CODEN:LUTEDX/(TEIE-5511)/1-90/(2024)

Industrial Electrical Engineering and Automation

Digitala tvillingar som testverktyg för automationssystem

Edvin Mårtensson

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

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering
Lund University
Lund 2024

Abstract

In water management and purification, various processes and technologies are used to ensure that water is safe to consume, and that wastewater is handled in an environmentally friendly way. To control these processes and ensure a stable flow and a safe process, various automated systems are used. These automation systems control everything from different flows through pumps and valves, to the amount of the different substances added to purify our water. Simulation has become a powerful method for understanding and optimizing complex processes in several widely different industries. By creating virtual models of various processes, scientists and engineers can test different scenarios, evaluate efficiency and identify potential improvements without having to intervene in real systems.

In this report, the connection between automation systems and simulation tools is studied. To achieve this goal, a simple digital twin is created. The work uses Siemens' SIMIT simulation tool to build up the digital twin. The automation system used in the work is part of the control of a real wastewater treatment plant programmed in ABB's system 800xA. To perform the connection between the automation system and the digital twin, an OPC connection is used.

The work shows that the connection between automation system and digital twin can be achieved in the desired way. It also shows how the digital twin can be used to test the function of the automation system and how the current process is affected in different scenarios. The report then discusses the produced results based on the aspects of technical implications, learning environment, sustainability perspective, and requirements picture. Finally, the report describes what conclusions can be drawn from this project. The overall objective of this work was to implement a simulation solution for a water treatment plant and connect the simulation to an existing automation solution. This may be considered possible and carried out. Furthermore, in the simulation, it would be possible to make adaptations in the process using SIMIT, where there would also be a graphical presentation of the simulation, and the impact would be shown in operator images for the automation solution. This, too, may in the same way be considered as completed and functioning. The connection between the existing automation system and the simulation model to be investigated could be carried out using an OPC connection as desired.

Sammanfattning

Inom vattenhantering och rening används olika processer och teknologier för att säkerställa att vatten är säkert att konsumera och att avloppsvatten hanteras på ett miljövänligt sätt. För att kontrollera dessa processer och säkerställa ett stabilt flöde och en säker process används olika automatiserade system. Dessa automationssystem kontrollerar allt från olika flöden genom pumpar och ventiler, till mängden av de olika ämnen som tillsätts för att rena vårt vatten. Simulering har blivit en kraftfull metod för att förstå och optimera komplexa processer inom flera vitt skilda industrier. Genom att skapa virtuella modeller av olika processer kan forskare och ingenjörer testa olika scenarier, utvärdera effektivitet och identifiera potentiella förbättringar utan att behöva ingripa i verkliga system.

I denna rapport studeras kopplingen mellan automationssystem och simuleringsverktyg. För att nå detta mål skapas en enklare digital tvilling. Arbetet använder sig av Siemens simuleringsverktyg SIMIT för att bygga upp den digitala tvillingen. Automationssystemet som används i arbetet är en del av styrningen till ett verkligt avloppsreningsverk som är programmerat i ABB:s system 800xA. För att utföra kopplingen mellan automationssystem och digital tvilling används en OPC-koppling.

Arbetet visar att kopplingen mellan automationssystem och digital tvilling går att åstadkomma på önskat sätt. Den visar även hur den digitala tvillingen kan användas för att testa automationssystemets funktion och hur den aktuella processen påverkas i olika scenarion. Rapporten diskuterar sedan det framtagna resultatet utifrån aspekterna tekniska implikationer, lärandemiljö, hållbarhetsperspektiv, samt kravbild. Slutligen redogör rapporten för vilka slutsatser som kan dras av detta projekt. Den övergripande målsättningen för detta arbete var att implementera en simuleringslösning för ett avloppsreningsverk samt koppla ihop simuleringen mot en befintlig automationslösning. Detta får anses som möjligt och utfört. Vidare skulle det i simuleringen vara möjligt att göra anpassningar i processen med hjälp av SIMIT, där det även skulle finns grafisk presentation av simuleringen, och påverkan skulle visas i operatörsbilder till automationslösningen. Även detta får på samma sätt anses som utfört och fungerande. Den koppling mellan existerande automationssystem och simuleringsmodell som skulle undersökas kunde utföras med hjälp av en OPC-koppling som önskat.

