Examensarbete

HÖGSKOLAN I SKÖVDE

LOKAL STANDARDISERING AV PLC PROGRAMMERING VERIFIERAD MED EMULERING

LOCAL STANDARDIZATION OF PLC PROGRAMMING VERIFIED WITH EMULATION

Examensarbete inom huvudområdet automatiseringsteknik
Grundnivå 15 högskolepoäng
Vårtermin år 2019

Författare: Joakim Ramn & Marcus Steneros Hermansson

Handledare Högskolan i Skövde: Mikel Ayani
Handledare Projektengagemang: Michael Johansson
Examinator: Amos Ng

Joakim Ramn
Marcus Steneros Hermansson

Äkthetsintyg

Denna examensrapport är inlämnad av Joakim Ramn och Marcus Steneros Hermansson till Högskolan i Skövde för examen vid institutionen för Ingenjörsvetenskap. Härmed intygas att allt material i denna rapport är vårt eget. Tydliga referenser ges till material som hämtats från annat håll.

Skövde 30/5 2019

Ort och datum

Signatur Joakim Ramn

Signatur Marcus Steneros Hermansson

Sammanfattning

Det finns idag ingen standard som beskriver hur tydliga program med självförklarande kod och bra struktur utformas. Företag som arbetar med programmering och levererar automatiserade processer skulle med fördel kunna nyttja en tydlig standard för att effektivisera sin process. För att undvika variation i programmeringsstrukturen som finns idag krävs det en lättillgänglig standard som är baserad på det bästa kända arbetssättet.

Arbetet utfördes på Projektengagemang och målet var att föreslå en lokal standard för PLC programmering att rätta sig efter vid programuppbyggnad. Standarden skulle enkelt kunna kompletteras i efterhand och vara baserad på Projektengagemangs tidigare projekt. Syftet med arbetet var att minimera variation, öka kvaliteten, underlätta upplärning och effektivisera Projektengagemangs PLC programmering.

Metoden Design and creation research strategy användes för att utföra arbetet. Processen började med att tydliggöra problemet och ta fram eventuella lösningsförslag för att sedan utveckla en preliminär lösning. När en preliminär lösning är framtagen påbörjas arbetet med att utveckla och verifiera artefakten som är en databaserad produkt, för att kontrollera att en korrekt produkt utvecklas för att sedan gå vidare till att validera, utvärdera och sammanställa resultatet.

Standarden som skapats innehåller ett bibliotek med funktionsblock, en projektfil i TIA portal där funktionerna styrs och en tillhörande emuleringsmodell för verifiering. Resultatet validerades två gånger, en gång av artefaktens utvecklare och en gång av Projektengagemang som användare av standarden. Efter utvärderingen kunde det bekräftas att de standardiserade funktionerna kan vara användbara men med viss komplettering. De standardiserade funktionerna kan underlätta långsiktigt vid felsökning då programmet får en känd struktur och ledtiden kan minskas vid programmering då en del av programkoden är klar och inga nya funktionsblock behöver skapas. Standarden kan även hjälpa till att minska variationen och öka kvaliteten genom återanvändning av standardiserade funktioner samt underlätta en upplärningsprocess. Finns grundläggande kunskaper i Simumatik3D är det möjligt att komplettera standarden på ett enkelt sätt.

Abstract

There is currently no standard that describes how well-defined programs with self-explanatory code and good structure are designed. Companies that work with programming and deliver automated processes could advantageously utilize a well-defined standard to streamline their process. To avoid variation in the programming structure that exists today, an easily accessible standard is required which is based on the best known working method.

The work was carried out at Projektengagemang and the goal was to propose a local standard for PLC programming to comply with program building. The standard could easily be supplemented retrospectively and be based on Projektengagemangs previous projects. The purpose of the work was to minimize variation, increase quality, facilitate tutoring and streamline Projektengagemangs PLC programming.

The method Design and creation research strategy were used to carry out the work, the process started by clarifying the problem and developing possible solution proposals in order to then develop a preliminary solution. When a preliminary solution is developed, the work continues with developing and verifying the artefact, which is a data-based product, to check that a correct product is being developed and then proceed to validate, evaluate and compile the result.

The standard created contains a library of function blocks, a project file in TIA portal where the functions are controlled and an associated emulation model for verification. The result was validated twice, once by the artifact's developer and once by Projektengagemang as a user of the standard. After the evaluation it could be confirmed that the standardized functions can be useful but with some supplementation. They can facilitate long-term troubleshooting when the program gets a known structure and the lead time can be reduced during programming since parts of the program code is ready and no new function blocks need to be created. The standard can also help reduce variation and increase quality through the reuse of standardized functions and facilitate a learning process. If you have basic knowledge of Simumatik3D, it is possible to supplement the standard in a simple way.

Innehållsförteckning

Äkthetsintyg i
Sammanfattning ii
Abstract iii
Figurförteckning vi
1. Introduktion 1
1.1 Företagspresentation 1
1.2 Problembeskrivning 1
1.3 Avgränsningar 1
1.4 Mål och syfte 2
1.5 Hållbar utveckling 2
1.6 Disposition 3
2. Metod 4
2.1 Design and creation strategy 4
2.2 Metodapplicering 5
3. Teoretisk referensram 6
3.1 Standarder och standardisering 6
3.1.1 Standarder genom historien 6
3.1.2 Möjligheter med standarder 7
3.2 Vad är en PLC 7
3.2.1 Användning av en PLC 8
3.2.2 PLC standard 9
3.2.3 Programmeringsspråk 9
3.2.3.1 Instruction list 10
3.2.3.2 Structured text 10
3.2.3.3 Ladder diagram 10
3.2.3.4 Function block diagram 11
3.2.3.5 Sequential function chart 11
3.2.4 Sammanfattning programmeringsspråk 12
3.3 Simulering och emulering 12
3.4 Verifiering och validering av modeller 13
3.5 Virtuell driftsättning 13
3.6 Simumatik3D 15
3.7 TIA portal 15
3.8 Sammanfattning teoretisk referensram 15

4. Genomförande 16
4.1 Tillvägagångssätt 16
4.2 Utrustning 17
4.3 Kommunikation 18
4.4 Val av funktioner 21
4.5 Uppbyggnad av standarden 22
4.6 Konstruktion av emuleringsmodell 30
4.7 Verifiering av standarden 38
5. Utvärdering 41
5.1 Validering 1 42
5.1 Validering 2 43
5.2 Sammanställning av valideringar 44
6. Diskussioner 45
6.1 Projektets helhet 45
6.2 Metoden 45
6.3 Konstruktion av emuleringsmodell 45
6.4 Begränsningar i Simumatik3D 45
7. Slutsatser 47
7.1 Framtida arbete 47
7.2 Återkoppling hållbar utveckling 47
7.3 Utvärdering av syfte och mål 48
7.3.1 Mål 48
7.3.2 Syfte 48
Referenser 49

Figurförteckning

Figur 1. Design and creation i fem steg. 5
Figur 2. Användning av olika styrsystem beroende på styrningens komplexitet. 9
Figur 3. Exempel på hur instruction list skrivs. 10
Figur 4. Exempel på hur structured text skrivs. 10
Figur 5. Exempel på hur ladder diagram skrivs. 11
Figur 6. Exempel på hur function block diagram skrivs. 11
Figur 7. Exempel på hur sequential function chart skrivs. 11
Figur 8. Minskad driftsättningstid. 13
Figur 9. Fyra kombinationer att arbeta med driftsättning. 14
Figur 10. Tillvägagångssätt för att verifiera funktioner. 17
Figur 11. Överblick över kommunikationen. 18
Figur 12. Arbetets parametrar för den virtuella PLC enheten. 19
Figur 13. Fönster för OPC UA inställning. 20
Figur 14. Parametrar för PLC enheten i Simumatik3D. 20
Figur 15. Kommunikation mellan datorer och mjukvaror. 21
Figur 16. Olika sorters block i TIA Portal. 23
Figur 17. Struktur på taglista i TIA portal. 24
Figur 18. Strukturen av standarden i TIA portal. 24
Figur 19. Biblioteket med standardiserade funktioner. 24
Figur 20. Styrning av Motordrift FB. 25
Figur 21. Styrning av Rörelse FB. 26
Figur 22. Styrning av Motordrift ET200pro FB. 27
Figur 23. Styrning av Grind FB. 28
Figur 24. Styrning av Detaljräkning FB. 29
Figur 25. Styrning av Cykeltider FB. 30
Figur 26. Systemet testar motordrift. 31
Figur 27. Cylinder med stoppfunktion. 31
Figur 28. 2D layout av systemet. 32
Figur 29. System med bana 1, vridbord och bana 2. 32
Figur 30. Fullständigt system. 33
Figur 31. Hissfunktion i Simumatik3D. 34
Figur 32. HMI panel för detaljräkning. 34
Figur 33. Flödet av en icke godkänd detalj. 35
Figur 34. OK detalj. 35
Figur 35. NOK detalj. 35
Figur 36. Ljusfyr. 36
Figur 37. Skyddskrets HMI. 36
Figur 38. Dörr med låsfunktion. 37
Figur 39. Manuellfunktioner bana 1 och vridbord. 37
Figur 40. Manuellfunktoner bana 2 och hiss 37
Figur 41.Projektträd med grupperingar. 38
Figur 42. In- och utgångar i Simumatik3D. 38

1. Introduktion

Dagens industrier har höga krav från kunden som kräver låga kostnader, snabba leveranser och hög kvalitet på produkterna. Dessa krav gör att industrin kontinuerligt måste utveckla sina processer att bli effektiva, resurssnåla och flexibla, men det gör även att utvecklingen av dessa processer måste göras på ett snabbt och effektivt sätt (Dahl *et al.*, 2016; Bondeson and Liss, 2018). Inom tillverkningsindustrin har målet med utveckling och förbättring av system varit att uppnå en högre ekonomisk vinning och uppnått detta genom en högre nivå av automatisering i sina processer (Bondeson och Liss, 2018).

En fjärdedel av tiden vid implementering av nya processer är driftsättning, 60–70% av driftsättningen består av programmeringskod som behöver förbättras och justeras (Bondeson och Liss, 2018). För att snabbt och effektivt kunna leverera och implementera nya system är det viktigt att det finns en tydlig standard för att underlätta Programmable Logic Controller (PLC) programmering och tydliggöra dess struktur. Det finns idag generella standarder inom PLC programmering som innehåller beskrivning av en PLC enhets hela livscykel däribland vilka typer av programmeringsspråk det finns men den beskriver inte hur tydliga program med självförklarande kod och bra struktur utformas.

Företag som arbetar med att leverera automatiserade processer och programmering skulle med fördel kunna nyttja en tydlig lokal standard för att effektivisera sin process. En lokal standard syftar på att den ska användas på det enskilda företaget, detta eftersom alla företag har olika kunder med olika automationsprocesser och standarden måste således anpassas individuellt för varje företag. Den föreslagna standarden som lämnas i detta arbete är inget som kan kopieras för att användning på andra företag som behöver en programmeringsstandard, däremot finns det en tydlig metod och ett tillvägagångssätt att följa vid skapandet av en ny programmeringsstandard.

1.1 Företagspresentation

Projektengagemang är ett företag vars affärsidé bygger på att leverera konsulttjänster inom arkitektur, industri och energi, installation samt bygg och anläggning som skapar ett värde för både kund och samhälle. Företaget grundades 2006 och är idag verksamma på 37 orter i Sverige med mer än 1200 anställda. Inom divisionen industri och energi arbetar Projektengagemang med att göra kundens process mer lönsam, hållbar och konkurrenskraftig. Kunderna är återkommande inom försvar, livsmedel, fordon, energi och verkstadsindustri. Projektengagemang tar bland annat på sig teknikroller inom industrin vid maskinanskaffning, där de hjälper andra företag med elektronik och mekanik ur ett underhållsperspektiv. De är även aktivt deltagande i för och efterbesiktning av maskinerna. Automatisering och automationsprojekt är ett område som Projektengagemang aktivt arbetar med, där de bistår med automationsingenjörer och projektledare.

1.2 Problembeskrivning

Projektengagemang har idag ingen egen standard för deras programmerare, men känner att behovet ökar eftersom de är i en expanderande fas och anställer många nya automationsingenjörer. Eftersom de ofta är flera ingenjörer som arbetar tillsammans i samma projekt finns det många åsikter om hur en funktion eller ett program ska programmeras. För att undvika variation i programmeringsstrukturen som finns idag krävs det en lättillgänglig standard som är baserad på det bästa kända arbetssättet.

1.3 Avgränsningar

- Förslaget som lämnas till Projektengagemang är en påbörjad standard som kontinuerligt bör uppdateras och kompletteras i efterhand för att uppnå full utnyttjandegrad.
- Fördefinition av vilka och hur många funktioner som standardiseras specificeras inte utan detta görs med Projektengagemangs personal och kommer att kunna variera beroende på tid.

- Emuleringsmodellen skapas utifrån de funktioner som ska verifieras i modellen och är inte baserad på ett verkligt system.
- Examensarbetet kommer att begränsas till funktioner i Siemens mjukvara TIA portal.
- Ingen undersökning av vilka emuleringsmjukvaror som finns tillgängliga genomförs.

1.4 Mål och syfte

Huvudmålet med arbetet är att föreslå en lokal standard för PLC programmering att rätta sig efter vid programuppbyggnad. Standarden ska enkelt kunna kompletteras i efterhand vid behov och vara baserad på Projektengagemangs tidigare projekt. Syftet med arbetet är att minimera variation, öka kvaliteten, underlätta upplärning och effektivisera Projektengagemangs PLC programmering. För att förtydliga detta huvudmål kan det delas upp i 3 delar.

- Föreslå en lokal standard för PLC programmering att rätta sig efter vid programuppbyggnad.
- Föreslå en lokal standard för PLC programmering som enkelt kan kompletteras i efterhand.
- Föreslå en lokal standard för PLC programmering som är baserad på Projektengagemangs tidigare projekt.

1.5 Hållbar utveckling

Det är inte längre en fråga om när eller om hållbarhet måste integreras i vardagen utan snarare hur den ska integreras (Jonker och Harmsen, 2012). En viktig definition av hållbarhet gavs av Bruntlandkommissionen i FN-dokumentet *Our common future* som med den svenska översättningen kallas *Vår gemensamma framtid* (Brundtland och Hägerhäll, 1988).

> *”En hållbar utveckling är en utveckling som tillfredsställer dagens behov utan att äventyra kommande generationers möjligheter att tillfredsställa sina behov.”*

Viktiga faktorer som nämns i rapporten är ekonomi, teknik, miljö och den sociala situationen (Gulliksson och Holmgren, 2018). Det finns inte oändligt med resurser i världen och frågan om hållbarhet är därför väldigt aktuell i alla branscher men ingenjörer har en speciell nyckelroll vid strävan att uppnå hållbar utveckling eftersom många lösningar är tekniska (Gröndahl och Svanström, 2011). Tekniken ses som den näst viktigaste faktorn för att nå ett hållbart samhälle vilket också märks i två delmål av FN:s hållbarhetsmål (SDG) (United Nations, 2018).

> *”17.7 Främja utveckling, överföring och spridning av miljövänlig teknik till utvecklingsländerna på gynnsamma villkor, inklusive koncessions och förmånsvillkor, på villkor som överenskommits mellan parterna.”*

> *”17.8 Senast 2017 fullt ut operationaliserat teknikbanken och mekanismen för kapacitetsuppbyggnad inom vetenskap, teknik och innovation för de minst utvecklade länderna samt öka användningen av gynnsam teknik, i synnerhet informations- och kommunikationsteknik.”*

Teknik och ingenjörsarbete är starkt förknippade med miljö, social och ekonomisk utveckling. Det är tekniken som till stor del måste kompensera för den globala standardökningen av befolkning som av sociala skäl är ett måste för stora delar av jordens befolkning (Gulliksson och Holmgren, 2018). Ingenjörer jobbar överallt i samhället med olika tekniska lösningar men det mesta av deras arbete sker på industrier med tillverkning av produkter. Globaliseringen och ökad öppenhet, tack vare bättre kommunikation, inspirerar och tvingar industrin att ta steg för steg mot hållbarhet. Detta gör att ingenjörer har en central roll i att skapa en hållbar utveckling (Jonker och Harmsen, 2012). Ett sätt ingenjörer kan påverka sitt arbete är att jobba med standardiserade metoder. Delaktighet och inflytande i arbetet med framtagande av standarder är nödvändigt för att tillgodose samhällets krav

på säkerhet, effektivitet och hållbar utveckling. Om det finns ett effektivt deltagande i standardiseringsverksamheten påverkar det konkurrenskraften och därmed nationell tillväxt. Sverige är ett litet exportberoende land och det är viktigt att utvecklingen av det standardiseringsarbetet som sker i vår omvärld stämmer överens med svenska intressen och prioriteringar (Regeringskansliet, 2015).

1.6 Disposition

Introduktionen innehåller bakgrunden till arbetet, en företagspresentation av Projektengagemang, arbetets problembeskrivning, vilka avgränsningar som fastlagts men även arbetets mål och syfte. Introduktionen innehåller även viktig information om hållbar utveckling och hur arbetet med standardisering är kopplat till en hållbar utveckling.

Metodkapitlet definierar vad en metod är för något och varför en metod ska användas. Det beskriver även den metod som används i arbetet och tillvägagångssättet för att nå ett önskvärt resultat.

Den teoretiska referensramen innehåller information som är användbar för läsare som inte är insatta i ämnet. Här beskrivs bland annat standarder och standardisering, vad en PLC är för något samt hur den används, skillnaden på verifiering och validering samt tydliggör vad simulering och emulering är.

Genomförandekapitlet innehåller hela beskrivningen för hur arbetet har genomförts. Hur kommunikationen mellan mjukvaror fungerar, uppbyggnad av standarden i TIA portal, hur emuleringsmodellen konstruerades och hur standarden verifierats.

Utvärderingen beskriver tillvägagångssättet för validering av standarden och innehåller en utvärdering av valideringen där resultatet sammanställs.

Diskussionen framför tankar och åsikter kring projektets helhet, användning av den valda metoden, vad det finns för begränsningar i Simumatik3D och hur det kan förbättras.

Slutsatsen innehåller en del om hur framtida arbete med standarden kan utföras och vad det finns för fördelar med att arbeta med standarden och tillhörande emuleringsmodell, samt en del där utvärdering av resultatet mot mål och syften görs.

2. Metod

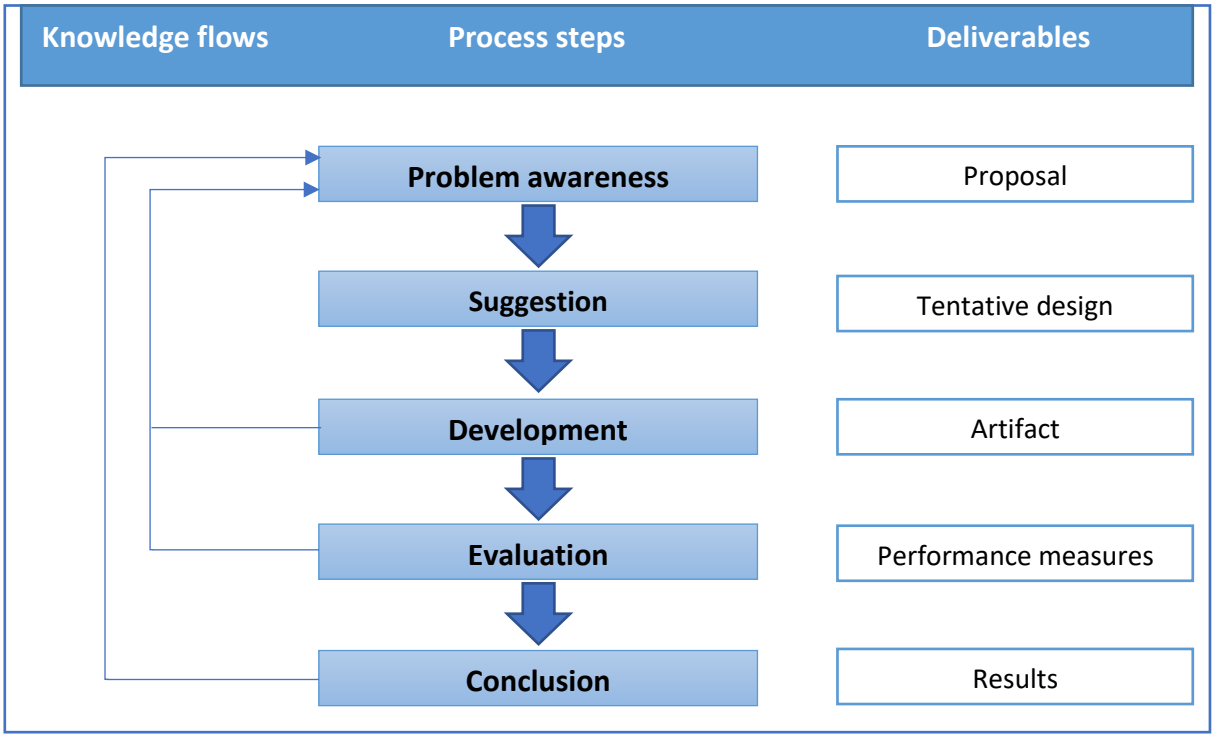
Nationalencyklopedin (2019) definierar metodik och metodologi som ett vetenskapligt tillvägagångssätt för att lösa problem, uppnå givna mål och generera ökad kunskap.

Ett forskningsarbete som undersöker möjligheterna med digital teknologi innehåller oftast analysering, design och utveckling av en datorbaserad produkt som till exempel en datoranimation, emuleringsmodell eller simuleringsmodell (Oates, 2006). Men för att sådant arbete ska anses vara forskning och inte bara en illustration av teknisk framgång ska det enligt Oates (2006) innehålla akademiska kvalifikationer såsom analyser, argumentation, bestyrkning av arbetet och noggrant granskade utvärderingar. Forskningsarbete behöver även generera en ökad kunskap. Oates (2006) definierar tre olika arbetskategorier med fokus på digital teknologi för att ge ökad kunskap:

- Forskningsprojekt med en datorbaserad produkt som är huvudmålet och är därmed bidraget till ökad kunskap.
- Forskningsprojekt som genererar en datorbaserad produkt, men bidraget till kunskap ligger i hur slutkunden använder och nyttjar produkten.
- Forskningsprojekt som genom projektet skapar en färdigutvecklad datorbaserad produkt men bidraget till kunskap är utvecklingsprocessen och hur produkten uppnått målet.

