figur 4 sätts “950 sm DB”.Axis.Pos00.Velocity till 100 och när det är genomfört inväntar programmet att “950 axis DB”.Pos00.InPos blir TRUE innan den går vidare till nästa tillstånd.



Figur 4. Exempel på SFC.

I figur 5 visas två FBD:s i TIA. FBD:s i TIA fyller stor funktion i programmet då de har ett stort användningsområde och kan vara till nytta för beräkningar av hastighet, tryck och vridmoment. I figur 4 nedan används två "MOVE" FBD:s, det är för att skicka en hastighet till rullbanden som de ska köras. Inuti dessa block finns flera ytterligare block som har programmerats av utvecklarna av programmet som gör beräkningar för hastighet, tryck och vridmoment. FBD:s har stor användning för styrning av rullbanden, bland annat för att skicka ut hastigheten som de ska köra i. De används även för att beräkna vridmoment och tryck för teknologiobjektet (avsnitt 2.3.2) som styr pushern.

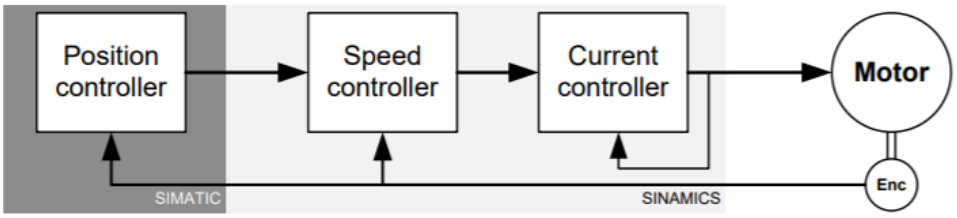


Figur 5. Exempel på FBD.

2.3.2 Teknologiobjekt

För att gynna användandet av tekniska funktioner som kommer med en SIMATIC kontroller så har teknologiobjekt implementerats i programmet av Siemens. En SIMATIC kontroller används för att kontrollera drivsystemet vilket innebär att den kontrollerar motorn desslikes, se figur 6, där en positionsregulator bland annat används för att styra motorn. Positionsregulatorn är en funktion som kommer med en SIMATIC-kontroller. Dessa teknologiobjekt används främst för att underlätta styrandet och behandlingen av axlar som rör på sig, till exempel en servomotor. Ett TO (teknologiobjekt) för rörelsekontroll i SIMATIC kan ha flera egenskaper och kan representera till exempel ett mjukvaruobjekt i kontrollen och mekaniska komponenter som använder de teknologiska funktionerna.

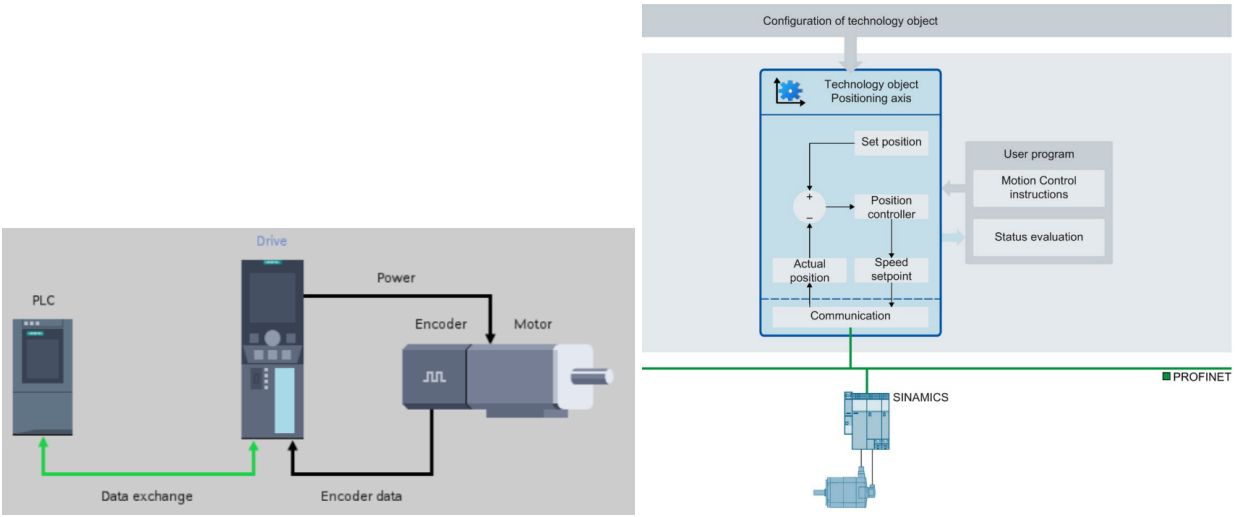
När man konfigurerar ett teknologiobjekt så är ett antal regulatorer involverade som arbetar i en sluten krets, se figur 6. Strömregulatorn är till för att få ut rätt mängd ström (börvärdet) för att reglera motorn för ett begärt vridmoment. Hastighetsregulatorn ger ut det börvärde för vridmomentet som behövs för att motorn ska komma upp till en efterfrågad hastighet. Positionsregulatorn, ger ut den hastighet (börvärde) som behövs för att motorn ska komma till den positionen som är önskad. [6]



Figur 6. Schema över kontrollerna som är involverade i användandet av teknologiobjekt. [3]

Teknologiobjekt för positionering kommunicerar via ett lagringsmedium och PLC:n, se figur 7a. Tillsammans med en motor skapar lagringsmediumet ett drivsystem där lagringsmediumet kan mata motorn med varierande spänning för att styra vilken hastighet som motorn kör i. Lagringsmediumet skickar ström till motorn som är kopplad till en pulsgivare. Pulsgivaren används för att mäta position och hastighet och skickar sedan tillbaka data till drivsystemet. Ett TO styr en axel, som styrs av motorn och kan få den till en viss position väldigt precist. När man simulerar ett TO så får man ut axelns nuvarande position men även hur snabbt den rör sig och dess acceleration. Detta går att styra ett TO genom att förflytta det till en viss position eller med

en viss hastighet via en kontrollpanel i TIA . Ett TO är nödvändigt för applikationer då man behöver få ut en exakt kontrollerad position eller för att styra en hastighet. För detta har Siemens gjort flera TO som går att implementera i ett program i TIA, dessa är till för simulering och styrning av axlar genom att använda kontrollpanelen i TIA för att styra dess hastighets- och positionsvärde.



Figur 7a [3] och 7b [7]. Schema över hur teknologiobjektet för positionering fungerar.

Pulsgivaren som syns i figur 7a kan konfigureras genom tre olika sätt:

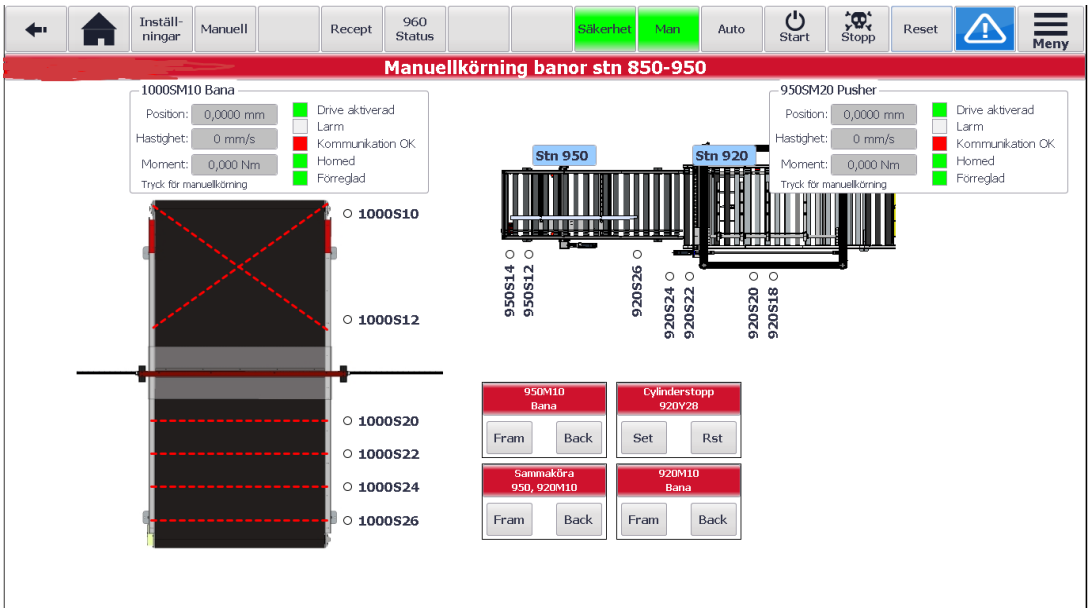
- Inkrementell, det verkliga värdet i PROFIdrivetelegramet är baserat på ett inkrementell värde som startar i position noll varje gång regulatorn startas.
- Absolut, det verkliga värdet i PROFIdrivetelegramet baseras på ett absolut värde som sparas i ett minne varje gång regulatorn startas och stoppas.
- Cykliskt absolut, positionen på axeln bestäms direkt av det nuvarande absolutvärdet hos pulsgivaren, som är det senast sparade värdet i minnet.

I figur 7b syns hur konfigurationen av ett TO fungerar. Kommunikationen mellan TIA (User program) och ett TO sker genom PROFINET. TIA bestämmer börvärde för hastighet och position och skickar det till PROFINET. Sedan skickar kommunikationslänken vad ärvärdet för positionen är och subtraherar det från börvärdet av positionen. Skillnaden mellan bör- och ärvärdet för positionen är reglerfelet, vilket önskas vara noll. Om det inte är noll regleras börvärdet av hastigheten för att få reglerfelet att bli noll. Samtidigt som allt detta sker skickar TO uppdateringar om vad som händer till användaren.