Tack

Jag vill rikta ett stort tack till alla som har bistått mig i detta projektarbete. Mina handledare Jörgen Svensson (studierektor och universitetslektor, IEA, LTH) och Arto Lind (Business Section Manager, AFRY) för att ni gett mig möjligheten att genomföra detta projekt samt bistått mig med vägledning och återkoppling. Daniel Karlsson och Nils-Inge Engström för all hjälp kopplat till simuleringen. David Olsson, Joakim Kock, Abanoub Ramzy och Patrik Lindestam för ert stöd kopplat till automationssystemet och ändringen av miljön. Även ett extra tack till Mazen Abed Alhadi för hans välkomnande och stöd under min tid på AFRY.

Innehåll

1 Introduktion **1**
- 1.1 Bakgrund . . . 1
- 1.2 Mål . . . 1
- 1.3 Metod . . . 2
- 1.4 Omfattning och begränsningar . . . 2
- 1.5 Översikt . . . 2

2 Litteraturstudie **3**
- 2.1 VA-processer . . . 3
  - 2.1.1 Vattenverk . . . 3
  - 2.1.2 Avloppsverk . . . 4
- 2.2 Automation . . . 5
- 2.3 Simulering . . . 6
- 2.4 Digital tvilling . . . 7
- 2.5 AFRY . . . 7
  - 2.5.1 Sydvatten . . . 8
  - 2.5.2 Käppalaförbundet . . . 8
  - 2.5.3 BillerudKorsnäs . . . 8
  - 2.5.4 GE Healthcare . . . 8
  - 2.5.5 Stockholm Vatten och Avfall . . . 8

3 Implementation **11**
- 3.1 Metod . . . 11
- 3.2 Arbetsprocess för automatisering . . . 11
- 3.3 Automationssystem - struktur . . . 13
  - 3.3.1 ABB 800xA . . . 13
  - 3.3.2 Automationssystem - processbeskrivning . . . 18
  - 3.3.3 Utvald process . . . 21
  - 3.3.4 Integration med simulering . . . 25
- 3.4 Simulering . . . 29
  - 3.4.1 SIMIT . . . 29
- 3.5 Avgränsningar . . . 34
  - 3.5.1 Tekniska begränsningar . . . 34
  - 3.5.2 Tidsbegränsningar . . . 35

4 Resultat **37**
- 4.1 Simulering . . . 37

5 Diskussion **49**
- 5.1 Tekniska implikationer . . . 49
- 5.2 Lärandemiljö . . . 50
- 5.3 Hållbarhetsperspektiv . . . 50
- 5.4 Kravbild . . . 50

6 Slutsatser och framtida arbete **53**

7 Referenser **54**

Bilagor **56**
- A. Demosystem . . . 56
- B. Uppbyggnad av templates . . . 65

C. Genererade objekt 70

Akronymer

SIMIT - Simuleringsprogram från Siemens.
ABB 800xA - Automationssystem från ABB.
VA - Vatten & Avlopp.
PLC - Programmable Logic Controller.
IoT - Internet of Things, samlingsnamn för de tekniker som gör att vardagsföremål, maskiner, fordon och byggnader, med inbyggd elektronik och Internetuppkoppling, kan styras eller utbyta data över nätet.
Big Data - Digitalt lagrad information av sådan storlek att det är svårt att bearbeta den med traditionella databasmetoder.
SCADA - Supervisory Control And Data Acquisition, system för övervakning och styrning av processer.
CPU - Central Processing Unit.
I/O - Input/Output.
VVS - Värme, Ventilation och Sanitet.
HMI - Human-Machine Interface.
P&ID - Process- och instrumentdiagram.
HW - Hardware.
OPC - Open Platform Communications.
RAS - Return-Activated Sludge.
RSPS - Return Sludge Pump Station.
WAS - Waste-Activated Sludge.
RDT - Real Digital Twin.
RDT Runtime - Bibliotek med komponenter från AFRY.
Flownet - Bibliotek med komponenter från Siemens.

1 Introduktion

1.1 Bakgrund

Vatten är jordens mest livsnödvändiga resurs och utgör en central del av människors dagliga liv och industriella verksamheter. I Sverige spelar vatten en särskilt viktig roll med tanke på landets omfattande vattenresurser och dess betydelse för ekonomi, miljö och samhälle.