2.1 Design and creation strategy

Design and creation research strategy är en forskningsmetod som inriktar sig på att utveckla nya datorbaserade produkter, dessa datorbaserade produkter kallas för artefakter. Metoden kan användas vid utveckling av flera olika typer av artefakter, men måste baseras på etablerade principer för systemutveckling. Normalt sätt används Design and creation som en problemlösningsmetod och är en iterativ process i fem steg, vilket betyder att stegen upprepas cykliskt tills rätt resultat uppnås. Steg ett i processen är att tydliggöra problemet och ta fram eventuella lösningsförslag för att sedan gå vidare till steg två med ett av förslagen och börja utveckla en preliminär lösning. I steg tre utvecklas och verifieras artefakten för att kontrollera att en korrekt produkt utvecklas. I de två sista stegen valideras artefakten mot ursprungsproblemet för att kontrollera att rätt produkt har utvecklats för att sedan sammanställa och utvärdera resultatet (Oates, 2006). Vaishnavi och Kuechler (2004) beskriver de fem stegen enligt figur 1.



Figur 1. Design and creation i fem steg.

Oates (2006) beskriver att efter utveckling av en artefakt så krävs en evaluering som testar om artefakten uppfyller de kriterier som krävs för att bli funktionsduglig. Vilka kriterier som ska uppfyllas beror på varför de utvecklades från början och det tänkta användningsområdet, exempel på kriterier är noggrannhet, funktionalitet, prestanda och tillgänglighet. Det finns två metoder för att testa ett system ur en teknisk aspekt, Black box och White box. Om ett program eller funktions innehåll inte behöver undersökas noggrant utan fokuserar på om det ger förväntat funktionalitet på systemet kan Black box metoden användas. När White box används undersöks programkoden för att testa olika lösningar för att uppnå ett specificerat resultat (Oates, 2006).

2.2 Metodapplicering

Arbetet har genomförts med metoden Design and creation strategy men med viss modifikation för lättare applikation. Den artefakt som skapas är ett bibliotek med standardiserade funktioner och en projektfil i TIA portal där styrningen av funktionerna sker. Detta kompletteras med en emuleringsmodell med syfte att verifiera funktionerna. PLC koden är testade enligt Black box metoden då funktionerna är programmerade och levererade av Projektengagemang. Funktionerna är verifierade genom funktionstester i emuleringsmodellen och en verifierad funktion läggs till i biblioteket med standardfunktioner som Projektengagemang kan återanvända. Bidraget till kunskap i detta arbete förväntas att vara utvecklingsprocessen.

3. Teoretisk referensram

3.1 Standarder och standardisering

År 1922 grundades svenska industrins standardiseringskommission som utvecklar, säljer och utbildar i standarder. SIS har bytt namn flera gånger men valt att behålla förkortningen och 1931 antog de namnet Sveriges standardiseringskommission på grund av att de inte längre jobbar med enbart industriell standardisering. År 2019 är namnet Swedish Standard Institute (SIS) och är Sveriges representanter i både den europeiska standardiseringskommissionen CEN och den globala organisationen International Standard Organization (ISO). SIS beskriver en standard som en rutin eller arbetssätt och en gemensam lösning till ett återkommande problem. En standard implementeras för att uppnå högre kvalitet, undvika missförstånd och slippa återuppfinna hjulet (SIS, 2019b, 2019a).

Standardisering kan enligt Brunsson och Jacobsson (2000) delas in i tre kategorier:

- Standarder för *vad vi gör:*

Dessa standarder beskriver till exempel hur ledarskap ska gå till, utförande i produktionsprocesser, vilka kurser som ska finnas i olika utbildningsprogram samt vett och etikett i sociala sammanhang.

- Standarder för *vad vi har:*

Dessa standarder refererar till vad en organisation borde ha, till exempel tydliga mål och en HR (Human resource) avdelning. De innefattar även standarder inom FN (förenta nationerna) om behovet att ha till exempel demokrati, regering och utbildningssystem.

- Standarder om *att vara något:*

Dessa standarder handlar om att benämna saker på samma sätt. Standarden beskriver vad en telefon eller dator måste innehålla och ha för funktion för att få kallas just telefon eller dator. Även det metriska systemet och Celsius temperaturskala är en sådan typ av standard.

ISO grundades 1947 genom att deltagare från 25 länder träffades för att samordna och ena industriella standarder. ISO grundades med tankesättet om att hitta det bästa sättet att genomföra en uppgift. År 2019 är 163 länder medlemmar i ISO och har standarder som täcker allt från livsmedelssäkerhet, sjukvård, lantbruk och i stort sett alla typer av industrier. ISO beskriver en standard som en världsklassig specifikation för service, system och produkter för att säkerhetsställa kvalité, säkerhet och effektivitet (ISO, 2019a).

3.1.1 Standarder genom historien

Genom tiderna som den teknologiska utvecklingen gått framåt har den efterföljts av standardutvecklingen. Hawkins, Mansell och Skea (1995) menar att standarder är oundvikliga i produktioner där hög kvalité och stabilt flöde efterfrågas. Hawkins, Mansell och Skea (1995, s.1.) definierar en teknisk standard:

> *”By the most basic definition, technical standards are agreed external points of reference to which the physical and performance characteristics of technologies can be compared”*

Fred Winslow Taylor skapade i början på 1900-talet en grund för att hitta det bästa sättet att utföra en uppgift genom nya tekniker, såsom tid och rörelsestudier. Taylor separerade planering och produktion vilket skapade tillverkningsingenjörer, som genom sina nya tekniker, kunde utveckla ett standardiserat arbetssätt för produktionen. Taylors arbetssätt om att hitta det bästa sättet att utföra en uppgift kom senare att kallas Taylorismen (Dennis, 2016).

Under samma tidsepok som Taylor, arbetade Henry Ford med att designa en bil som var lätt att tillverka och reparera. Dennis (2016) skriver att det inte var monteringslinan som var nyckeln till Fords massproduktion utan hans arbete med utbytbara och standardiserade komponenter som var lätta att montera som gjorde att massproduktionen var möjlig. Det var efter att Ford standardiserat komponenterna som han anammade Taylorismen och förflyttade komponenterna till arbetaren och därmed skapade löpandebandet. Cykeltiderna som mättes 1908 var på flera timmar hade minskats ner till några minuter i Fords nya monteringslina 1913. Under 1920 producerade Ford 2 miljoner bilar per år och hade minskat kostnaden för kunden med två tredjedelar. Fords arbetssätt och standardisering lockade även andra intressenter, däribland Eiji Toyoda från The Toyota Motor Company, som besökte Fords fabrik i Rouge, Detroit. Eiji Toyoda tillsammans med Taiichi Ohno kom senare att skapa The Toyota Production System, som utanför Toyota kallas för Lean som många tillverkande företag fortfarande jobbar efter (Dennis, 2016).

3.1.2 Möjligheter med standarder

Fördelarna med att standardisera sin process, sitt arbetssätt och sina produkter är många. Bicheno *et al.* (2013) menar att implementera standardiserat arbete handlar i grund och botten om att skapa repetitiva procedurer och processer som är pålitliga, kapabla och ligger till grund för förbättring. En välstrukturerad standard är en standard som är så pass bra att det känns självklart att utföra uppgiften på det sättet och dumt att genomföra den på något annat sätt (Bicheno *et al.*, 2013). Det finns vissa saker som är viktiga att förstå när det gäller standardiserat arbetssätt och Bicheno *et al.* (2013) beskriver det i tre nyckelaspekter:

1. Standardiserat arbete är inte oföränderligt, utan när bättre metoder hittas uppdateras standarden.
2. Standardiserat arbete främjar stabilitet genom minskad variation och underlättar även att hitta dessa variationer.
3. Standardiserat arbete är en central och nödvändig del i förbättringsarbete och processen i att kunna utveckla sina metoder till de bättre.

Oavsett inom vilket område en standard implementeras är syftet att effektivisera en specifik uppgift, metod, process eller produktion. Standardiserat arbetssätt implementeras i produktionen för att uppnå högre effektivitet och en mer resurssnål process (SIS, 2019b). Den metriska mätskalan och Celsius temperaturskala standardiserades för att undvika missförstånd och standardisera mätning (Brunsson och Jacobsson, 2000). ISO skapade standarder för bland annat kvalitetsstyrning och livsmedelshantering för att säkerhetsställa att produkter upprätthåller kvalitet som möter kundens förväntningar (ISO, 2019b). International Electrotechnical Commission (IEC) utvecklade IEC 61131 för hantering av en PLC enhets livscykel och underlätta för ingenjörer att programmera (PLCopen, 2013a; Bolton, 2015).

3.2 Vad är en PLC

För att kunna styra maskiner och andra automatiserade utrustningar används ett styrsystem som hanterar ingående signaler från givare, tryckknappar, sensorer och utgående signaler till reläer, kontaktorer, frekvensomriktare för att styra motorer, ventiler och ställdon. Detta styrsystem är ofta en PLC.

Den första PLC enheten utvecklades 1969 men före implementeringen av PLC kopplades styrningen med fysiska reläkretsar med NO (Normalt öppen) och NC (Normalt stängd) kontakter för att få önskad funktion. Nackdelarna med detta var att flexibiliteten var oerhört liten eftersom kablaget behövde kopplas om varje gång en funktionsändring skulle ske. Komponenterna blev även fler och mer plats i apparatskåpet behövdes för att få plats med relästyrningen (Bolton, 2015). Om en befintlig

relästyrning byts ut mot ett styrsystem som en PLC kan ofta en stor platsbesparing i apparatskåpet göras. Fördelarna med en PLC styrning jämfört med relästyrning är enligt Bolton (2015):

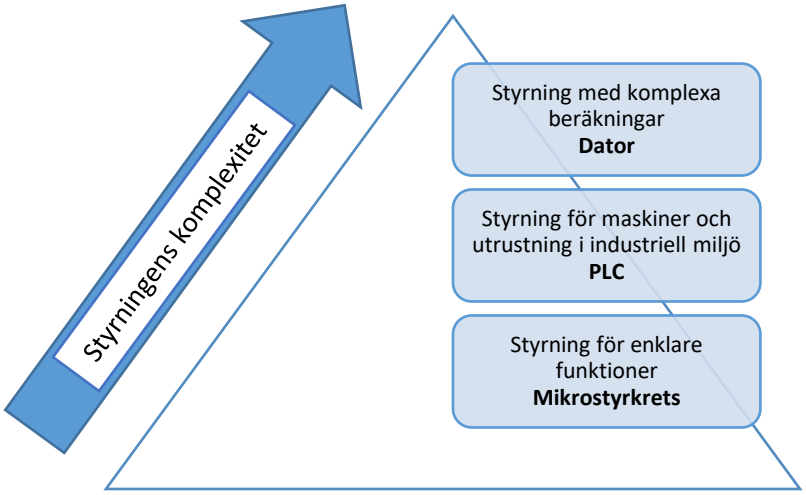
- Det är enkelt att implementera nya funktioner och ändringar.
- Kan enkelt expanderas genom att sätta dit fler moduler (om en moduluppbyggd PLC används).
- Den är mer robust och pålitligt i en industriell miljö.
- Den är mer kompakt och exekverar instruktioner snabbare.
- Den kräver mindre underhåll.

En PLC är baserad på en mikroprocessor och använder ett programmerbart minne för att lagra instruktioner och för att implementera funktioner som logik, sekvenser, räknare och tidtagning (Bolton, 2015). Den är tillverkad för användning av ingenjörer men gör sig ofta även förstådd av elektriker då programmeringen ofta kan likna reläkretsar. En PLC har likheter med en dator men medan datorer är optimerade för uträkningar och visningsuppgifter så är en PLC gjord för industriell miljö och att kontrollera specifik data (Bolton, 2015).

En PLC är ofta uppbyggd av en Central processing unit (CPU), en strömförsörjningsenhet, en minnesenhet, en programmeringsenhet, ett interface för in- och utsignaler (I/O) och ett interface för kommunikation. En CPU innehåller en mikroprocessor för att tolka insignalerna och utföra styrningsåtgärder enligt det program som är lagrat i minnesenheten. Strömförsörjningsenheten behövs för att konvertera den inkommande växelströmmen (AC) till den likström (DC) som krävs för processorer och kretsar. Programmeringsenheten används för att skriva in det önskade programmet till minnesenheten. Processorn hämtar information från externa enheter genom I/O sektionen, de signaler som förekommer vid in och utsignaler är analoga eller digitala. En digital signal skickar en spänning som vid ett visst minimivärde ses som en binär etta av PLC enheten, om spänningen är lägre än detta minimivärde ses signalen som en binär nolla av PLC enheten. Vid en analog signal är spänningen proportionell mot den variabel som används, om en temperatur mäts kommer det att skickas en spänning som är proportionell mot temperaturen som mäts (Bolton, 2015).

3.2.1 Användning av en PLC

Idag är användningen av PLC enheter stor men den används inte vid alla applikationer. Många industriella maskiner behöver en lättanvänd och säker styrning och då är styrning med en PLC en av de vanligaste styrningarna (ABB, 2019). Vid enklare styrningar som används i hemmabruk, exempelvis styrningen till en tvättmaskin så används ett enklare och billigare styrsystem som en mikrostyrkrets. Vid mer krävande och komplexa styrningar som exempelvis styrningen till ett flygplan så används en dator då den har möjlighet att göra mer avancerade matematiska beräkningar med snabbt gensvar (Bolton, 2015). I figur 2 tydliggörs användning av styrsystem beroende på styrningens komplexitet.



Figur 2. Användning av olika styrsystem beroende på styrningens komplexitet.

3.2.2 PLC standard

Komponenter som innehåller och använder mikroprocessorer såsom en PLC kräver goda programmeringskunskaper. Men eftersom en PLC är en komponent som är tänkt att användas av ingenjörer som inte har expertkunskap inom programmering utvecklades programmeringsspråket Ladder (Bolton, 2015). De flesta PLC tillverkare tog till sig detta programmeringsspråk men utvecklade en egen version av det. Därav utvecklades 1993 standarden IEC 1131–3 som nu går under namnet IEC 61131–3 och är delen som fokuserar på programmeringsspråk. Standarden innefattar programmeringsspråken Ladder Diagram (LAD), Sequential Function Chart (SFC), Structured Text (ST), Instruction List (IL) och Function Block Diagram (FBD), dessa beskrivs mer ingående i kapitel 3.2.3. Programmeringsspråk (PLCopen, 2013a; Bolton, 2015)

IEC 61131 standarden innefattar hela livscykeln av en PLC och har 9 delar (PLCopen, 2013b; Bolton, 2015):

- Generella definitioner och grundläggande terminologi.
- Elektroniska och mekaniska utrustningskrav samt verifieringstest.
- Programmeringsspråk.
- Användarmanual.
- Mjukvara som behövs för kommunikation mot andra utrustningar.
- Kommunikation genom fältbussar.
- Fuzzy control programming.
- Riktlinjer för implementering av programmeringsspråken.
- I/O Link, kommunikation med sensorer och ställdon.

3.2.3 Programmeringsspråk

För programmering av en PLC finns det många olika mjukvaror och språk. För att underlätta framtida programmering och undvika att inte för många språk används, skapades standarden IEC 61131–3 som är den tredje delen av IEC 61131 standarden. Standarden specificerar bland annat fem programmeringsspråk och hur de kan användas. Instruction list (IL), Structured text (ST), Ladder diagram (LD), Function block diagram (FBD) och Sequential funtion chart (SFC), av dessa fem är tre grafiska språk och två är textbaserade. Ett program kan använda moduler som är skrivna i dessa fem språk beroende av behov (Ramanathan, 2014). För att illustrera och jämföra skillnaderna i de olika språken kommer ett exempel visas i en figur under varje språkbeskrivning. Dessa exempel använder samma funktion, det är en AND funktion som kräver att alla ingångar är sanna för att utgången ska bli

sann. Tabell 1 visar en sanningstabell för AND funktionen som används, detta för att visa skillnaderna på de olika språken nedan.

Tabell 1. Sanningstabell för AND funktion.

Output	Input_1	Input_2
0	0	0
0	1	0
0	0	1
1	1	1

3.2.3.1 Instruction list

Instruction list är ett enkelt textbaserat lågnivåspråk som skrivs som en kedja med instruktioner. IL använder mindre utrymme än LD och exekveras därför också snabbare. Språket har varit mycket populärt i Europa och kan även programmeras med en handhållen enhet. Det är passande att använda sig av IL när det är en enkel funktion som ska upprepas många gånger i projektet. Det är inte passande att använda IL vid komplexa projekt eftersom det är svårt att förstå programmets flöde utan utförliga kommentarer. Det kan även vara problematiskt att konvertera språk från IL till grafiska språk (Ramanathan, 2014). Endast en instruktion kan skrivas per rad och nästa instruktion specificeras på nästa rad. Ett exempel på hur IL skrivs visas i figur 3.

```
LD Input_1
AND Input_2
OUT Output
```

Figur 3. Exempel på hur instruction list skrivs.

3.2.3.2 Structured text

Structured text är ett textbaserat språk som liknar C och Pascal som är högnivåspråk. Programmeringen är mer flexibel än vad som är möjligt med grafiska språk som LD. Den största fördelen med ST är att komplexa program kan delas in i mindre delar vilket gör att ST är mindre benäget för fel och enkelt att felsöka i. Det är även lämplig att använda ST för sortering och hantering av data (Ramanathan, 2014). Ett exempel på hur ST skrivs visas i figur 4.

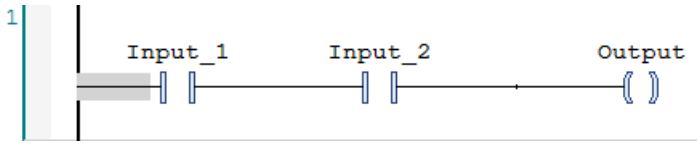
```
1  WHILE Input_1 AND Input_2 DO
2  Output:=1;
3  END_WHILE
```

Figur 4. Exempel på hur structured text skrivs.

3.2.3.3 Ladder diagram

Ladder diagram är ett grafiskt lågnivåspråk med symboler för att underlätta visualiseringen av programmeringen. Detta är ett av de äldsta och mest traditionella programmeringsspråken till PLC. Det är designat för att efterlikna reläkretsarna som PLC enheten skulle ersätta och blev därför accepterat runt om i världen tack vare sin likhet till de elektriska kretsarna. Elektriker, underhållspersonal och annan berörd personal som inte har en djupgående kompetens inom programmeringsspråk kan utan större svårigheter läsa, programmera och felsöka med hjälp av det grafiska språket. Ladder diagram har fått sitt namn efter att det har samma struktur som en stege där stegen är elektriska kretsar som passerar genom ett eller flera villkor för att utgången ska bli sann.

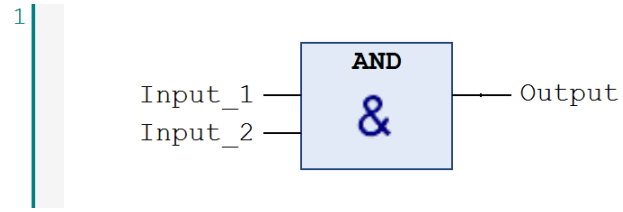
Varje steg kallas för ett nätverk och utförs från vänster till höger, uppifrån och ner (Ramanathan, 2014). Ett exempel på hur LD skrivs visas i figur 5.



Figur 5. Exempel på hur ladder diagram skrivs.

3.2.3.4 Function block diagram

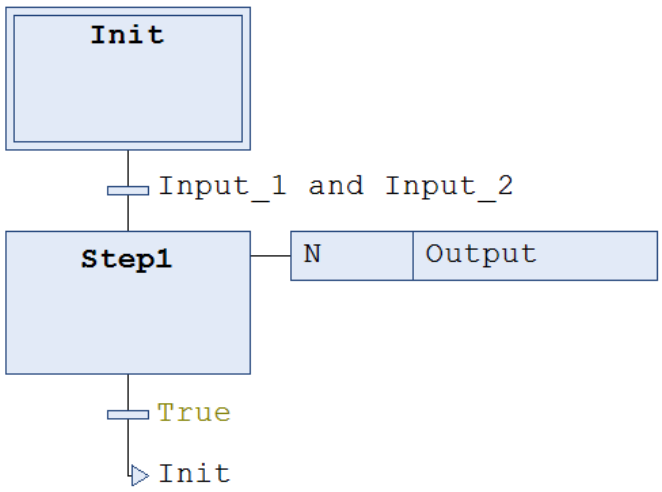
Function block diagram är ett grafiskt språk som består av funktionsblock som kopplas till ingångar, utgångar och andra funktioner. Det liknar elektroniska kretsar på det sätt att du kopplar ihop funktioner och funktionsblock med horisontella och vertikala linjer. En bra egenskap hos FBD är att det kan skrivas och kombineras tillsammans med de andra IEC 61131–3 språken (Ramanathan, 2014). FBD är mer kompakt än LD och ett exempel på hur FBD kan skrivas visas i figur 6.



Figur 6. Exempel på hur function block diagram skrivs.

3.2.3.5 Sequential function chart

Sequential function chart är ett språk gjort för att programmera efter specifika sekvenser. Det är mycket användbart med SFC programmering för att tydligt se i vilket steg en process befinner sig i vilket också gör det förhållandevis enkelt att felsöka beroende av komplexiteten av hela programmet. En av anledningarna till att SFC är med som ett standardiserat språk i IEC 61131–3 är att det enkelt kan integreras med de andra standardiserade språken. Språket är uppbyggt av rektangulära lådor som kallas Steps och kräver något slags villkor i en Transition för att gå vidare till nästa Step. Programmeraren väljer alltså vad som ska ske i varje Step genom Actions och vilka villkor som måste uppfyllas för att gå vidare till nästa Step (Ramanathan, 2014). Ett exempel på hur SFC kan skrivas visas i figur 7.



Figur 7. Exempel på hur sequential function chart skrivs.

3.2.4 Sammanfattning programmeringsspråk

Med de olika språken i IEC-61131-3 standarden kan välstrukturerade program skrivas. Användandet av de olika språken tillsammans ger flexibilitet att beroende på användningsområde använda det mest lämpade språket för uppgiften. Programmet kan delas in i moduler som dels kan återanvändas och testas var för sig. På så sätt kan ett stort program av moduler byggas upp och testas var för sig för att sedan fogas samman till en enhet. Möjligheten att använda de olika språken tillsammans gör att stora, välstrukturerade och kraftfulla program kan skrivas (Karvosenoja och Gyllenspång, 2001).

3.3 Simulering och emulering

Simulering kan beskrivas som ett virtuellt modellerat system eller del av systemet som modellerats för att uppnå samma beteende som det specifika verkliga systemet. Detta innebär att tester och experiment kan utföras i en virtuell miljö utan att på något sätt riskera att det verkliga systemet tar skada. Simulering baseras på dynamiska och matematiska modeller som byggts upp med data från verkligheten och simulerar det verkliga systemet över tid (Oppelt och Urbas, 2014). Zlajpah (2008, s.880) beskriver syftet med simulering:

> *”The principle “learning” is essential in simulation. Using simulation we can learn about something in a very effective way and while modifying “rules” we can observe the effects of our interaction”*

Emuleringsmodeller och simuleringsmodeller liknar varandra både till dess syfte och dess konstruktion men det skiljer mycket i hur de olika modellerna används och fungerar. Simuleringsmodeller används för att testa olika lösningar, hitta eventuella flaskhalsar, balansera flödet och se hur det verkliga systemet förändras över tid med de parametrar som matas in i modellen (McGregor, 2003; Banks, 2010). Simulering används ofta i ett experimentellt syfte där nya idéer kan kombineras med erfarenhet och modellen levererar ett resultat för att se om det ska implementeras i verkligheten (McGregor, 2003). Till skillnad från en simuleringsmodell är en emuleringsmodell anpassad för att användas till en mer specifik uppgift; att testa styrsystemet i olika situationer men även till att utbilda operatörer och underhållspersonal. Ytterligare en skillnad mellan modellerna är att emuleringsmodellen arbetar i realtid och passar därmed inte till samma experimentella syfte som en simuleringsmodell gör (McGregor, 2003).