2.3.3 Human Machine Interface

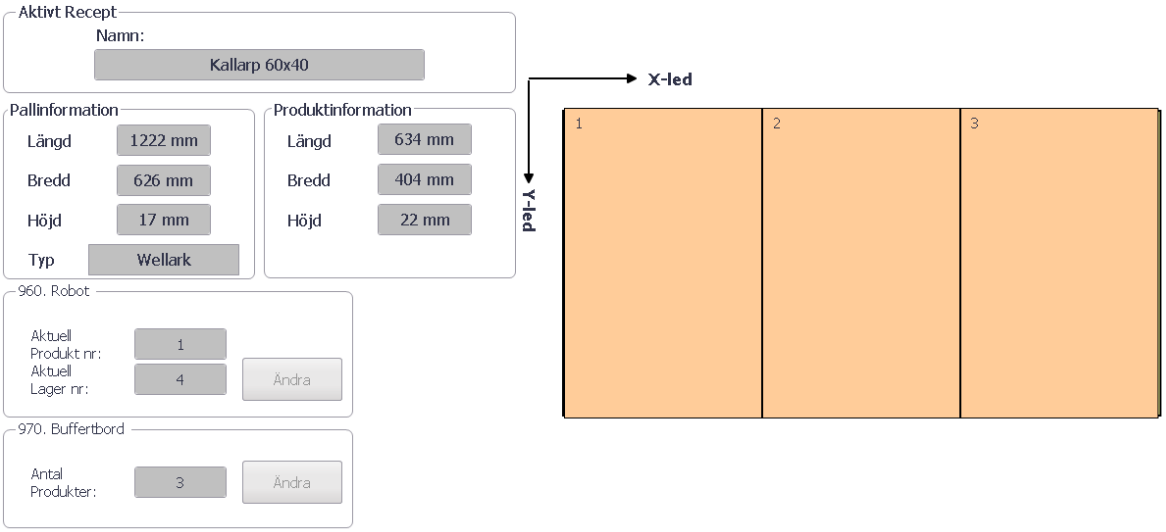
För att simulera ett HMI i TIA använder det sig av ett program som heter “WinCC Runtime” som redan finns i TIA. Detta används för att göra en visualisering av HMI:t och visa designen som har gjorts i TIA i ett separat fönster som kommer fram när en simulering av HMI:t startas. I figur 8 visas går det att se en del av HMI:t som användes i detta examensarbete.

HMI:t är programmerat av Automationsteknik AB och det har ett flertal funktioner som är till för att styra rullbanden och pushern. I det simulerade HMI:t går det att välja mellan att köra den digitala tvillingen i två olika lägen. Det finns ett autoläge som innebär att allt körs automatiskt enligt koden i PLC:n och det räcker med att starta den i auto och sedan starta den via start-knappen i HMI:t. I autoläge kommer rullbanden och pushern köras automatiskt med hjälp av PLC-koden. Manuellt läge innebär att rullbanden och pushern styrs genom knapparna som syns i figur 8 men roboten körs via PLC-koden. I manuellt läge kan man köra rullbanden (920, 950 och 1000) framåt och man kan även samköra rullband 920 och 950. I manuellt läge körs rullbanden med samma hastighet hela tiden och detsamma gäller pushern. I autoläge är rullbanden och pushern programmerade att köra med olika hastigheter t.ex då ett paket har kommit till en av sensorerna på rullband 950, i vilket fall pushern ska börja köra långsammare precis innan den kommer i kontakt med paketet. I HMI:t kan man även återställa roboten och rullbanden så att de är i sina grundlägen och återställa deras minnen. Det gör att roboten inte längre vet hur många paket den har hämtat och att rullbanden är i sin första sekvens innan den börjar köra. Det går även att köra pushern och rullband 1000 till deras startpositioner i vilka de behöver vara innan processen startas.



Figur 8. Visualisering av HMI:t.

Det går även att ställa in mönstret, i vilken roboten ska placera paketen på det nya rullbandet (1000). I figur 9 finns användargränssnittet för dessa inställningar, där man kan ställa in pall- och produktinformation såsom längd, bredd och höjd för pall respektive produkt som väljs att användas i TIA. Flera mönster kan skapas, namnges och sparas exempelvis Det används för att bestämma hur paketen ska placeras på rullband 1000 och anger information för hur stort det nuvarande paketet är och minnet för hur många paket roboten har placerat och lagt på buffertbordet respektive på rullband 1000.



Figur 9. Recept för palletering av paketen.

2.4 PLCSIM Advanced

PLCSIM Advanced är ett program som är utvecklat av Siemens AG som kan användas med TIA. Det används för att göra simuleringar av PLC:n för att se hur den beter sig i den virtuella världen. Med PLCSIM Advanced kan man välja två olika sätt att starta upp en sammankoppling mellan PLCSIM och TIA för att simulera PLC:n. Den ena metoden kallas PLCSIM vilket enbart tillåter lokal kommunikation mellan programmen på en dator (t.ex TIA och SIMIT) och den andra kallas PLCSIM Virthual Ethernet Adapter där kommunikationen sker via nätverket. PLCSIM Virthual Ethernet Adapter tillåter därför att PLCSIM Advanced och TIA (samt dess PLC-kod) inte behöver befinna sig på samma enhet under simulering.

När en simulering i programmet startas går det att koppla upp TIA till PLCSIM Advanced, vilket gör att signalerna i TIA kan granskas och se deras nuvarande status (om de är ett eller noll). Det går även att kontrollera PLC:ns signaler genom att tvinga bitar till att sättas till ettor eller nollor, detta kan man göra direkt i blocken eller genom ett HMI som TIA kan simulera. Instansen som PLCSIM Advanced skapar kopplas vidare till SIMIT och signaler i TIA som har adresserats och kan utnyttjas i SIMIT:s egna simulering (avsnitt 2.5.5). Detta program kan ses som en brygga mellan TIA och SIMIT.

2.5 SIMIT Simulation Platform

SIMIT är ett program från Siemens AG som används som en koppling mellan simuleringen av PLC:n och visualiseringen av den digitala tvillingen i RobotStudio. Programmet fungerar som en brygga mellan TIA Portal och RobotStudio då signaler mellan dessa program sammankopplas i SIMIT.

Med SIMIT kan man starta en simulering med bland annat “Charts” (se avsnitt 2.5.3), kopplingar och visualiseringar i 2D. I detta projekt används “Charts” och olika kopplingar för att bygga upp en simulering med signaler som skapas i SIMIT, men även signaler som kan hämtas från andra program. En sammankoppling uppstår mellan SIMIT och andra program, t.ex. RobotStudio och TIA. Kopplingen ska utföra ett utbyte av signaler mellan programmen där SIMIT är bryggan mellan för själva utbytet. En utsignal i PLCSIM Advanced blir en insignal i SIMIT, som sedan ska bearbetas och kopplas vidare till SIMIT:s utsignaler. SIMIT:s utsignal ska sedan kopplas

vidare till RobotStudio där den har sitt ändamål, denna process ska fungera åt båda hållen. SIMIT kan även simulera teknologiobjekten med hjälp av PROFIdrive (avsnitt 2.5.6).

2.5.1 Time Slices

Simuleringar är beräknande cykliskt, dvs att beräkningarna följer en cykel över vad som sker och gör det i ett repeterande mönster. Cykeltiden anger den tidsram där beräkningar och utbyte av data ska utföras. Cykeltiden ställs in med hjälp av en av åtta tillgängliga time slices. Varje komponent i en chart och varje koppling blir tilldelad till en av de åtta tillgängliga time slices som man kan ställa in till olika starttider då de ska exekveras. Varje koppling är tilldelad till en time slice, som bestämmer cykeln för hur snabbt kopplingen ska utföra utbyte av data. En komponent i en chart har även sub-time slices (A-H), för varje time slice, och varje sub-time slice körs med samma cykeltid som den time slice den är kopplad till. [9]

2.5.2 Driftlägen

Kopplingarna och simuleringsmodellen kan redigeras i tre olika driftlägen i SIMIT. Det finns synkront, asynkront och buss-synkront driftläge.

I asynkront läge så är beräkningarna av time slices och bearbetningen av kopplingarna tidsstyrda. Om simuleringsmodellen av en time slice inte är beräknad på den förväntade tiden så är en eller flera cykler i bearbetningen förlorade och då kommer simuleringen sluta med att beräkna de resterande time slices. Beräkningen av simuleringen kommer inte börja igen förrän alla time slices kan beräknas igen. [8]

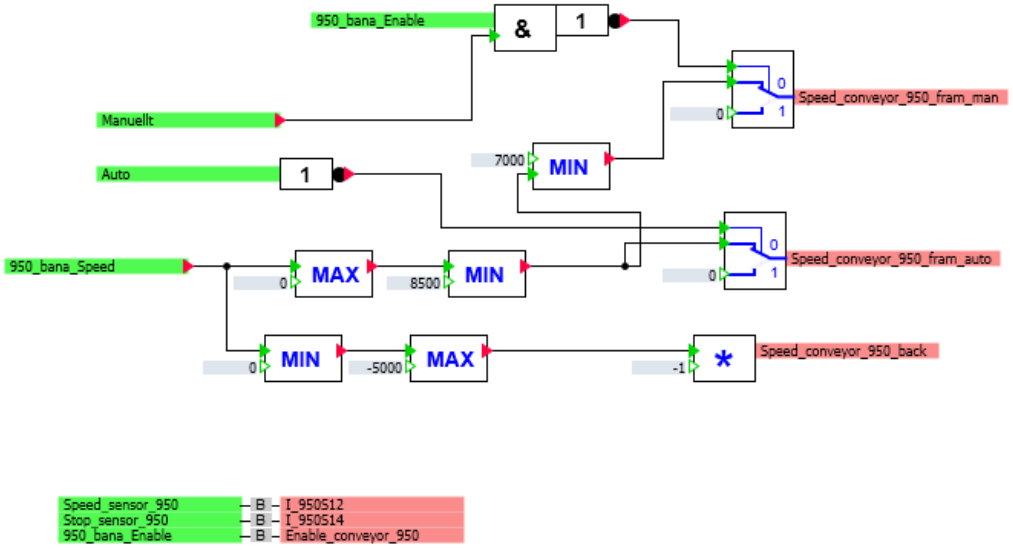
I synkront läge så är alla moduler och kopplingar beräknade i en exakt specificerad sekvens. Nästa aktion i simuleringen ska inte utföras förrän den föregående är helt klar. Detta resulterar i bättre svarstider med korta cykeltider, varje modul och koppling kan dock stoppa hela simuleringen om något går fel.

I buss-synkront läge så ser SIMIT till att alla komponenter i simuleringen har samma synkroniserad utgång av simuleringen. Detta leder till att just detta driftläge är väldigt användbart om användaren är i realtidsbehov. Detta innebär också att minst en time slice måste ha samma tid

inställd som cykeltiden är för hela projektet för att göra det möjligt att veta vilken period och starttid som den ska utgå ifrån.

2.5.3 Charts

Charts är ett verktyg i SIMIT som är till för att skapa en simuleringsmodell genom att använda signaler från kopplingar som Shared Memory (avsnitt 2.5.4) och PLCSIM Advanced (avsnitt 2.5.5). I figur 10 nedan går det att se ett exempel på hur en chart i SIMIT kan se ut.



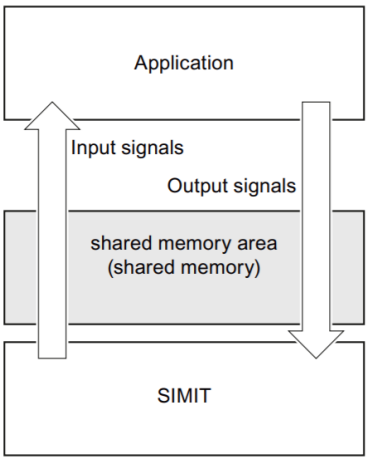
Figur 10. En chart för manuell styrning av rullband 950.