Inom vattenhantering och rening används olika processer och teknologier för att säkerställa att vatten är säkert att konsumera och att avloppsvatten hanteras på ett miljövänligt sätt. Vattenverk och avloppsreningsverk är centrala anläggningar som ansvarar för att utföra dessa processer, vilka innefattar allt från mekanisk filtrering och kemisk behandling till biologiska reningstekniker. För att kontrollera dessa processer och säkerställa ett stabilt flöde och en säker process används olika automatiserade system. Dessa automationssystem kontrollerar allt från olika flöden genom pumpar och ventiler, till mängden av de olika ämnen som tillsätts för att rena vårt vatten.

Simulering har blivit en kraftfull metod för att förstå och optimera komplexa processer inom flera vitt skilda industrier. Genom att skapa virtuella modeller av olika processer kan forskare och ingenjörer testa olika scenarier, utvärdera effektivitet och identifiera potentiella förbättringar utan att behöva ingripa i verkliga system. Simulering möjliggör också utveckling och testning av automationssystem genom att tillåta för övervakning och studier av system i realtid. Simuleringar kan även användas som testmiljöer för upplärning av operatörer, samt fungera som en möjlighet att testa olika felscenarion utan att påverka det verkliga systemet.

I denna rapport kommer kopplingen mellan automationssystem och simuleringsverktyg att studeras, för att utreda vilka fördelar som finns med att använda simulerade miljöer i samband med VA-processer, samt hur denna koppling kan göras på ett fungerande sätt. För att nå detta mål kommer en enklare digital tvilling att försöka skapas. Arbetet utförs i samarbete med AFRY.

1.2 Mål

Den övergripande målsättningen för arbetet är att implementera en simuleringslösning för ett avloppsreningsverk samt koppla ihop simuleringen med en befintlig automationslösning. I simuleringen skall det vara möjligt att göra anpassningar i processen med hjälp av simuleringsprogrammet SIMIT, där det även finns grafisk presentation av simuleringen, och påverkan ska visas i operatörsbilder till automationslösningen. För att nå den övergripande målsättningen kommer följande delmål att fungera som en viktig del av arbetsprocessen.

Delmål:

- Studera och skapa förståelse för VA-processer.
- Studera och förstå hur automationslösningar är uppbyggda med till exempel ABB 800xA.
- Studera simuleringsverktyget SIMIT för att kunna använda och implementera till VA-processer.

Frågeställningar:

- Vilka typer av VA-processer finns?
- Hur ser en typisk VA process ut?
- Hur ser en automationslösning ut för en VA process?
- Hur kan en VA process simuleras?
- Hur ska en automationslösning sammanlänkas med en simulering på bästa sätt?

1.3 Metod

Grunden till rapporten består av den litteraturstudie som kommer att inleda arbetsprocessen. Denna bidrar till en ökad förståelse kring hur VA-processer fungerar och är uppbyggda. Utifrån denna kommer också en inblick skapas angående vilka delar i en VA process som är relevanta att simulera.

Vidare kommer befintliga automationslösningar för olika VA-processer att studeras, för att skapa en förståelse för hur dessa är uppbyggda. Fokus kommer att vara på lösningar som använder ABB:s automationssystem, ABB 800xA.

Därefter kommer simuleringsverktyget SIMIT studeras och testas för att skapa tillräcklig kännedom och kunskap av programmet för att kunna implementera modeller för VA-processer. Vidare kommer fokus att vara på hur SIMIT ska kombineras med befintliga ABB 800xA automationslösningar på bästa sätt.

Slutligen kommer SIMIT att implementeras för att simulera en anpassningsbar VA process med en befintlig automationslösning.

1.4 Omfattning och begränsningar

För att genomföra arbetet inom tilltänkt tid simuleras inte hela avloppsreningsverket. Istället studeras en viss del av verket som en isolerad instans och används för att bygga simuleringen kring denna. Denna del bestäms senare i arbetet när en tydligare bild har uppnåtts kring vilket arbete och vilken tid som krävs för att bygga de modeller och kopplingar som innefattas i arbetet. Även de tekniska faktorer som påverkar vilken del av processen som är lämplig för en isolerad simulering inverkar på vilken del som väljs. Då vissa delar av verket är starkare knutna till omgivningen än andra kommer detta påverka valet av den isolerade sektionen.

1.5 Översikt

Denna rapport är indelad i sex huvudsakliga kapitel. En översikt av dessa presenteras nedan.