Emulering kan definieras som att efterlikna eller härma ett verkligt system så väl virtuell som fysisk process, detta betyder att när samma data laddas in i det verkliga systemet och emuleringsmodellen ska liknande eller samma resultat uppnås (Oppelt och Urbas, 2014; Koninklijke Bibliotheek, 2019). I både simuleringsmodeller och emuleringsmodeller finns det dock mätbara skillnader mellan verkligheten och modellerna, denna skillnad kallas ”trovärdighetsgap”. Med en emuleringsmodell försöks detta gap att minskas genom att efterlikna det verkliga systemet så mycket som möjligt genom att ta med delar från det verkliga systemet (McGregor, 2003). McGregor (2003) menar för att en modell ska vara användbar måste den vara noggrann, men hur noggrann den ska vara varierar för olika projekt och hur modellen ska användas. McGregors (2003, s.1683) svar på frågan hur noggrann en modell ska vara, lämnar noggrannheten öppen för tolkning och anpassningsbar:

> *”The most appropriate answer to this is, sufficiently accurate such that the results remain statistically meaningful”*

Eftersom en emuleringsmodell reflekterar ett verkligt system i realtid kan stora delar av styrsystemets funktioner verifieras i förväg. McGregor (2003) beskriver att testa styrsystemet med hjälp av emuleringsmodell är lönsamt under flera omständigheter och nämner några:

- När de verkliga testerna på något sätt riskerar att sätta projektet i ett kritiskt läge.
- När det inte finns tid tillgänglig att genomföra ett fullt test innan uppstart.
- När det inte finns tillräckligt med resurser i systemet för att testa styrsystemet.
- När kostnaden av verklig test överstiger kostnader för emuleringstest.

3.4 Verifiering och validering av modeller

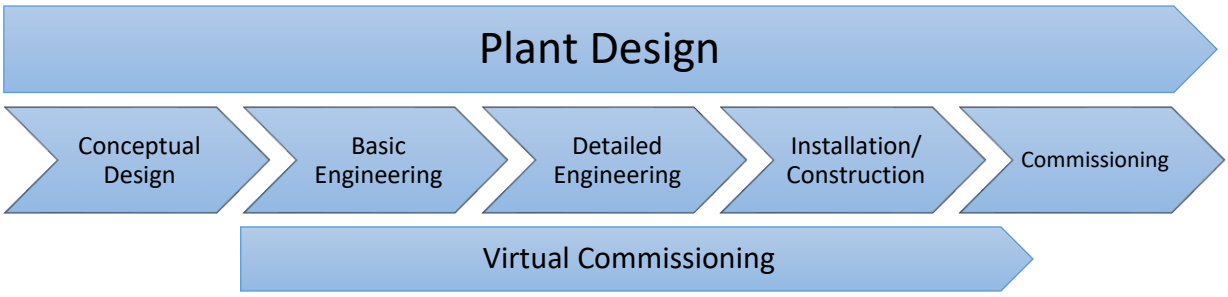
Banks (2010) beskriver verifiering och validering som en av de svåraste och viktigaste uppgifter en modellutvecklare har på grund av att ingenjörer, analytiker och ledning tar viktiga beslut utifrån modellens resultat. Det är därför viktigt att modellutvecklaren kan utveckla modellen med hög pålitlighet och gör detta genom att verifiera och validera sin modell.

Verifiering handlar om att bygga modellen korrekt genom att noggrant kontrollera ingående parametrar och att den logiska strukturen av modellen är korrekt representerad (Eriksson, 2008; Banks, 2010). Att verifiera en modell är en process för att få en trovärdig representation av de system som den baseras på (Banks, 2010).

Validering handlar om att tillverka rätt modell, vilket betyder att en modell ska ha ett så likt beteende som möjligt av det verkliga systemet för att kunna använda den till att utföra experiment och analyser med trovärdigt resultat (Eriksson, 2008; Banks, 2010). Validering är en kontinuerlig process där modellen och det verkliga systemet jämförs och justeringar görs i modellen för att uppnå samma resultat som i verkligheten. Om en modell inte är validerad kan den anses vara icke trovärdig och därmed antas ge missvisande resultat (Banks, 2010).

3.5 Virtuell driftsättning

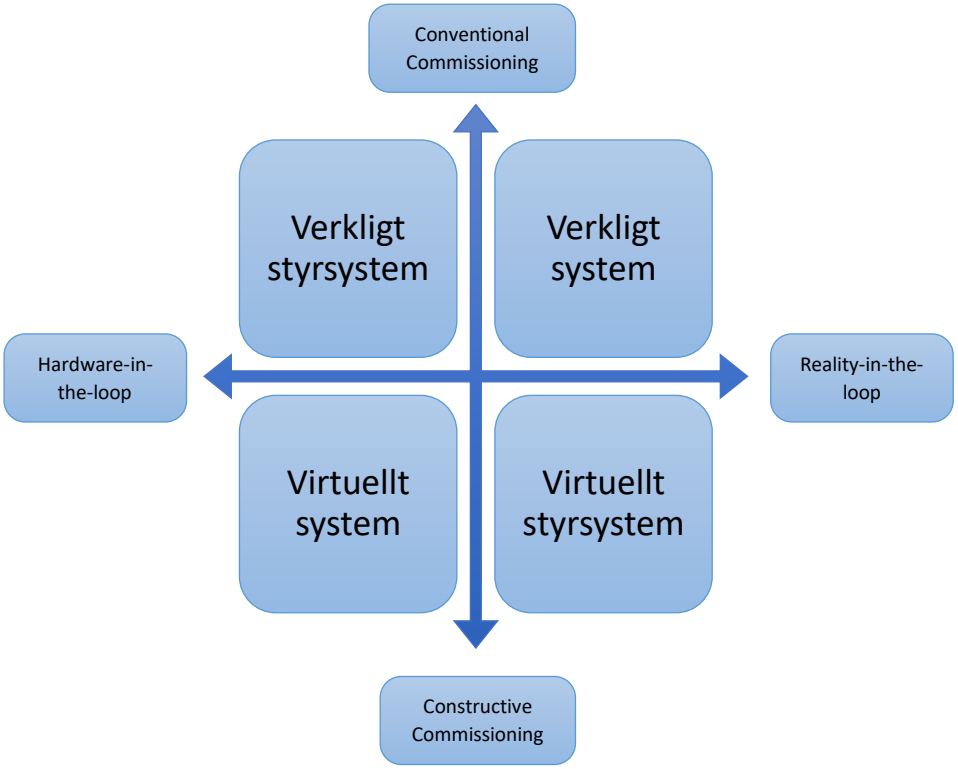
Att få en utrustning installerad och redo för produktion kallas för driftsättning. Bondeson och Liss (2018) skriver att ungefär 25% av den totala projekttiden är driftsättning, av de 25% är 60–70% på grund av felaktig eller defekt programkod. Detta betyder att felaktig eller defekt programkod motsvarar 15% av den totala projekttiden. Genom att arbeta med virtuell driftsättning kan programmerare verifiera sin programkod i en virtuell miljö parallellt med övriga ingenjörer i framtagningsfasen, vilket skulle kunna minska driftsättningstiden med 75%, se figur 8 (Bondeson och Liss, 2018).



Figur 8. Minskad driftsättningstid.

Det finns olika sätt att arbeta med driftsättning, Lee och Park (2015) definierar enligt figur 9, fyra olika kombinationer:

1. Konventionell driftsättning, med verkligt system och verkligt styrsystem.
2. Virtuell driftsättning, hardware-in-the-loop, med virtuellt system och verkligt styrsystem.
3. Virtuell driftsättning, reality-in-the-loop, med verkligt system och virtuellt styrsystem.
4. Constructive commissioning, med virtuellt system och virtuellt styrsystem.



Figur 9. Fyra kombinationer att arbeta med driftsättning.

Virtuell driftsättning görs genom att simulera driftsättningen i en riskfri virtuell miljö där elektronik -, mekanik-, och automationsingenjörernas arbete sammanställs i en identisk modell av det verkliga systemet. Styrsystem, mekaniska rörelser och elektroniken kontrolleras i modellen för att sedan felsöka och rätta till fel innan de implementeras i verkligheten (Bondeson and Liss, 2018; Siemens, 2019). Att arbeta med virtuell driftsättning betyder att programmerare får mer tid att konstruera sin PLC kod vilket ger en ökad kvalité på koden samtidigt som tiden för den verkliga driftsättningen blir betydligt kortare. Det är inte alltid lönsamt att arbeta med virtuell driftsättning utan att ta hänsyn till vissa aspekter. Det kan vara tidskrävande att skapa en modell för virtuell driftsättning, speciellt för företag som inte har mycket erfarenhet av simulering (Oppelt och Urbas, 2014; Dahl *et al.*, 2016). Mindre företag kan även sakna behörigheter till mjukvaran som modellerna byggs i, vilket betyder att viss investeringskostnad kan behövas. Större företag kan ha specialbyggda maskiner som är anpassade efter en specifik produkt, dessa kräver ofta mycket tid och ansträngning att korrekt modellera. Dock finns det vision inom industrin om att alla komponenter ska levereras med en färdig modell (Dahl *et al.*, 2016).

3.6 Simumatik3D

Simumatik3D är en emuleringsmjukvara som är anpassat för industriella automatiseringsapplikationer och ger möjlighet att testa både PLC och robotprogram i en virtuell miljö. Simumatik3D kan bland annat användas till virtuell driftsättning och verifiering av PLC kod.

3.7 TIA portal

Totally integrated automation (TIA) portal är en mjukvara från Siemens och ger obehindrad tillgång till Siemens kompletta sortiment av digitaliserade automationstjänster. Det är en modern lösning för ett brett spektrum av automationsuppgifter samtidigt som det är grundpelaren för framtidens ännu mer integrerade mjukvarulösningar för automation och industriell IT. Mjukvaran kan framförallt användas för programmering av PLC enheter och Human machine interface (HMI) paneler (Folke, 2019).

3.8 Sammanfattning teoretisk referensram

Den teoretiska referensramen har genomförts för att få ökad kunskap inom de område som examensarbetet har utförts. Definitioner av standarder, hur standardisering genomförs och möjligheter med standarder har generat kunskap om hur en standard bör se ut och vilka krav som bör uppfyllas för att uppnå en optimal standard.

Kapitlet om PLC förtydligar vad en PLC är för något, hur den används och vilka programmeringsspråk som finns. Detta är relevant då det är en standard för PLC programmering som görs Här finns även ett kapitel om den existerande PLC standarden IEC 61131. Standarden innefattar en PLC enhets hela livscykel och definierar däribland de fem olika programmeringsspråk som beskrivs i kapitlet. Detta är en viktig standard då de fem olika språken har gjort det lätt för programmerare och ingenjörer att känna igen PLC program oberoende vart i världen arbetet utförs. Standarden beskriver dock inte hur programmeringen ska utföras för att det ska bli tydligt och strukturerat, detta gör att det skiljer sig mycket i programstrukturen beroende på programmerare. I en lokal standard för ett enskilt företag, likt de som skapats i detta examensarbete, skapas en gemensam struktur för alla programmerare. En lokal standard kan innehålla standardiserade funktioner med tydlig struktur som är lätta att återanvända i flera automationsprojekt vilket gör att programmet har samma uppbyggnad oberoende vem som programmerat.

Den teoretiska referensramen beskriver även emulering och simulering, samt skillnaden mellan dem. Det finns även ett kapitel om verifiering och validering av modeller samt ett kapitel om virtuell driftsättning. I detta arbete har en emuleringsmodell byggts upp, därav krävdes kunskap om vad en emuleringsmodell är, hur den används och hur dessa skiljer sig från simuleringsmodeller. Det krävdes även kunskap om hur verifiering och validering av dessa modeller utförs och hur detta kan ge en mer trovärdig och användbar modell. En emuleringsmodell eller simuleringsmodell kan även användas till virtuell driftsättning istället för att verifiera standardiserade funktioner som det gjorts i detta arbete. Därav beskrivs virtuell driftsättning med dess fördelar, vilka kombinationer av driftsättning de finns och i vilka sammanhang virtuell driftsättning är lönsamt.

En kort beskrivning av de mjukvaror vi använder, TIA portal och Simumatik3D, finns med i den teoretiska referensramen för att tydliggöra vad mjukvarorna kan användas till.

4. Genomförande

I kapitlet genomförande beskrivs grundligt hur arbetet genomförts för att uppnå en standardiserad PLC kod. Följande rubriker behandlas i detta kapitel:

- Tillvägagångssätt, beskriver övergripande hur arbetet genomförts och hur metoden Design and creation research strategy har applicerats.
- Utrustning, beskriver vilken typ av utrustning som använts, både mjukvaror och hårdvaror.
- Kommunikation, tillvägagångssättet för att få en stabil och fungerande kommunikation mellan all utrustning.
- Val av funktioner, en beskrivning av vilka funktioner som valts samt varför dessa funktioner har valts.
- Uppbyggnad av standarden, en beskrivning av hur standarden byggts upp i TIA portal.
- Konstruktion av emuleringsmodell, kapitlet beskriver hur emuleringsmodellen konstruerats samt hur systemet har utvecklats från första funktion till sista.
- Verifiering av standard, beskriver tillvägagångssättet för att verifiera funktionerna.

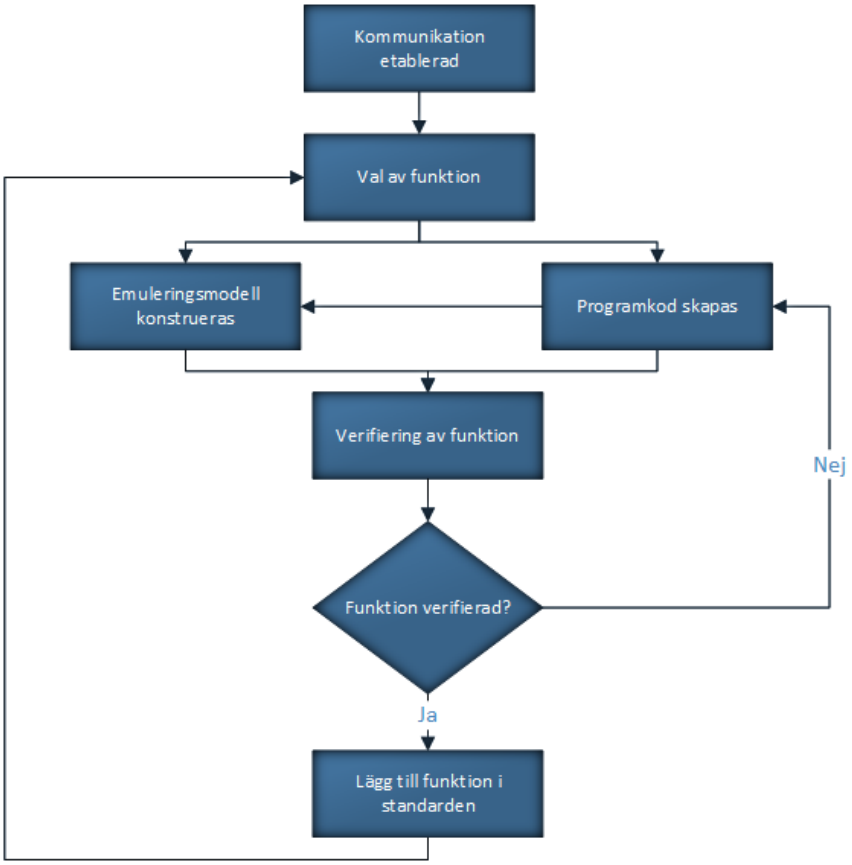
4.1 Tillvägagångssätt

Under hela genomförandet har metoden Design and creation research strategy använts som ett grundläggande arbetssätt. Design and creation är som tidigare nämnt en iterativ process vilket betyder att vissa steg upprepas för att rätt artefakt ska kunna uppnås.

Arbetet påbörjades genom att planera tillsammans med Projektengagemang om hur standarden skulle utformas, hur den skulle se ut och vad den bör innehålla. Det var många detaljer som var svåra att svara på tidigt i arbetet men en planering utformades och mål sattes för att kunna uppfylla Projektengagemangs förväntningar och arbetets mål. En grundläggande kunskap för de program och mjukvaror som skulle användas skapades. Detta gjordes genom att läsa manualer, fråga anställda på Projektengagemang om tips och genom olika tester. Att göra sig bekant med mjukvarorna gjordes för att få en större och djupare förståelse för hur programmen är uppbyggda och vilka möjligheter som finns men även för att underlätta att skapa den standardiserade PLC koden och emuleringsmodellen.

När en förståelse för mjukvarorna och deras möjligheter skapats enskilt gjordes ett test för att få kommunikationen att fungera men även för att förstå hur kommunikationen skulle ske. Det fanns manualer för hur de olika mjukvarorna fungerade på egen hand men enbart lite och begränsad information gick att hitta om hur kommunikationen mellan dessa mjukvaror skulle ske.

Efter en fungerande kommunikation etablerats påbörjades en process där funktioner valdes tillsammans med Projektengagemang. När funktionerna valts skapades en PLC kod och emuleringsmodell parallellt. När PLC koden och modellen skapats verifierades funktionerna i modellen med hjälp av funktionstester. Detta har gjorts en funktion i taget, figur 10 visualiserar hur arbetet gått till.



Figur 10. Tillvägagångssätt för att verifiera funktioner.

Blir inte verifieringen av funktionen godkänd och systemet inte fungerar upprepas det föregående steget och det sker justeringar av PLC koden eller emuleringsmodellen. När funktionen verifierats kan den ingå i standarden och processen börjar om med steget val av funktion och arbete med en ny funktion inleds. Detta är en iterativ process och är en del av utvecklingssteget i metoden Design and creation research strategy. När de funktioner som valts ut verifierats påbörjades utvärderingssteget där två valideringar gjordes. Eftersom arbetet utförs enligt Design and creation är slutanvändaren och hur de kan använda den utvecklade artefakten en del av arbetet, detta beskrivs i utvärderingen och slutsatsen.

4.2 Utrustning

För att utföra examensarbetet och uppfylla målen har det krävts olika hårdvara och diverse mjukvaror. Dessa är följande:

- Stationär PC med kraftfull processor och tillbehör.
- Bärbar PC med tillbehör.
- Mjukvaran: VMware.
- Mjukvaran: Siemens TIA portal V15.
- Mjukvaran: Siemens S7-PLCSIM Advanced V2.0 SP1.
- Mjukvaran: Simumatik3D.

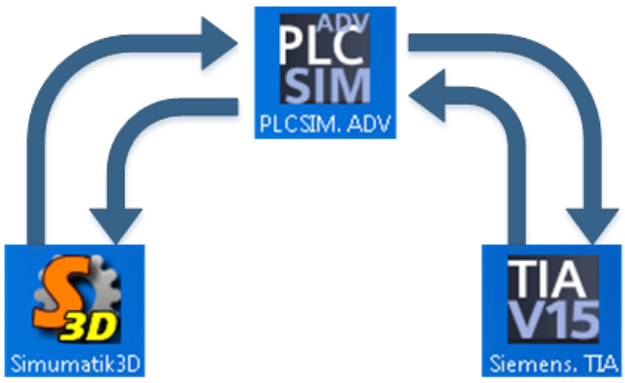
Mjukvaran Simumatik3D krävs för att verifiera funktionerna genom emulering. Mjukvaran PLCSIM Advanced är till för att skapa en virtuell PLC som kan användas i en extern mjukvara som Simumatik3D. Mjukvaran TIA portal används för att programmera och styra funktionerna i PLC koden. Eftersom

Projektengagemangs licenser och utrustningar används behövdes mjukvaran VMware för att skapa en virtuell dator som har tillgång till TIA portal och PLCSIM Advanced.

En stationär PC med kraftfull processor krävs för att klara av att köra både PLC koden och emuleringen samtidigt. En bärbar PC krävs för att kunna ha flexibilitet i arbetsmiljön och vid behov kunna ta med sig projektet till andra platser än Projektengagemangs kontor. Både PLC logiken, PLCSIM Advanced och emuleringsmodellen kördes i början på samma dator men när emuleringsmodellen och PLC logiken blev större krävdes det att de tillgängliga datorerna kopplades samman för att fördela de krävande programmen. Den bärbara PC:n har licens för TIA portal och PLCSIM Advanced medan Simumatik3D är licensfritt och kunde användas på båda datorer.

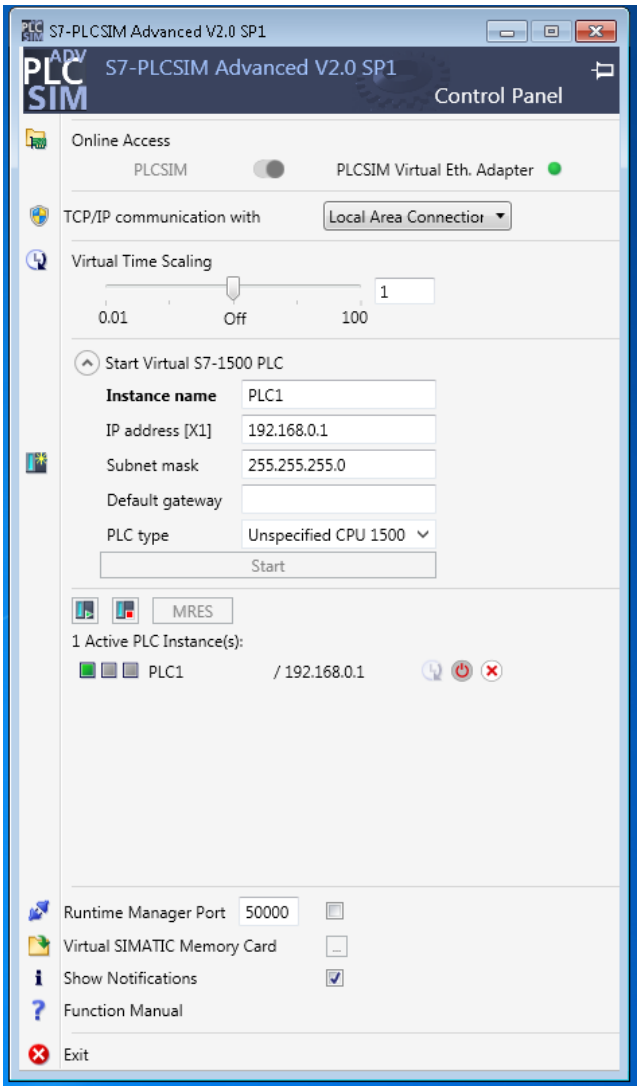
4.3 Kommunikation

För att verifiera TIA portals PLC kod med hjälp av emuleringsmodellen i Simumatik3D behöver en kommunikation mellan dessa mjukvaror skapas. Detta görs med hjälp av PLCSIM Advanced där en virtuell PLC skapas och kopplas ihop med både TIA portal och Simumatik3D. Figur 11 är en överblick över hur kommunikationen fungerar mellan mjukvarorna. Programmet som skapas i TIA portal laddas alltså ner till den virtuella PLC enheten. När Simumatik3D startar emuleringen kopplas I/O från Simumatik3D ihop med tagarna från det nedladdade programmet i den virtuella PLC enheten. När exempelvis givare eller tryckknappar blir påverkade i Simumatik3D kommer signalerna skickas in till den virtuella PLC enheten och kan efter nästa cykelvarv monitoreras från TIA portal.