Den chart som visas i figur 10 består av block som är bland annat logiska grindar (t.ex AND, OR, NOT, SET/RESET) och matematiska funktioner (t.ex multiplikation, division, integration). Det är block som kommer med i SIMIT:s grundbibliotek när programmet laddas ner. Det finns även mer komplicerade block som används för att simulera motorer och som bland annat beräknar vridmoment och kraft. Signaler som används i charts kommer från signaler (gröna) som är skapade för SHM (avsnitt 2.5.4) och signaler (röda) som hämtas från PLCSIM Advanced koppling (avsnitt 2.5.5). SIMIT bryggan mellan RobotStudio och TIA och det innebär att sammankoppling av signaler från de två programmen sker i SIMIT. Det sker genom de dragna linjerna mellan programmen och blocken som är kopplade mellan signalerna. Blocken används för att manipulera signalerna för att få ut ett önskvärt resultat av sammankopplingen. I

verkligheten fungerar det inte på samma sätt utan då är det en koppling mellan PLC:n och hela systemet. Detta innebär att i verkligheten finns inget mellanställe som SIMIT där signalerna går genom utan det är snarare enbart en sammankoppling mellan RobotStudio och TIA.

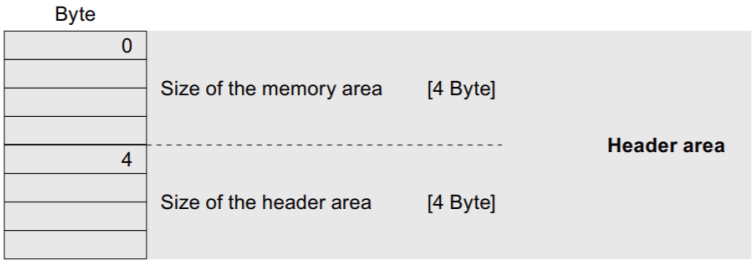
2.5.4 Shared Memory (SHM)

SHM är ett minne som skapas i SIMIT och används som en koppling mellan RobotStudio och SIMIT i detta projekt. SHM gateway används för att göra ett utbyte av signaler mellan SIMIT och det andra programmet som man väljer att sammankoppla till SHM. Detta minne tillhandahåller en välfungerande samverkan av signaler som tillåter andra program att kommunicera med SIMIT. Insignalerna från SIMIT är skrivna till minnet och utsignalerna läses av SIMIT från minnet, som figur 11 visar.



Figur 11. Hur SIMIT kommunicerar med SHM och applikationen som är kopplad till minnet. [7]

Minnets är uppdelat i två delar, header och informationsdata. Headern är minst åtta bytes stor. De första fyra innehåller information om hur stort hela minnet är och de fyra nästa innehåller headerns storlek. I delen av minnet som innehåller all data, bland annat signalerna som ska kopplas till SHM genom adresser, finns det en viss struktur. Varje signal i SHM kopplingen har en unik adress i datan. Eftersom att en signal på en viss adress enbart kan definieras som en in- eller utsignal så kan inte samma adress användas till en insignal och till en utsignal.



Figur 12. Hur headern är uppbyggd i SHM. [7]

Det finns en faktor till som spelar stor roll i SHM, det är vilken datatyp som in- och utsignalerna har. Det finns flera datatyper som tar upp olika mycket plats på en adress i minnet. Till exempel, kan en signal vara en BOOL som tar upp en bit, BYTE som tar upp en byte, en WORD eller INT som tar upp två bytes och DWORD, DINT och REAL som tar upp fyra bytes. Detta är viktigt att ha i åtanke när man ska adressera signalerna i SHM och sedan koppla dem till RobotStudio. En signal i RobotStudio kommer ha sin motsvarighet i SIMIT som fungerar som ett mellanstopp innan den signalen ska skickas till TIA. Det är viktigt att den signalen har samma adress i SHM som signalen i RobotStudio har på grund av att när signalen i RobotStudio blir true/false så ska det ske för motsvarande signal i SIMIT samtidigt. Om det inte är rätt adresserat kommer kommunikationen mellan programmen bli fel då förmodligen en signal i RobotStudio kommer aktiverar fel signal i SIMIT. Det kan leda till att fel sekvens i PLC:n kommer exekveras vilket gör att roboten utför en felaktig rörelse. Hur man adresserar signalerna och vilka värden de kan ha går att se i figur 13. I RobotStudio behöver signalerna också adresseras på ett liknande sätt, största skillnaden är att de inte behöver skrivas som M eller MB utan skrivs istället som en signal med ett visst antal bitar som den ska täcka.

Data type	Variable	Notation	Value range
BOOL	1 bit	M<byte>.<bit>	True/False
BYTE	1 byte (8 bits)	MB<byte>	0 ... 255 or −128 ... 127
WORD	2 bytes	MW<byte>	0 ... 65,535
INT	2 bytes	MW<byte>	−32,768 ... 32,767
DWORD	4 bytes	MD<byte>	0 ... 4,294,967,295
DINT	4 bytes	MD<byte>	−2,147,483,648 ... 2,147,483,647
REAL	4 bytes	MD<byte>	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$

Figur 13. De olika datatyperna man kan använda i SHM. [7]

Observera att signalerna inte får krocka eller överlappa varandra. Till exempel, om man anger en insignal till SHM som en BYTE till adressen MB0 (första BYTE:n) så kan man inte lägga en annan insignal som är en BOOL på adresserna M0.0-M0.7. Eftersom MB0 tar upp den första BYTE:n i minnet betyder det att den tar upp de första 8 bitarna (M0.0-M0.7). Det blir ett problem då SIMIT inte kommer kunna köra en simulering av programmet då flera signaler är skrivna på samma adress. När en signal ska skrivas till en adress börjar det alltid med M för "Memory" sedan beroende på vilken datatyp som signal är, skrivs signalen antingen som MW eller MD och det bestämmer då vilka värden signalen kan ha och hur många bitar/bytes signalen kommer ta upp (se figur 13).

2.5.5 PLCSIM Advanced koppling

PLCSIM Advanced koppling i SIMIT tillåter att data utbytes cykliskt mellan PLCSIM Advanced och SIMIT. Det fungerar genom ett färdigt program från TIA importeras till SIMIT genom att välja PLCSIM Advanced koppling, i SIMIT, som gör att alla adresserade signaler i TIA överförs till SIMIT. [9]

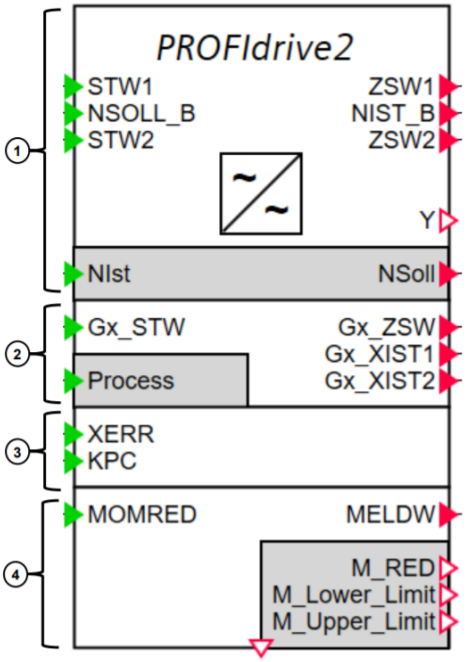
Med PLCSIM Advanced kopplingen skapas en väg mellan SIMIT och TIA via vilken signaler kan överföras. Detta innebär att när en signal blir en etta eller nolla i TIA kommer den även bli det i SIMIT vilket sedan gör det möjligt att sammankoppla en signal från TIA i SIMIT och sedan skicka den till RobotStudio genom SHM (avsnitt 2.5.4). Signalerna kommer samlas i en tabell för in- och utsignaler där det går att se adresser samt om de är true eller false.

2.5.6 PROFINET & PROFIdrive

PROFINET är ett kommunikationsprotokoll som är utformat för att överföra data mellan kontroller och enheter i en automatiseringsmiljö som appliceras på Ethernet i olika industrier. [8] Det används för att ha kommunikation mellan PLCSIM Advanced och SIMIT, framförallt för teknologiobjekten då Profidrive används i SIMIT för att använda signalerna från teknologiobjektet.

Profidrive är en komponent i SIMIT som kommunicerar via PROFINETför att hämta data som skickas via telegram från PLCSIM Advanced. I figur 14 visas hur PROFIdrive2 ser ut i SIMIT.

PROFIdrive1 efterliknar PROFIdrive2, men utnyttjar inte STW2 och ZSW2 signalerna som används för positionsreglering i detta projekt.



Figur 14. Profidrive2 komponent i SIMIT [8].

Figur 14 är uppdelad i 4 olika block, dessa visar beteendet för "PROFIdrive2" och representerar hur det kan se ut för t.ex telegram 105. Blocken är:

1. "PROFIdrive2", kontroll/status signalerna som skickas i en word signal (2 bytes). Detta block skickar är- och börvärdet för hastigheten på teknologiobjektet.
2. "Sensor", kopplingen till encodern.
3. "DynamicServoControl", positionsregulator för driven.
4. "SiemensMomentumReduction", begränsar vridmomentet.

2.6 RobotStudio

RobotStudio är ett program som är utvecklat av ABB. Det är en applikation som är till för 3D-modellering, offline programmering och simulering av robotar . I RobotStudio går det köra i olika lägen, antingen en helt virtuell robot som programmers offline eller så har en verklig robot som är kopplad till en fysisk robot vilket då innebär att användaren onlineprogrammerar. RobotStudio har ett bibliotek med ett antal olika robotmodeller som alla kan köras antingen online eller offline. Kör man online så måste man givetvis ha tillgång till den fysiska roboten.

Programmering av roboten kan uföras på diverse sätt. Till exempel kan roboten programmeras så att den enbart ska gå till vissa riktpunkter eller mönster som den ska följa. Roboten har även ett I/O-system där man kan ha en stor samling av in- och utsignaler som styr roboten beroende på vad signalerna har för värden och hur roboten är programmerad. Allt detta måste sedan överföras till RAPID (avsnitt 2.6.2).

2.6.1 Riktpunkter och mönster

Man kan lägga in riktpunkter till roboten som den ska gå till genom att antingen skriva in koordinater för just den punkten eller genom att man styr roboten med en funktion, som heter freehand, som tillåter att flytta robotens olika axlar i olika riktningar (x, y eller z riktning eftersom att man arbetar med 3D-modeller). Om man styr roboten med freehand så kan man välja själv utan att skriva in koordinater var roboten ska stå och i vilken riktning, sedan använder man sig av en funktion som skapar en riktpunkt där roboten står just nu. Genom att skapa ett flertal riktpunkter för roboten så kan man överföra de punkterna till ett mönster som roboten ska följa och då även skapa instruktioner för hur roboten ska ta sig till de punkterna. När man har skapat ett mönster för roboten så kan användaren själv välja vilken start- och slutpunkt roboten ska ha i just det mönstret.