- **Kapitel 1.** Detta kapitel fungerar som en introduktion till arbetet. I detta avsnitt ges en bakgrund till projektet, samt en kort beskrivning av de mål, frågeställningar och begränsningar som på förhand anses påverka arbetet. Även en överblick till den metod som kommer att användas ges i detta avsnitt.
- **Kapitel 2.** I detta kapitel presenteras den bakomliggande teori som framkommer i litteraturstudien. Här beskrivs bland annat VA-processer, automationslösningar och digitala tvillingar för att skapa en grund för det vidare arbetet.
- **Kapitel 3.** Här beskrivs implementationen av det tänkta arbetet. Denna del innehåller en mer detaljerad beskrivning av metoden som används för arbetet, arbetsprocessen som har använts, en genomgång av automationssystemets uppbyggnad och hur simuleringen är designad.
- **Kapitel 4.** I detta kapitel presenteras resultaten av arbetet.
- **Kapitel 5.** Detta kapitel diskuterar de resultat som har tagits fram, vilka implikationer som följer och vad dessa innebär kring olika aspekter av arbetet.
- **Kapitel 6.** Slutligen redogör detta kapitel för de slutsatser som kan dras samt vilket framtida arbete som kan följa efter detta arbete.

2 Litteraturstudie

Projektet syftar till att ta fram en simuleringsmodell av en avloppsreningsprocess med hjälp av ett automationssystem av ett avloppsreningsverk skapat i ABB 800xA. Med en integration mellan simulerings- och automationssystem strävar projektet efter att förbättra förståelsen för vattenprocesser och optimera dess effektivitet och tillförlitlighet. Genom att utforska möjligheten att länka samman simulering och automationssystem hoppas projektet bidra till framsteg inom automation, vattenhantering och rening för en mer hållbar framtid.

VA industrin i Sverige och världen står inför stora utmaningar och investeringar för att kunna tillgodose vattenförsörjningen till en växande befolkning samt rening av avloppsvatten på ett hållbart sätt. Det investeras stora belopp för att öka kapacitet, förbättra processer och att införa effektivisering via digitalisering och mer avancerad automation. Att kunna simulera processer öppnar upp möjligheter för att kunna effektivisera lösningar på olika sätt. Till exempel kan man med hjälp av simulering testa framtagna automationslösningar så att projektgenomförandet, speciellt idrifttagning kan ske på ett snabbare och mer effektivt sätt. Därtill kan simulering av hela processen vara ett verktyg för att kunna optimera driften av anläggningen.

2.1 VA-processer

Den simulering som är i fokus för resultat- och diskussionsdelarna senare i denna rapport fokuserar på en specifik VA-process. För att bygga en förståelse för de olika VA-processer som finns och som i regel bygger på avancerade automationslösningar, och därigenom är högst aktuella att simulera för optimering, handlar detta första avsnitt om VA-processer i allmänhet. De två största processerna inom VA är vattenverk och avloppsverk. Genom dessa sker den reningsprocess som är nödvändig både för att kunna försörja Sveriges befolkning med rent dricksvatten, men också för att ta hand om smutsigt avloppsvatten.

Det kretslopp som utgör Sveriges VA-system bygger på att det vatten som ska bli vårt dricksvatten hämtas från någon av de cirka 2000 vattentäkter som finns runt om i landet. Därefter renas vattnet i ett vattenverk innan det distribueras i de ledningsnät och vattentorn som följaktligen fördelar ut vattnet till respektive hushåll. När vattnet sedan har använts skickats det via avloppsledningsnät till ett avloppsverk. Där renas vattnet återigen för att därefter kunna spolas tillbaka ut i naturen (Svenskt Vatten 2019).

I Sverige försörjer kommunala vatten- och avloppsanläggningar 90 procent av befolkningen. Systemet består av 1750 vattenverk, 1700 avloppsverk och 101 000 kilometer avloppsrör, varav majoriteten byggdes under 1950-, 60- och 70-talen och närmar sig sin tekniska livslängd (NCC 2024). Sveriges yta består till nio procent av sjöar, och svenska hushåll använder endast mellan en halv och en procent av det sötvatten som finns tillgängligt (Svenskt Vatten 2019). Sveriges tillgång till rent vatten är därför mycket bra. Trots detta uppgav en av fyra kommuner att vattenbristen hade förvärrats och 40 procent av alla kommuner hade drabbats av vattenbrist någon gång under de senaste fem åren enligt en undersökning som Ekot gjorde 2019 (Sveriges Radio 2019). För att kunna klara av framtidens behov av vatten och avlopp trots klimatförändringar och ökad befolkningsmängd börjar behovet av modernisering och utbyggnad därför bli påtagligt, ett arbete som pågår för fullt.