Figur 11. Överblick över kommunikationen.

Kommunikationen i PLCSIM Advanced sker genom PLCSIM Virtual Ethernet Adapter och TCP/IP inställd på den port som kommunikationen ska gå genom, se figur 12. PLCSIM Advanced, TIA portal och Simumatik3D kan finnas på samma dator men kan även användas på olika datorer genom att föra över informationen via lokala nätverk. Fördelen med att ha programmen på olika datorer är att belastningen från programmen fördelas.

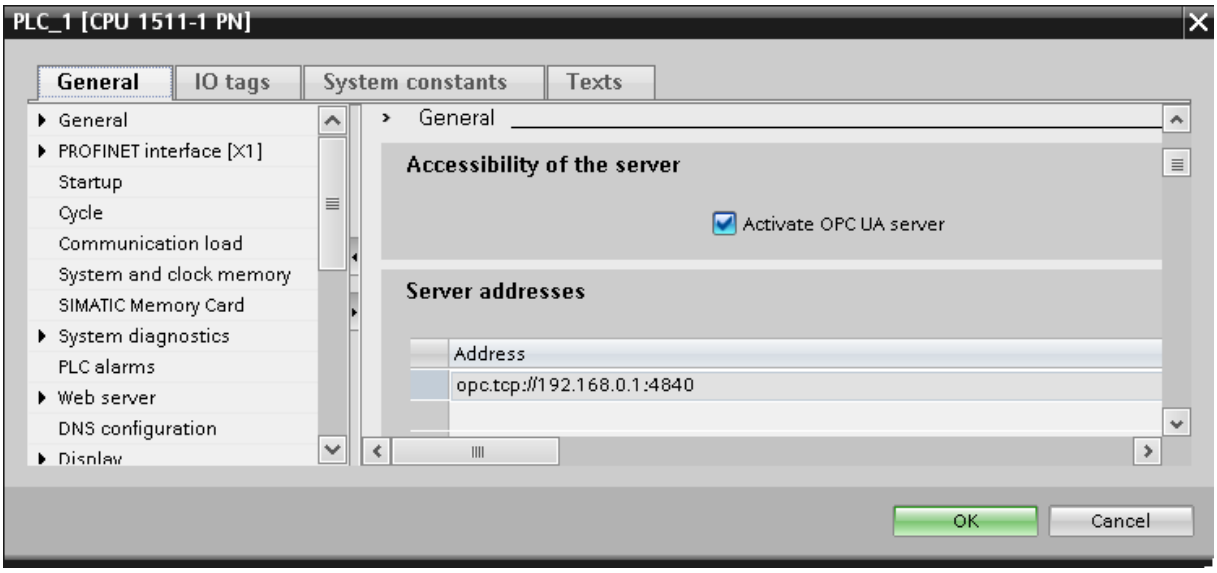


Figur 12. Arbetets parametrar för den virtuella PLC enheten.

Den virtuella PLC enheten för projektet skapas i PLCSIM Advanced genom kontrollpanelen som syns i figur 12 och nedan beskrivs tillvägagångssättet:

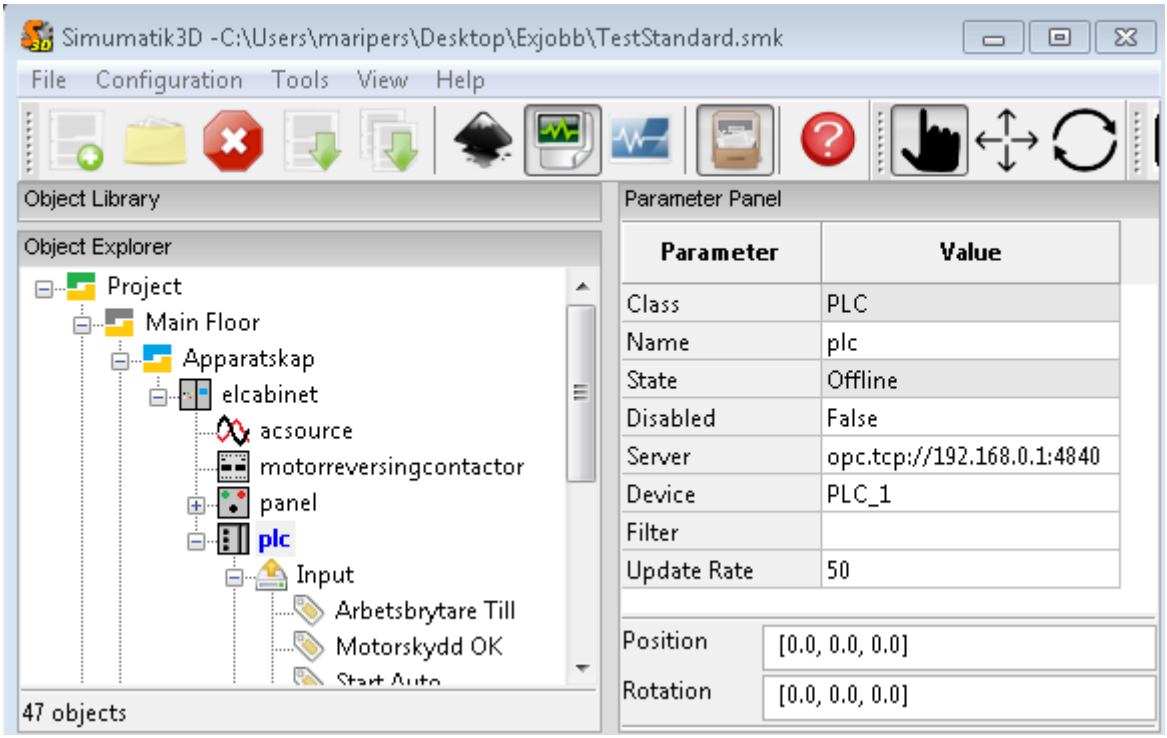
1. Välj PLCSIM Virtual Eth. Adapter i Online access.
2. Välj Local area connection under TCP/IP.
3. Skriv i ett unikt namn i Instance name, detta är den virtuella PLC enhetens namn.
4. Skriv i rätt IP adress, det är viktigt att adressen är samma i samtliga mjukvaror vid denna typ av kommunikation.
5. Skriv i Subnet mask, denna finns att hämta i datorns nätverk- och delningscenter.
6. Välj vilken typ PLC som ska användas, denna ska stämma överens med TIA portal.
7. Tryck Start och den virtuella PLC enheten kommer dyka upp under Active PLC Instance.

Efter att den virtuella PLC enheten skapas i PLCSIM Advanced kommer det vara möjligt att ladda ner PLC koden från TIA portal till den virtuella PLC enheten. Emuleringsmodellen i Simumatik3D och PLC koden i TIA portal kommunicerar genom OPC-UA vilket betyder att detta behöver aktiveras i TIA portal för att fungera. OPC UA aktiveras genom att högerklicka på den PLC som programmeras, bocka i en ruta under properties > general > OPC UA på vår PLC enhet i TIA portal, figur 13 visar hur fönstret för OPC UA inställningen ser ut men även vilket IP adress och port som används.



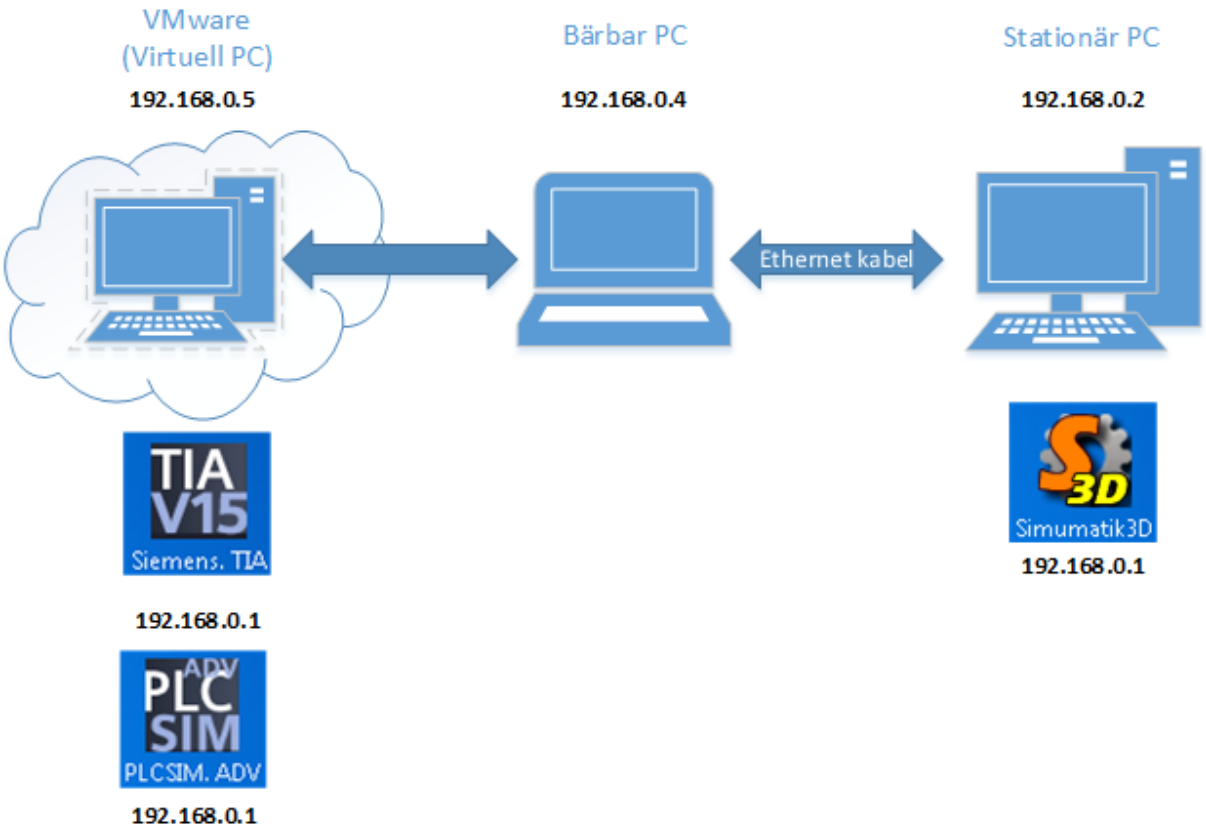
Figur 13. Fönster för OPC UA inställning.

När OPC-UA är aktiverat behöver rätt namn och IP adress skrivas på PLC enheten i Simumatik3D, det är viktigt att både IP adressen och namnet på PLC enheten stämmer överens i både TIA portal och Simumatik3D, detta gäller även namnen på alla in- och utgångar. Figur 14 visar hur parametrarna för PLC enheten i Simumatik3D är konfigurerade.



Figur 14. Parametrar för PLC enheten i Simumatik3D.

På grund av hög belastning på den bärbara PC som användes krävdes det att mjukvarorna användes på olika datorer för att kunna använda programmen smidigare, utan krångel och långa väntetider. För att denna kommunikation skulle fungera krävs det att rätt TCP/IP Adresser används. Figur 15 visar hur konfigurationen skett för att etablera en bra kommunikation.



Figur 15. Kommunikation mellan datorer och mjukvaror.

4.4 Val av funktioner

Det fanns möjlighet att använda projekt och ett lokalt bibliotek med funktioner från Projektengagemangs kontor i Lidköping i arbetet. Tillsammans med Projektengagemang valdes funktioner ut från detta bibliotek och andra program som är vanligt förekommande i Projektengagemangs automatiseringsprojekt. För att komma fram till vilka funktioner som ska ingå i standarden fördes diskussioner med handledaren på Projektengagemang som visste vilka funktioner som var mer vanligt använda än andra. Mängden funktioner styrdes utifrån den tid som fanns att lägga på arbetet, om mindre tid hade tillhandahållits hade färre funktioner funnits med i standarden och om mer tid tillhandahållits hade fler funktioner funnits med.

Den första funktion som användes i standarden var en motordrift som kan användas för att styra en bred variation av elmotorer. Motordriftfunktionen kan styras både i automatisk drift och manuell drift med hjälp av tryckknappar. Motordriften kan styra en motor framåt, bakåt och i högfart. För att emulera motordriften i Simumatik3D användes motorer och banor men även ett vridbord och en hiss för att kunna utnyttja motordriftfunktionerna fullt ut.

Den andra funktionen i standarden var en rörelsefunktion för styrningar av exempelvis ventiler och cylindrar. Funktionen kan exempelvis styra en 2-läges ventil som i sin tur styr en cylinder, funktionen är till för att kunna styra en bred variation av rörelser och ventiler och inte bara en specifik variant. Rörelsefunktionen kan styras i både automatisk drift samt manuell drift med hjälp av tryckknappar. För att emulera funktionen användes en ventil som styr ett cylinderstopp i Simumatik3D.

Den tredje funktionen är till för att styra motordrifter på ett ET200pro rack. ET200pro är ett moduluppbyggt I/O system som kommunicerar med PROFINET eller PROFIBUS. Det är möjligt att med hjälp av moduler addera både frekvensomriktare, RFID moduler, analoga I/O, digitala I/O, motorskydd

och annan utrustning. Denna funktion är dock bara till för själva motorstyrningen på ett sådant rack. Funktionen kan styra motorer i både automatisk drift och i manuell drift med tryckknappar. Den största skillnaden på denna motordriftsfunktion jämfört med en vanlig är att det finns ett summalarm och återställning samt en utgång för broms. Denna funktion emuleras i Simumatik3D genom en ET200pro motor.

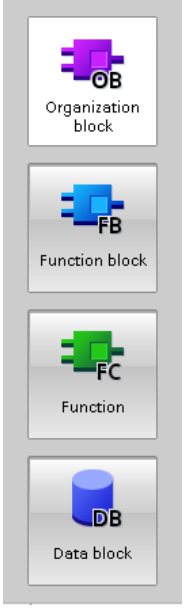
Den fjärde funktionen är till för att kunna öppna och låsa en grind vid ett tillfälle som är passande för den automatiserade cellen. Grindfunktionen är till för att fungera med en bred variation av grindar och kan användas enskilt men även som en del vid ett större säkerhetssystem. Funktionens syfte är att hålla en grind låst tills någon begär inträde, då låses grinden upp vid ett tillfälle då det är säkert för en person att gå in i cellen. Ingen produktion i det området kommer att starta förrän personen lämnat cellen, stängt grinden och återställt den. Då låses grinden och produktionen kan fortsätta. Denna funktion emuleras i Simumatik3D genom en grind med låsfunktion som visar när det är möjligt att öppna eller inte.

Den femte funktionen är en statistisk funktion som räknar detaljer som producerats. Detaljräknaren räknar alla detaljer men räknar i fyra olika kategorier. Den räknar de detaljer som är rätt producerade (OK), de detaljer som inte är rätt (NOK) men även totalen av dessa två. Dessa tre värden går att återställa enkelt genom en tryckknapp men det finns även ett totalvärde som fortsätter räkna och kan bara återställas i PLC koden. Detta emuleras i Simumatik3D genom displayer som visar vad de olika räknevärdena är och även tryckknapp för återställning. För att kunna använda funktionen fullt ut är det möjligt att under emuleringen välja att använda OK eller NOK detaljer.

Den sjätte och sista funktionen är också en statistisk funktion men denna är till för att mäta cykeltid. Funktionen cykeltid är till för att mäta takttiden eller cykeltiden för hela produktionen eller delar av produktionen. Detta genom att ett villkor för att börja räkna cykeltiden uppfylls, exempelvis automatiskt driftläge. Sedan kommer cykeltiden att flytta det aktuella värdet när detaljen kommer till en brytpunkt. Det finns möjlighet att se de tio senaste cykeltiderna men vid emulering av funktionen används endast en av de tio senaste värdena. Funktionen gör det även möjligt att jämföra den nuvarande cykeltiden med en rekommenderad cykeltid för att visa om cykeltiden är överskriden eller inte. Cykeltidsfunktionen emuleras i Simumatik3D genom displayer för att visa den aktuella cykeltiden och den senaste cykeltiden.

4.5 Uppbyggnad av standarden

Uppbyggnaden av standarden gjordes i TIA portal. För att få grundläggande kunskap i TIA portal och hur Projektengagemang programmerar och strukturerar sina projekt har genomgångar gjorts av deras tidigare projekt. De tidigare projekten användes även för att kontrollera hur Projektengagemang har använt de funktioner som standardiseras. Vid uppbyggnad av projekt i TIA portal är det viktigt att hålla ordning på de olika block som används, de ser ofta liknande ut i andra PLC mjukvaror men med olika namn och egenskaper. I figur 16 visas de fyra olika block som kan användas vid programuppbyggnad i TIA portal.

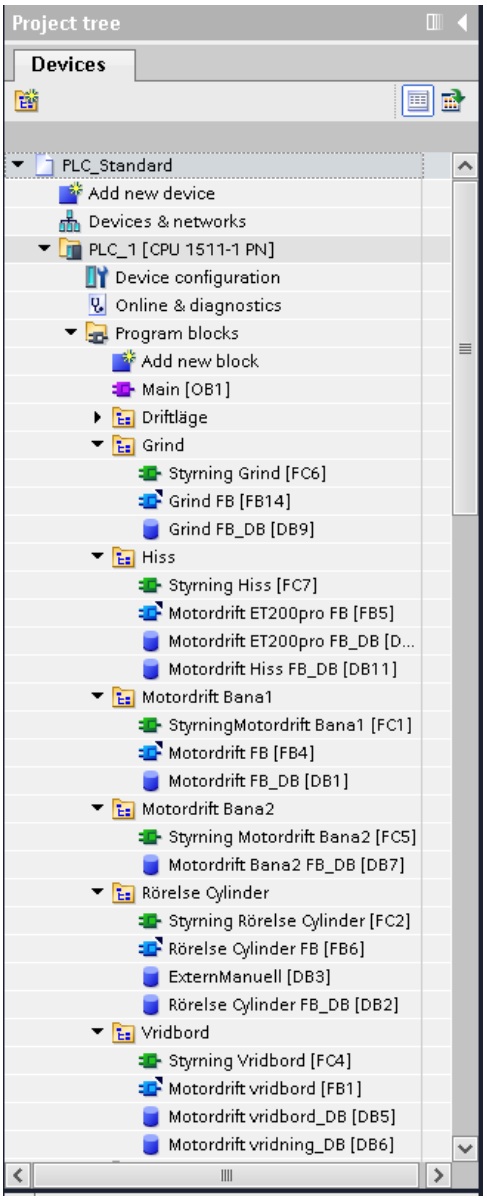


Figur 16. Olika sorters block i TIA Portal.

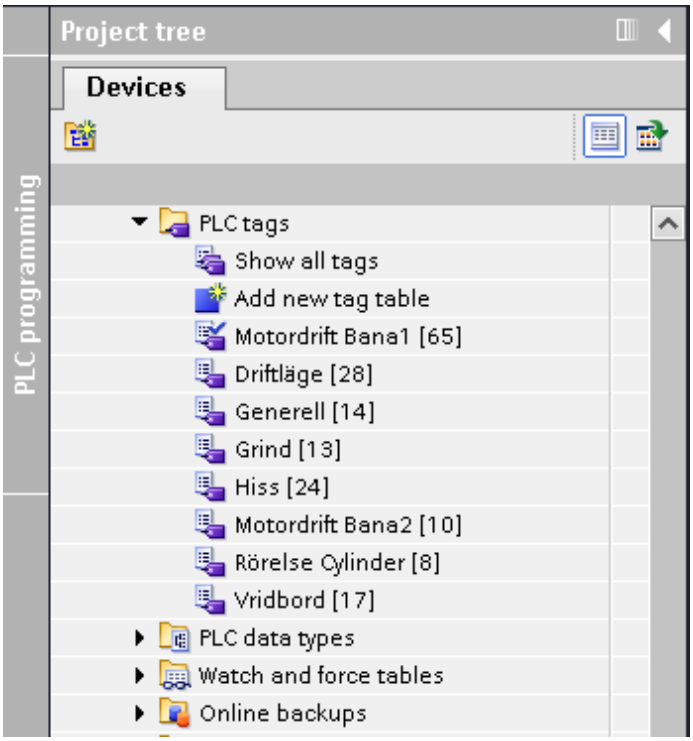
Organization block (OB) läses och exekveras cykliskt och är huvud (Main) blocket för hela programmet. Functions (FC) används som programmeringsblock eller subrutin och har inget tillhörande minne. Ett FC måste använda sig av ett globalt data block för att kunna spara värden. Ett Data block (DB) är till för att lagra program data och värden, globala DB kan användas från alla FB och FC medan instans DB är lokala för ett specifikt FB. Function block (FB) används som programmeringsblock som sparar värden permanent i instans data block. Detta för att kunna behålla värden efter att blocket använts. FB används för att slippa upprepa kod då ett FB kan användas flera gånger i ett program men för varje gång samma FB används måste ett nytt instans DB skapas för att kunna spara värden.

