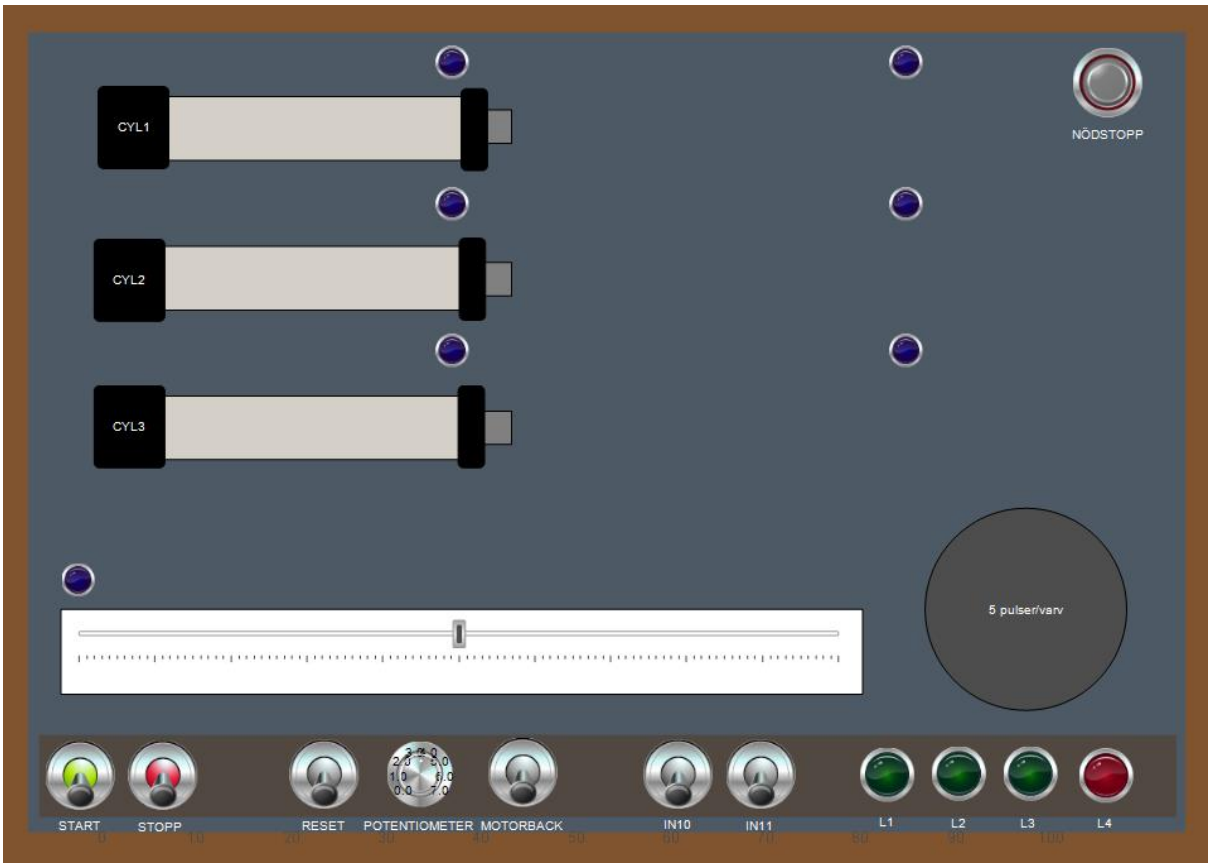
CHALMERS



Simulatorutveckling i Codesys

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

Alexander Ryytty Nilsson

INSTITUTIONEN FÖR Elektroteknik

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2021
www.chalmers.se

Förord

Denna rapport handlar om det examensarbete på Institutionen för Elektroteknik vid Chalmers tekniska högskola jag har ägnat vintern 2020/2021 åt att utföra. Examensarbetet utfördes på Chalmers Lindholmen med Veronica Olesen som handledare och examinator.

Jag skulle vilja tacka Veronica för att hon givit mig möjligheten att utföra detta projekt och för det stöd du har givit mig under hela vintern.

Sammanfattning

Rapporten beskriver arbetet för att utveckla simulatorer som skall underlätta studier för Chalmers studenter på Lindholmen. Under pågående pandemi har tillgängligheten till laborationssalar minskat gentemot tidigare år. För att kursinnehållet skall förbli det samma och inte behöva anpassas efter omständigheterna simuleras laborationsutrustning i två kurser. Simuleringarna skapas i Codesys som även är den mjukvara som används i laborationssalen för att styra utrustningen. Resultatet av simulatorerna är att studenter kan skriva kod på distans för att sedan med enkla medel överföra koden till laborationssalen. Med denna lösning har pandemin minimal påverkan på utbildningen samtidigt som det underlättar för framtida studenter då möjligheten att testa kod inte längre är begränsad.

Abstract

This report describes the work to develop simulations that will facilitate studies for Chalmers students at Lindholmen. During the ongoing pandemic the availability of laboratory rooms has decreased compared to previous years. In order for the course content to remain the same and not adapt to the circumstances the laboratory equipment of two courses is simulated. The simulations are created in Codesys which is also the software used to control the equipment at Chalmers. The result of the simulators is that students can write code remotely and transfer the code to the laboratory room with simple means. With this solution the pandemic has a minimal impact on the education while making it easier for future students to test their code.

Innehåll

1 Inledning

För att underlätta de praktiska studierna på Chalmers är simulatorer ett bra alternativ. Eftersom tillgången till laborationssalar och dess utrustning är begränsad kan simulatorer underlätta arbetet mellan lektioner och även ersätta lektioner. Detta projekt genomfördes därmed för att ge ett extra stöd till studenter framför allt nu när pandemin begränsar tillgängligheten ytterligare.

1.1 Bakgrund

Under pågående pandemi har beslut tagits på Chalmers om att hålla undervisning på distans. Det blir problematiskt vid laborationer. Lösningen har varit att minska antalet personer i laborationssalen vilket leder till att studenterna får färre tillfällen i skolan. Detta leder även till att studenterna inte längre har tillgång till labbutrustningen som skolan tillhandahåller på samma sätt som tidigare studenter samt går miste om praktisk undervisning.

1.2 Syfte

Med hjälp utav Codesys simuleringsverktyg kan laborationsutrustningen simuleras vilket gör att studenterna kan utföra sina laborationer på distans med likvärdiga resultat. Syftet blir därmed att skapa simuleringar som är så lika den verkliga labbutrustningen som möjligt. Labbutrustningen som simuleras är från kursen Styrteknik där en korvskivare och ett tilluftssystem symboliseras utav två Mecman-plattor. I kursen Industriella styr och övervakningssystem används en testplatta för en produktionskedja som även den simuleras fast i två delar.

1.3 Avgränsningar

Följande avgränsningar är vad som gäller för projektet:

Hisstyrning ingår i kursen men är redan simulerad i samma miljö så kommer ej utvecklas.

Delar av simulatorerna förenklas då de endast har en visuell påverkan på slutresultatet.

Det finns ett moment i motorn som gör att den fortsätter gå några millimeter efter den stoppas, detta har inte simulerats.

1.4 Precisering av mål

Projektets delmål:

- Samla information om laborationsutrustningen.
- Visualisera laborationsutrustningen.
- Skapa rörelser i visualiseringen som speglar samlad information.
- Göra det möjligt för stundeter att överföra skriven kod från simulatorn till laborationssal utan komplikationer.

2 Teoretisk bakgrund

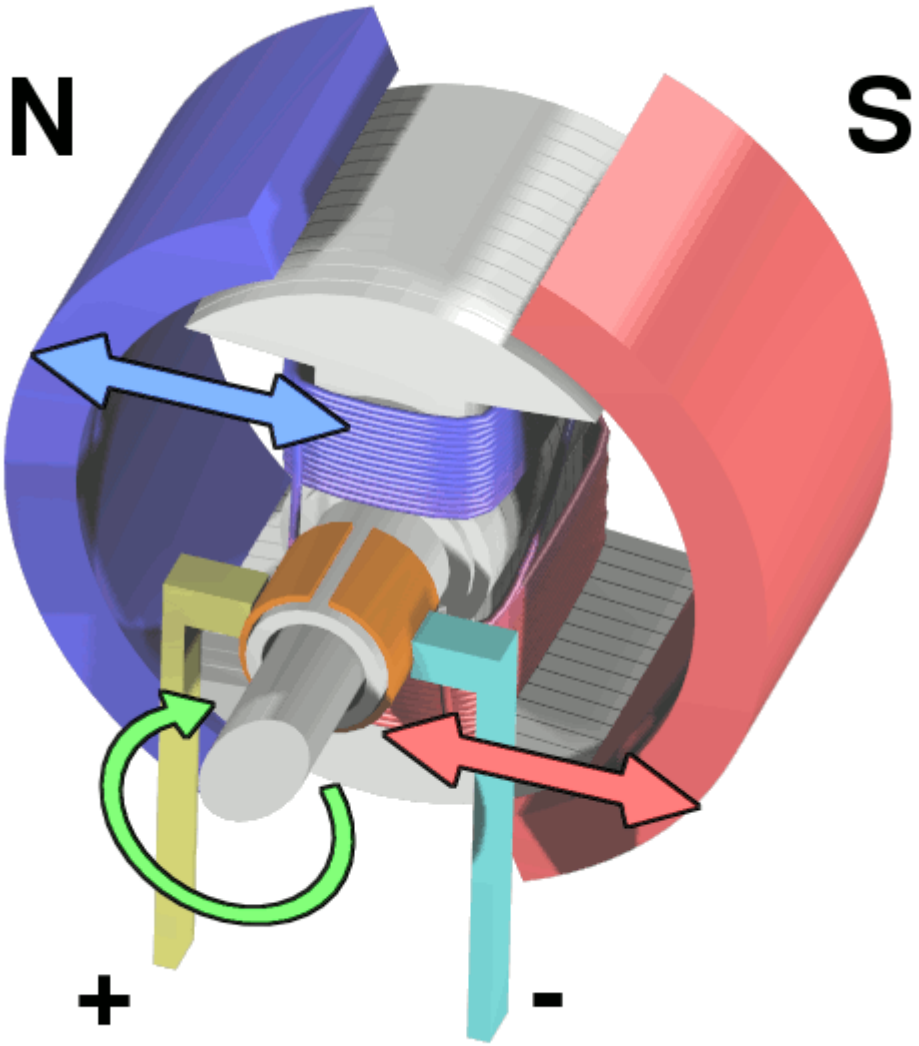
I följande kapitel beskrivs bakgrundskunskaper för projektet.

2.1 Mecman-platta

En Mecman-platta är en testplatta där olika elektriska komponenter styrs. Plattan är tillverkad av företaget Mecman som utvecklar pneumatiska komponenter. Mecman-plattans främsta syfte är att simulera verkliga processer med representativa verktyg. På Mecman-plattan sitter en likströmsmotor, cylindrar, knappar, lampor och sensorer som beskrivs ytterligare i detta kapitel.
Mecman tillverkar även en testplatta för en produktionskedja. Den består av cylindrar, sensorer, hissar och en sugkopp som tillsammans förflyttar puckar för att simulera produktionskedjan.