För att sedan få roboten att följa mönstret man har skapat och flytta sig till punkterna så behöver man synkronisera dem till RAPID (avsnitt 2.6.2).

2.6.2 RAPID

RAPID är språket som robotar programmeras i RobotStudio, antingen offline eller online. Man kan skapa flera moduler i RAPID som man kan anropa i sitt huvudprogram, eller så skriver man koden direkt i huvudprogram beroende på vad man föredrar. Alla mönster som man har skapat kommer att finnas i RAPID-kontrollen där de definieras som en förflyttning från en punkt till en annan genom att använda koordinaterna som har överförts och vad för slags konfiguration roboten ska ha för just den punkten. I RAPID-kontrollen lokaliseras sedan all kod och data som är samlat för roboten.

Man utnyttjar även I/O-systemet här då det finns funktioner som bland annat väntar tills en digital insignal blir ett eller noll, eller sätter en digital utsignal till ett eller noll. Beroende på hur koden är strukturerad kan man då göra väldigt olika saker med hjälp av I/O-systemet. I figur 15 visas ett exempel på hur RAPID-kod kan se ut. Figur 15 visar instruktioner till roboten för att förflyttas till hempositionen.

```
!Hem körning
PROC Home()
    !Path_10;
    !Spara nuvarande position för roboten.
    pMoveHomePos:=CRobT(\Tool:=tGripper\WObj:=wobj0);
    !Sätt Z v?rdet
    pMoveHomePos.trans.z:=pMoveHomePos.trans.z+100;
    !Kör upp till positionen.
    MoveL pMoveHomePos,v200,Fine,tGripper\WObj:=wobj0;
    PHome_Joint:=CalcJointT(pMoveHomePos,tGripper);
    PHome_Joint.robax.rax_1:=0;
    MoveAbsJ PHome_Joint\NoEOffs,v200,z200,tGripper\WObj:=wobj0;
    !Kör till hemma positionen
    MoveL pHome,v500,z0,tGripper\WObj:=wobj0;
    !Sätt HomeposDone
    SetDO doHomePosDone,1;
ENDPROC


ENDMODULE
```

Figur 15. RAPID-kod i RobotStudio

2.6.3 I/O - systemet

I I/O-systemet läggs alla in- och utsignaler in. Här finns sedan olika nätverk som man kan skriva till, till exempel PROFINET, lokala nätverket och ethernet. För robotsignalerna så använder man till exempel “PROFINET Internal Device”.

När man skapar en signal så kan man välja antingen digital in- eller utsignal, analog in- eller utsignal och grupp in- eller utsignal. Signalens device mapping måste bestämmas, vilket innebär vilken bit i I/O-systemets minne som man vill att signalen ska vara definierad till. Detta är viktigt att ha i åtanke när man ska sammankoppla RobotStudio och SIMIT. En digital insignal kan till exempel ha en device mapping på noll eller 20. En gruppsignal är annorlunda då de kan skrivas till flera bits, till exempel vara skriven från bit noll till bit sju eller från bit noll till 14 beroende på hur många bitar eller bytes signalen behöver ta upp.

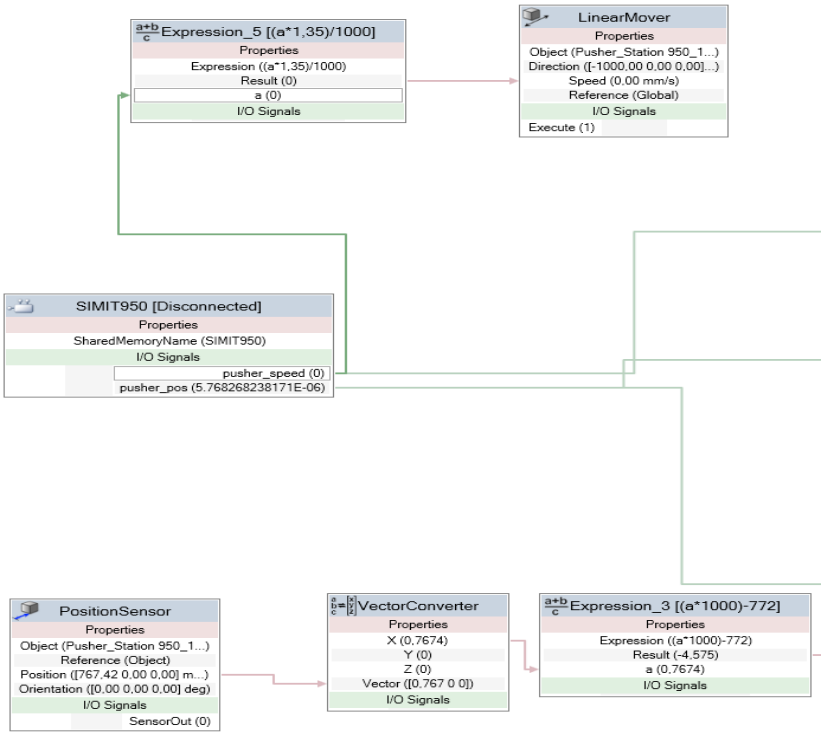
Sammankopplingen mellan I/O-systemen i RobotStudio och SIMIT fungerar genom att använda SHM (avsnitt 2.5.4). Ett SHM skapas i SIMIT och kopplas till RobotStudio genom smartkomponenten “SIMITConnection”. Det leder till att minnet kommer att delas mellan programmen och då kopplas respektive programs I/O-system ihop. Då gäller det att signalerna som motsvarar varandra i programmen och är skrivna på samma adress så att rätt signaler är kopplade till varandra. En signal i RobotStudio adresseras genom att skriva ett värde på device mapping. En digital insignal och en digital utsignal kan ha samma device mapping då det finns varsin bit att använda för in- och utsignaler. Det är för att PROFINET har delat upp adressernaför in- och utsignaler. Till exempel kan en insignal ha device mapping ett och det kan även en utsignal. En device mapping i I/O-systemet i RobotStudio kommer att motsvara en adress i SHM i SIMIT, då är det viktigt att de är skrivna på samma bit. En digital insignal kan skrivas på bit ett i RobotStudio och det behöver den även göras i SIMIT, men där kommer den även skrivas på en byte i minnet (se figur 13). Det gäller då att hitta rätt adress i respektive program för att koppla ihop rätt signal med varandra vilket främst görs genom att skriva dem på samma bit i minnet. Till exempel om en signal i RobotStudio är skriven på bit ett i RobotStudio behöver deb skrivas på adress M0.1 i SIMIT.

2.6.4 Station logic

Station logic är en modul inom RobotStudio som fungerar som ett redigeringsprogram. Där kan man lägga upp ett schema med smartkomponenter och in- och utsignaler. Genom detta går det att styra en simulering på valfritt sätt och behandla signaler. T.ex finns det smartkomponenter som:

- Linear Mover
- SIMIT Connection
- Sensorer och givare
- Logiska grindar

Dessa smartkomponenter används för att efterlikna simuleringen till verkligheten så mycket som möjligt. SIMT Connection används för att få signalerna från PLC:n till RobotStudio och signaler från RobotStudio till PLC:n. Sedan används bland annat Linear Mover för att få en linjär rörelse på t.ex pushern. Sensorer och givare som finns i verkligheten för produktionslinjen placeras i RobotStudio genom att använda smartkomponenter som har linjesensorer och plansensorer som används för att efterlikna sensorerna. I figur 16 visas en station logic i RobotStudio.



Figur 16. Station logic i RobotStudio.

2.6.5 Fysik

RobotStudio används även som en fysikmotor. Denna fysikmotor är det som tillåter paketen att förflytta sig på banden, samt att pushern kan fösa paketen till långsidan av bandet. Objekt i simuleringen kan ha tre olika inställningar:

- Dynamisk (dynamic)
- Fixerad (fixed)
- Inaktiv (inactive).

Det dynamiska betyder att objektet är med i fysiksimuleringen och kan påverkas av andra objekt samt gravitation. Ett exempel på ett dynamiskt objekt är paketen som åker på rullbanden, dels påverkas de av både bandet själv, men även pushern. Objekt som är fixerade är rullbanden, roboten och pushern.

Att vara ett fixerat objekt betyder att ett objekt är med i fysiksimuleringen och kan påverka de dynamiska objekten. Ett fixerat objekt kan dock inte påverkas av annat än deras mekaniska

instruktioner från RobotStudio. Ett paket kan därför inte påverka roboten, rullbanden eller pushern, men alla tre objekt kan påverka paketet.

Objekt som inte har någon betydelse för simuleringen ställs in till inaktiva. Dessa objekt är inte med i fysiksimuleringen och är enbart där för visuell representation. Om ett dynamiskt eller fixerat objekt skulle bemöta ett inaktivt objekt kommer de enbart passera rakt genom detta objekt. Ett konkret exempel på inaktiva objekt är staketen runt omkring roboten i projektet eftersom staketen enbart används för människans säkerhet, människan som då inte finns med i simuleringen.

3 Metod

I detta kapitel kommer val av metod samt processer av examensarbetet att förklaras. De faser som examensarbetet har gått igenom samt källkritik kommer även presenteras i detta kapitel.

3.1 Planering & förstudier

I början av examensarbetet diskuterades och analyserades metoder och planering för hur arbetet ska struktureras för att uppnå det önskade resultatet, vilket är att få fram en fungerande digital tvilling som ska representera den verkliga modellen så mycket som möjligt. Planeringen gjordes mellan framförallt studenterna men även ihop med handledaren från företaget. Handledaren visste vilka program och områden som det förmodligen skulle behöva läggas ner mest tid på då de redan hade erfarenhet av programmen som användes under detta examensarbete. Ett möte tillsammans med ABB och Siemens genomfördes för att få en ingång till deras program. Innan mötet kom företagen med exempel på vad de tyckte att studenterna behövde ha koll på innan ett möte kunde upprättas, det innebar att förstudier av programmen behövdes.

Förstudier ägnades framförallt åt RobotStudio och SIMIT i början av examensarbetet. ABB har handledningar inom grundläggande områden. I programmet kommer även testprojekt som kan studeras för att få en bra förståelse för RobotStudio. I mötet med ABB gick de igenom projekt som de har gjort i RobotStudio som ansågs vara relevanta för detta examensarbete. I SIMIT gjordes förstudier genom att läsa handledningar som förklarade hur sammankopplingen mellan alla programmen fungerade samt vad de olika funktionerna i programmet gör. I mötet med Siemens gick de igenom projekt som de har gjort och grundläggande moment för att ge en bra förståelse för programmet.