Uppbyggnaden av standarden startades med att skapa en tydlig struktur som skulle göra det lätt att hitta i programmet och göra ändringar vid behov. De olika funktionerna är uppdelade i separata mappar för att enkelt kunna både felsöka och ändra styrningen. Det används ett OB som kallas Main där olika FC körs för styrningarna till de standardiserade funktionerna. Och varje FC styr olika FB. De funktioner som används i standarden är alla FB. Eftersom arbetet följer black box metoden, gjordes inga ändringar på något FB utan bara individuella styrningar i FC för att kunna använda dessa FB. Styrningarna är programmerade i FBD för att det är ett kompakt och användarvänligt språk för arbetet som utförts. För att addera de utvalda funktionerna följdes det tillvägagångssätt som beskrivs i kapitel 4.1 *Tillvägagångssätt.* Varje funktion lades in i programmet en i taget för att enkelt kunna verifiera just den funktionen med hjälp av Simumatik3D. För att skapa ett system där alla funktioner som valts ut skulle bli relevanta användes ibland samma standardiserade FB fler än en gång. Detta gjorde dock att extra säkerhet på att funktionen fungerar i olika situationer kunde upprättas. Ett transportsystem av exempelvis en låda gör att alla de funktioner som skulle finnas i standarden kunde användas på rätt sätt. Det första FB som användes var motordrift FB, från början bara för att testa kommunikationen men det blev också standardens första verifierade funktion. Den fullständiga strukturen av programmet kan ses i figur 18. Cykeltider FB och Detaljräknare FB ligger båda under mappen Driftläge, detta på grund utav att de är statistiska funktioner. För att göra det tydligt och lätt att hitta och redigera tagar, delades taglistan upp i flera sektioner, se figur 17. En för varje ny mapp från strukturen. Alla tagar är dock globala och kan användas vart som helst i projektet. Hela bytes skickas till och tas emot från Simumatik3D. I standarden ingår även ett bibliotek med de FB som är standardiserade, biblioteket

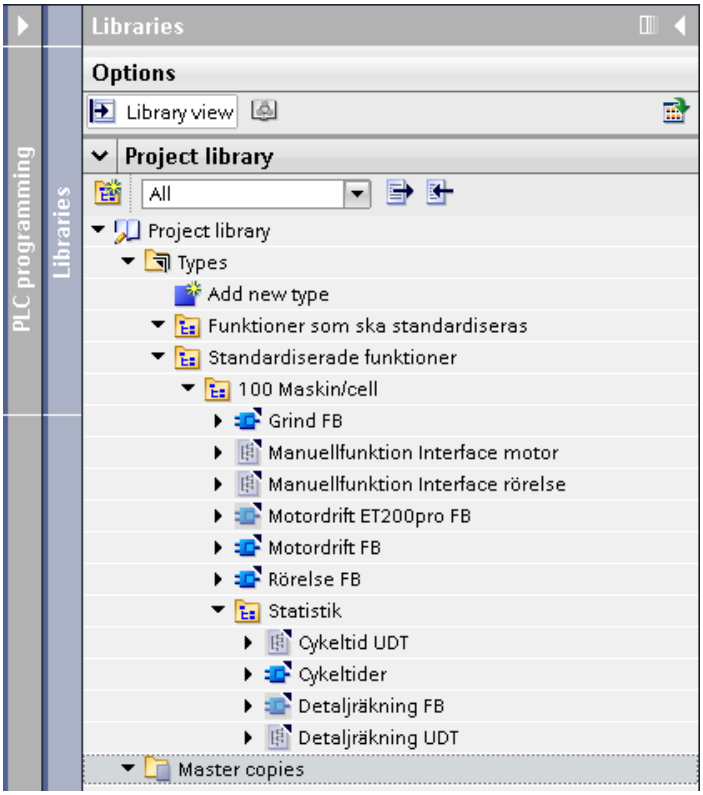
ska användas av programmerare för att enkelt kunna använda de standardiserade funktionerna i nya projekt. Biblioteket visas i figur 19.



Figur 18. Strukturen av standarden i TIA portal.

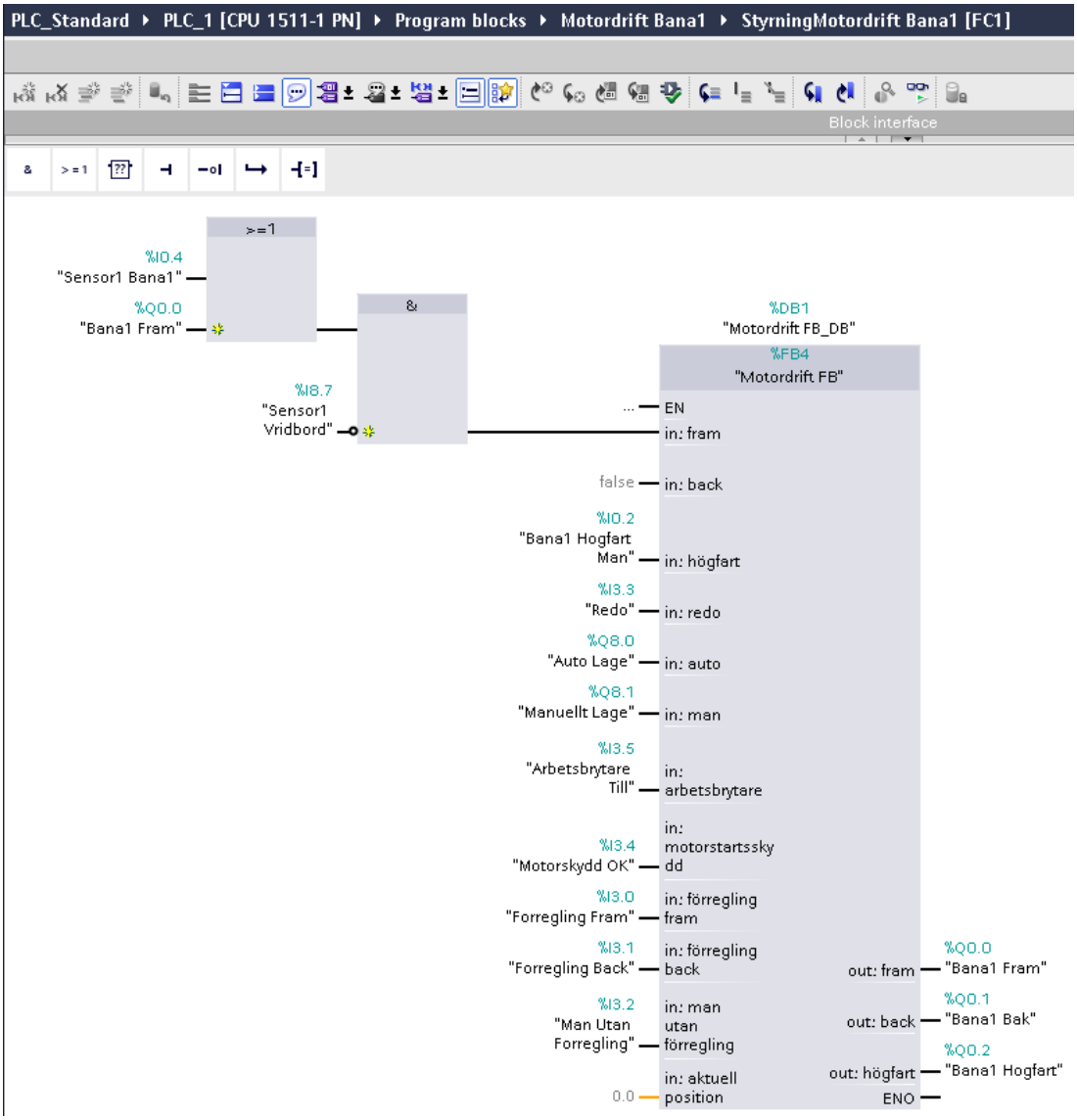


Figur 17. Struktur på taglista i TIA portal.



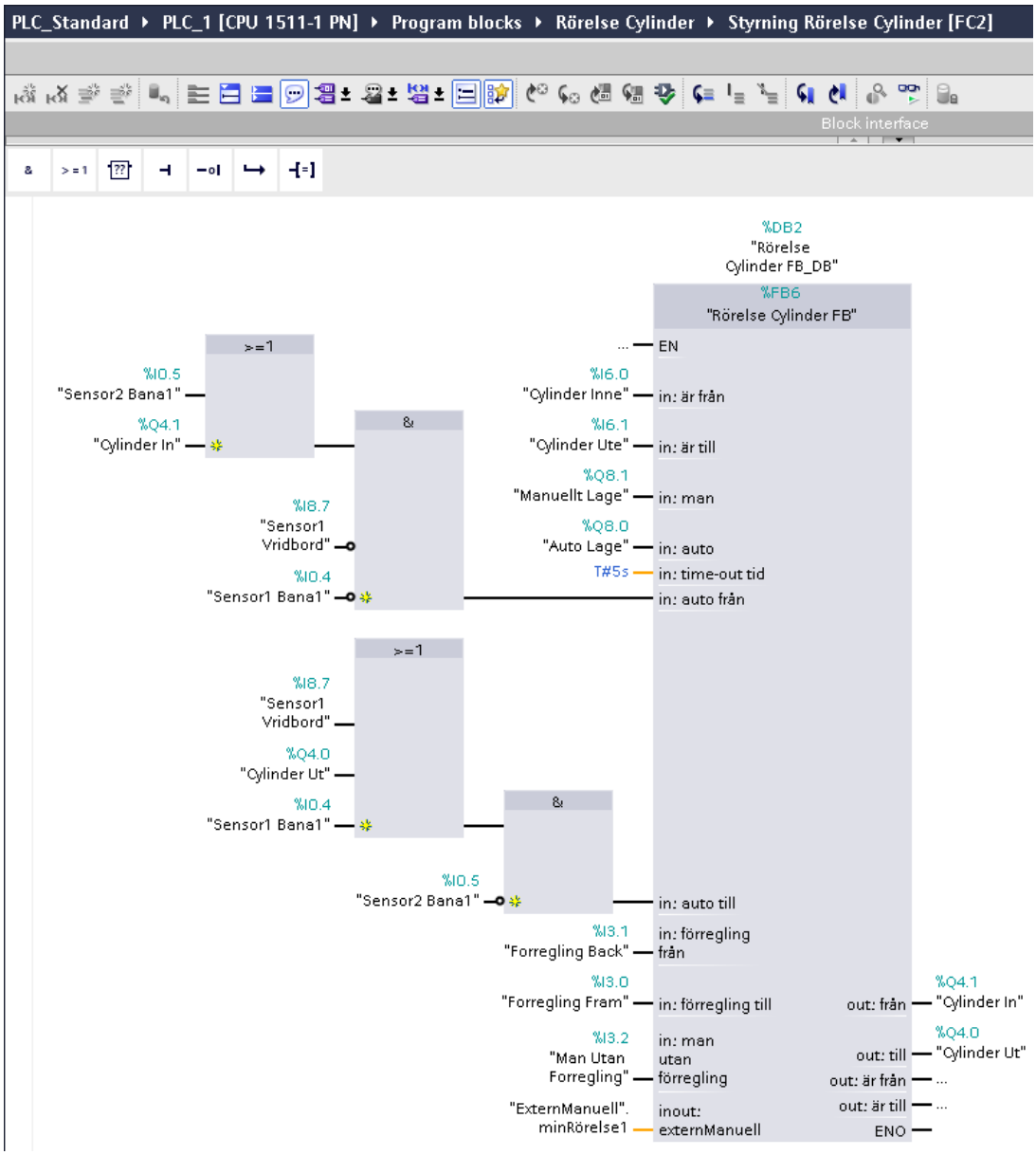
Figur 19. Biblioteket med standardiserade funktioner.

Figur 20 visar hur styrningen av detta FB har programmerats för att styra motordriften för bana1 i Simumatik3D. Motordrift FB är till för olika typer av motordrifter och gör det enkelt att styra en motor i fyra olika lägen: framåt, bakåt, framåt i högfart och bakåt i högfart. Den kan styras i ett automatiskt läge och ett manuellt läge. För det automatiska läget styrs motordrift FB genom att signaler skickas in på in:fram och in:back beroende på om användaren vill att motorn ska drivas framåt eller om den ska drivas bakåt. I manuellt läge ska motordriften styras genom knappar på en HMI panel som är kopplade till det tillhörande instans DB för blocket. För att styra högfarten ska en signal skickas in på in:högfart, det spelar ingen roll om driften är i automatiskt eller manuellt läge för detta. Motordrift FB används på fem olika platser och för tre olika ändamål. Den används för en vridning till ett vridbord, till en hiss och tre olika bandrifter. Figur 20 visar hur den används för att styra motorn till bana1. I styrning motordrift bana1 FC så är motordrift FB till för att köra OK och NOK lådor till vridbordet. Vid banan och vridningen av vridbordet används motordrift FB för att både vrida bordet fram och tillbaka men även att köra lådor på och av banan. För hissen används motordrift FB för att köra hissen upp och ner både i låg och högfart.



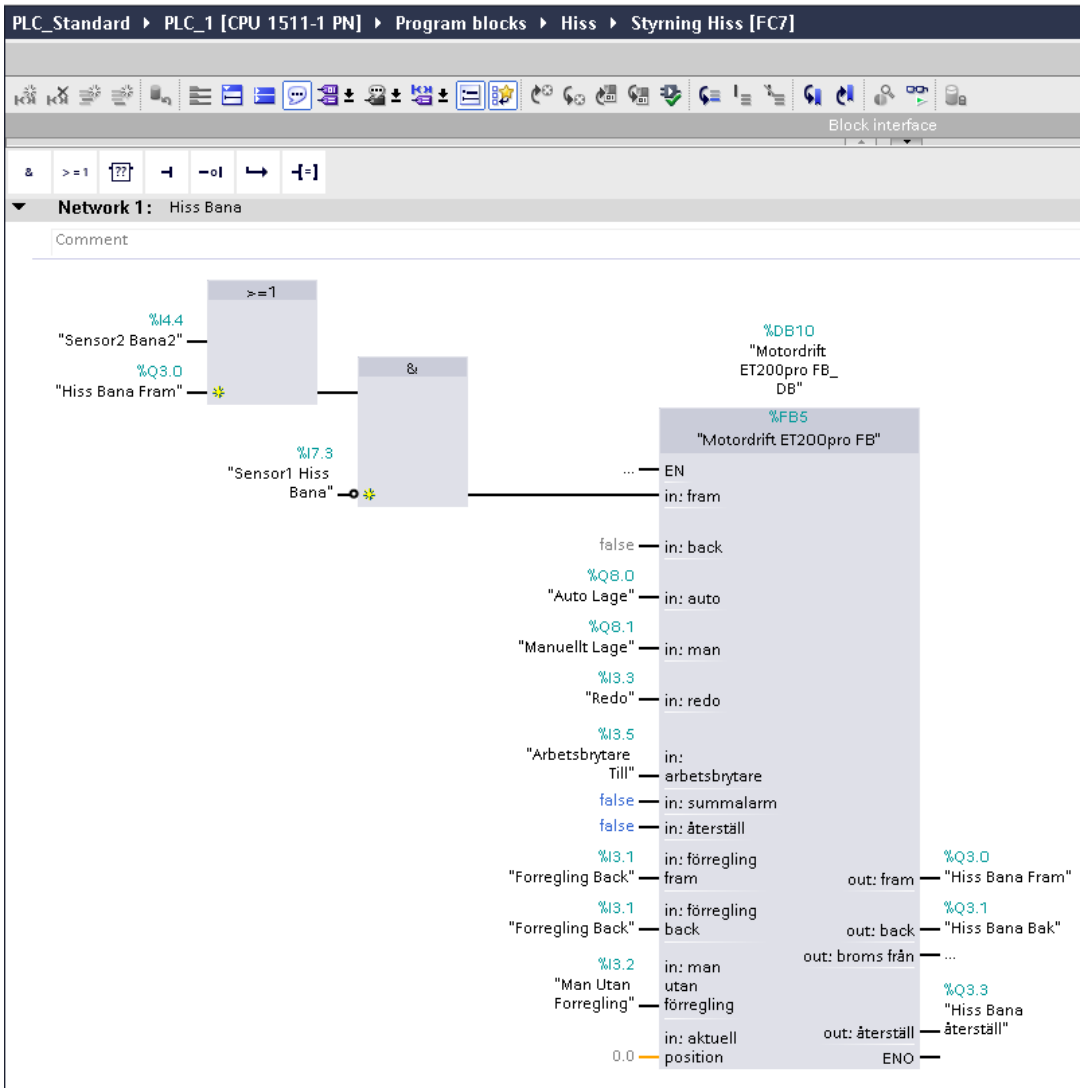
Figur 20. Styrning av Motordrift FB.

Figur 21 visar hur styrningen av detta Rörelse FB har programmerats. Rörelse FB kan användas till olika typer av rörelser exempelvis lyft, låsningar, indexeringar, stopp och andra rörelser. Rörelse FB kan styra en rörelse i två lägen, ett hemmaläge och ett arbetsläge. Den kan styras i ett automatiskt läge och ett manuellt läge. För det automatiska läget styrs rörelse FB genom att signaler skickas in på in:auto från och in:auto till, beroende på om användaren vill att rörelsen ska gå till hemmaläge eller till arbetsläge. I manuellt läge kan motorn styras genom knappar på en HMI panel som är kopplade till det tillhörande instans DB för blocket eller genom den externa manuellstyrningen som är kopplad inout:externManuell. Rörelse FB används på en plats, detta är vid cylinderstoppet mellan bana1 och vridbordet. Figur 21 visar hur den används för att styra cylindern. Cylindern styrs med två hållkretsar, en för att komma till sitt arbetsläge som är till och en för att komma till sitt hemmaläge som är från.



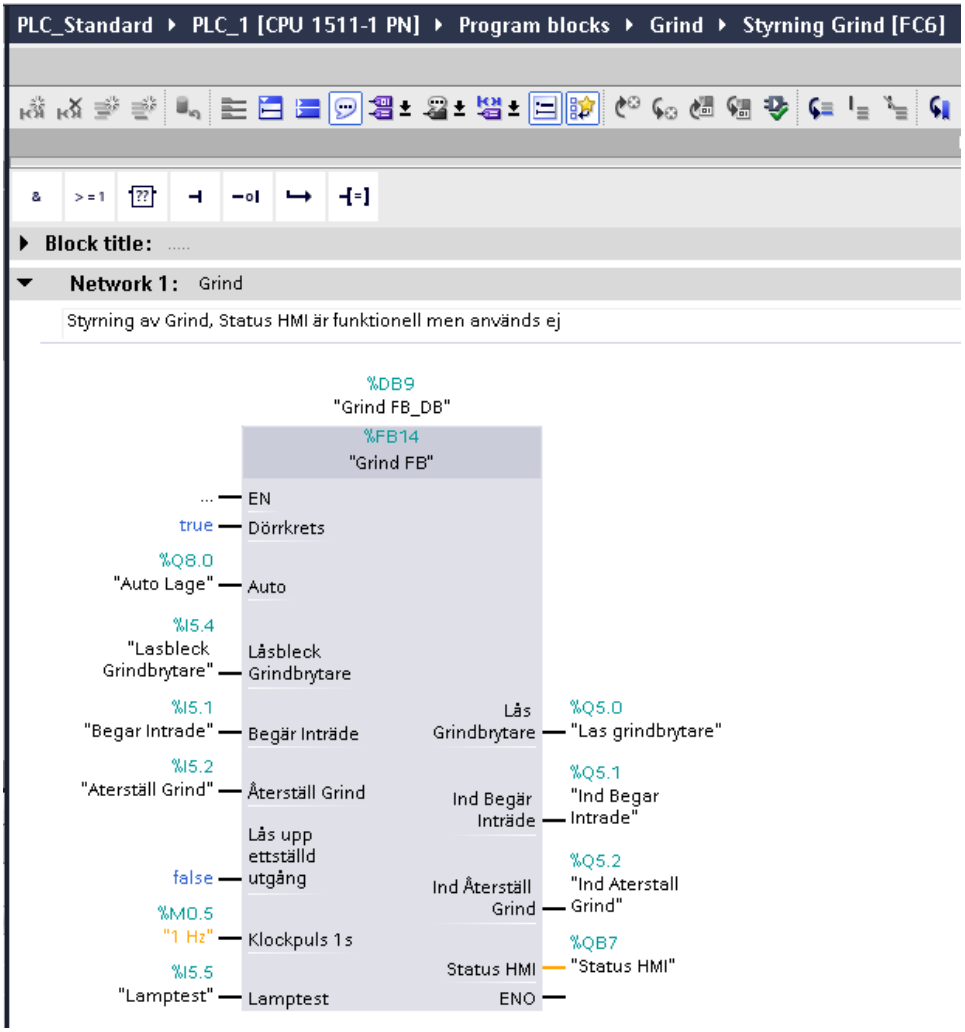
Figur 21. Styrning av Rörelse FB.

Figur 22 visar styrningen för motordrift ET200pro FB. Motordrift ET200pro FB kan användas för att styra motordrifter på ett ET200pro rack. Funktionen gör det möjligt att styra en sådan motor i två lägen, framåt och bakåt. Den kan styras i ett automatiskt läge och ett manuellt läge. För det automatiska läget styrs motordrift FB genom att signaler skickas in på in:fram och in:back beroende på om användaren vill att motorn ska drivas framåt eller om den ska drivas bakåt. I manuellt läge ska motordriften styras genom knappar på en HMI panel som är kopplade till det tillhörande instans DB för blocket. Det finns även möjlighet att använda summalarm och återställning av detta för det enskilda blocket. Motordrift ET200pro FB används för att styra banan på hissen. I Automatisk drift styrs den genom en hållkrets som startas när lådor kommer till sensor2 på bana2 och slutar när de är framme på hissen. I manuell drift styrs den av tryckknappar som är kopplade till de HMI tagar som styr de manuella funktionerna.



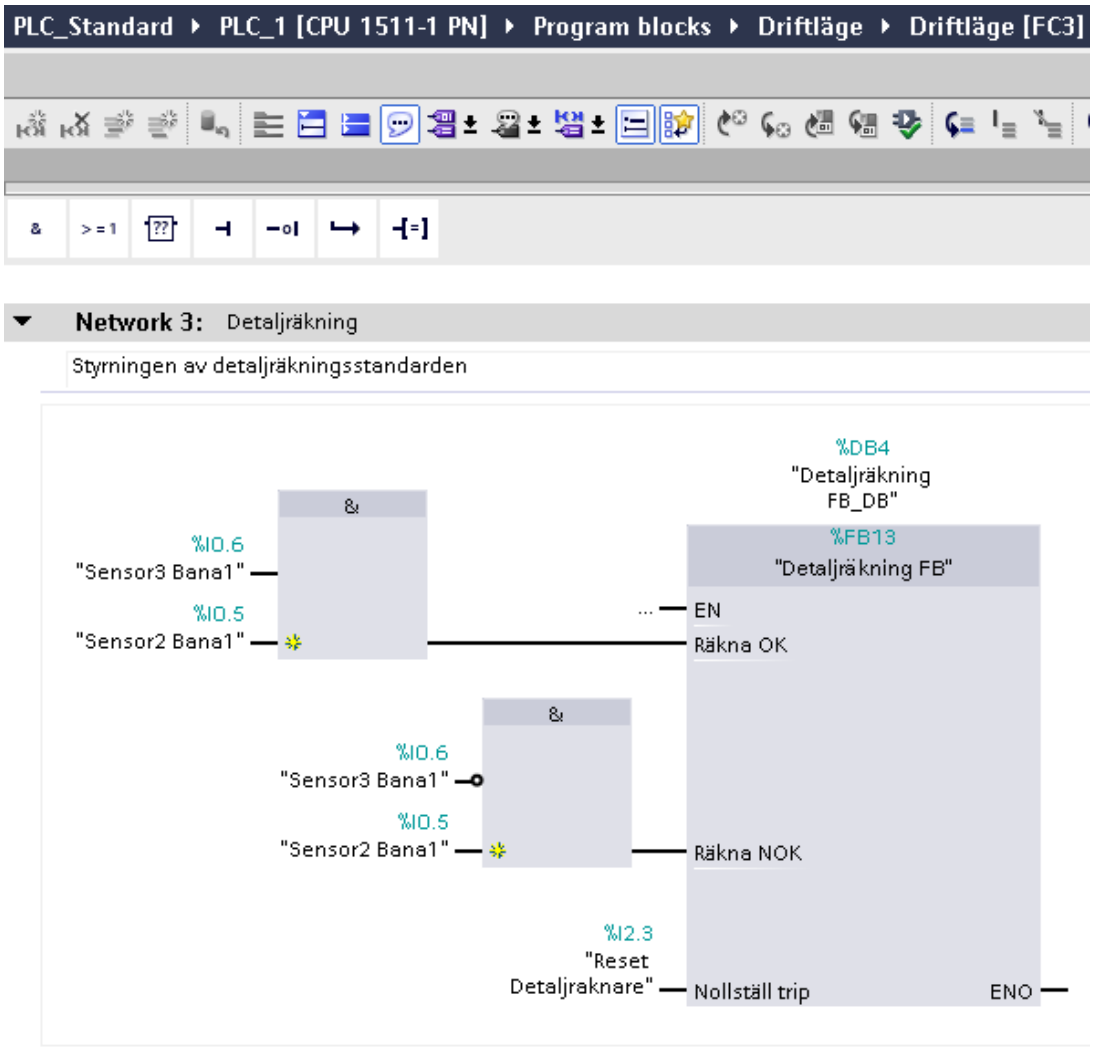
Figur 22. Styrning av Motordrift ET200pro FB.