2.2 Likströmsmotor

Likströmsmotorn är en motor vars uppgift är att omvandla elektrisk energi till mekanisk energi. Likströmsmotorn består av två delar, en fast del som heter statorn och en rörlig del som heter rotorn. I den fasta delen finns två magneter som skapar ett magnetfält över rotorns position. Rotorn är uppbyggd av en rektangulär lindning som strömsätts från en extern källa. När ström går i lindningen gör magnetfältet att den roterar ett halvt varv. För att rotationen skall fortsätta och rotera ett helt varv behöver strömmen ändra riktning, det görs med hjälp av en strömvändare, vilket är den orangea delen som roterar med den gröna pilen i figur 1. Ändring av strömriktning i lindningen skapar alltså en rotationsrörelse.



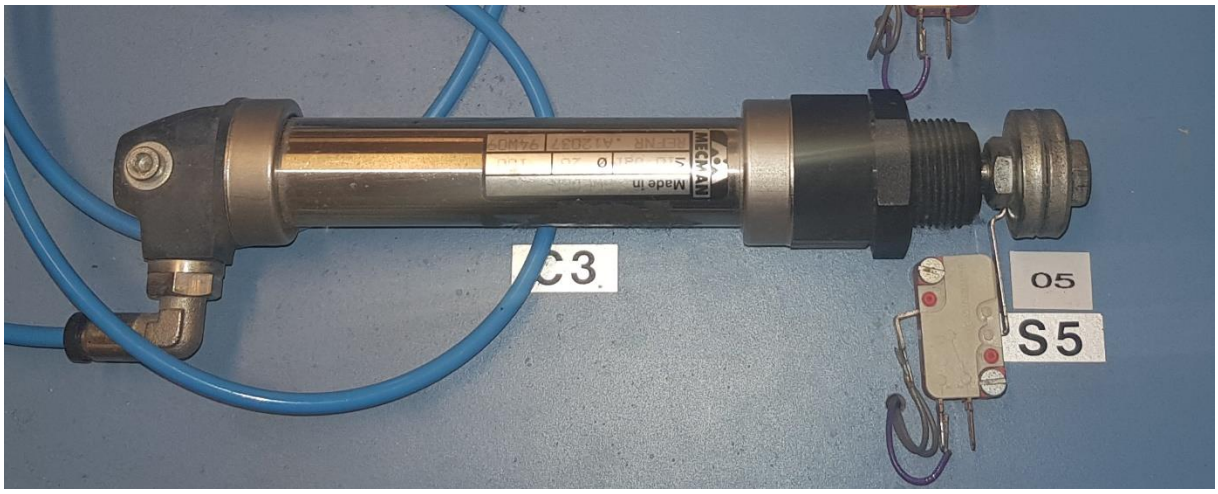
Figur 1. Bild som visar likströmsmotorns rotation [1].

På strömvändaren sitter små borstar. Dessa slits över tid på grund av friktionskrafter då strömriktningen ändras vilket försämrar motorns livslängd. Lösningen är att använda borstlösa likströmsmotorer där magneterna är placerad i den roterande delen och lindningen i den statiska. Konstruktionen skapar ett roterande magnetfält som följer lindningen vilket avlägsnar friktionskraften från systemet [2]. Likströmsmotorns konstruktion gör den till en flexibel motor där hastigheten beror på matningsspänningen från den externa källan. Den kan även tillverkas i olika storlekar för att anpassa den efter dess användningsområde.

2.3 Pneumatisk cylinder

En pneumatisk cylinder är en cylinder som styrs av någon form av gas, oftast luft. Cylindern styrs av ett kolvorienterat system som innehåller en kammare för att tryckluften ska kunna komma in i systemet, en väg

för luften att lämna systemet och en kolv. När cylindern arbetar tvingas komprimerad luft genom en magnetventil i kolvens baksida från kammaren. Luften pressas sedan med stor kraft på kolvytan vilket pressar fram kolven som syns till höger i figur 2. För att kolven skall återvända till sitt ursprungliga lägen används antingen en fjäder som trycker tillbaka kolven eller ett dubbelverkande lufttrycksystem som använder ventiler för att injicera luftströmmar från båda ändarna av kolven [3].



Figur 2. Bild på cylinder av märket Mecman

Cylindern styrs oftast av ett elektriskt system i form av en PLC och cylindern har en linjär rörelse. Tryckluft är ett säkert medium och är enkelt att komprimera, det leder till att pneumatiska cylindrar funkar i de flesta miljöer utan att vara någon större risk för omgivningen. Däremot är komprimeringen av luft en energikrävande process med låg verkningsgrad vilket är både dyrt och miljöbelastande [4].

2.4 Nödstopp

Ett nödstopp är en röd knapp vars uppgift är att stoppa strömtillförsel till processen vid en nödsituation, detta görs manuellt av att knappen aktiveras. Vid aktivering skall den befinna sig i sitt läge tills den återställs via en rotationsrörelse på knappen. Återställningen skall ej återställa processen utan endast göra en återstart möjlig. Nödstoppet kopplas så strömförsörjningen till systemet går via nödstoppet ut till processen, därmed bryts strömmen till hela systemet vid aktivering [5].

2.5 Programmable logic controller

I en automatiserad industriell miljö är PLC:n (programmable logic controller) hjärnan i operationen. Den tar in information från processen och skickar ut styrsignaler som är förprogrammerade i valda sekvenser. Styrsignalerna jobbar med 24V spänning och har digitala in- och utgångar samt analoga signaler [6]. En digital signal är en signal som bara har ett högt och ett lågt läge. Signalen kan alltså bara vara av eller på. De digitala utsignalerna funkar därmed genom att PLC:n antigen skickar en hög eller låg signal till en utgående komponent för att styra den. Ingångarna till PLC:n tar in signaler som antingen är låga eller höga, ett exempel på en sådan signal är signaler ifrån givare. De analoga signalerna kan ha samtliga värden i sitt spänningsintervall. Är spänningen som i detta fallet 24V kan den analoga signalen ha ett värde mellan 0 och

24. PLC-system finns i flera olika konfigurationer och i detta projekt används en CREVIS NA-9372 och utvecklingsmiljön Codesys.

PLC:n ersatte reläpaneler i början på 70-talet och har sedan dess varit en stor del av industrin. Anledningen till att PLC:n används så frekvent är att den kan anpassas efter dess miljö. Extrema miljöförhållanden som höga temperaturer, hög luftfuktighet, elektromagnetiska fält och mekaniska vibrationer är inget problem för PLC:n. Dess struktur som tillåter snabba installationer och byten av moduler gör den även till ett flexibelt verktyg [7].

2.6 Codesys

Codesys är ett programmeringsverktyg för att styra PLC:n. Programmering av styrsignaler kan hanteras med fem olika programmeringsspråk. Instruction list är ett lågnivåspråk likt assembler fast anpassat för PLC programmering. Structured text är ett högnivåspråk baserat på programmeringsspråket Pascal som även är likt C. Ladder diagram är ett språk som representerar ett grafiskt diagram baserat på relälogik. Function block diagram är ett grafiskt språk som beskriver funktioner mellan ingångs variabler och utgångs variabler. Sequential function diagram är också ett grafiskt språk som används för att stega programmerade processer [8]. Dessa programmeringsspråk kan kopplas direkt till simuleringar i samma mjukvara. Kopplingen mellan ett verkligt system och en simulering blir därmed väldigt enkel då det endast behövs ställas in i Codesys om en simulering körs eller inte.

Codesys samlar flera funktioner i en och samma mjukvara samtidigt som de följer standarden IEC 61131-3 [9]. Standardens uppgift är att underlätta kommunikationen mellan olika system samtidigt som det underlättar för användare med tidigare erfarenhet av standarden i andra mjukvaror.

2.7 Visualization Manager

I Codesys kan man köra simulationer för att testa sin kod. För att få en bättre bild av vad koden är kapabel till finns ett visualiseringsverktyg med färdiga verktyg som lampor, switchar och olika geometriska former. De kan anpassas efter storlek, form och färg. Det går även att koppla variabler till objektens egenskaper för att förändringar i simuleringen ska ske ifall det programmerade villkoret för variabeln uppfylls. Med hjälp av det kan man simulera en större industriell process, testa eventuella svagheter i koden och få en visuell representation av detta.
Image pool är ett verktyg i Codesys där man kan importera bildfiler in i programmet för att sedan använda i Visualization Manager.

2.8 HMI

HMI står för Human-Machince Interface. Det är en koppling mellan olika delar av processen till knappar eller andra visuella föremål som skall vara styrbart av en människa. Via en PLC styrs de elektriska komponenterna i processen.

Ett HMI har inte några fasta riktlinjer utan kan anpassas efter användningsområdet. Däremot finns det en del designaspekter som framhävs för att få ett så användarvänligt gränssnitt som möjligt [10]. En enkel design underlättar då nya arbetare introduceras till gränssnittet. Överkompliceradd design eller överrepresenterade processdelar är en distraktion som saknar funktion och bör därmed undvikas. Användning av vanliga igenkännbara instrument underlättar också då det blir mindre nytt att lära samt igenkänningsfaktorn är högre.

Användningen av olika färger skall även minimeras. Mängden färg är inget problem men att använda flera olika färger skapar otydlighet genom flera fokuseringspunkter. Detta gäller framförallt färger med hög kontrast. Färgen röd skall även undvikas helt då den förknippas med nödsituationer.

Bakgrundfärgens uppgift är även att skapa kontrast till instrumenten i gränssnittet. Till en början användes det mycket svart som bakgrundsfärg, Det gjorde så att de övriga färgerna reflekterade ljus rakt i ansiktet på arbetarna, för att minska det sänktes ljusstyrkan i lokalen men det bidrog till ökad ansträngning för ögonen samt försämrad sömn för de anställda. Lösningen var att använda grå som bakgrundsfärg då det funkar lika bra som kontrast till andra färgen och minskar det reflekterande ljuset från skärmen.

Designa sin HMI för användning på touchscreen är även fördelaktigt för en mer erfaren arbetar då de kan använda båda händerna för att snabbare styra processen.

2.9 Elektriska Sensorer

Sensorer är till för att definiera föremål genom att till exempel se ifall de är elektriskt ledande eller om de reflekterar ljus utan att ha någon direkt kontakt med föremålets yta.
En induktiv sensor detekterar ifall ett föremål är metalliskt. Det görs genom att sensorn skapar ett starkt magnetfält vid dess ände. När ett metalliskt föremål passerar magnetfältet försvagas det vilket skickar vidare en signal att det befinner sig ett metalliskt föremål vid sensorn [11]. Denna typ av sensor används frekvent till exempel på flygplatser i form av metalldetektorer men också vid trafikljus för att se om fordon väntar vid rödljus.

En kapacitiv sensor är också en beröringsfri givare men denna sensor detekterar kapacitans. Denna sensor ger en signal varje gång ett föremål kan lagra en elektriskladdning bättre än luft. Den ger alltså en signal varje gång ett föremål befinner sig på den sökta positionen.