3.2 Programmering

Efter förstudierna var det tid för att börja programmera på den riktiga modellen som till slut skulle bli den digitala tvillingen. Då det redan fanns färdig PLC-kod för den digitala tvillingen, behövde det inte läggas ner mycket tid på programmering i TIA. Det som istället behövde göras i programmet var att lära sig hur man navigerar sig i det samt att göra eventuella ändringar i

PLC-koden. En del ändringar skedde i PLC-koden för att få den digitala tvillingen att fungera. Teknologiobjekten i TIA kommunicerade med SIMIT via PLCSIM Advanced kopplingen men även genom PROFINET. Funktionsblocket "ProfiDrive2" användes i SIMIT för att hämta signalerna från TIA som beräknade position och hastighet för t.ex pushern, tillsammans med andra funktionsblock som finns i SIMIT:s bibliotek för charts.

I RobotStudio konstruerades den virtuella miljön för simuleringen av den digitala tvillingen. Detta gjordes genom att använda smartkomponenter i RobotStudio samt anpassning av fysik till olika objekt. I RobotStudio importerades CAD-modeller av den verkliga modellen, se figur 1, från AutoCAD som har designats av personal på Automationsteknik innan examensarbetet började. I RobotStudio importerades sedan en robot från biblioteket i programmet, där alla robotar som ABB har tillverkat finns. Från det biblioteket valdes den robot som är av samma modell som Automationsteknik AB använder i sin verkliga produktionslinje.

I SIMIT skapas en brygga mellan alla program där alla signaler och deras värden passerar för att kommunicera mellan programmen. SIMIT:s kommunikation till RobotStudio görs genom en "Shared Memory (SHM)" (avsnitt 2.5.4) och till TIA genom PLCSIM Advanced (avsnitt 2.5.5). Det behövs också en miljö där signalerna samlas och behöver eventuellt regleras eller styras med hjälp av logiska grindar (avsnitt 2.5.3), se figur 10. För att signalerna skulle kunna samverka och reagera på vad som händer i de olika programmen behövs alla relevanta signaler sammankopplas. Detta gjordes genom att använda I/O-system i respektive program, dels för de signaler som görs i PLC:n för att styra roboten och dels för signaler t.ex som hastighet för rullbanden. Sensorerna skapades i RobotStudio genom att använda smartkomponenten "Line sensor", vilket skapar en sensor som representeras av en rak linje som tillges en längd och vilken position i rummet den ska ha. Signalen som sensorn genererade skickades vidare till PLC:n genom först SHM till SIMIT och sedan till TIA genom PLCSIM Advanced kopplingen. Två sensorer placerades på rullbanden, en lågfartsgivare som signalerar att rullbanden skulle köra långsammare för att undvika en kraftig kollision samt en sensor för att ange när ett paket har ankommit till plockpositionen för roboten vid station 950.

För att roboten ska initiera rörelserna inväntar den en styrsignal, denna styrsignal kommer från PLC-koden. För att genomföra sina uppgifter såsom att plocka ett paket från sin plockposition, kopplades sensorsignalerna vidare till SIMIT som vidarebefordrade signalerna till PLC:n. För att

slutföra uppgiften behövde PLC:ns styrsignaler kopplas vidare till RobotStudio för att fullborda sin process.

Sedan utnyttjades en smartkomponent som heter “Physics Control”, som applicerades på respektive rätblock för att tilldela en ythastighet (surface velocity), som bestäms av PLC:n, samt en riktning i rummet längs en av koordinataxlarna i rummet som går att bestämma själv. Ythastigheten gör att när ett annat objekt kommer i kontakt med ovansidan av rätblocket kommer den transporteras i den riktning och hastighet som har bestämts av smartkomponenten. Rätblocket kan sedan göras osynligt i simuleringen. Rätblock tillverkades i RobotStudio för att efterlikna paketen som Automationsteknik AB använder i verkligheten. Detta gjordes genom att undersöka HMI:t där all paketeringinformation finns för att få rätt dimensioner för paketet

När sammankopplingen av I/O-systemen var klar, behövdes den digitala tvillingen programmeras så att den kunde köras manuellt genom HMI:t, se figur 8. Då den digitala tvillingen körs manuellt innebär det att användaren styr allt. Detta gjordes genom att starta en simulering av HMI:t i TIA och sedan ta ut de nödvändiga signalerna som behövdes för att köra manuellt. Dessa signaler var att köra rullbanden framåt och bakåt, en signal som avgör om det är i auto eller manuellt samt signalen för hastigheten som rullbanden ska köra i beroende på om de kör framåt eller bakåt. När det gick att köra den digitala tvillingen i manuellt läge var nästa steg att få den att köra i autoläge. Det fungerade genom att användaren valde att köra i auto i HMI:t och sedan trycka start och då börjar paket transporteras via rullbanden till plockpositionen för roboten och efter det ska hela processen för den digitala tvillingen köras av sig själv genom PLC:n.

3.3 Presentation av den digitala tvillingen

När programmeringen av den digitala tvillingen var färdig och projektet började närma sitt slut, var det tid för att ha en presentation av hela examensarbetet tillsammans med personalen på Automationsteknik samt för personal på Siemens och ABB. Där presenterade studenterna sitt examensarbete och förklarade hur det hade gått, vad som hade gått bra samt problemen som hade dykt upp. Tankegången kring hela examensarbetet förklarades och det diskuterades vad studenterna tycker det finns för brister i programmen samt förslag från företagen andra metoder för att lösa problemen.

3.4 Källkritik

[1] anses vara trovärdig då det tas upp i ett annat examensarbete och det som tas upp i den rapporten känns relevant till vad detta examensarbete handlar om.

[2] anses vara en trovärdig källa då den kommer från ABB som är utvecklarna av RobotStudio. Är även ett stort multinationellt företag.

[3], [4], [6], [7] & [8] anses vara trovärdiga då de kommer från Siemens AG som är utvecklarna av alla program förutom RobotStudio som har använts i detta examensarbete. Är även ett stort multinationellt företag.

[5] & [9] anses vara trovärdiga källor då informationen som har tagits därifrån är grundläggande och stämmer överens med överblicken angående ämnet som har fåtts från företaget.

[10] anses vara trovärdig då det är en vetenskaplig artikel.

4 Analys

I detta kapitel kommer mål och krav för detta examensarbete att presenteras samt om de har uppfyllts. Analys av problemen som har uppstått under examensarbetets gång och deras lösningar

kommer också att diskuteras. De val som har gjorts för att lösa problemen som har dykt upp kommer att presenteras i detta kapitel.

4.1 Mål & krav

I detta examensarbete gavs en kravspecifikation enligt följande:

- Det ska gå att styra rullbanden manuellt med hjälp av HMI:t
- Allting i den slutgiltiga simulering ska skötas automatiskt med hjälp av PLC-koden.
- Det ska gå att göra olika tester och se vad som händer i simuleringen när ett fel sker.
- Undvika ändringar i PLC-koden

Det första kravet blev uppfyllt då det går att styra rullbanden manuellt med hjälp av HMI:t. Det går att köra den digitala tvillingen i autoläge men det fungerar inte felfritt då det problemen som beskrivs i avsnittet *3.2 Programmering* gör att det inte är helt automatiskt. Tester blev också svårt att göra på grund av de problemen som har nämnts men det går att simulera så att PLC:n skickar ut ett meddelande i HMI:t som meddelar vart ett fel har uppstått. En del ändringar gjordes i koden för att komma fram till det resultat som är idag. Det hade förmodligen gått att lösa utan att göra ändringarna i PLC-koden men med den kunskap och tidsram som fanns så blev detta den bästa lösningen för projektet.

Målet var att ta fram en digital tvilling enligt dessa kraven genom att använda programmen som har beskrivits tidigare.

- Robotstudio, som sköter simuleringen och miljön där allt utspelar sig. Tar även in all data från PLC:n.
- SIMIT, bryggan mellan simuleringen och PLC:n. Alla signaler går genom SIMIT och skickas vidare till de andra programmen.
- TIA, programmering av PLC:n.

4.2 Analys av problemen

Kommunikationen i gränssnittet mellan TIA och SIMIT fungerade oftast felfritt medan det fanns ett större problem i kommunikationen i gränssnittet mellan RobotStudio och SIMIT. I/O-systemen för den digitala tvillingen sammankopplas genom SHM och PLCSIM Advanced kopplingen. Överföringen av signaler mellan RobotStudio och SIMIT fungerade inte som förväntat och det

gick inte att få fram det önskvärda resultatet. Istället blev det en långsam simulering och signalerna för 1000 bandet och pushern, som styrs av teknologiobjekten (avsnitt 2.3.2) i TIA, inte matchade med varandra i RobotStudio och TIA. RobotStudios positioner hade alltid en eftersläpning och ingen lösning hittades på detta problem mer än att implementera en "reset" funktion just för pushern. Denna funktion var ett gäng smartkomponenter som kollade var pushern befann sig i xyz-led och vilken hastighet den hade, om hastigheten var noll och den befann sig rimligt nära dess hemposition. Om detta var uppfyllt så sätts pushern till dess hemposition med hjälp av smartkomponenten "Positioner". Detta behövdes då den alltid hade en felmarginal på några millimeter, en felmarginal som var olinjär och ej hade ett mönster.

När vi började med att överföra teknologiobjektens signaler till SIMIT fick vi till en början inte fram några värden alls på hastighet och position för pushern. I TIA reagerade signalerna på när teknologiobjektet skickade ut att pushern ska flytta på sig men i SIMIT reagerade inte signalerna efter att de hade överförts. Lösningen på detta blev att göra ett chart enligt figur 18 för att få ut värden på hastighet och position för pushern och rullband 1000 som styrs av teknologiobjekt. Detta var en liknande koppling för ett chart som hittades i en handledning från SIMIT ([8]) men med lite förändringar.

När värden för hastighet och position för pushern hade överförts korrekt till SIMIT märktes ytterligare ett problem. Varje gång en simulering startades var alltid startpositionen för pushern noll i SIMIT medan i HMI:t varierade den ofta mellan olika värden som skiljde sig från noll. Vår hypotes för detta var att teknologiobjektet (alternativt profidriven) lagrade positionen, medan SIMIT alltid utgick från noll. Lösningen på detta blev att ändra så att pulsgivaren för teknologiobjektet ändrades till att vara inkrementell istället för cyklist absolut (se avsnitt 2.3.2) för att den ska börja räkna från noll varje gång den startades istället.