I figur 23 visas styrningen av grind FB. Grind FB kan användas till en bred variation av grindar och säkerhetsgrindar även tillsammans med övrig säkerhet i cellen. För att öppna en grind styrd av grind FB måste det automatiska läget vara frånkopplat, Kretsen till de andra dörrarna vara hel, och inträde måste vara begärt. Användaren begär inträde genom tryckknappen begär inträde, knappen kommer då att blinka tills det automatiska läget är från. När det automatiska läget är från och lampan för inträde begärt lyser med ett fast sken kommer det vara möjligt att öppna dörren. För att återgå till ett automatiskt läge måste grinden återställas, detta görs genom tryckknappen återställ grind. När knappen trycks in kommer en indikering för återställning av grind börja att blinka, tills dörren är stängs då det istället lyser ett fast sken och om dörrkretsen är till låses grinden och det är möjligt att starta det automatiska läget igen. Grind FB har använts för att styra en grind med låsfunktion i Simumatik3D. Eftersom inga dörrkretsar används är ingången Dörrkrets alltid sann. Klockpulsen för indikeringarna är just nu på 1Hz men kan ändras efter den blinkfrekvens som önskas.



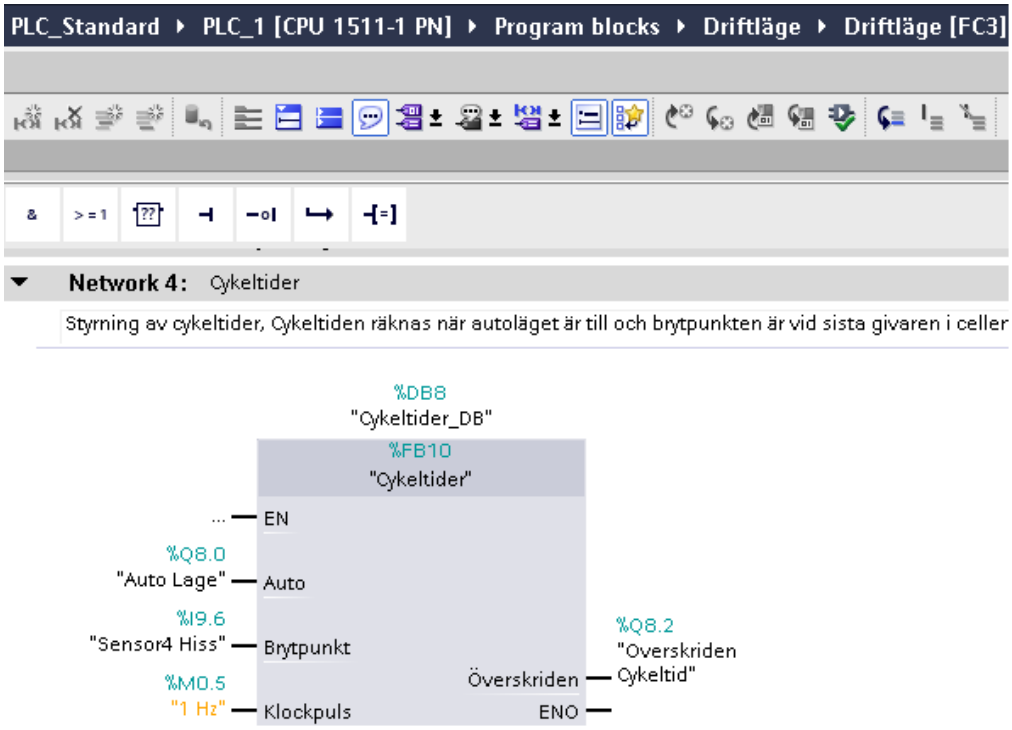
Figur 23. Styrning av Grind FB.

I figur 24 visas styrningen för Detaljräkning FB. Detaljräkning FB kan användas för att räkna detaljer i en hel produktion eller delar av den. Den styrs genom att signaler skickas in på Räkna OK för att räkna detaljer som är bra eller genom att skicka en signal in på Räkna NOK för att räkna detaljer som inte är bra. Det finns fyra värden som kan användas för information om antal detaljer. Det är trip antal OK, trip antal NOK, trip alla TOT och alla TOT. De första tre går att återställa genom att skicka en signal in på Nollställ trip. Funktionen används för detaljräkning i hela systemet, vid stoppcylindern avgörs det om det är en OK eller NOK detalj och en signal skickas in till detaljräkning FB.



Figur 24. Styrning av Detaljräkning FB.

I figur 25 visas styrningen av cykeltider FB. Cykeltider FB kan användas för att mäta cykeltider i en hel produktion eller delar av den. Den styrs genom att en signal skickas in på Auto, det startar räknaren av cykeltiden och sålänge den signalen är till kommer cykeltiden att fortsätta räkna. Det krävs även en signal in på Brytpunkt när den nuvarande cykeltiden ska flyttas till den senaste cykeltiden och börja räkna på en ny cykeltid. Upp till de senaste tio cykeltiderna kan sparas och användas. Cykeltider FB används för att räkna cykeltiden för hela cellen. Brytpunkten är när lådan har nått sista givaren med hissen. Två displayer används för att visa den nuvarande cykeltiden och den senaste cykeltiden.



Figur 25. Styrning av Cykeltider FB.

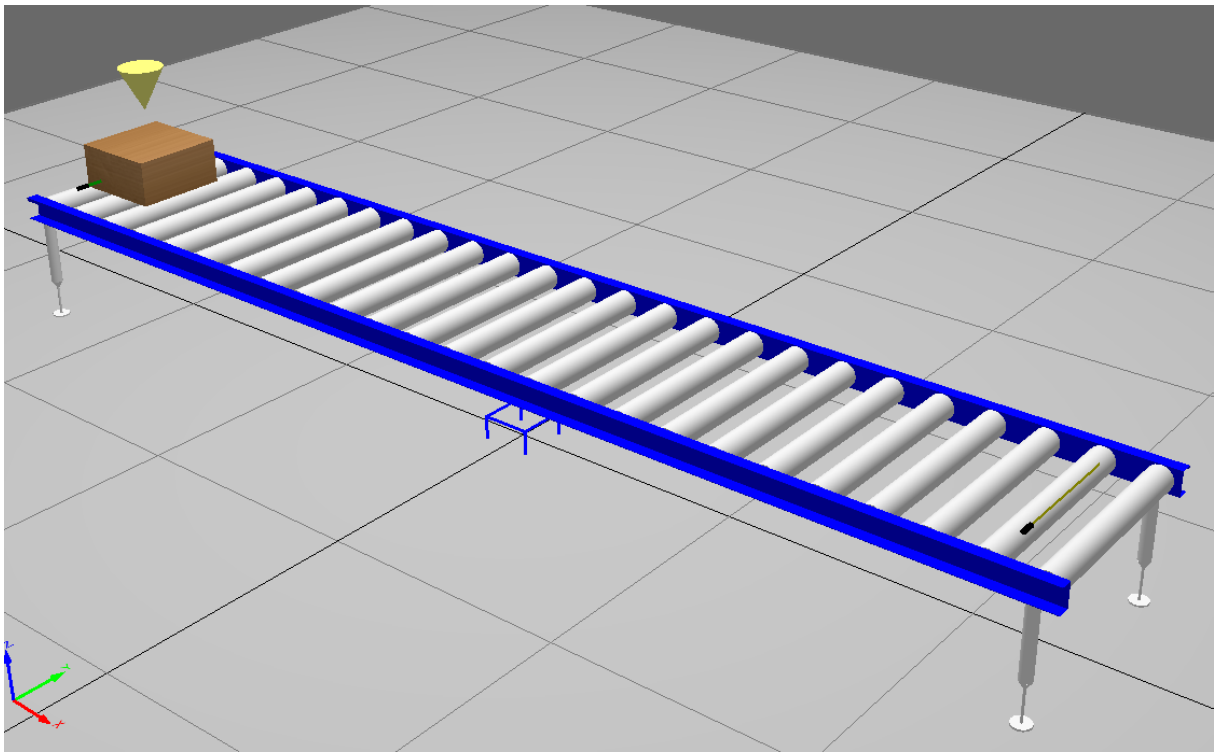
4.6 Konstruktion av emuleringsmodell

Emuleringsmodellen byggs helt utifrån de funktioner som valts ut i kapitel 4.4 *Val av funktioner*. Syftet med emuleringsmodellen är kunna verifiera att funktionerna utför rätt saker, kunna använda funktionerna på rätt sätt och skapa en riskfri testmiljö.

Hela emuleringsmodellen konstruerades i Simumatik3D, där en komponent i taget lades till för att emulera den PLC kod som programmeras i TIA portal. I Simumatik3D finns ett färdigt bibliotek med olika komponenter som enkelt kan läggas till i projektträdet. I ett nytt projekt är det enda som finns ett Main floor där hela systemet kommer att stå och sedan behöver nya grupper skapas för att kunna lägga till komponenter. Simumatik3D hjälper till att styra vilka komponenter som ska användas vart, då det enbart är möjligt att lägga till de komponenter som kan användas på det specifika stället. Till exempel, om en bana skapas kan enbart applicerbara komponenter läggas till såsom sensorer, cylindrar, stopp och rörelseaxlar.

Det första som byggdes i Simumatik3D i detta arbete var apparatskåpet för att lägga till en PLC, HMI panel, kontaktorer och strömförsörjning. Eftersom den första funktionen som testades var en motordrift påbörjades en konstruktion av en bana som kunde transportera produkten fram och tillbaka, både i manuell och automatisk drift. I banas båda ändar finns en sensor som detekterar produkten och ändrar drivningens riktning. Även en Productinput har lagts till i ena änden för att

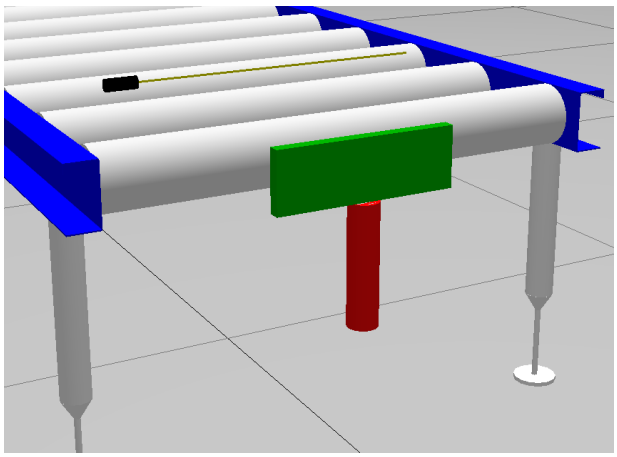
försörja systemet med produkter i form av enkla lådor och styrs av en tryckknapp. I figur 26 visualiseras hur systemet såg ut efter verifiering av första funktionen.



Figur 26. Systemet testar motordrift.

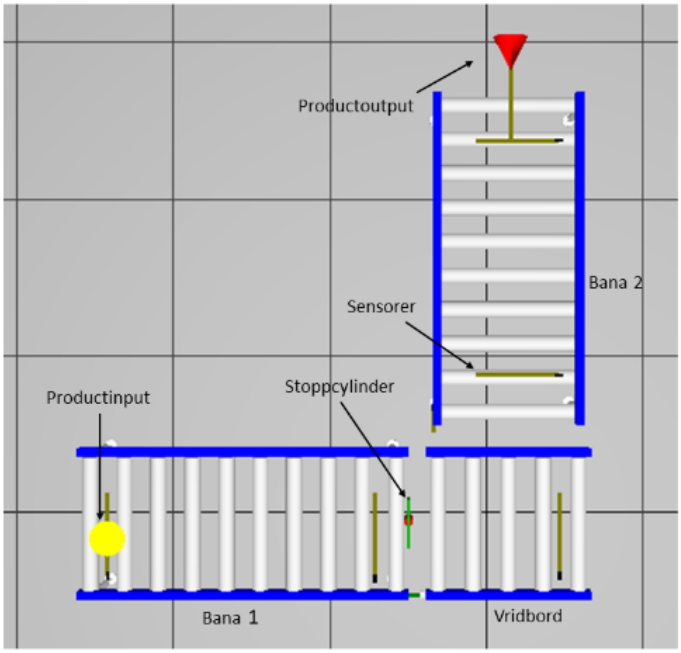
Systemet växer, anpassas och förändras desto mer funktioner som läggs in, men genom hela arbetet har en simpel och lätthanterlig modell med syfte att verifiera funktionerna valt att användas.

Efter den första motordriften som styr banan konstruerades en pneumatisk rörelse i form av en dubbelverkande cylinder vars funktion är att stoppa lådan i slutet på banan. För att få detta att fungera behövdes även ett pneumatikskåp läggas till med en kompressor och ventil för att styra cylindern. Cylinderns upp- och ner rörelse styrs med hjälp av en sensor som indikerar om lådan är på rätt plats och stoppets inbyggda ändlägessensorer, stoppet visualiseras i figur 27.



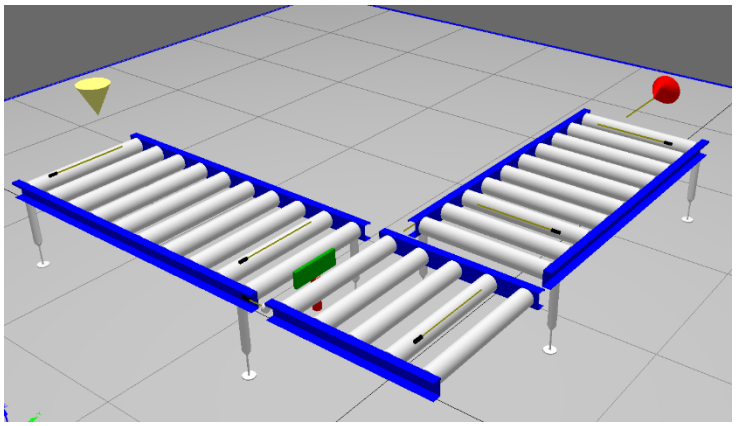
Figur 27. Cylinder med stoppfunktion.

När stoppet uppnått rätt funktioner expanderades systemet genom att lägga till ett vridbord och ytterligare en bana, figur 28 är en 2D layout som tydliggör vad varje komponent heter och hur systemet ser ut i detta steg.



Figur 28. 2D layout av systemet.

Vridbordet styrs med två motorer, den första styr banan på vridbordet och den andra styr vridningen. På varje bana finns sensorer som indikerar vart lådan befinner sig och hjälper till att kontrollera vilka banor som ska vara igång. Vridbordet har även två sensorer med tillhörande reflektorer som indikerar vilken position den står i, tillbaka eller framme. Detta betyder att motorn kan köras i båda riktningar i automatisk drift vilket verifierar den delen av funktionen. I figur 29 står den i tillbaka läge riktad mot bana 1. I slutet på bana 2 har en Productoutput lagts till som gör att lådan försvinner och ytterligare en låda kan startas i början på bana 1.

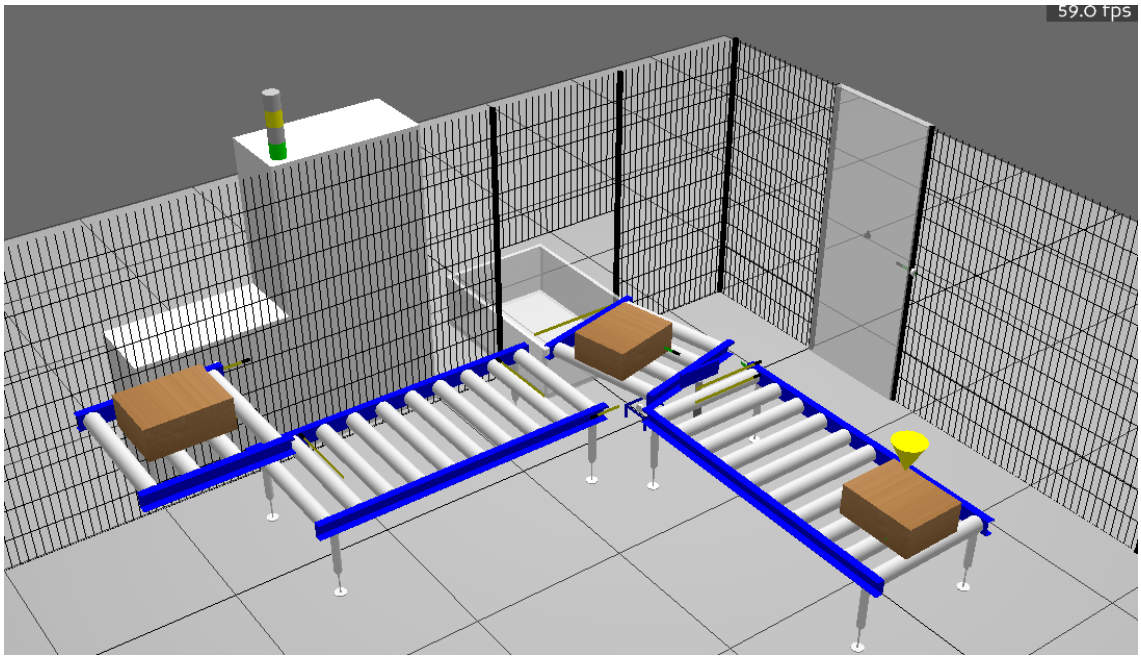


Figur 29. System med bana 1, vridbord och bana 2.

När alla funktioner som behandlas av detta arbete enligt kapitlet 4.4 *Val av funktioner* är införda i systemet har följande konstruerats i Simumatik3D:

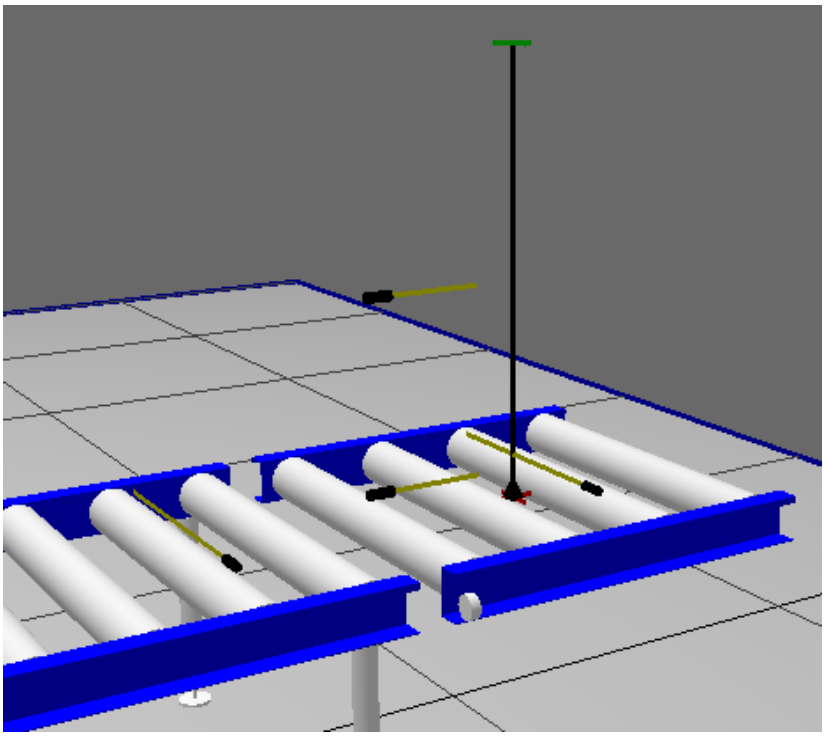
- 4st banor styrda med motordrift.
- Dubbelverkande cylinder i form av ett stopp, styrs med pneumatik.
- Vridbord som styrs av motordrift, både framåt och bakåt.
- Hiss, styrs av en motordrift, fungerar upp, ner och med en hastighetskontroll.
- Kontroll av produkt med detaljräknare vars funktion är att sortera bort fel sorts detaljer.
- Cykeltidsräknare som räknar både aktuell och senaste cykeltid.
- Ljusfyr som indikerar vilket läge systemet är i.
- Skyddskrets med låsning.
- 3st HMI paneler, manuellfunktion, driftläge och inträde/låsning av skyddskrets.

Figur 30 visar hur det fullständiga systemet ser ut.



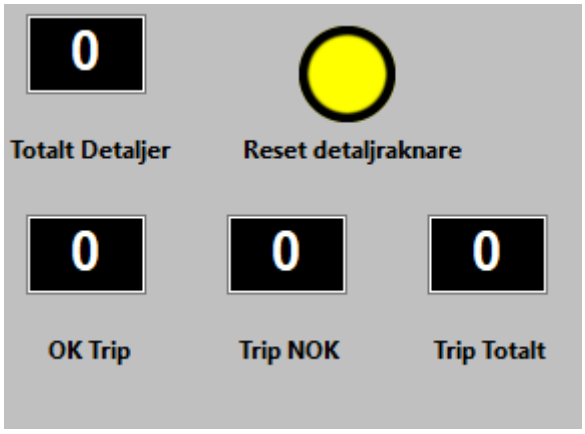
Figur 30. Fullständigt system.

Hissen förflyttar sig upp och ner med hjälp av en motor som styr en Electric Translational Joint. Hissen använder sig utav sex sensorer. En sensor innan hissen som indikerar att en detalj är på ingång och startar banan på hissen. En sensor på hissen som stoppar banan och indikerar att detaljen är rätt placerad för att kunna starta hissens rörelse uppåt, samma sensor styr även när hissen ska åka ner genom att indikera att den inte längre har en detalj i hissens övre läge. Hissen använder de andra fyra som ändlägesgivare samt styrning utav hastigheten. Två sensorer känner av en reflektor som sitter monterad på hissen när den åker förbi och startar/stoppar högfartshastigheten. Hissen visas i figur 31.