En optisk sensor är även den beröringsfri. Den fungerar genom att sensorn skickar ut ljus och ser ifall ljuset reflekterar tillbaka. Kommer ljuset tillbaka till sensorn ger den en signal annars inte. Den kan alltså detektera ett föremåls reflektionsförmåga.

3 Metod

För att uppnå målen i projektet testades laborationsutrustningen i Chalmers laborationssal Tanken. Varje variabel testades enskilt och mätningar på avstånd och tid gjordes på samtliga signaler för att beskriva

rörelsemönster. Med hjälp av bilder på utrustningen skapades visuella representationer av utrustningen i Codesys simuleringsmiljö. Variabler kopplades till de visuella delarna i simuleringen och rörelsemönster för dessa programmerades i programmeringsspråket FBD(Funktions block diagram). Olika exporteringslösningar testades för att hitta den smidigaste kodöverföringen från simulator till laborationsutrustningen.

4 Laborationsutrustning

I kursen Styrteknik används två Mecman-plattor och i kursen Industriella styr och övervakningssystem används en testplatta för en produktionskedja. I detta kapitel kommer uppbyggnaden av utrustningen demonstreras samt vad som skiljer dem åt.

4.1 Pneumatisk Cylinder

På de två Mecman-plattorna finns tre cylindrar som alla är kopplade till tryckluft. Tryckluften styrs av PLC:n där de får en 24-voltssignal som släpper in luft i systemet. Är det ingen signal släpps luften ur systemet. Cylindrarna är även utrustad med två kontakter som signalerar cylindrarnas position. De är positionerade så en får kontakt då cylindern är vid sin startposition respektive slutposition. Därmed kan PLC:n arbeta med de olika lägena på cylindrarna.

I Chalmers laborationssal är samtliga Mecman-plattor kopplade på samma tryckluftssystem. Det leder till att samtliga cylindrar arbetar med samma tryck och cylindrarna får en aktiveringstid på 0,6 sekunder samt en deaktiveringstid på 0,3 sekunder.

4.2 Motor

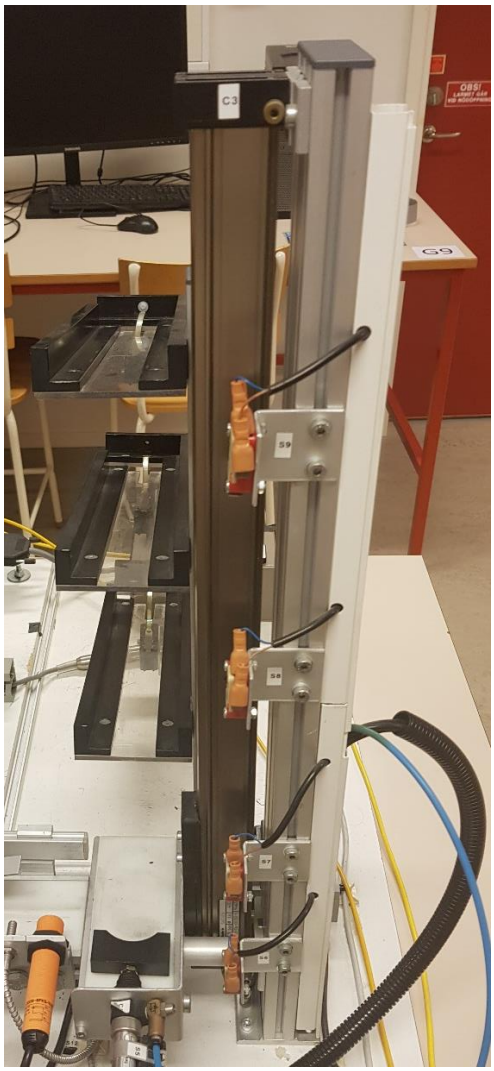
PLC:n kan köra motorn i tre lägen, snabbt framåt, långsamt framåt och snabbt bakåt. Lägena aktiveras via enskilda digitala portar. När motorn är aktiv roterar en 100 mm lång skruv i vald hastighet och riktning. På en av Mecman-plattorna sitter en mutter på skruven som indikerar dess rotation. På den andra plattan är inte distansen muttern roterat intressant så där är ingen mutter installerad. Där skall istället motorns hastighet regleras med hjälp av matningsspänningen.

På skruvens ände sitter en cirkelplatta som roterar med skruven. På Mecman-platta två har cirkelplattan 15 hål längs med cirkelns kant som genererar en pulssignal in till PLC:n vid varje hålpassering, det vill säga 15 pulser per rotation. På den första plattan är cirkelplattan konstruerad på samma sätt men har endast fem hål vilket ger fem pulser per rotation.
PLC:n har ingen kontakt med muttern, det är endast en visuell representation av distansen.

4.3 Hiss

Hissarna körs med två styrsignaler, en för att köra upp hissen och en för att köra ner. Hissarna har även fyra olika våningar som är positionsindikerade med knappar som trycks in av hissens rörliga komponent när deras position möts. Knapparna är insignaler till PLC:n vilket ger information om hissens position. PLC:n vet dock bara om hissen befinner sig på dessa fyra olika positioner. Den har alltså ingen information mellan våningarna.

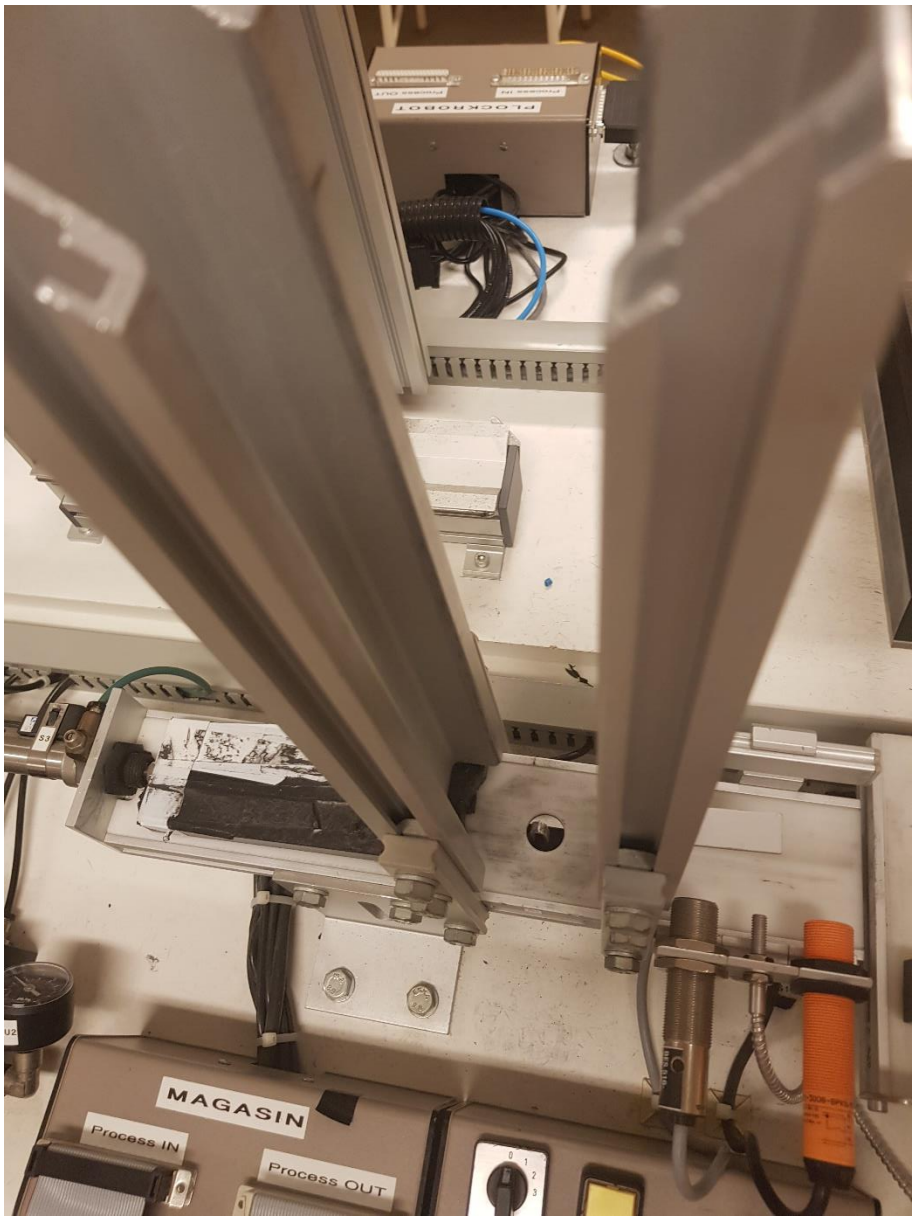
På hissen i figur 3 sitter även en platta placerad på de tre översta våningarna. Plattorna lutar neråt så om något placeras på plattan glider det ner till den lägsta punkten där det också sitter en knapp som indikerar om något befinner sig där. Dessa knappar är också insignaler till PLC:n. Hissens rörliga del består av en cylinder och en plan platta. Den plana plattans uppgift är att förflytta föremål vertikalt medan cylindern förflyttar föremål från den plana plattan till någon av de lutande plattorna.



Figur 3. Bild på hiss med plattor och indikeringsgivare.

4.4 Rotationsrörelse

Den andra hissens rörliga del består av en cylinder där en sugpropp är placerad på dess ände. Sugproppen har två styrsignaler, en för att aktivera dess sugfunktion och en för att avaktivera. Syftet är att sugproppen tillsammans med cylindern ska hämta föremål på den lutande plattan från den andra hissen genom att aktivera både cylinder för att nå fram till föremålet och sugproppen för att föremålet skall fästa. Hissen är även utrustad med en cylinder som förflyttar hela hissen närmare den positionen. När ett objekt är fäst vid sugproppen skall den placera objektet i ett vertikalt magasin. Magasinet är placerat så hissen behöver rotera 90 grader medsols. Rotationen styrs av två styrsignaler, en för att aktivera rotationen och en för att rotera tillbaka. Rotationstiden är en sekund för respektive rotation. Under sugproppen sitter även en knapp som indikerar om hissen nuddar magasinets bottenposition eller ett annat föremål i magasinet. Magasinet har även en knapp i botten som indikerar om det befinner sig något föremål i magasinet, se figur 4 . Dessa knappar är insignaler till PLC:n.



Figur 4. Bild på magasin med knapp i botten

Befinner sig hissarna i sitt bottenläge och hissarnas styrsignal för att köra upp aktiveras tar det 2,5 sekunder innan högsta punkten är nådd. Körtiden för hissarna att åka ner från topposition till botten är också 2,5 sekunder. Hissen kör med konstant hastighet.

4.5 Sensorer

Från magasinet förflyttar en cylinder föremål till den första hissen. Under cylinderaktiveringen passerar föremålet tre sensorer, en induktiv, en kapacitiv och en optisk. Dessa ger insignaler till PLC:n och med hjälp av dessa sensorer vet systemet om föremålet är metalliskt, ljusreflekterande och om det faktiskt är något föremål alls i denna del av processen.