När många olika delar och moment är inblandade i fysiksimuleringen och skulle samverka så blev det inte alltid det förväntade resultatet. Ett problem som konstant dök upp var när roboten hämtade ett paket från 950 och sedan placerade paketet på bord 970 eller rullband 1000. PLC:n skickar konstant uppdateringar på positioner som roboten ska hämta och lämna paket, under den process skedde det en del fel. När roboten lämnade ett paket på en av stationerna i RobotStudio trycktes paketet ner för mycket vilket gjorde att paketet flög iväg. Detta försökte vi lösa igenom att sänka alla rullband och bord som paketen placeras på men eftersom att kommunikationen

mellan RobotStudio och SIMIT inte fungerar optimalt fungerade inte det. Problemet grundar sig på att fysiken i simuleringarna i RobotStudio inte alltid fungerade optimalt och på grund av det var det svårt att komma fram till en lösning på detta problem.

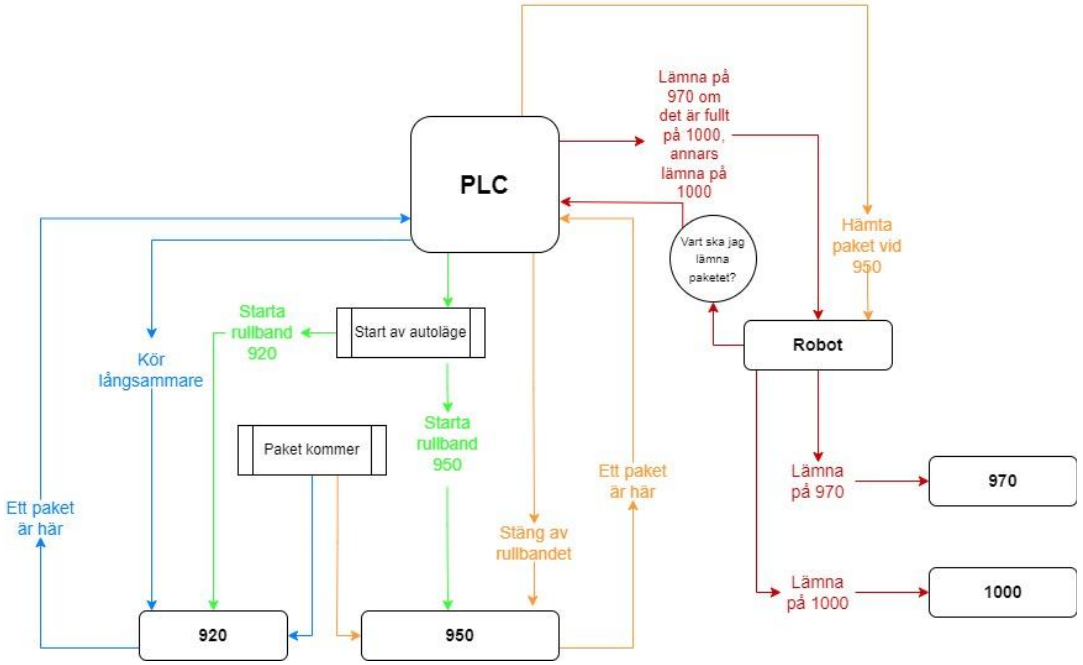
Ibland kunde PLC:n fastna i ett steg på grund av att en signal i en sekvens inte hade det önskade värdet för att gå vidare i processen. Oftast handlade detta om att roboten eller rullbanden behövdes återställas och att allt behövdes köras tillbaka till sina grundpositioner. Det var svårt att avgöra exakt varför detta skedde men det antogs att det grundar sig på att PLC:n läser i sitt minne efter ett värde på en signal och då finns ett värde från en föregående simulering. Genom att återställa allt säkerställer man att minnet har nollställts och att alla signaler har deras grundvärden som de borde ha när en simulering startas. En annan lösning på om PLC:n fastnar i en sekvens är att det går att ändra värdet på en signal manuellt i TIA till det önskade värdet så att PLC:n går vidare till nästa steg i sekvensen. Problemet med det dock är att man inte kör den digitala tvillingen i ett helt automatiskt läge längre.

Ett problem som dök upp var att signalerna för bland annat hastigheten för rullbanden inte kom med när man gjorde en PLCSIM Advanced koppling mellan TIA och SIMIT. Det var för att i PLC-koden som hade gjorts av Automationsteknik AB så hade de signalerna inga adresser, vilket gjorde att de inte kom med i kopplingen mellan programmen. För att lösa det skapades nya signaler som först kopplades till de redan befintliga signalerna som fanns, och sedan skrevs de på en ledig adress så att de kom med i kopplingen. Ett problem som tillkom till detta var dock att de nya signalerna hoppade mellan det önskade värdet och två andra värden som inte kunde förklaras var de kom ifrån. Signalens värde hoppade mellan olika värden hela tiden vilket innebär att det blev som ett brus på signalen. Detta löstes genom att använda olika funktionsblock i ett chart som begränsade signalen till att ligga i ett område som låg mellan noll och det önskade värdet och så fort det låg utanför de gränserna sattes det till den undre eller övre gränsen, se figur 10 för att se hur lösningen ser ut. I I/O-systemet för SIMIT där alla signalers adresser och värden finns(när man är online) gick det att kontrollera om signalernas värde ligger inom det önskade området.
Ett annat problem som uppstod i kopplingen mellan RobotStudio och SIMIT var att SHM smartkomponenten i RobotStudio inte kan ta emot negativa värden. Negativa värden används då rullbanden ska köra baklänges i manuellt läge. Detta löstes genom att göra en chart som i figur 10 och sedan genom att i RobotStudio multiplicera den signalen med minus ett igen efter att RobotStudio har tagit emot signalen.

Innan paketen når station 950, finns där ett par stopp som ska hindra paketen från att åka vidare på rullband 920. Detta är för att stoppa inkommande paket när roboten och pushern är upptagna med ett tidigare paket för att undvika krockar med andra paket. Figur 19 visar hur den mekaniska princip ser ut för stoppen i detta projekt. Fysiken i en simulering i RobotStudio ej optimal, därför gick det inte att få med stoppen på ett önskvärt sätt i en simulering.

5 Resultat

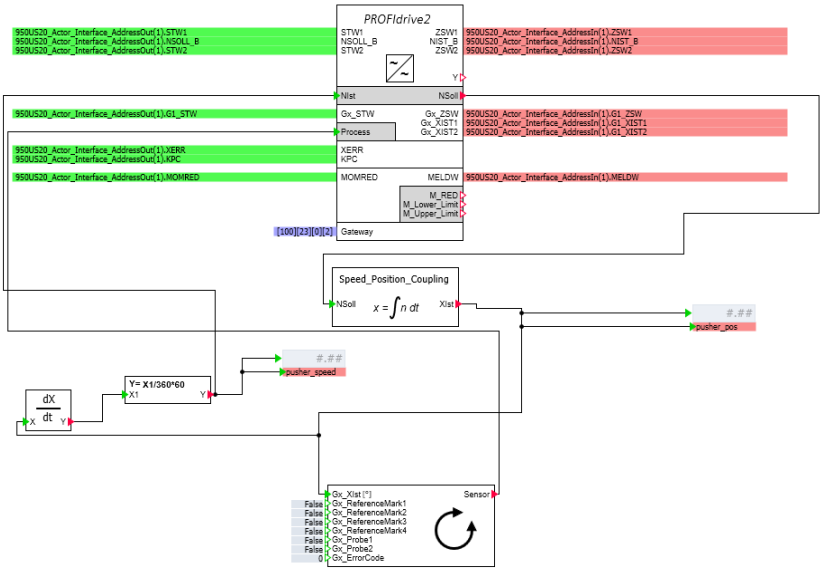
I figur 17 går det att se ett förenklat flödesdiagram för hela processen när det är i autoläge. Det är PLC:n som styr allt beroende på vad den får för signaler från Robotstudio. Linjer som är i samma färg motsvarar tillståndsövergångarna i samma sekvens.



Figur 17. Förenklat flödesdiagram för den digitala tvillingen i autoläge.

Först startas autoläge i HMI:t och sedan sätter PLC:n igång rullband 920 med en hastighet och rullband 950 med en annan hastighet. Sedan kan ett paket komma och då börjar det på 920 tills det kommer till en lågfartsgivare på rullbandet som säger till PLC:n att ett paket är här och PLC:n svarar med att 920 ska köra långsammare. Sedan kommer paketet till 950 och det vet den då det finns en lågfartsgivare i slutet av rullbandet och det skickas till PLC:n och den svarar med att 950 ska köra långsammare. Strax efter lågfartsgivaren på 950 finns det en sensor som känner av om ett paket är vid den, då skickas det till PLC:n och den svarar med att stänga av 950. PLC:n skickar sedan till roboten att den ska hämta paketet vid 950. Efter roboten har hämtat paketet skickar den en förfrågan till PLC:n var den ska lämna det. Då finns det två utfall, det ena som kan hända är roboten lämnar paketet på rullband 1000 om det finns plats och om rullbandet är aktiverat och fungerar. På 1000 rullbandet får det plats med tre paket jämte varandra i sidled. Det andra som kan hända är att om det antingen är fullt på 1000 eller om det inte fungerar, då ska roboten lägga paketet på 970 bordet. PLC:n och roboten har koll på hur många paket som finns på bord 970 och 1000 genom att i PLC:n finns det en räknare för båda borden som räknar uppåt varje gång ett paket lämnas där samt räknar neråt när ett paket hämtas från 970. Efter att roboten har lämnat ett paket på antingen 970 eller 1000 går den tillbaka till sitt utgångsläge. Denna process sker då det inte har uppstått något fel t.ex att roboten slutar fungera eller om något av rullbanden stängs av.

Genom att göra ett chart som visas i figur 18 nedan gick det att få ut värden för hastighet och position för pushern och rullband 1000 i SIMIT. Ett positions teknologiobjekt ger ut en position och i vilken hastighet som den rör sig med mot den önskade positionen. I figur 18 nedan ser man en PROFIdrive2 (avsnitt 2.5.6) som tillsammans med de andra komponenterna fungerar som ett reglersystem. Från PLC:n skickas ett börvärde till PROFIdrive2 som sedan tas vidare till "Speed_Position_Coupling" som beräknar vilken position som den är på just nu, ärvärdet. Ärvärdet tas sedan vidare till en komponent som deriverar signalen med avseende på tiden och för att få ut den linjära hastigheten behövs komponenten direkt efter som först dividerar signalen med 360 och multiplicerar med 60. Efter det skickas signalen tillbaka till PROFIdrive2 för att få ut reglerfelet. Genom detta kan systemet sedan veta när den har kommit till rätt position då reglerfelet är lika med noll.



Figur 18. Koppling för pushern i SIMIT i ett chart.