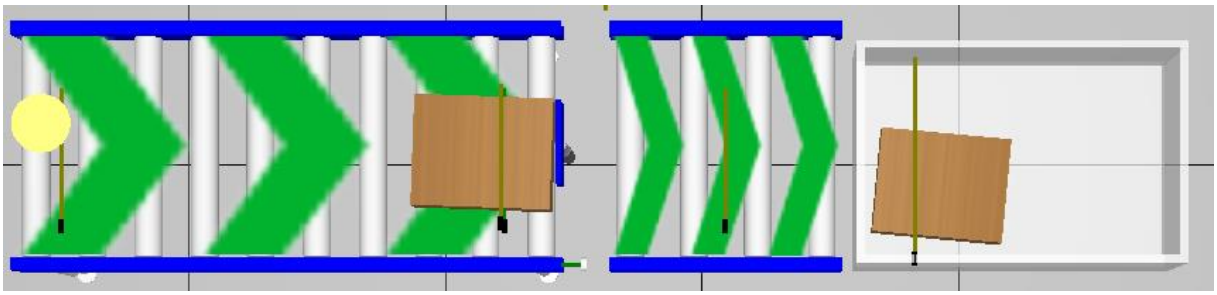
Figur 31. Hissfunktion i Simumatik3D.

I systemet finns en produktkontroll som kontrollerar om detaljen håller en specifik höjd. I detta läge räknas det även upp i HMI panelen för detaljräkning och visualiseras med hjälp av displayer. I HMI panelen räknas det godkända detaljer, icke godkända detaljer och totalt antal körda detaljer och kan återställas med en tryckknapp. Det finns även en funktion som räknar totalt antal körda detaljer som inte kan återställas och kommer räkna från den dag funktionen skapades för att se hur många detaljer som körs genom systemet. Figur 32 visar hur HMI panelen ser ut för detaljräkning.



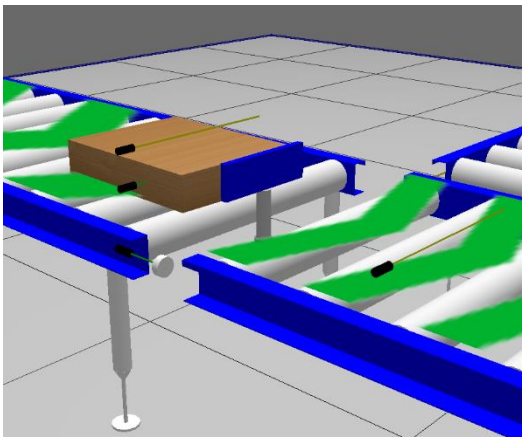
Figur 32. HMI panel för detaljräkning.

När kontrollen inte godkänner en detalj skickas den vidare till en container förbi vridbordet som är tänkt att operatören ska kunna tömma manuellt. I containern finns en sensor som känner av när den detalj som inte godkänts har åkt ner i containern. Se figur 33 för flödet av en icke godkänd detalj.

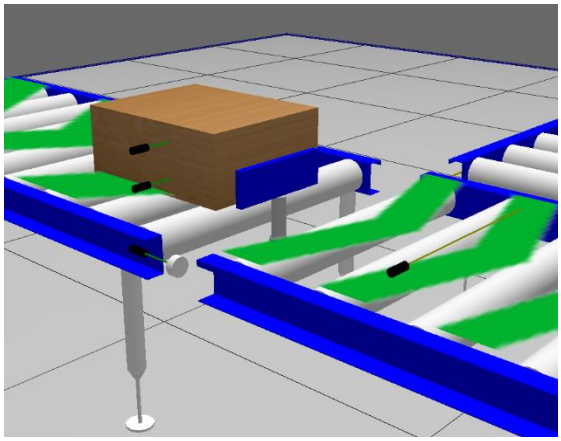


Figur 33. Flödet av en icke godkänd detalj.

Kontrollen sker med hjälp av två sensorer. Om enbart den nedre är påverkad är detaljen inte godkänd och är båda två påverkade är det en godkänd detalj. Figur 34 och 35 visar skillnaden mellan en godkänd detalj och en icke godkänd detalj.



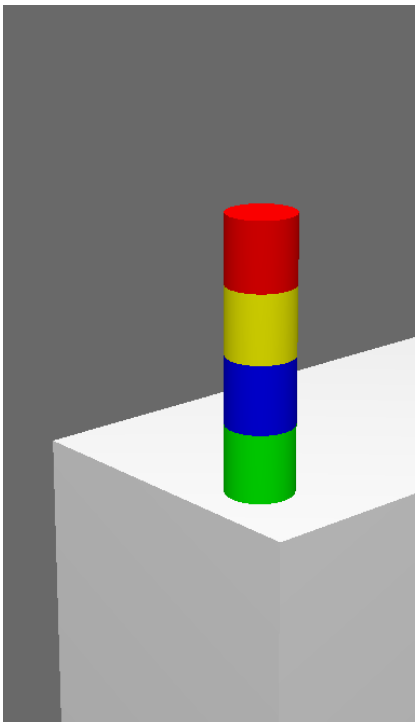
Figur 35. NOK detalj.



Figur 34. OK detalj.

Cykeltidsräknaren är implementerad i HMI panelen och visas genom två olika displayer, där en visar den aktuella cykeltiden och en som visar den föregående cykeltiden. Cykeltidsräknaren aktiveras genom att en produkt startas och sensorn i början på bana 1 aktiveras. Räknaren är klar när produkten har nått systemets slut, vilket är i hissens övre läge. Det finns även en cykeltidsreferens som den aktuella cykeltiden refererar till, om den aktuella cykeltiden överskrider referensen indikeras att cykeltiden är överskriden, detta visas med hjälp av ljusfyren.

Ljusfyren visar information om systemets status genom att lamporna indikerar när systemet är i specifika lägen men är inte en standardiserad funktion. Det finns fyra olika lampor i ljusfyren, grön, gul, röd och blå. Den gröna lampan indikerar att systemet är i automatisk drift. Gul lampa betyder att cykeltiden är överskriden vilket betyder att produkten har tagit för lång tid att komma till systemets ändläge. Röd lampa betyder i detta system att manuellt läge är aktiverat, i manuellt läge kan alla manuella funktioner användas, såsom att styra alla banor, vridbord och hiss. Blå lampa används inte i detta system utan den är bara förberedd att kunna användas om systemet expanderar. Figur 36 visas hur ljusfyren ser ut.

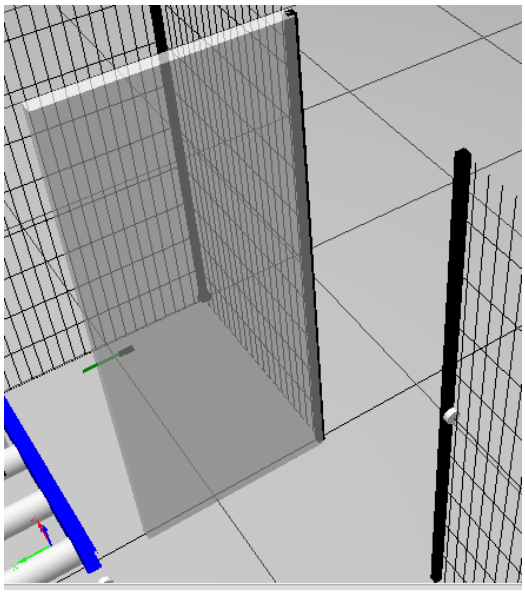


Figur 36. Ljusfyr.

I modellen har en dörr in till systemet lagts till tillsammans med grindar och dessa agerar som en skyddskrets som normalt sätt förhindrar operatören att gå in i riskområdet utan att stanna systemet. I modellen implementeras skyddskretsen genom en Doorjoint där sedan en 3D modell konstrueras som i detta fall föreställer en dörr. På 3D modellen adderades ett lås som styrs med två tryckknappar, Begär inträde och Återställ grind. När tryckknappen Begär inträde aktiveras genom HMI panelen går systemet ur autoläge och stannar därmed systemet och låset på dörren låses upp så dörren kan öppnas. För att kunna starta systemet igen behöver kretsen återställas genom knappen Återställ grind. Då knapparna har ljusfunktion, såsom blinka när operatören begärt inträde eller att knappen Återställd grind lyser när kretsen är redo att återställas, har en funktion för att testa om lamporna fungerar lagts till. Figur 37 visar hur HMI panelen för skyddskretsen ser ut.

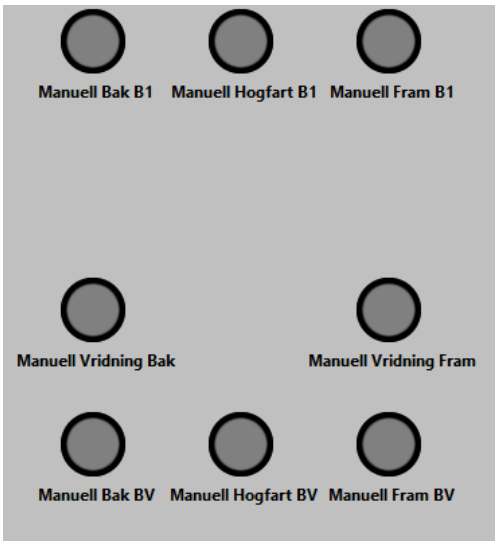
Figur 37. Skyddskrets HMI.

På dörren har en sensor adderats med tillhörande reflektor som känner av att skyddskretsen inte är bruten. Om skyddskretsen är bruten går det inte att återställa kretsen och låsa dörren. I figur 38 visas dörr och sensor.

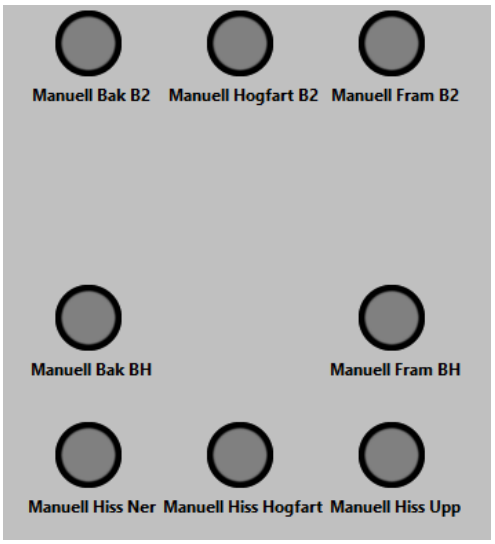


Figur 38. Dörr med låsfunktion.

Alla funktioner som styr en rörelse kan köras manuellt genom en HMI panel. Det finns två kriterier för att få köra manuellt i systemet; skyddskretsen måste vara till och manuellt läge av hela systemet måste vara till. När dessa två kriterier är uppfyllda kan alla funktioner köras manuellt genom tryckknapparna på HMI panelen enligt figur 39 och figur 40.



Figur 39. Manuellfunktioner bana 1 och vridbord.

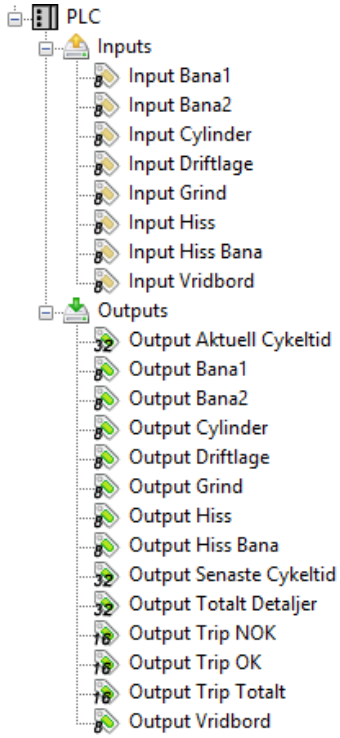


Figur 40. Manuellfunktoner bana 2 och hiss

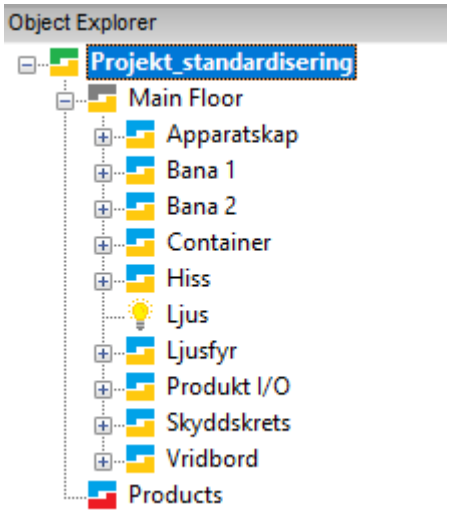
Manuellfunktionerna är indelade i fyra delar som styr bana 1, bana 2, vridbord med bana och vridning samt hiss med bana och rörelse upp och ner. Knapparna styr följande:

- Bana 1 styrs med knapparna benämnt B1.
- Bana 2 styrs med knapparna benämnt B2
- Bana vridbord styrs med knapparna BV och vridningen styrs med vridning fram/bak.
- Bana hiss styrs med knapparna BH och upp/ner med Hiss ner och Hiss upp med funktion för högfart.

Konstruktionen i Simumatik3D är skapat på ett sådant sätt att det är lätt att expandera och lägga till ytterligare funktioner. Genom hela arbetet har det gjorts lätt att överskådligt se vilka komponenter som hör hemma till vilken del i systemet. In- och utgångar är uppdelade på olika in- och utgångskort som tillhör en viss del av systemet, detta har gjorts för att det ska vara lätt att följa hur allt i Simumatik3D är i hopkopplat samt att det ska vara lätt att komplettera. Figur 41 och 42 visar hur det har gjorts överskådligt i Simumatik3D.



Figur 42. In- och utgångar i Simumatik3D.



Figur 41.Projektträd med grupperingar.

4.7 Verifiering av standarden

I tidigare steg tydliggörs att PLCSIM Advanced är ihopkopplat med både Simumatik3D och TIA portal. I PLCSIM Advanced finns en virtuell PLC skapad som är ihopkopplad med Simumatik3D:s PLC vilket betyder att när programkod laddas över från TIA portal till PLCSIM Advanced laddas det även in i Simumatik3D:s PLC. Detta gör det möjligt att styra Simumatik3D:s emuleringsmodell med TIA portals programkod. Men för att detta ska fungera på ett korrekt sätt behöver de in- och utgångar som finns i TIA portal benämnas på samma sätt i de in- och utgångar som skapas i Simumatik3D:s PLC.

Projektengagemangs funktioner verifieras genom funktionstester i emuleringsmodellen och ser till att den fungerar utan att på något sätt ändra funktionens ursprungskod. Emuleringsmodellen anpassas och konstrueras efter de funktionerna som valts. Detta beskrivs mer i kapitel 4.5 *Uppbyggnad av standard*. Verifiering i detta sammanhang handlar om att se till att funktionerna som Projektengagemang skapat fungerar att använda i emuleringsmodellen utan att ändra i funktionsblocken och att funktionerna utför rätt saker. Till exempel verifieras att funktionsblocket Motordrift kan kompletteras med programkod utanför funktionsblocket och kan kopplas ihop med Simumatik3D och styra en bana. Som nämnt i kapitel 4.1 *Tillvägagångssätt* justeras PLC koden och emuleringsmodellen tills funktionen har blivit verifierad innan den standardiseras och systemet expanderas. En verifierad funktion blir sedan en standardiserad funktion.

För att en funktion ska anses vara verifierad måste den först genomgå ett funktionstest med godkänt resultat. Varje funktion testas med ett individuellt funktionstest som är utformat för att testa just den funktionen. Ett exempel på funktionstest kan ses i tabell 2 där en del av Motordriftens funktionstest visas där bland annat test av kommunikation, autodrift och manuelldrift genomförs.

Tabell 2. Funktionstest av motordrift.

Funktionstest Motordrift				Utf: Marcus Hermansson & Joakim Ramn
Test	**Kontrollera:**	**Verifiera att:**	**Godkänt/ icke godkänt**	**Kommentarer**
Kommunikationstest	TIA portal online, virtuella PLC online Simumatik har startad simulering.	In -och utgångskort får signal OK.	Godkänt	
Test Autodrift	Signal fram till, motorskydd OK, förregling fram till, arbetsbrytare till.	Startar motor fram.	Godkänt	
	Signal bak till, motorskydd OK, förregling bak till, arbetsbrytare till	Startar motor bak.	Godkänt	
	Signal fram till, signal högfart till motorskydd OK, förregling fram till, arbetsbrytare till	Startar motor fram högfart	Godkänt	Används endast i vissa delar av systemet.
	Signal bak till, signal högfart till motorskydd OK, förregling bak till, arbetsbrytare till	Startar motor bak högfart	Godkänt	Används endast i vissa delar av systemet.
	Om autoläge, motorskydd OK, förregling fram till, arbetsbrytare till bortfaller.	Motor fram stoppas.	Godkänt	
	Om autoläge, motorskydd OK, förregling bak till, arbetsbrytare till bortfaller.	Motor bak stoppas	Godkänt	
	Autoläge till efter bortfall av autoläge.	Startar efter återupptaget autoläge	Godkänt	Vid vissa positioner, kan krävas manuell körning.
Test Manuelldrift	Manuellt läge, tryckknapp fram aktiverad, motorskydd OK, arbetsbrytare till, förregling fram till	Motor fram startar	Godkänt	
	Manuellt läge, tryckknapp bak aktiverad, motorskydd OK, arbetsbrytare till, förregling bak till	Motor bak startar	Godkänt	
	Manuellt läge, tryckknapp bak aktiverad, tryckknapp högfart, motorskydd OK, arbetsbrytare till, förregling bak till	Motor startar högfart bak	Godkänt	Används endast i vissa delar av systemet.
	Manuellt läge, tryckknapp fram aktiverad, tryckknapp högfart, motorskydd OK, arbetsbrytare till, förregling fram till	Motor startar högfart fram	Godkänt	Används endast i vissa delar av systemet.
	Om manuellt läge, tryckknapp fram aktiverad, motorskydd OK, arbetsbrytare till, förregling fram till bortfaller.	Motor fram stoppas.	Godkänt	
	Om manuellt läge, tryckknapp bak aktiverad, motorskydd OK, arbetsbrytare till, förregling bak till bortfaller.	Motor bak stoppas.	Godkänt	

5. Utvärdering

Artefakten som är utvecklad i detta arbete är ett bibliotek med standardfunktioner och en projektfil i TIA portal där styrningen av funktionerna finns. Detta kompletteras med en emuleringsmodell för att verifiera funktionerna. Funktionerna i biblioteket är verifierade i emuleringsmodellen med hjälp av funktionstester. Arbetet har följt metoden Design and creation som i sin iterativa process har utvärdering som näst sista steg. I utvärderingen ska den utvecklade artefakten granskas, den ska bedömas om den är värdeskapande men även på vilket sätt den är värdeskapande och det förväntade resultatet ska jämföras med det uppnådda resultatet.

Utvärderingen baseras på valideringen och kommer att genomföras två gånger; första gången där validering utförs av artefaktens utvecklare och andra gången där Projektengagemang validerar standarden. Valideringen behandlar följande huvudpunkter:

- Om standarden är användbar.
- Om standarden skapar värde.
- Om standarden tillsammans med emuleringsmodellen kan bidra vid upplärning av nya automationsingenjörer.
- Om standarden är lätt att komplettera med nya verifierade funktioner.

Standarden kommer att valideras genom ett frågeformulär som både artefaktens utvecklare och Projektengagemang svarar på, detta sammanställs sedan till en gemensam utvärdering. Frågeformuläret kan ses i tabell 3.

Tabell 3. Frågeformulär till validering av standard.

Validering av standard			
Huvudfråga	**Om svar JA på huvudfråga**	**Kommentarer**	**JA/ NEJ**
1. Är standarden användbar?			
	1.1 Vid nya projeket?		
	1.2 Vid upplärning?		
	1.3 Vid felsökning i pågående projekt?		
	1.4 Är den enkel att använda?		
	1.5 Är de standardiserade funktionerna användbara?		
	1.6 Finns möjlighet till effektivare programmering?		
	1.7 Övriga kommentarer om användning?		
2. Skapar standarden värde?			
	2.1 Skapar den värde långsiktigt?		
	2.2 Skapar den värde kortsiktigt?		
	2.3 Minskas ledtid vid programmering?		
	2.4 Minskas ledtid vid felsökning?		
	2.5 Finns möjlighet till minskad variation i programkod?		
	2.6 Finns möjlighet till ökad kvalitet av programkod?		
	2.7 Övriga kommentarer på värdeökning?		
3. Kan standarden bidra vid upplärning?			
	3.1 Är projektet i TIA portal lätt att förstå?		
	3.2 Är emuleringsmodellen lätt att förstå?		
	3.3 Är det en bra testmiljö?		
	3.4 Övriga kommentarer vid upplärning?		
4. Kan standarden kompletteras?			
	4.1 Är det enkelt att standardisera nya funktioner?		
	4.2 Är det enkelt lägga in nya funktioner i TIA portal?		
	4.3 Är det enkelt att expandera emuleringsmodellen?		
	4.4 Övriga kommentarer vid komplettering?		

5.1 Validering 1

Den första valideringen görs av de som utvecklat artefakten. Första huvudfrågan att svara på var om standarden är användbar. Huvudfrågan besvarades med ett Ja och även dess följdfrågor men med ett antal kommentarer.

- Fråga 1.3, vid felsökning i pågående projekt kommer standarden vara användbar på lång sikt eftersom det kan ta tid innan en felsökning sker på en anläggning där de standardiserade funktionerna har använts.
- Fråga 1.5, de standardiserade funktionerna är användbara och det är vanligt förekommande funktioner som använts men för att nå full utnyttjandegrad kommer en ständig komplettering av standarden behöva göras.

- Fråga 1.6, det finns möjligheter till effektivare PLC programmering då det finns ett bibliotek tillgängligt med lättillgängliga standardiserade funktioner att föra in direkt i nya projekt.
- Fråga 1.7, övriga kommentarer om användning är att det rekommenderas att bekanta sig med kommunikationen mellan mjukvarorna för verifiering av standarden då detta kan vara ett avancerat moment.

Den andra huvudfrågan att svara på var om standarden skapar värde. Huvudfrågan besvarades med ett Ja och även följdfrågorna men med ett antal kommentarer.

- Fråga 2.3, det finns möjlighet att minska ledtiden vid programmering genom att de standardiserade funktionerna återanvänds istället för att nya funktionsblock måste skapas.
- Fråga 2.5, variationen kan minskas i programkoden då programmerare använder sig av samma funktioner som en grund och bygger sina program utifrån den grunden.
- Fråga 2.6, eftersom funktionerna som standardiseras är tillhandhållna från Projektengagemang och är funktioner som används i flera automatiseringsprojekt bör dess kvalitet vara god och desto mer dessa funktioner används desto mer kommer den generella kvaliteten av programkod öka.
- Fråga 2.7, det kan vara nödvändigt att bekanta sig med projektet och biblioteket för att standarden ska bli värdeökande.