4.6 Puckar

Puckarna är föremålen som rör sig i produktionskedjan. Det finns tre typer av puckar där alla har samma form, endast ytan varierar. Puck ett från vänster i figur 5 är målad vit, puck två har en metallyta och puck tre är målad svart.

Figur 5. Bild på puckarna som rör sig i produktionskedjan

5 Visualisering

I projektet har fyra simuleringar skapats. I följande kapitel beskrivs visualiseringen av komponenterna beskrivna i kapitel 4 i den simulerade miljön.

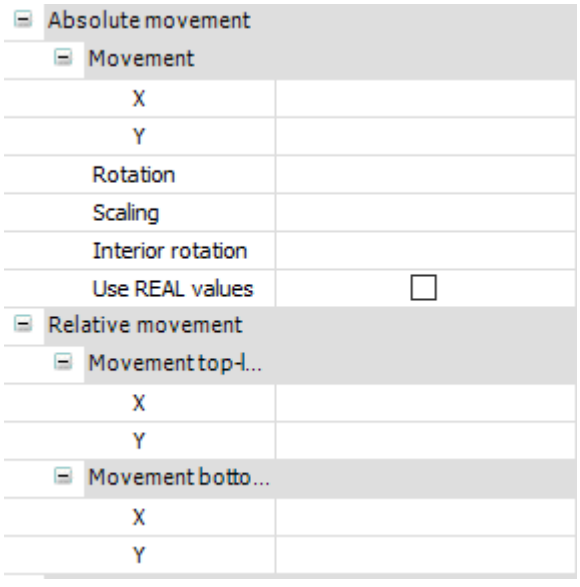
5.1 Objekt

I Codesys visualization finns en ToolBox med färdiga objekt som kan kopplas till olika variabler. Rektanglar, lampor, knappar och potentiometrar kan anpassas och förflyttas fritt i visualiseringen. Knapparna är väldigt enkla i visualiseringen, trycks en knapp in blir variabeln den är kopplad till TRUE och släpps den blir den FALSE. Det går även att anpassa knapparna så variabeln förblir TRUE tills knappen trycks in igen. Det bestäms genom att koppla knappens variabel till antingen en Togglefunktion eller Tapfunktion i knappens properties.

Till lamporna kopplas en variabel där lampans uppgift är att tändas om variabeln är TRUE. Det går alltså inte styra visualiseringen genom lampan då den endast kan läsa koden.

Potentiometern är direkt kopplad till en variabel som kan ha flera värden. Det går alltså inte att koppla en BOOL som endast kan vara TRUE eller FALSE. Under Properties för potentiometern ställs skala samt minsta och högsta värdet in. När programmet körs kan den kopplade variabel styras genom att rotera potentiometern mellan de valda gränsvärdena med den applicerade skalan.

Rektanglarna i visualiseringen kan anpassas efter form, position och färg. Under properties kan de även kopplas variabler som ändrar rektanglarna under tiden programmet körs. Figur 6 visar var rörelsevariablerna kan kopplas beroende på vilken rörelse som skall utföras. Under Absolute movement kopplas variabler för att förflytta, rotera eller skala rektangeln och under Relative movement kopplas variabler för att förlänga eller förkorta rektangeln relativt det valda hörnet på rektangeln.

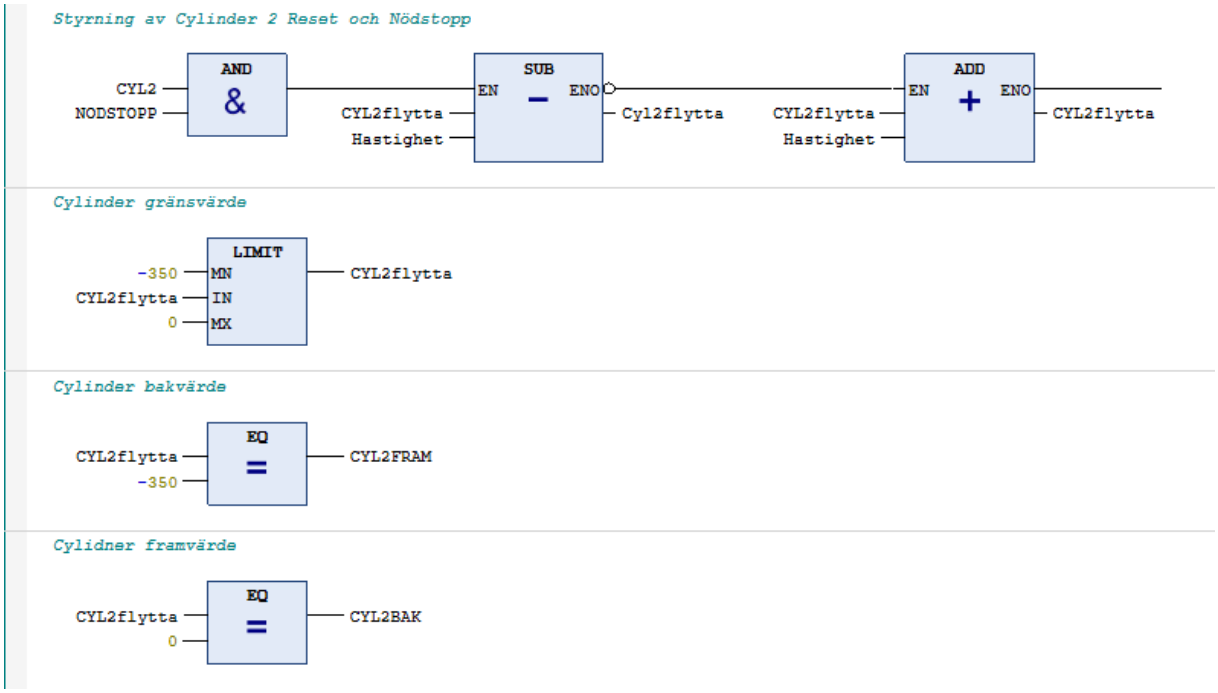


Figur 6. Propertiesfönster för rektanglarnas rörelse.

5.2 Rörelse

Rörelserna i simuleringarna har samma struktur där villkoren för rörelserna skiljer. Varje rörelse har en egen variabel kopplad till objektets förflyttning eller förlängning. I figur 7 är villkoren att CYL2 och NODSTOPP ska vara aktiva för att CYL2flytta ska börja subtrahera med hastigheten. Denna subtraktion inträffar var femte millisekund så länge villkoret är uppfyllt då det är uppdateringsfrekvensen för programmet. För att subtraktionen inte ska pågå längre än önskat har variabeln gränsvärden -350 och 0. Eftersom cylindern ska återgå till sin ursprungliga position när villkoret inte uppfylls har en invertering skapats vid ENO. ENO är aktiv när blocket körs men med denna invertering är signalen aktiv när villkoret ej är uppfyllt. Det resulterar i att förflyttningsvariabeln adderas tills den når gränsvärdet på 0.

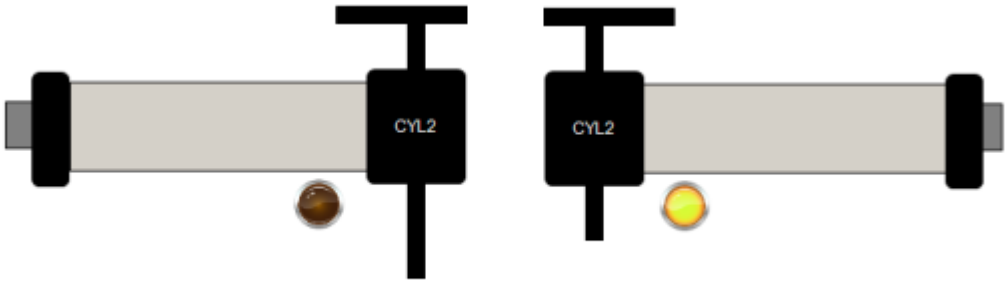
Med subtraktion och addition av förflyttningsvariabler får objekten en konstant hastighet där gränsvärden avslutar beräkningarna med hjälp av LIMIT block se figur 7. Den valda hastigheten är baserad på tiden det tar för beräkningen att avslutas så aktivering och deaktiveringstiden efterliknar laborationsutrustningen. Hastigheten beräknas först genom att dividera det önskade tidsförloppet med uppdateringsfrekvensen. Kvoten är antalet beräkningar som görs innan tiden är nådd. Rörelsedistansen divideras sedan med det beräknade värdet för att få fram hastigheten.



Figur 7. Kod för cylinder 2:s rörelse, gränsvärden, max- och minpunktvariabler.

På simuleringsplattan för produktionskedjan finns en rotationsrörelse där en hiss roterar 90 grader runt sin egen axel. Eftersom simuleringen är i en två dimensionell miljö är rotationen istället en speglingseffekt i simulatorn. Rotationen styrs av två variabler, en för att aktivera rörelsen och en för att rotera tillbaka. I denna rörelse behöver alltså hela cylindern förflytta sig. Eftersom cylindrarna är uppbyggda av tre rektanglar behöver alla tre delar förflyttas. Speglingseffekten gör dock så rektanglarna behöver förflytta sig olika långt beroende på hur långt ifrån speglingens position de är. Därav skapades tre variabler som beräknades med olika

hastigheter för att rörelserna på rektanglarna skall avslutas samtidigt. Eftersom rörelsen endast sker i en riktning kopplades dessa variabler till rektanglarnas förflyttningsvariabel i x-led. Figur 8 demonstrerar rotationens aktiva och inaktiva lägen. Det aktiva läget representeras av den lysande lampan.



Figur 8. Rotationsrörelsens aktiva och inaktiva läge.