Programmen kommunicerar med varandra genom att en simulering startas i SIMIT. En instans startas samtidigt i PLCSIM Advanced och på den kan man koppla upp sitt PLC program från TIA. På detta sätt kan SIMIT få signaler från PLC:n och där används sedan SHM för att lagra signalerna. Via detta minne går det att skicka signaler mellan RobotStudio och SIMIT genom smartkomponenten "SIMIT SharedMemory" som finns i RobotStudio. Signalerna kopplas genom att para ihop en adress i det gemensamma minnet med rätt bit i RobotStudios I/O-system. I/O-systemet i RobotStudio kommunicerar med det gemensamma minnet mellan RobotStudio och SIMIT via Profinet. Som det har nämts tidigare i denna rapport fungerar inte SHM optimalt

mellan RobotStudio och SIMIT. Detta gör att det var svårt att styra t.ex pushern på ett realistiskt sätt då den ibland kunde förflyttas längre än sitt mekaniska stopp. I verkligheten finns det väggar på rullband 950, som ligger vid kanten på långsidan av rullbandet, som pushern flyttar sig mellan. I RobotStudio kan pushern gå igenom dessa väggar då de har satts till att vara inaktiva i simuleringen. Detta baseras på att kommunikationen mellan RobotStudio och SIMIT fungerar till en viss grad vilket gör att pushern inte rör sig exakt mellan sina programmerade stopp. SHM klarade av att föra över digitala signaler på ett bra sätt men när det handlar om en signal som konstant uppdateras blir det fel, t.ex en hastighet eller position. RobotStudio klarar inte av att ta emot det uppdaterade värdet på signalen som SIMIT skickade hela tiden vilket gjorde att simuleringen i den virtuella miljön är långsammare än vad värdena som skickas från TIA och SIMIT är. Det börjar med att teknologiobjektet i TIA får signaler till sig som säger att den ska röra sig till en viss position som skickas från PLC:n, det skickas sedan till SIMIT och sedan till RobotStudio. I SIMIT (och TIA) kör pushern med en hastighet men på grund av att SHM kopplingen inte fungerar optimalt så kommer RobotStudio inte upp i den hastigheten lika snabbt utan ligger alltid lite efter. Detta leder till att när pushern ska stå stilla enligt SIMIT så kan den fortfarande flytta på sig i RobotStudio. Detta gör att pushern kan gå förbi sina ändpunkter i simuleringar vilket är ett stort problem, då pushern kan då nästa gång den ska köras starta från ett annat läge än sin startposition som har programmerats i PLC:n. Detta leder till att pushern inte då kommer uppfylla sin funktion med att flytta paket intill kanten på rullband 950 vilket kan göra att roboten sedan inte kommer hämta det på rätt position. Det är ett fel som blir värre efter varje delmoment i proccessen.. Detta diskuterades med ABB och Siemens då de lyssnade på presentationen av arbetet och även då sa de att det är ett problem som de vet om. Kommunikationen mellan RobotStudio och SIMIT är ett område som båda företagen jobbar på att förbättra, och i denna stund så är det inget som fungerar optimalt vilket gör det svårt att få fram det önskade resultatet.

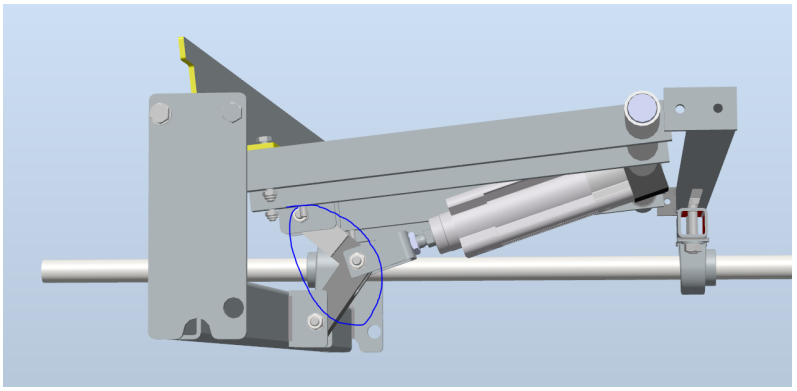
När simuleringen är igång och man försöker visa ett fel, t.ex som att ett paket aldrig kommer fram till slutet av 950 kommer PLC:n reagera på det och skicka ut ett felmeddelande. För detta fall finns det en timer som går igång när PLC:n är i den sekvens då den har skickat till roboten att den ska hämta ett paket vid 950. När timern har gått en viss tid innan den får en återkoppling från roboten att paketet har hämtats så skickas en felsignal som visas som ett felmeddelande sedan på HMI:t. Detta går att simulera genom att när en simulering har startats och ett paket är på väg till 950 så kan man ta bort det paketet från rullbanden mitt i simuleringen.

Eftersom allt som finns i den virtuella miljön är CAD-modeller förutom roboten, så är det inga rullband som kan användas i RobotStudio utan enbart modeller som går att applicera fysik på (se avsnitt 2.6.5). För att ett paket ska transporteras på rullbanden används en funktion i RobotStudio som skapar objekt, antingen ett rätblock eller en kub, som går att dimensionera på önskvärt sätt. Det blev ett rätblock som dimensioneras i rummet på så vis att det blev lika långt och brett som rullband 920 och 950. Dessa rätblock fick en fixerad fysikinställning (avsnitt 2.6.5) för att de ska kunna vara delaktiga i simuleringen men enbart i en konstant position under simuleringen.

Då roboten ska hämta ett paket och sedan placera det antingen på bord 970 eller rullband 1000 uppstår det problem. I RobotStudio går det att välja vilket material som paketet ska vara gjort av. Det går dock inte att välja att det ska vara av papperskartong vilket är vad Automationsteknik AB använder i verkligheten. Detta gör att två hårda objekt trycktes mot varandra och tillslut flyger paketet iväg. Detta problem grundas på att RobotStudio inte representerar en verklighetstrogen fysik i just detta fall. Detta sågs redan vara ett problem under ett av mötena som vi hade med ABB då de visade ett projekt som de hade gjort i RobotStudio. De visade en process där föremål flyttades via ett rullband, samt en annan process där en robot ska flytta ett objekt och sedan placera det på en annan position. De hade samma problem som har uppstått i detta examensarbete när en robot tar ett objekt och lägger det på t.ex ett bord och att det sedan flyger iväg vid direkt kontakt. De visste om att det var ett problem i programmet och att det är något som de jobbar med att förbättra i RobotStudio. Detta ses därför som ett problem som är svårt att göra något åt utan lämnas åt sidan för tillfället.

Tidigare i rapporten har det nämts att fysiken i RobotStudio är begränsad vilket gör att det inte gick att lösa problemet med stoppen. I figur 19 visas det hur dessa stopp ser ut i RobotStudio. Den mekaniska rörelsen som önskades få fram är inringat. Det fungerar på det sätt att den är cylinderkolv som trycks ut genom att luft trycker ut den. Det ska sedan leda till klossarna som sitter ihop ska förflytta sig i samma riktning som cylinderkolven trycks ut, vilket gör att stoppet ska åka uppåt. För fysiken i RobotStudio var detta alldeles för mycket. Ett flertal experiment genomfördes av oss studenter och av personal på Automationsteknik AB men ingen av oss kunde lösa detta i RobotStudio. Det blev sen även bekräftat av personal på ABB att den mekaniska rörelse vi strävade efter var svår att simulera i RobotStudio. Som resultat av detta använde vi inte oss av några stopp överhuvudtaget, då hela poängen med implementeringen av dessa stopp var att

få fullt fungerande mekaniska stopp. Tester genomfördes enbart på denna CAD-modell som finns i figur 19 och inga andra CAD-modeller eller liknande mekaniska rörelser testades utan det valdes att istället tilldela ett objekt en fysik. Då detta inte gick att lösa på ett önskvärt sätt togs de bort istället för att användas då RobotStudio inte klarar av den fysiska rörelsen med alla delar som ska vara inblandade.



Figur 19. Stoppet i RobotStudio.

Ett negativt värde skickades från PLC:n då ett rullband kör baklänges, detta löstes genom att lägga in ett funktionsblock i SIMIT som först multiplicerade signalen med minus ett innan den skickades till RobotStudio. Sedan när den har tagits emot av RobotStudio och ska styra rullbanden multipliceras signalen med minus ett igen innan den bestämmer hastigheten för rullbandet i smartkomponenten “Physics control”. Se figur 10 för att se hur det ser ut i en chart i SIMIT. I överföringen av signaler mellan RobotStudio och SIMIT skickas enbart positiva värden men när ett rullband körs baklänges vet man att den digitala tvillingen är i manuellt läge. Genom detta vet man då att den digitala tvillingen styrs av HMI:t och då kommer en knapp i HMI:t trigga en signal som säger att rullbandet ska köra baklänges. Signalen som blir triggad skickas till RobotStudio för att veta om rullbandet ska köra baklänges, vilket är när signalen är en etta och då vet man att den digitala tvillingen är i manuellt läge.

Då det finns I/O-system i respektive program med ett visst antal signaler så behövdes de sammankopplas och mötas upp på ett och samma ställe. Det gjordes genom att använda gränssnitten som beskrivs i *2.5.4 Shared Memory* och *2.5.5 PLCSIM Advanced koppling.* Dessa gränssnitt tillät I/O-systemen att mötas i SIMIT där sammankopplingen av signalerna sker, se figur 10. I SIMIT gjordes ett I/O-system med signaler för SHM och PLCSIM Advanced kopplingen med signaler som är kopior av dem som redan finns i RobotStudio och TIA. T.ex

gjordes ett I/O-system för SHM som används för att skicka sensorsignalerna till SIMIT, medan det finns ett annat I/O-system i SIMIT som används för att skicka robotsignalerna från TIA via PLCSIM Advanced kopplingen. De två gränssnitten som har nämnts, möts i SIMIT sedan i ett chart (se avsnitt 2.5.3) för att sammankoppla signalerna.

6 Slutsats

I detta kapitel kommer den problemformulering som skrevs i inledningen av rapporten att reflekteras över. Framtida utvecklingsmöjligheter av projektet kommer att diskuteras och sist följer en reflektion över etiska aspekter.

6.1 Reflektion av problemformulering

1. Hur skapar man en digital tvilling?

En digital tvilling går att ta fram på flera olika sätt. Det behövs en virtuell miljö där man har den virtuella kopian av det verkliga objektet. Sedan behövs indata och beteende hos det verkliga objektet som går att ta fram med hjälp av programmering och sensorer och givare bland annat.

2. Hur simuleras servoaxlarna som styr pushern och det ena rullbandet?

Servoaxlarna styrs av teknologiobjekten i TIA och kommunicerar via PROFINET med SIMIT för att ta emot signaler från PLC:n och sedan för att sedan skicka vidare dem till RobotStudio. Det innebär att alla programmen behövs för att simulera servoaxlarna och för att få det i en sluten krets där alla program kommunicerar med varandra.