Den tredje huvudfrågan att svara på var om standarden kan underlätta upplärning. Huvudfrågan besvarades med ett Ja och följdfrågorna men med en kommentar.

- Fråga 3.4 det finns många olika möjligheter för upplärning tillsammans med standarden och den kompletterande emuleringsmodellen. Det går att bara iaktta och se hur det ska se ut, kolla i koden och skriva ett förväntat resultat för att sedan kolla i emuleringsmodellen och se det faktiska resultatet. Eller till och med ta bort all styrning men ge tillgång till de standardiserade funktionerna för att den nyanställde ska programmera en fungerande styrning till emuleringsmodellen.

Den fjärde huvudfrågan att svara på var om standarden kan kompletteras. Huvudfrågan besvarades med ett Ja och även på alla följdfrågor med kommentarer förutom 4.4.

- Fråga 4.1, för att standardisera nya funktioner måste funktionerna verifieras i Simumatik3D genom ett funktionstest. Det blir ett tydligt resultat om funktionen kan standardiseras eller inte och det blir enkelt för användaren att standardisera nya funktioner.
- Fråga 4.2, det är enkelt att lägga till nya funktioner i TIA portal på grund av den goda strukturen och uppdelning av funktioner som gjorts, om fortsatt arbete sker i samma struktur kommer det att fortsätta vara tydligt.
- Fråga 4.3, för att expandera emuleringsmodellen krävs viss kunskap i Simumatik3D, denna kunskap går dock snabbt att lära sig genom att bara öppna en tom fil och testa sig fram, Simumatik3D begränsar konstruktionen till vad som går att göra vilket gör det svårt att göra fel.

Standarden anses av artefaktens utvecklare som helhet vara validerad, alltså har rätt standard byggts för ändamålet.

5.1 Validering 2

Den andra valideringen är utförd av Projektengagemang ur ett perspektiv som användare. Den första huvudfrågan att svara på är om standarden är användbar. Projektengagemang svarade Ja på huvudfrågan och alla följdfrågor utom 1.3.

- Fråga 1.3, felsökningen sker i det pågående projektet och standarden används inte vid den felsökningen, men det finns möjlighet att återkoppla fel i standarden för att uppnå en bättre standard.

- Fråga 1.5, för att de standardiserade funktionerna ska bli mer användbara behöver kompletteras med övervakningslarm och liknande.

Den andra huvudfrågan att svara på är om standarden skapar värde. Projektengagemang svarade Ja på huvudfrågan och samtliga följdfrågor.

- Fråga 2.1, det skapar ett stort långsiktigt värde att ha standardiserad kod.
- Fråga 2.3, ledtiden kan minskas vid programmering om det finns mycket färdig kod, det betyder att fokus kan ligga på huvuduppgiften och lösa problem istället.
- Fråga 2.4, ledtiden vid felsökning kan minskas om alla projekt är lika då alla vet vart felet kan hittas.

Den tredje huvudfrågan att svara på är om standarden kan bidra vid upplärning. Projektengagemang svarade Ja på huvudfrågan och samtliga följdfrågor.

- Fråga 3.2, emuleringsmodellen kan bli lätt att förstå då en grundutbildning i mjukvaran genomförts.

Den fjärde huvudfrågan att svara på är om standarden kan kompletteras. Projektengagemang svarade Ja på huvudfrågan och alla följdfrågor utom 4.1 och 4.3 där det råder osäkerheter.

- Fråga 4.1, det är osäkert om det är enkelt att standardisera nya funktioner då det finns bristande kunskap i mjukvaran Simumatik3D.
- Fråga 4.3, det är osäkert om det är enkelt att expandera emuleringsmodellen då det finns bristande kunskap i mjukvaran Simumatik3D

5.2 Sammanställning av valideringar

När de två valideringarna jämförs är det mycket lite som skiljer de åt, detta anses som positivt. Den första punkten där valideringen skiljer sig åt är fråga 1.3 där artefaktens utvecklare ansåg att standarden kan vara användbar vid felsökning av pågående projekt. Projektengagemang ansåg att den inte var till användning eftersom felsökningen sker i det pågående projektet och standarden används inte vid den felsökningen. I ett kortsiktigt perspektiv har Projektengagemang rätt. Men ur ett långsiktigt perspektiv kommer standarden ha ett exponentiellt ökande värde desto längre tid den används. Om samma funktioner används i tio år kommer troligtvis alla på företaget känna igen koden och vid en felsökning i ett pågående projekt skulle det kunna vara användbart att det finns en kod som känns igen.

En sammanställning gjordes utifrån resultatet från validering ett och två, då har följande punkter kunnat tydliggöras:

- Det standardiserade funktioner kan vara användbara, men viss komplettering skulle behöva genomföras för att uppnå högre utnyttjandegrad.
- Det kan underlätta långsiktigt vid felsökning genom att programmet får en känd struktur som alla kan hitta i.
- Ledtiden kan minskas vid programmering då en del av programkoden redan är klar och nya funktionsblock behöver inte skapas.
- Minskad variation och ökad kvalitet kan uppnås genom återanvändning av standardiserade funktioner.
- Standarden med tillhörande emuleringsmodell kan underlätta mycket vid upplärning om grundläggande kunskap i Simumatik3D finns.
- Standarden med tillhörande emuleringsmodell är lätt att komplettera och att föra in nya funktioner om grundläggande kunskap i Simumatik3D finns.

6. Diskussioner

6.1 Projektets helhet

Arbetets mål är att skapa en standard för PLC programmering och även om en grund har skapats för hur en standard till PLC programmering kan se ut har kunskapen som erhållits under arbetet varit det mest givande. Projektet har till stor del varit ett praktiskt arbete och baserats på teoretiska grunder. Den teoretiska referensramen gav förberedande kunskap samt förståelse för hur arbetet kunde genomföras och vad som skulle uppnås. Något som ökar mycket inom industrin är användning av emulering- och simuleringsmodeller, detta gör arbetet aktuellt och är något som kommer kunna nyttjas i arbetslivet framöver.

Ny kunskap har erhållits inom områden som automatisering och produktion, till exempel hur standardisering kan gå till, hur en emuleringsmodell kan konstrueras, hur en anpassning av redan färdiga funktionsblock till ett system kan utföras och hur virtuella PLC enheter kan konfigureras. En av utmaningarna under arbetet var att få fyra mjukvaror, TIA portal, Simumatik3D, PLCSIM Advanced och VMWare, att kommunicera med varandra. Denna utmaning gav större förståelse och mer kunskap inom IT kommunikation.

6.2 Metoden

Design and creation research strategy har varit en bra metod för arbetet då det har gett en struktur att följa när en artefakt utvecklas. Eftersom Design and creation genomförs i fem övergripande steg och metoden tydligt redogör för vad som förväntas i varje steg men ändå lämnar utrymme för att enkelt kunna anpassas har metoden passat bra för arbetet. Om en annan metod valts, exempelvis Lean's metod PDCA eller Sex Sigma's metod DMAIC hade utvecklingen och framtagningen av standarden inte varit lika tydlig. Arbetet var till stor del ett praktiskt genomförande och Design and creation fokuserar på utveckling och användning av en artefakt, som i detta fall är standarden. Vid användning av PDCA och DMAIC ligger mycket arbete i planeringsstadiet och insamling av data. Hade dessa metoder använts hade mer tid lagts på detta än utvecklingen av standarden.

6.3 Konstruktion av emuleringsmodell

I projektet och emuleringsmodellen återanvänds en del av funktionerna för att kunna få ett flöde i systemet och ett system som har något typ av syfte. Systemet kommer med största sannolikhet att förändras desto mer funktioner som implementeras och för att bibehålla flödet och ett system som är verklighetstroget kommer flera funktioner att behöva användas mer än en gång. Sex olika funktioner har implementerats i systemet. Mängden funktioner var inte specificerat innan projektet startade, detta var bra då det möjliggjorde att kunna justera antalet funktioner efter den utsatta tiden för arbetet.

6.4 Begränsningar i Simumatik3D

Simumatik3D har varit en relativt lätthanterlig emuleringsmjukvara för arbetet som har standardiserat vanligt förekommande funktioner då det är lätt att lägga till nya komponenter direkt ur Simumatik3D:s tillhörande bibliotek och koppla ihop dessa. När väl kommunikationen är etablerat och korrekt infört är det enkelt att testa programkoden. Men det finns begränsningar i Simumatik3D som hade underlättat både vid konstruktion och verifiering om de förbättrats.

Det finns ett begränsat urval av komponenter i Simumatik3D, vilket kan betyda att visa förenklingar och antagningar kan behöva göras för att systemet ska uppnå den programmerade funktionen. Många komponenter går att göra inställningar på för att ändra dess funktion vilket gör komponenter anpassningsbara, men det finns bara ett specifikt urval av komponenter så om en verklig maskin eller

robotcell har en specifik uppgift begränsas Simumatik3D:s möjligheter. Detta har dock inte begränsat arbetet då funktioner som är vanligt förekommande använts.

Vid konstruktion hade det underlättat om det fanns som komplement till koordinatpositioner möjlighet att dra ut komponenter till arbetsytan samtidigt som den aktuella positionen visas och automatiskt kunna fästa till golvet eller andra komponenter. Detta gäller även vid konstruktion av HMI paneler då det även där enbart finns koordinatpositionering. Detta hade kunnat minska ledtiden något vid framtagning av modellen samtidigt som det skulle upplevas mer lätthanterligt.

I Simumatik3D finns det en funktion för att ta en sensor ur funktion för att kunna testa hur systemet reagerar när sensorer går sönder. Dock finns det ingen möjlighet att direkt i Simumatik3D ändra en sensors status från noll till ett som skulle kunna ge möjlighet att testa hur systemet reagerar om spånor, smuts eller något liknande fastnat på sensorn.

I vissa funktionsblock som standardiserats finns det möjlighet att lägga till övervakningslarm som aktiveras om specifika villkor uppfylls. Dessa larm brukar visualiseras på operatörens HMI panel, men det finns ingen möjlighet att lägga till detta i Simumatik3D, så verifiering av detta blir komplicerat då en extern HMI panel behöver installeras, virtuell eller fysisk. Det hade varit ett bra komplement om fanns möjlighet att föra en display likt detta i Simumatik3D.

Eftersom ingen undersökning av tillgängliga emuleringsmjukvaror gjordes har ingen jämförelse gjorts om någon annan mjukvara skulle vara bättre lämpad för arbetet och standardisering av PLC programmering. Simumatik3D har varit en bra emuleringsmjukvara för arbetet eftersom det varit enkelt att använda och förstå mjukvaran. Funktionerna som standardiserats kunde funktionstestas på ett bra sätt med hjälp av Simumatik3D

7. Slutsatser

7.1 Framtida arbete

I standarden som utvecklats i detta arbete har ett antal funktioner standardiserats och även denna standard bör kontinuerligt uppdateras och förbättras för att uppnå en så optimal användning som möjligt. För att uppnå högre flexibilitet och effektivitet kan ett bredare urval av funktioner standardiseras. Även mer kompletta funktionsblock med övervakningslarm för att få larmhanteringen att fungera skulle ge ökad effektivitet. Det är dock viktigt att verifiera funktionerna igen när någon ändring görs eller om nya funktioner läggs till för att kunna förlita sig på att förväntat resultat uppnås genom standarden.

En del av syftet med att utveckla en standard för PLC programmering är att underlätta för programmerare att uppnå en fullgod programkod och minska variationen, minska ledtiden och öka kvaliteten på programkoden. Men för att uppnå detta är det viktigt att personalen tar till sig och använder det nya standardiserade arbetssättet, är med och kompletterar standarden med nya funktioner och ständigt förbättrar den.

I detta arbete används emuleringsmodellen till att verifiera funktionerna och i framtiden kunna vara till hjälp vid upplärning av nya automationsingenjörer genom att visualisera sin programmering. Men om högre kunskap inom emulering och simulering uppnås kan en högre grad av arbete med virtuell driftsättning utföras. Eftersom virtuell driftsättning blir allt mer vanligt och utvecklingen av mjukvaror ökar kan det vara lönsamt att investera i kunskapshöjande processer såsom utbildningar inom detta område för att kunna bli konkurrenskraftig. Det finns många emuleringsmjukvaror tillgängliga på marknaden, om en högre grad användning av virtuell driftsättning vill uppnås kan en noggrann undersökning om emuleringsmjukvarorna vara lämpligt.

Simumatik3D kommer inom snar framtid att släppa en relativt omfattande uppdatering på sin mjukvara. Detta medför större urval av komponenter, mer detaljerade komponenter och förbättrad koppling inuti Simumatik3D. Det kommer även innebära att användaren kommer kunna få ut mer användbara data ur modellen. Eftersom emuleringsmodellen i standarden är konstruerad i Simumatik3D kan det vara värt för Projektengagemang att uppdatera till den nya mjukvaran då möjlighet finns att uppnå högre utnyttjandegrad och effektivitet ur emuleringsmodellen.

Bäst möjliga resultat nås om Projektengagemang fortsätter att arbeta med standarden och fortsätter utveckla den. Det kommer aldrig att finnas en slutgiltig standard eftersom en standard kontinuerligt behöver uppdateras och kompletteras för att nå full utnyttjandegrad. Förutom att Projektengagemang ska arbeta enligt den föreslagna standarden finns det även stora möjligheter att göra ett uppföljande examensarbete för andra studenter. Studenterna skulle kunna undersöka om det finns andra emuleringsmjukvaror som är bättre lämpade för att verifiera funktioner och komplettera standarden för att sedan implementera detta. Om fortsatt utveckling av standarden sker av både projektengagemang och studenter ses det som en positiv riktning att följa.

7.2 Återkoppling hållbar utveckling

Om Projektengagemang fortsätter att arbeta med den föreslagna standarden har det en positiv effekt både lokalt och nationellt. En större ekonomisk hållbarhet är möjlig att uppnå på en lokal nivå, detta genom att en ökad effektivitet kan uppnås. Genom att följa regeringskansliets prioriteringar kring standardisering samt att vid delaktighet och inflytande i arbetet kring standardisering kan

samhällets krav på säkerhet, effektivitet och hållbar utveckling uppfyllas. Detta har en positiv inverkan av hållbar utveckling på en nationell nivå.

7.3 Utvärdering av syfte och mål

I detta kapitel görs en utvärdering utifrån resultatet jämfört med de specificerade mål och syfte som sattes i början av arbetet. Anledningen till detta är för att se om projektet har förändrats längs med arbetets gång eller om projektet uppnått de mål som förväntades.

7.3.1 Mål

Huvudmålet med arbetet var att föreslå en lokal standard för PLC programmering att rätta sig efter vid programuppbyggnad som enkelt kan kompletteras i efterhand och som är baserad på Projektengagemangs tidigare projekt. För att förtydliga detta huvudmål kan det delas upp i 3 delar.

- Föreslå en lokal standard för PLC programmering att rätta sig efter vid programuppbyggnad
- Föreslå en lokal standard för PLC programmering som enkelt kan kompletteras i efterhand
- Föreslå en lokal standard för PLC programmering som är baserad på Projektengagemangs tidigare projekt.

Under arbetets gång har standardens utformning förändrats mot den förväntade utformningen och på grund av detta är en av dessa delar inte lika relevant. I målet som sattes innan arbetet påbörjades skulle standarden vara ett sätt att kontrollera sitt program emot och rätta sig efter vid programuppbyggnad. När standarden utvecklades upptäcktes ett större värde i att standarden skulle användas på ett annat sätt. Projektfilen i TIA portal visualiserar hur de standardiserade funktionerna kan användas men det är inte längre tänkt att nya automatiseringsprojekt ska kontrollera sig emot den PLC kod som skrivits. Istället ska alla nya projekt använda de funktioner som verifierats och standardiserats samt använda projektfilen för att se hur de standardiserade funktionerna kan styras. Även om den aktuella planen för hur standarden ska användas är bättre än den som planerades kommer den första delen av huvudmålet inte att uppnås.

För att bekräfta att de andra två delarna av huvudmålet uppfylldes gjordes två valideringar av standarden, en av artefaktens utvecklare och en av Projektengagemang som användare. Resultatet av arbetet och valideringen sammanställt och utvärderat mot arbetets mål presenterats enligt följande:

- Standarden är lätt att använda och hämta funktioner ifrån.
- Användbara funktioner har standardiserats.
- Det är lätt att komplettera och lägga till nya funktioner i efterhand.
- Funktionerna är hämtade från Projektengagemangs tidigare projekt och bibliotek.

I övrigt är standarden enligt tidigare utvärdering lätt att komplettera och funktionerna är hämtade från ett bibliotek med funktioner som använts i flera tidigare projekt och är vanligt använda.

7.3.2 Syfte

Syftet med arbetet kan delas upp i fyra delar:

- Minska variation.
- Öka kvaliteten på programkoden.
- Underlätta vid upplärning.
- Effektivisera Projektengagemangs PLC programmering.

För att säkerhetsställa att standarden som utvecklats uppfyller de syfte som specificerats är den utvärderad och validerad. Resultatet av arbetet och den sammanställda valideringen utvärderat mot arbetets mål presenteras enligt följande:

- Minskad variation i programkoden kan uppnås då programmerare använder sig av en gemensam grund av funktioner att bygga sina program utifrån.
- Kvaliteten av programkoden kan generellt öka, då funktionerna som är standardiserade ska vara baserade på det bästa kända arbetssättet, om en bättre programmerad funktion uppkommer ska den ersätta den tidigare versionen.
- Standarden med tillhörande emuleringsmodell kan underlätta en upplärning genom att kolla hur funktionerna är programmerade i projektfilen och se hur det styr emuleringsmodellen. Det finns även möjlighet att ta bort all styrning i projektfilen låta den nyanställda programmera emuleringsmodellen med hjälp av standardfunktionerna i biblioteket.
- En ökad effektivitet kan uppnås genom att använda standarden, framförallt om funktionerna kompletteras med ytterligare data som övervakningslarm så funktionerna blir mer kompletta. Med ett bibliotek med standardiserade funktioner som är lättillgängligt kan en effektivare programmering uppnås.

Referenser

ABB (2019) *Styrsystem i det mindre formatet*.

Banks, J. (2010) *Discrete-event system simulation*. Upper Saddle River, N.J.: Pearson Education.

Bibliotheek, K. (2019) *What is emulation?* Available at: https://www.kb.nl/en/organisation/research-expertise/research-on-digitisation-and-digital-preservation/emulation/what-is-emulation (Accessed: 21 February 2019).

Bicheno, J. *et al.* (2013) *Ny verktygslåda för Lean : Filosofi, transformation, metoder och verktyg*. Göteborg: Revere.

Bolton, W. (2015) *Programmable Logic Controllers*. Amsterdam [Netherlands]: Newnes.

Bondeson, P. E. R. and Liss, S. (2018) 'Roadmap of Virtual Commissioning Inertia: An Investigation of Technical and Non-Technical Fields of Action'.

Brundtland, G. H. and Hägerhäll, B. (1988) *Vår gemensamma framtid : [rapport från] Världskommissionen för miljö och utveckling under ordförandeskap av Gro Harlem Brundtland*. Stockholm: Prisma :

Brunsson, N. and Jacobsson, B. (2000) *A world of standards*. Oxford: Oxford University Press.

Dahl, M. *et al.* (2016) 'Integrated Virtual Preparation and Commissioning: supporting formal methods during automation systems development', *IFAC-PapersOnLine*, 49(12), pp. 1939–1944. doi: 10.1016/j.ifacol.2016.07.914.

Dennis, P. (2016) *Lean production simplified : a plain-language guide to the world's most powerful production system*. New York: Productivity Press.

Eriksson, U. (2008) *Test och kvalitetssäkring av IT-system*. 2nd edn. Holmbergs i Malmö AB.

Folke, R. (Siemens) (2019) *Varför TIA portal?*

Gröndahl, F. and Svanström, M. (2011) *Hållbar utveckling : en introduktion för ingenjörer och andra problemlösare*. Stockholm: Liber.

Gulliksson, H. and Holmgren, U. (2018) *Hållbar utveckling : teknik, samhälle och livskvalitet*. 3rd edn. Lund: Studentlitteratur.

Hawkins, R., Mansell, R. E. and Skea, J. (1995) *Standards, innovation and competitiveness : the politics and economics of standards in natural and technical environments*. Aldershot: Elgar.

ISO (2019a) *About ISO*. Available at: https://www.iso.org/about-us.html (Accessed: 15 January 2019).

ISO (2019b) *Popular standards*. Available at: https://www.iso.org/popular-standards.html (Accessed: 21 February 2019).

Jonker, G. and Harmsen, J. (2012) *Engineering for sustainability : a practical guide for sustainable design*. Amsterdam: Elsevier.

Karvosenoja, K. and Gyllenspång, H. (2001) *Verifiering av PLC logik mot en simuleringsmodell*. Skövde.

Lee, C. G. and Park, S. C. (2015) 'Survey on the virtual commissioning of manufacturing systems', *Journal of Computational Design and Engineering*. Elsevier Masson SAS, 1(3), pp. 213–222. doi: 10.7315/jcde.2014.021.

McGregor, I. (2003) 'The relationship between simulation and emulation', pp. 1683–1688. doi: 10.1109/wsc.2002.1166451.

Nationalencyklopedin (2019) *metodologi*. Available at: https://www-ne-se.libraryproxy.his.se/uppslagsverk/encyklopedi/lång/metodologi.

Oates, B. J. (2006) *Researching information systems and computing*. London: SAGE.

Oppelt, M. and Urbas, L. (2014) 'Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering', *IECON Proceedings (Industrial Electronics Conference)*. IEEE, pp. 2564–2570. doi: 10.1109/IECON.2014.7048867.

PLCopen (2013a) *Benefits for Users of IEC61131-3*. Available at: http://www.plcopen.org/pages/benefits/benefits_for_users_iec61131/ (Accessed: 31 January 2019).

PLCopen (2013b) *TC1-Standards*. Available at: http://www.plcopen.org/pages/tc1_standards/ (Accessed: 31 January 2019).

Ramanathan, R. (2014) 'The IEC 61131-3 programming languages features for industrial control systems', in *2014 World Automation Congress (WAC)*, pp. 598–603. doi: 10.1109/WAC.2014.6936062.

Regeringskansliet (2015) 'Regeringens skrivelse ägande', pp. 1–115.

Siemens (2019) *Virtual commissioning of automation solutions*. Available at: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/virtual-commissioning.html (Accessed: 25 February 2019).

SIS (2019a) *Fakta och organisationen*. Available at: https://www.sis.se/om-sis/faktaochorganisation/ (Accessed: 15 January 2019).

SIS (2019b) *Vad är en standard?* Available at: https://www.sis.se/standarder/vad-ar-en-standard/ (Accessed: 15 January 2019).

United Nations (2018) *Goal 17: Revitalize the global partnership for sustainable development*.