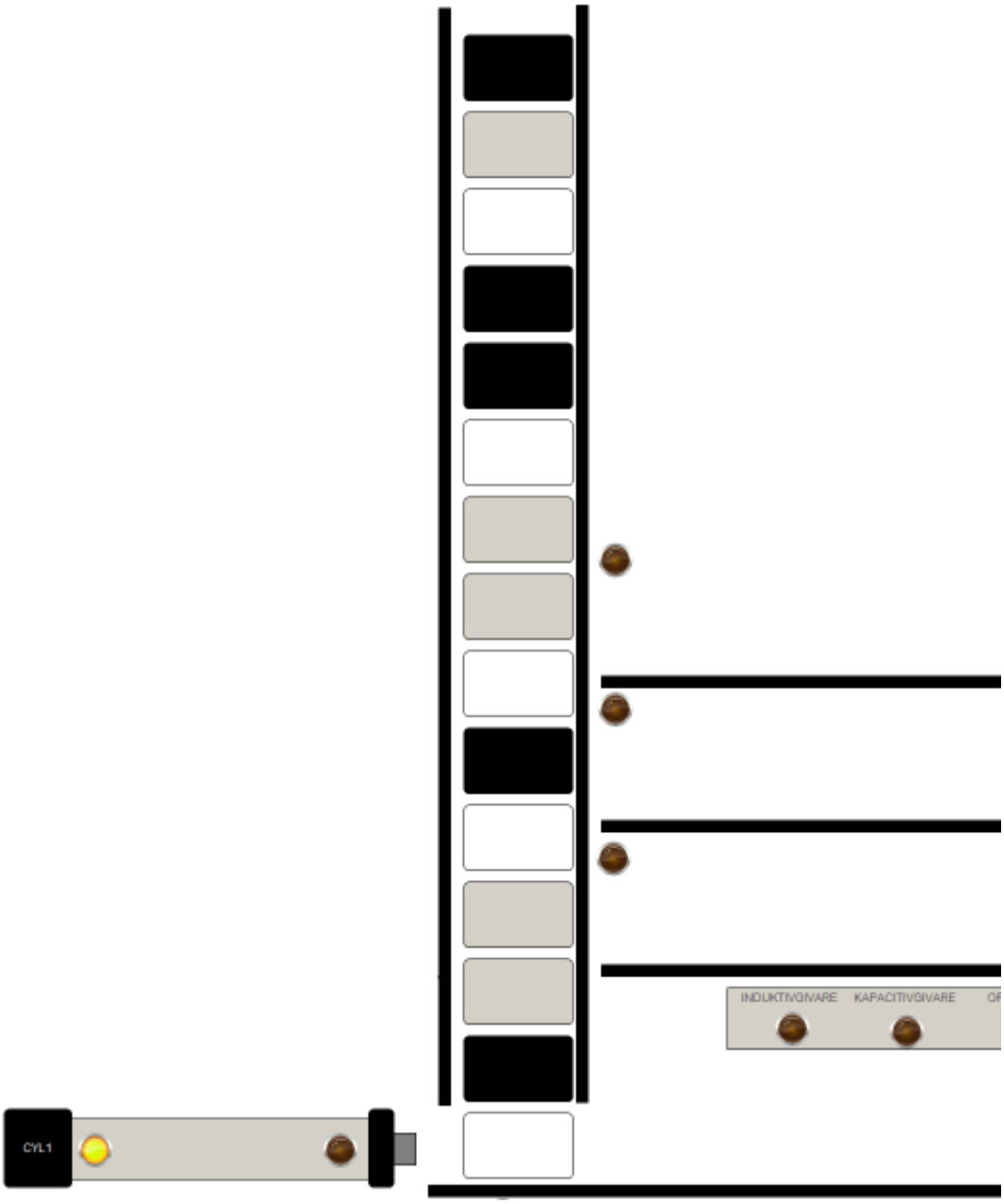
5.3 Ändläges- och våningsgivare

Positionsvariabler i simuleringen funkar genom att se om objektens rörelsevariabler har värden som överensstämmer med positionen som sökes. De flesta rörelser i processen har variabler vid sina max och minpunkter. Eftersom gränsvärden är skapade för dessa aktiveras variabeln genom att kolla om rörelsevariabeln är lika med gränsvärdet. För cylinder 2 är cylindern i sitt främre läge vid -350 vilket aktiverar CYL2FRAM variabeln och i sitt bakre läge vid 0 vilket aktiverar CYL2BAK.

Hissens har även variabler som definierar våningens höjd. Där definieras inte våningarna som en specifik punkt utan av ett intervall. Variabeln blir därmed aktiv om hissens rörelsevariabel befinner sig mellan våningens bottenpunkt och toppunkt.

5.4 Puckar

Puckarna är föremålen som skall röra sig i processkedjesystemet. Puckarna symboliseras i visualiseringen av 15 rektanglar som har samma startposition och storlek där 5 är svarta, 5 är metallfärgade och 5 är vita som visas i figur 9.



Figur 9. Bild på simulerat magasin med puckar i.

Deras uppgift är att följa med cylindrar och hissar i processerna om villkor uppfylls, till exempel att de befinner sig i rätt position eller att sugproppen är aktiv. Varje puck har två egna variabler, en för att förflytta pucken i X-led och en för förflyttning i Y-led. Dessa är REAL variabler alltså decimaltal detta för att hastigheterna inte blir tillräckligt precisa om uträkningar med heltal hade utförts. För att förflyttningen ska ske över en tid och inte initialt är variablerna kopplade till ett enable addition eller subtraktionsblock. Villkoren är kopplade till enable signalen vilket adderar eller subtraherar förflyttningsvariabeln med en förvald konstant hastighet tills villkoren inte längre är TRUE. Villkoren är uppbyggda så det först kollar om pucken är i rörelsens startposition följt av styrsignalen för delprocessen. För att stanna pucken vid sin slutposition kollar även villkoret om signalen inte är vid slutpositionen. Villkoret blir alltså FALSE när rörelsen skall avslutas vilket stannar pucken.

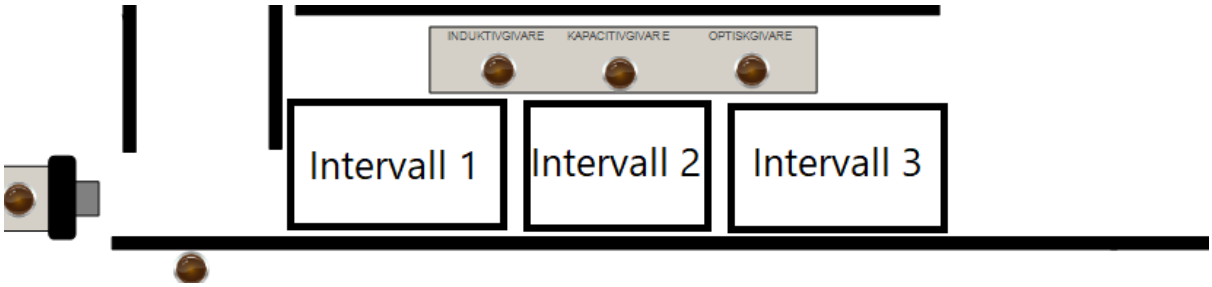
5.5 Kösystem

Kösystem har implementerats på våningarna och i magasinet. Skillnaden mellan kösystemen är att våningarna har en horisontell kö medan magasinet har en vertikal kö. Lösningarna är därmed väldigt lika varandra. Vid första kallelse av objekt förflyttas den första pucken till den första positionen i kön. Nästa objekt placeras på samma position men kollar också om någon av objekten redan ligger på den positionen. Är det redan ett objekt på positionen subtraheras rörelsevariabeln med 120 på våningarna respektive 80 i magasinet. De valda värdena på 120 och 80 är baserade på puckens mått plus 10 för att få mellanrum i kön.

Systemet kollar också kontinuerligt om pucken framför i kön finns kvar för att se om puckarna behöver flytta fram ett steg. Om en puck saknas på positionen framför i kön adderas rörelsevariabeln med avståndet till den positionen för att kön skall röra sig framåt.

5.6 Sensorer

Eftersom föremålen är givna i form av tre olika puckar är även sensorernas aktivering given för varje puck. Lösningen blev därmed att placera sensorerna vid den första cylinder bredvid varandra och kolla ifall puckarna som ger utslag befinner sig i ett intervall vid respektive sensor se figur 10. Vid utslag aktiveras variabeln som tänder lampan.



Figur 10. Visuell intervallbeskrivning av sensorer.

5.7 Nödstopp

I simuleringen körs nödstoppet som en parallell insignal till samtliga styrande funktioner via AND-block. Nödstoppet är initialt TRUE vilket gör att vid aktivering av nödstopp blir signalen FALSE. Det deaktiverar AND-blocket och stoppar hela styrningen även om den andra insignalen är TRUE.

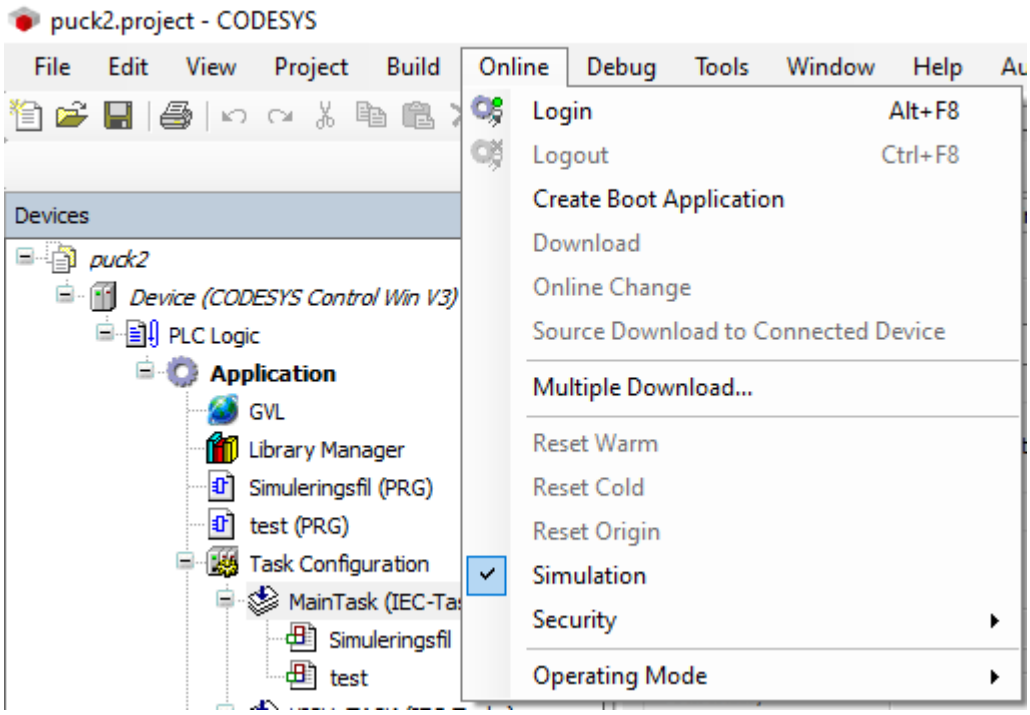
5.8 Sugkopp

Sugkoppen är skapad som en variabel som behöver vara TRUE för att förflyttningen av puckarna skall funka. Vid varje rörelsemoment för puckarna är sugkoppens variabel med i ett AND-block likt nödstoppet. Är inte sugkoppen TRUE kommer inte AND-blocket vara TRUE vilket stoppar puckens rörelse även om styrsignalen körs. Sugkoppen har två styrsignaler, en för att aktivera sugkoppen och en för att stänga av sugkoppen. Om sugkoppen sätts på är den aktiv tills avstängningssignalen aktiveras.