3. Hur ska alla mjukvaruprogram kommunicera med varandra?

Programmen kommunicerar med varandra genom att SIMIT startar en simulering som även startar en uppkoppling i PLCSIM Advanced. I TIA laddar man sedan upp hela PLC programmet där alla sekvenser, teknologiobjekten och HMI:t finns. För att få tillgång till signalerna för PLC:n i SIMIT behöver man hämta PLC programmet genom att använda en PLCSIM Advanced uppkoppling som gör att alla signaler som är skrivna på en adress i TIA går att föra över till SIMIT. I SIMIT kan man sedan använda ett minne som kan användas gemensamt med Robotstudio, då det finns en smartkomponent i Robotstudio som utnyttjar det gemensamma minnet och hämtar signalerna som finns där från SIMIT. När alla uppkopplingar är färdiga så kan alla programmen kommunicera med varandra och då är SIMITen brygga mellan simuleringen i Robotstudio och PLC:n i TIA. Se figur 2 för en visualisering av hur programmen kommunicerar med varandra.

4. Vad för slags fel går att simulera med den digitala tvillingen?

Den digitala tvillingen klarar av att simulera fel som när ett paket försvinner och när roboten eller något av rullbanden slutar fungera.

6.2 Framtida utvecklingsmöjligheter

Det finns stor potential för framtida utvecklingsmöjligheter inom detta projekt. Detta examensarbete har endast arbetat med en del av företagets arbetslinje och en robot. Det är lättare att göra en digital tvilling på något man har mycket kunskap om och data på men har man lyckats med ett projekt som detta, så går det att utveckla en digital tvilling på mer omfattande objekt. Företaget kan erbjuda sina tjänster och göra en digital tvilling åt andra då detta är ett område som ständigt kommer utvecklas och fler företag inom framförallt automationsteknik kommer satsa på inom de närmsta åren och ett tag framåt. Det finns stora utvecklingsmöjligheter för denna teknik inom många olika områden.

En digital tvilling kan användas bland annat i arbetsområden som infrastruktur, medicin och läkemedel, automationsteknik, tillverkningsverksamheter och till stationer för förnyelsebara energikällor. Det kan även användas inom byggnad, planering och reglering av städer. Inom alla dessa områden används digitala tvillingar för att underlätta design, planering och förbättring av systemen Är ett företag tidigt ute med att börja använda sig av detta kan det leda till många samarbetsmöjligheter med andra stora företag för att hjälpa dem med att utveckla en digital tvilling.

Att jobba med en digital tvilling kan innebära besparning av tid och pengar om man jobbar på ett smart sätt med den. Det är en smart arbetsteknik då man kan utnyttja att man har stor kontroll under en simulering och kan utföra tester och felsökningar i den virtuella miljön istället för att göra det i den verkliga. Detta leder till att man sparar pengar och tid för att på ett snabbare sätt hitta fel som man sedan kan åtgärda direkt på den verkliga roboten.
Det finns sätt att utveckla just detta projekt på och förbättra det. Som det har nämnts i denna rapport tidigare har det funnits svårigheter med att sätta upp en bra och snabb kommunikation mellan alla program då de har utvecklats av olika företag. Kommunikationen hade förmodligen fungerat bättre om ett program som NX MCD från Siemens hade använts istället för Robotstudio. Detta då MCD är utvecklat av Siemens och anses att programmet kan fungera bättre tillsammans

med SIMIT och TIA. Sedan kan det även utvecklas på det sätt att man gör en digital tvilling på ett större område än vad detta projekt har handlat om. Automationsteknik AB har en stor verkstad med flertal robotar och rullband, ett mål i framtiden kan vara att göra en digital tvilling på hela verkstaden. Detta kan leda till att de sparar tid och pengar på att kunna göra simuleringar och tester med alla sina robotar, göra felsökningar och pröva nya arbetsområden för deras verkstad. Företaget åker runt till olika företag för bland annat konsultarbeten och för att installera robotar hos dem, detta hade kunnat effektiviserats genom att använda sig av digitala tvillingar. Genom att göra en digital tvilling för ett annat företag kan de göra tester på den och få en glimt av hur det objektet kan se ut i deras miljö. Detta är gynnsammt för båda företagen och är effektivt, snabbt och utgör en modern arbetsmetod.

6.3 Reflektion över etiska aspekter

Detta projekt kan redan nu vara till stor nytta för ett företag som Automationsteknik. Detta är ett ekonomiskt- och tidssnålt arbetssätt för företaget då de kan spara tid och pengar på att göra felsökningar och tester på den digitala tvillingen istället för att göra det på den riktiga roboten och produktionslinjen. Med den digitala tvillingen kan man göra tester för att bland annat hitta nya arbetsområden och möjligheter till att utveckla produktionslinjer. Att arbeta med digitala tvillingar har redan påbörjats av vissa företag inom automation- och reglerteknik bland annat. Detta kommer vara det stora arbetsätt för företag som Automationsteknik att jobba på i framtiden då branschen kommer att digitaliseras mer.

Om flera branshcer hade tagit till sig att jobba med digitala tvillingar hade det strävat mot ett mer hållbart samhälle. Det är en miljövänlig arbetsteknik då man för den mesta delen jobbar i en virtuell miljö vilket inte påverkar den yttre världen. Digitala tvillingar behöver inte enbart appliceras på robotar eller hos företag som jobbar inom automation, det kan även appliceras på företag som jobbar med fastigheter och annan teknik som fordon och vitvaror. På en digital tvilling kan man hitta vart ett fel uppstår på en robot och se vilken komponent det är som t.ex behöver bytas ut.

Att jobba med digitala tvillingar kan hjälpa till med att utveckla sjukhus och medicinsk innovation. Det går att göra en digital tvilling av ett sjukhus, det kan man göra för att effektivisera byggnaden för att effektivisera. Bland annat genom att se till så att kapaciteten utnyttjas till max

och för att effektivisera bemmanings- och vårdmodeller. Genom att jobba med digitala tvillingar kan man sträva mot snabbare medicinsk innovation, undersökning om olika sjukdomar och läkemedel och sänka material- och medicinkostnader och öka patientsäkerheten. Användning av en digital tvilling för just detta leder till att människor i behov kan få hjälp snabbare och för att utveckla forskningen kring medicinteknik. [11]

En nackdel med att jobba med digitala tvillingar kan vara att allting lagras på en dator och i olika program. Om man inte har ett effektivt system med avseende på hur man sparar sina filer eller en tillräckligt bra dator att jobba på kan detta leda till att slutresultatet påverkas negativt. Datorn måste ha bra prestanda då det kan vara krävande att göra en digital tvilling, t.ex kan ett bra grafikkort och en bra processor vara nödvändigt för att starta ett sådant här projekt. Att jobba med digitala tvillingar handlar mycket om internetuppkoppling och säkerheten för uppkopplingen. Finns det inte en säker internetuppkoppling kan det innebära att det finns faror för ens arbete.

6.3.1 Sveriges ingenjörers hederskodex

Som ingenjör har man ett ansvar att se till att arbeten sköts enligt Sveriges ingenjörers hederskodex. Ingenjörer är bärare och förvaltare av den tekniska kunskapen. Detta ger ett särskilt ansvar att verka för att tekniken används för samhällets och mänsklighetens bästa och för att den i förbättrad form förs vidare till kommande generationer. Hederskodexen omfattas av tio punkter och i detta examensarbtete har speciellt följande punkter följts:

- Ingenjören bör i sin yrkesutövning känna ett personligt ansvar för att tekniken används på ett sätt som gagnar människa, miljö och samhälle.

- Ingenjören bör sträva efter att förbättra tekniken och det tekniska kunnandet i riktning mot ett effektivare resursutnyttjande utan skadeverkningar

7 Terminologi

Rummet - matematiskt uttryck för att beskriva en 3D-miljö
P-regulator - ett regleringssystem med enbart en proportionerlig(förstärkande eller försvagande) del

Reglerfel - skillnaden mellan bör- och ärvärdet

I/O-system - ett system med in- och utsignaler

CAD - digitalt baserad design och skapande av tekniska ritningar

SHM - Shared Memory

TIA - Totally Integrated Automation

Drive - objekt som används för att styra en motor genom att skicka lagrad data till motorn för att styra hastigheten

Drivsystem - System som består av en drive och en motor

SIMATIC - En funktion i TIA som används till bland annat positionsreglering

Teknologiobjekt - En funktion i TIA som underlättar styrning av motorer bl.a

Smartkomponent - En funktion i RobotStudio som kan representera bl.a en sensor eller utföra en funktion så som rörelser och matematiska operationer.

8 Källförteckning

[1] Edig. (23 augusti 2017). *Virtuell driftsättning Smarta Fabriker - examensarbete*. https://odr.chalmers.se/bitstream/20.500.12380/250419/1/250419.pdf [Hämtad: 2021-01-28]

[2] ABB. (26 november 2021). *Operating Manual - RobotStudio*. https://search.abb.com/library/Download.aspx?DocumentID=3HAC032104-001&LanguageCode=en&DocumentPartId=&Action=Launch [Hämtad: 2021-01-28]

[3] Siemens AG. (2017). *The Technology Objects (TO) SIMATIC S7-1500(T)*. https://cache.industry.siemens.com/dl/files/134/109743134/att_928448/v2/109743134_S7-1500T_TechnologyObjects_DOC_v10_en.pdf [2021-02-10]

[4] Siemens AG. (Juni 2018). SIMIT Simulation Platform(V10.0). Operating Manual. https://cache.industry.siemens.com/dl/files/317/109759317/att_956837/v1/SIMIT_enUS_en-US.pdf [Hämtad:2021-02-10]

[5] Siemens AG. (Januari 2013). *Shared Memory Gateway - User Manual.* https://cache.industry.siemens.com/dl/files/351/73136351/att_111459/v1/SIMIT_SHM_Gateway_e.pdf [Hämtad: 2021-02-15]

[6] Nelly Ayllon, PI North America. (10 februari 2021). *What is Profinet? - Profinet explained*. https://us.profinet.com/profinet-explained/ [Hämtad: 2021-02-30]

[7] Siemens AG. (December 2019). S7-1500/S7-1500T Axis functions V5.0 in TIA Portal V16. Function Manual. https://support.industry.siemens.com/cs/attachments/109766462/s71500_s71500t_axis_function_manual_en-US_en.US.pdf [Hämtad: 2021-03-30]

[8] Siemens AG. (April 2020). *SIMATIC Machine Simulator V3.0, Getting Started - Virtual commissioning of machines.* https://cache.industry.siemens.com/dl/files/943/109758943/att_1024310/v1/s71500_simatic_machine_simulator_getting_started_v3.0_EN.pdf [Hämtad: 2021-04-10]

[9] Siemens AG. (27 november 2017). *SIMATIC S7-1500 S7-1500T Motion Control V4.0 in TIA Portal V15*.
https://support.industry.siemens.com/cs/mdm/109749263?c=104218926091&lc=en-DE [Hämtad: 2021-9-17]

[10] IBM. *What is a digital twin?*
https://www.ibm.com/se-en/topics/what-is-a-digital-twin [Hämtad: 2021-10-30]