6 Användning av simulator

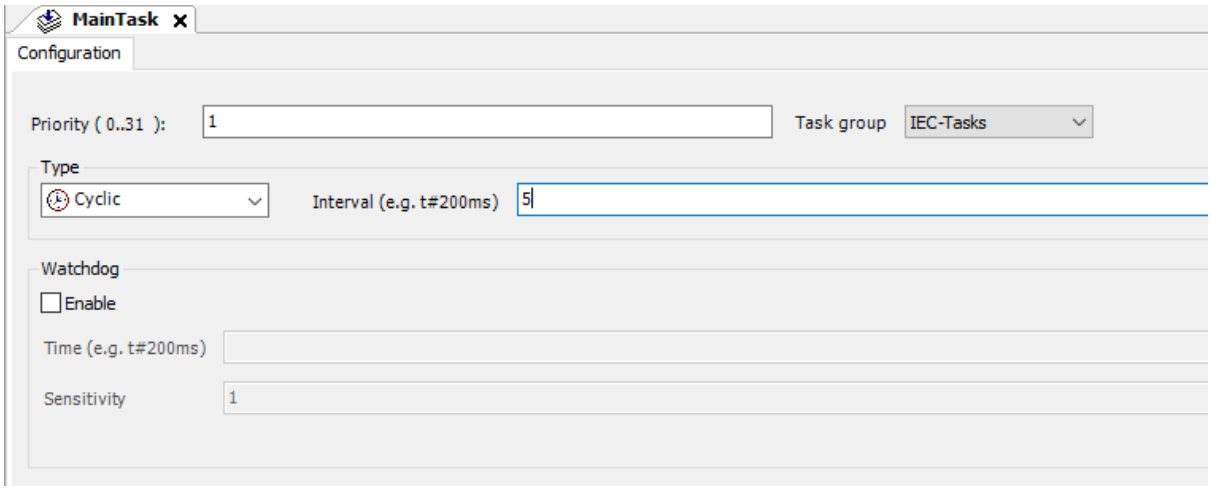
Följande kapitel är en användarhandledning för simulatorerna.

Öppna simulationsfilen i Codesys, ställ sedan in Codesys så mjukvaran vet att en simulering ska köras. Det görs genom att klicka på online högst upp och sedan klicka på simulation som i figur 11.



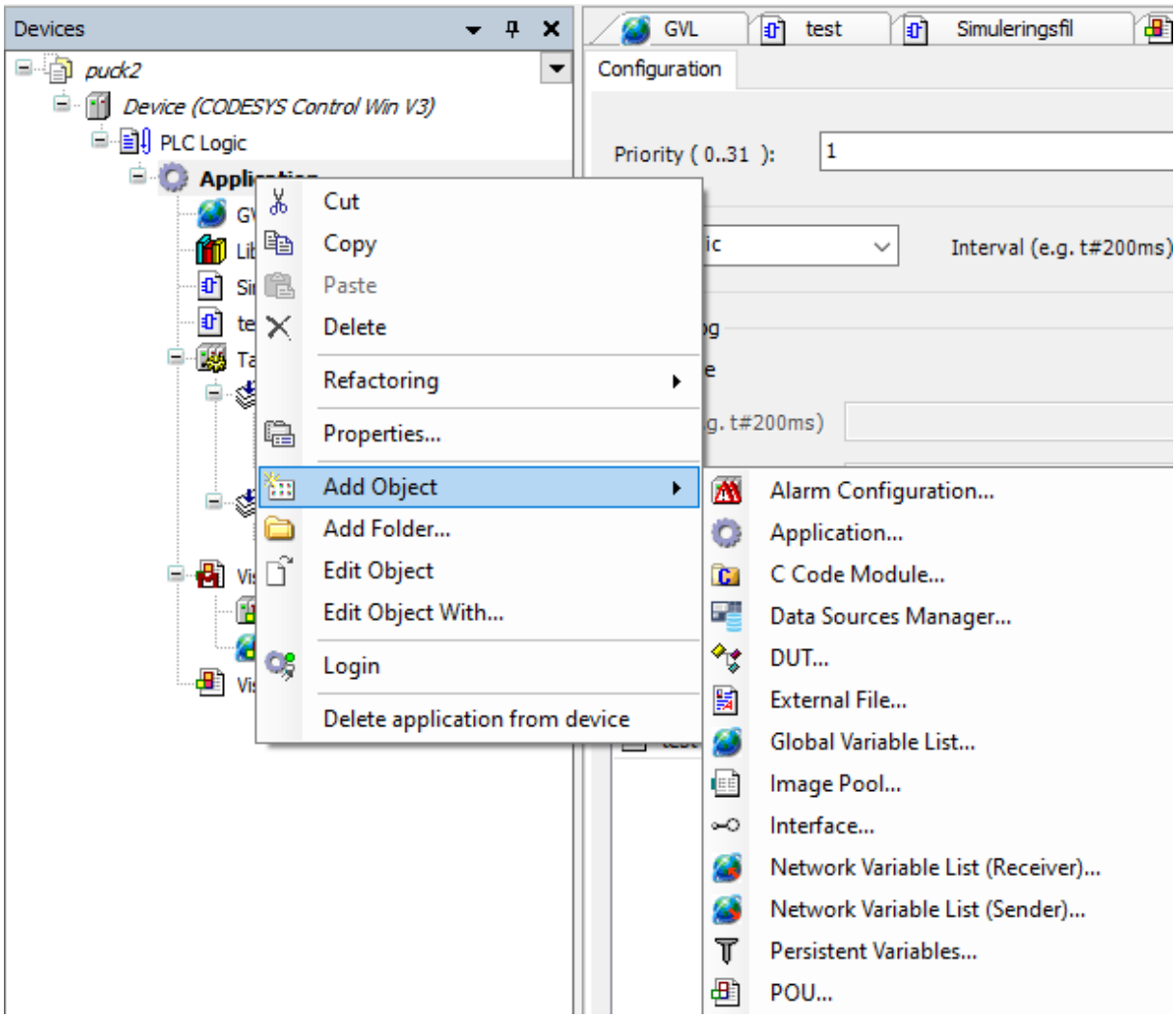
Figur 11. Aktivering av simuleringsmiljön.

Sen behöver programmets uppdateringsfrekvens ändras. Dubbelklicka på MainTask under Task Configuration och skriv in 5 i interval fältet, se figur 12. Det görs för att samtliga delar i simuleringen ska köras i en hastighet som liknar verkligheten.



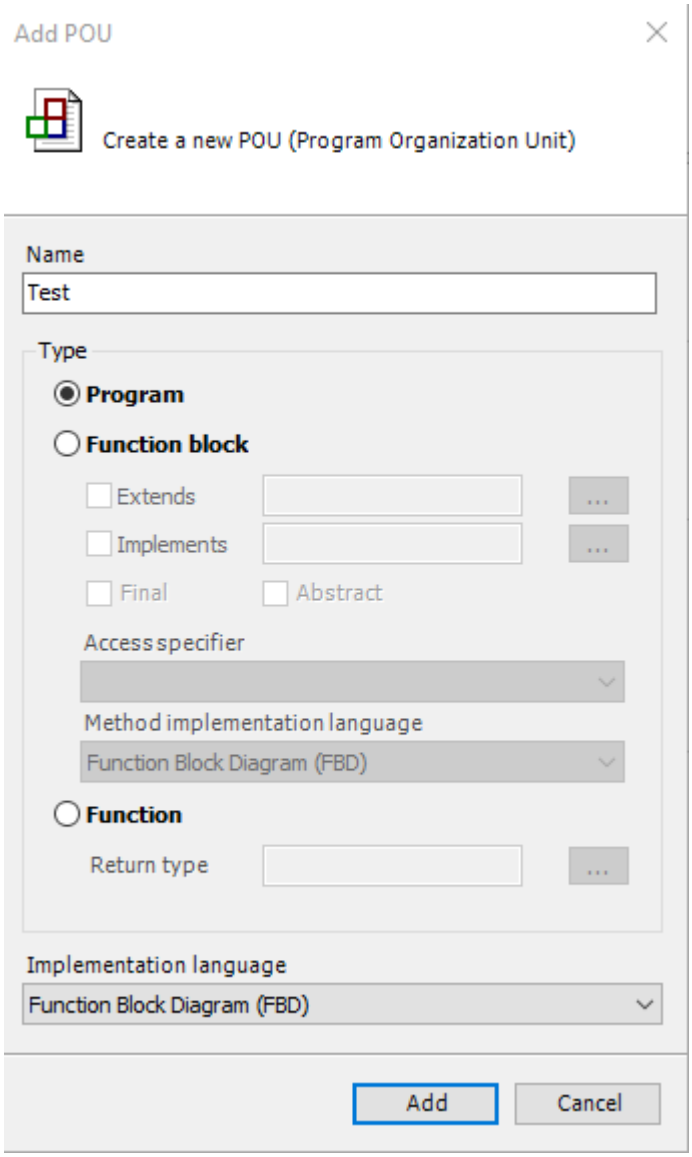
Figur 12. MainTask fönster där uppdateringsfrekvensen väljs.

För att skriva programmerbar kod behöver det först skapas en fil. Högerklicka på Application i Devices fönstret gå till Add Objects och välj POU... som är längst ner till höger i figur 13.



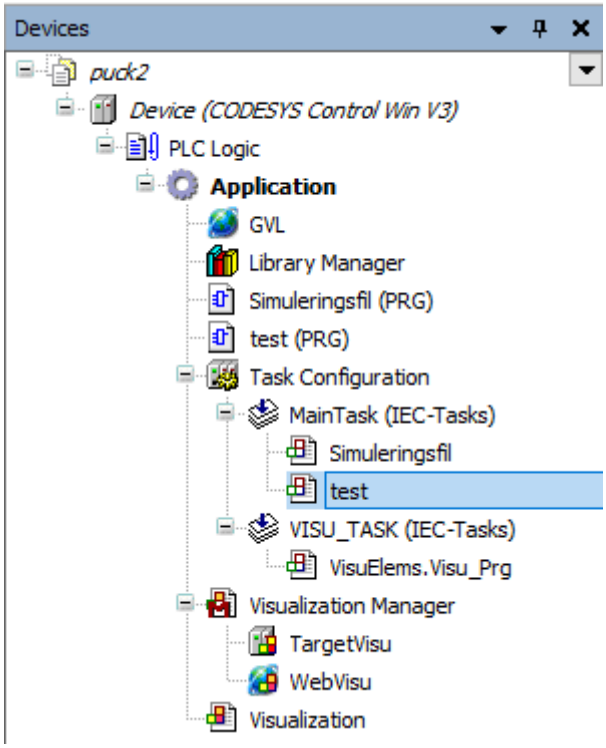
Figur 13. Instruktionsbild för hur man skapar egen POU fil.

Döp filen och välj ditt önskade programmeringsspråk under implementation language, klicka sedan Add för att lägga till filen, se figur 14.



Figur 14. POU fönster där namn och programmeringsspråk väljs.

Nu ligger filen under Application i Devices fönstret men för att koden ska kunna köras behöver du även markera filen och dra den till MainTask så det ser ut som i figur 15.



Figur 15. Förflyttning av programfil till MainTask.

Koden som skrivs skall använda variablerna i GVL (Global variable list) eftersom dessa är kopplade till simuleringen, det går inte att ta bort dessa men det är möjligt att lägga till variabler i listan. Skapa en egen variabel lista genom att högerklicka på Application, gå till Add Object och välj GVL.
Själva simuleringen sker i Visualization, dubbelklicka på Visualization för att komma dit.

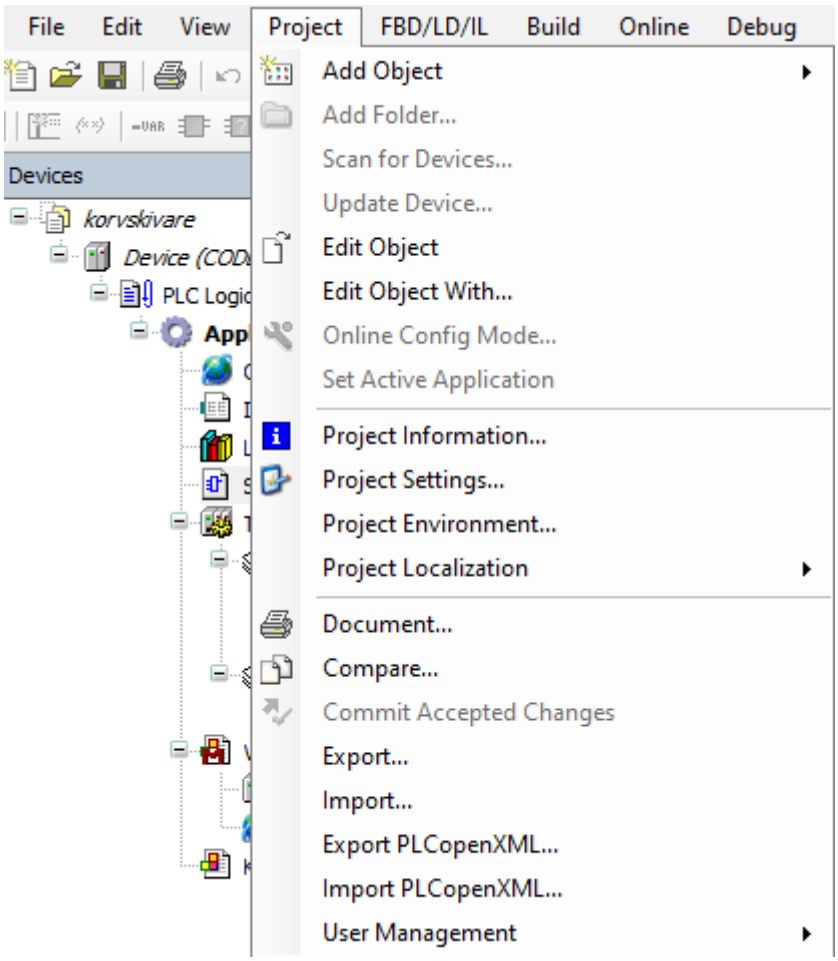
Följande filer är endast simulering och ska inte ändras av studenten:

- Simuleringsfil
- VISU_TASK
- Visualization Manager

7 Anpassning från Simulator till labbsal

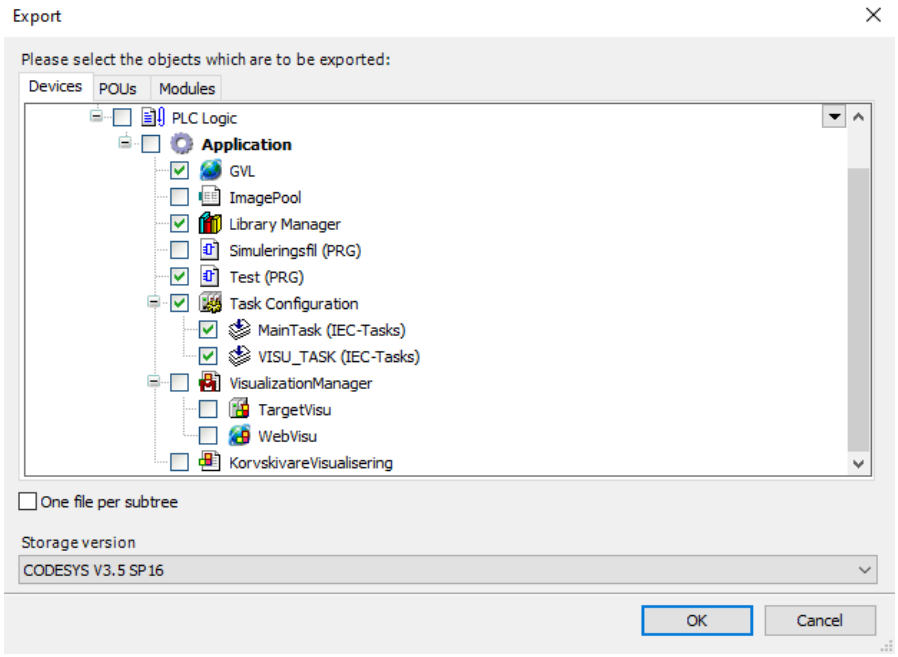
Följande kapitel är användarhandledning för överföring av kod skriven i simulatorn till laborationssalens utrustning.

När koden skall överföras från simulatorn till datorerna i laborationssalen behöver koden exporteras genom att klicka på Project högst upp följt av Export... se figur 16.



Figur 16. Exportering av projekt.

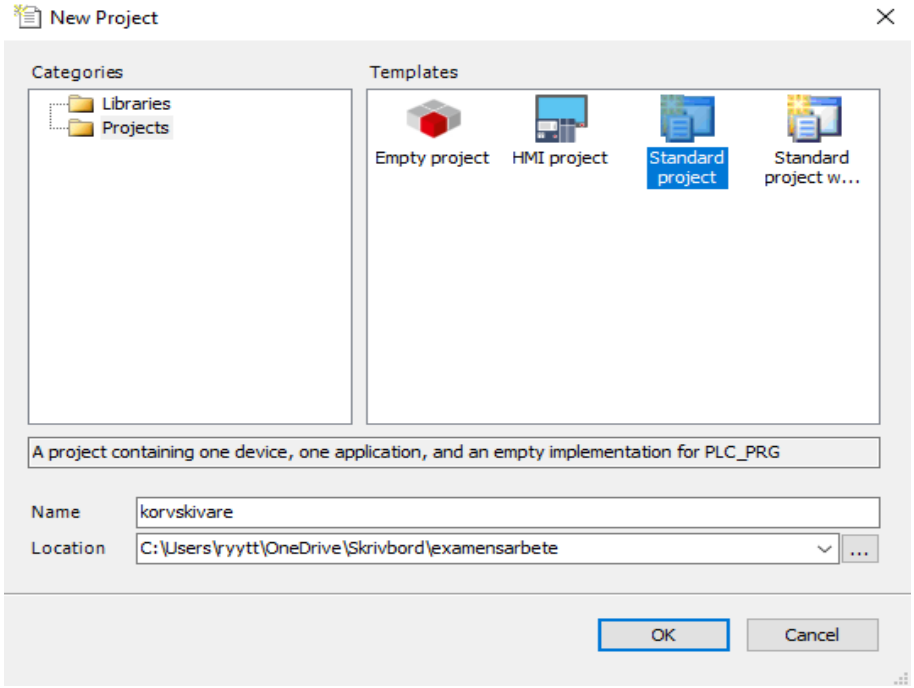
Markera dina skapade filer men även de filer som är markerade i figur 17. VISU_TASK exporteras för att visualiseringens uppdateringsfrekvens ska vara samma innan och efter exportering.



Figur 17. Val av filer som skall exporteras.

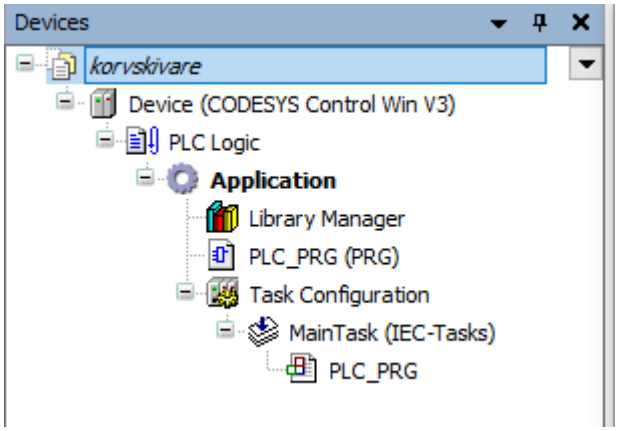
Spara sedan filen och ta med den till laborationssalen.

Öppna Codesys på valfri dator i laborationssalen och klicka på New Project...
Välj sedan Standard project och välj ett namn på projektet, se figur 18. Klicka OK

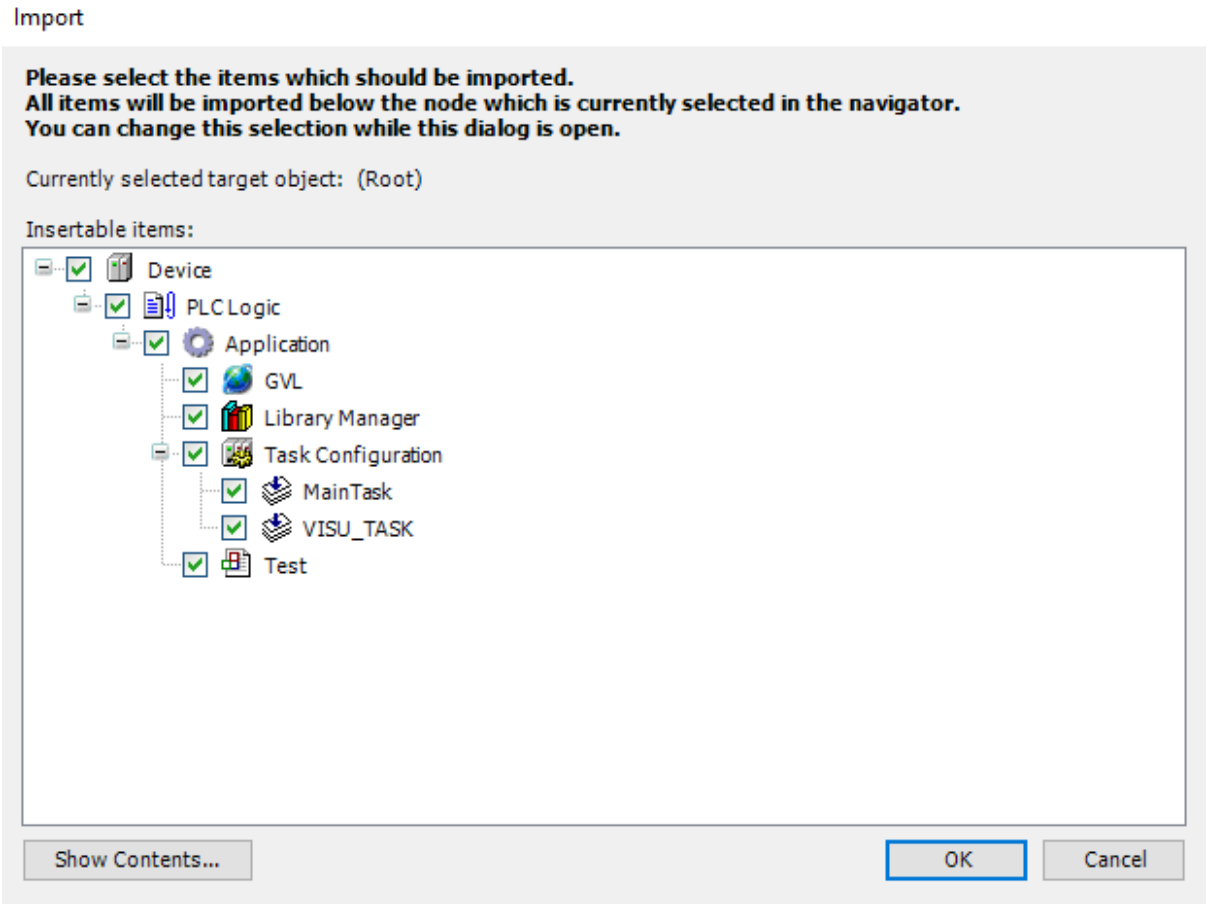


Figur 18. Uppstart av projektet i skolan.

Klicka sedan på Project högst upp och sedan Import…
Välj den exporterade filen och klicka Open.
Nu öppnas Import fönstret. Står det "*There are no objects in the export file which can be imported at the currently selected location."* Klicka på filnamnet i device fönstret uppe till vänster så kommer filerna upp, se figur 19 och 20.

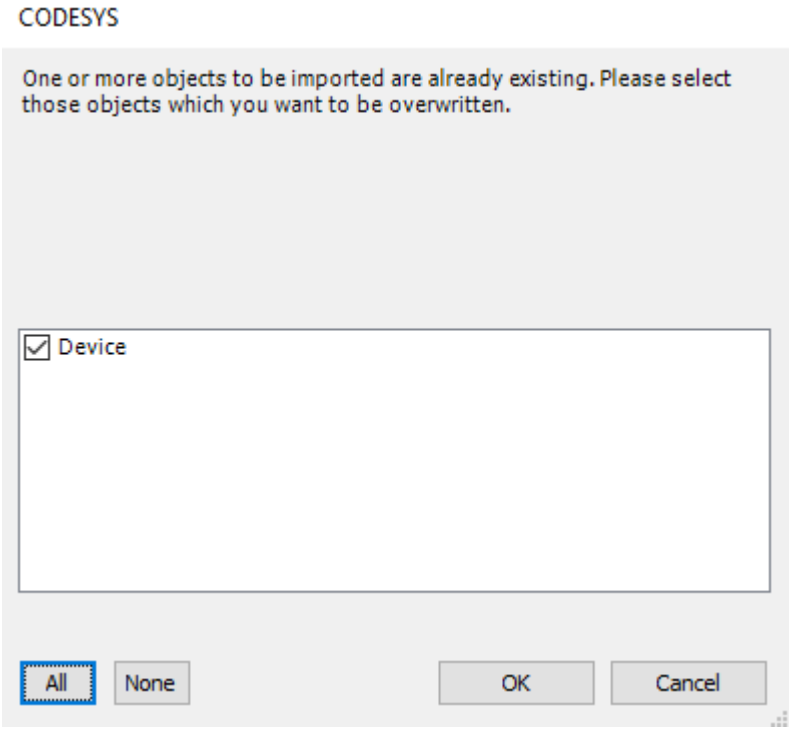


Figur 19. Devicesfönster för nya projektet.



Figur 20. Importeringsfönster där val av filer görs.

Markera alla filer i Import fönstret och klicka OK. Nu kommer Codesys fråga om du vill ersätta de befintliga filerna med de nya välj All och klicka OK som i figur 21.



Figur 21. Codesys popupfönster vid importering.

Står det nere till höger SIMULATION med röd bakgrund klicka på online högst upp och avbocka Simulation.

8 Resultat

Målet med projektet var att skapa simulatorer som ska användas av studenter för att underlätta den praktiska undervisningen på Chalmers campus Lindholmen. De fyra simulatorerna som skapats fungerar och kan kopplas direkt från simulator till det verkliga systemet via PLC:n utan ytterligare komplikationer. Eftersom en av simulatorerna har en roterande del som roterar 90 grader blev den visuella representationen i en två dimensionell miljö istället en spegling vid aktivering. Lösningen påverkar inte koden som skrivs men kan behöva testas av studenten för att förstå att det representerar rotationen.

Videoklipp som demonstrerar simulatorernas funktion:
https://www.youtube.com/watch?v=dZUZrrn2WS4&feature=youtu.be&ab_channel=Ryytty

9 Diskussion och slutsats

Flertal saker i detta projekt hade kunnat hanteras annorlunda. Hårdkodade lösningar som inte är anpassningsbara för andra framtida simulatorer är ett förbättringsområde. Varje puck fick även sin egen kod för dess rörelse vilket gjorde att simuleringsfilen blev väldigt stor. Även detta är ett förbättringsområde. Simulatorerna är även skapade för att köras precis som det verkliga systemet, rörelsebegränsningar utöver dessa har ej skapats vilket kan leda till varierande resultat om simulatorn körs på ett annat sätt. Exempelvis är puckarnas rörelser baserade på att deras rörelse börjar vid en specifik punkt. Är inte pucken vid denna exakta punkt blir resterande rörelser påverkade av felmarginalen vilket kan leda till större felmarginal vid nästa rörelse.

Eftersom simulationerna skulle spegla verklig utrustning blev HMI aspekten begränsad. För stora visuella skillnader mellan simulator och verklighet hade endast bidragit negativt då igenkänningsfaktorn försvinner, som ansågs vara viktigast i detta projekt. En grå bakgrund som anses vara idealt gav ingen kontrast då stora delar av utrustningen redan var gråfärgad.

Valet av programmeringsspråk hade ändrats nu i efterhand för de systemen med puckar. Detta för att minska mängden kod samt läsbarheten i koden. Fördelen att ha kvar funktionsblock som språk var att det gick att se exakt var i koden felet låg under utvecklingsprocessen. På Mecman-platta simulatorerna var därför funktionsblock ett bra språk då dessa ej krävde en stor mängd kod. Programmeringsspråket structured text hade troligtvis förenklat koden för puckarna.

Om man bortser från puckarna är de övriga lösningarna i simulatorerna anpassningsbara för framtida simulatorer. Variabler för simuleringens rörelsehastigheter är skapade och gränsvärden för dess start och slutposition, därmed behövs endast parametrar ändras för att anpassa de till nya simuleringar.

Då simulatorerna för processkedjan är sammanfogade i verkligheten hade en framtida utvecklingspotential för arbetet varit att även sammanfoga simulatorerna. Ytterligare utvecklingspotential finnes i korvskivaren och tilluftssystemet där en mer verklig process kan simuleras rent visuellt. Istället för att cylindrar skall representera knivar och ventiler kan det ersättas med simuleringar av verkliga verktyg.

Eftersom samtliga mål är uppfyllda är det förväntade resultatet likt det faktiska. Med den förbättrade tillgängligheten till simulatorn och laborationsutrustningen kan utbildningen förändras. Testa nya funktioner för att eventuellt bygga ut testplattorna är möjlig samt att göra ändringar i kursen.

9.1 Etik och hållbar utveckling

Det är ingen nyhet att simuleringar har sina fördelar. Tillgängligheten samt priset gör det till ett lysande testverktyg då eventuella fel kan rättas innan koden sätts i bruk och eventuellt skadar utrustningen. I detta fall när undervisningen på plats i skolan är begränsad underlättar det även för studenter då tillgängligheten till utrustningen istället blir obegränsad med simulatorn. I ett större perspektiv som utländsk support kan ett bolag med hjälp av simulatorer testa processer på andra sidan planeten utan att behöva resa dit. Simulatorernas påverkan på industrin är därmed väldigt positiv då det både kan minska miljöpåverkan i form av effektivisering av processer och behovet av resor för personal.

Nackdelar med simuleringar kan vara att mer komplexa system kan ta väldigt lång tid att simulera, samt i utbildningssyfte kan problematik uppstå med hårdvara som kan vara svår att simulera, till exempel glappande givare. Långa utvecklingsprocesser kostar pengar i form av lön till utvecklare och användning av enbart simulerade processer kan leda till försämrad utbildning då studenten inte får lära sig att koppla samman utrustningen.

I fall där simulatorer helt kan ersätta utrustning besparas miljön under både framställningen och driften. I detta fall där pneumatik används som är en energiöverföring med låg verkningsgrad är skillnaden extra tydlig. Brist på återvinningsförmåga av hårdvara är även miljöbelastande då nya produkter tillverkas utan att kunna återvinna till 100%. Det är även etiskbelastande eftersom de gamla produkterna i vissa fall skickas till utvecklingsländer som inte har kapaciteten att ta hand om avfallet på ett säkert sätt som skyddar de anställda och miljön. Kan man då begränsa tillverkningen av hårdvara med hjälp av simulatorer förbättras detta.

Referenser

[1] Electric motor cycle 2.png, Wikimedia [Online] Tillgängligt på:
https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_2.png
[Hämtad: 15-April-2021]

[2] A. Hughes, *Electric Motors and Drives: Fundamentals, Types and Applications*. Third Edition., Oxford, England: Elsevier Ltd, 2006. [Online] Tillgängligt på:
http://www.emic-bg.org/files/Electric_Motors___Drives.pdf
[Hämtad: 11-Januari-2021]

[3] Hydraulic cylinder, Wikipedia [Online] Tillgängligt på:
https://en.wikipedia.org/wiki/Hydraulic_cylinder
[Hämtad: 24-Febuari-2021]

[4] Pneumatik, Wikipedia [Online] Tillgängligt på:
https://sv.wikipedia.org/wiki/Pneumatik
[Hämtad: 15-April-2021]

[4] Maskinstyrningar i praktiken, RISE, rapport 2018:02 [Online] Tillgängligt på:
https://www.ri.se/sites/default/files/2019-10/RISE%20Rapport%202018-02%20Maskinstyrningar%20i%20praktiken.pdf
[Hämtad: 14-Januari-2021]

[5] Crevis NA-9371_2_3_UserManual, Crevis [Online] Tillgängligt på:
http://www.crevis.co.kr/eng/product/view01.php?str_bcode=030120001&str_no=383&page=1 Download Manual
[Hämtad: 8-Februari-2021]

[6] Kelvin Erickson: Programmable logic controllers. [Online] Tillgängligt på:
https://ieeexplore.ieee.org/document/481370
[Hämtad: 8-Februari-2021]

[7] Codesys, Wikiperdia [Online] Tillgängligt på:
https://en.wikipedia.org/wiki/CODESYS
[Hämtad: 15-April-2021]

[8] The System: Why Codesys, Codesys [Online] Tillgängligt på: https://www.codesys.com/the-system/why-codesys.html
[Hämtad: 24- Februari-2021]

[9] HMI Design Best Practices: The Complete Guide, dataPARC [Online] Tillgängligt på:
https://www.dataparc.com/blog/hmi-design-best-practices-complete-guide/

[Hämtad: 8-Januari-2021]

[10] Sensorer, Wikipedia [Online] Tillgängligt på:
https://sv.wikipedia.org/wiki/Sensor
[Hämtad: 15-April-2021]

INSTITUTIONEN Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2021
www.chalmers.se

CHALMERS