2.1.1 Vattenverk

Det svenska systemet bygger som tidigare nämnts på de 1750 vattenverk som finns runt om i landet för att rena och distribuera dricksvatten runt om i Sverige. Hälften av det vatten som blir dricksvatten kommer från ytvatten runt om i Sverige, medan den andra hälften är jämnt fördelad mellan konstgjort och naturligt grundvatten (Svenskt Vatten 2016). Förutsatt att det tas om hand på rätt sätt ger både ytvatten och grundvatten dricksvatten av god kvalitet.

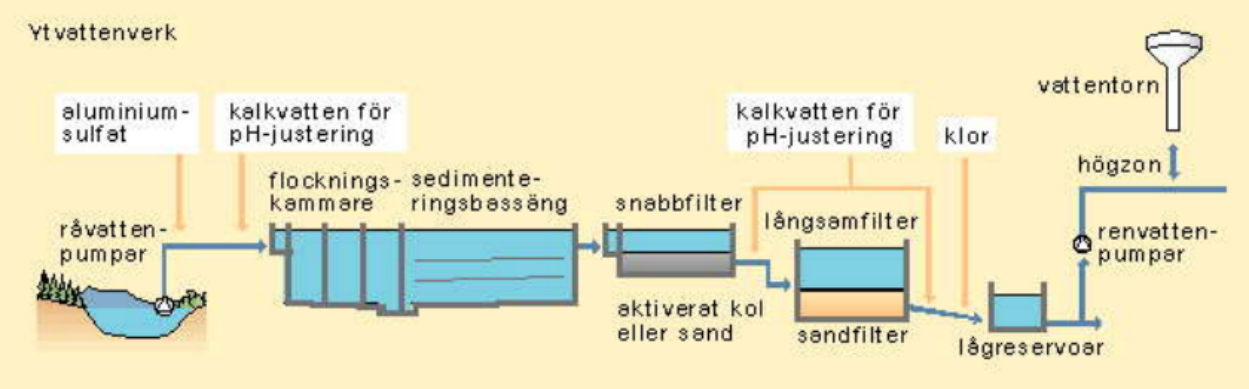
2.1.1.1 Grundvattenverk

Grundvatten har vanligtvis en högre kvalitet från vattentäkten och behöver därför normalt färre steg för att renas. De drygt 1450 grundvattenverken står därför för majoriteten av de vattenverk som finns i Sverige. De flesta av dessa försörjer färre än 2000 personer med sitt dricksvatten (Svensk Vatten 2016). En del verk kan vara så pass enkla att de endast består av ett pumpsystem, detta kräver dock väldigt god kvalitet på råvattnet. Vanligtvis består reningen av grundvattnet av att järn och mangan tas bort, varefter vattnet filtreras och desinficeras innan vattnet distribueras (Svenskt Vatten 2016).

2.1.1.2 Ytvattenverk

Ytvattenverken är betydligt färre än grundvattenverken, men är betydligt större. Hälften av Sveriges dricksvatten produceras i de endast 170 ytvattenverk som finns runt om i landet (Svenskt Vatten 2016). Att rena ytvatten för att producera dricksvatten är en betydligt mer komplicerad process än den för grundvattnet, med en hel del involverade steg.

Normalt sett passerar först vattnet genom en grovfiltrering, där större föremål rensas bort. I detta steg sker ofta även en pH-justering av vattnet. Vattnet går därefter vidare till en flockningskammare, i vilken kemikalier tillsätts. Kemikalierna gör att de ämnen som ska rensas bort samlas i så kallade flockar. I nästa steg, sedimenteringsbassängen, sjunker flockarna till botten av bassängen och filtreras därigenom bort. Därefter passerar vattnet genom ett snabbfilter och ett långsamfilter. Båda dessa består av sandbäddar där kvarvarande partiklar försvinner. I långsamfiltret finns även bakterier som tar bort lukt och smak från vattnet. I vissa verk ersätts dock långsamfiltret med ett kolfilter som har samma funktion. Till sist desinficeras vattnet genom att klor eller ozon tillsätts, eller att vattnet utsätts för UV-ljus för att renas. Därefter går vattnet vidare för att distribueras (Svenskt Vatten 2016).



Figur 1: Processer i ett ytvattenverk.

2.1.2 Avloppsverk

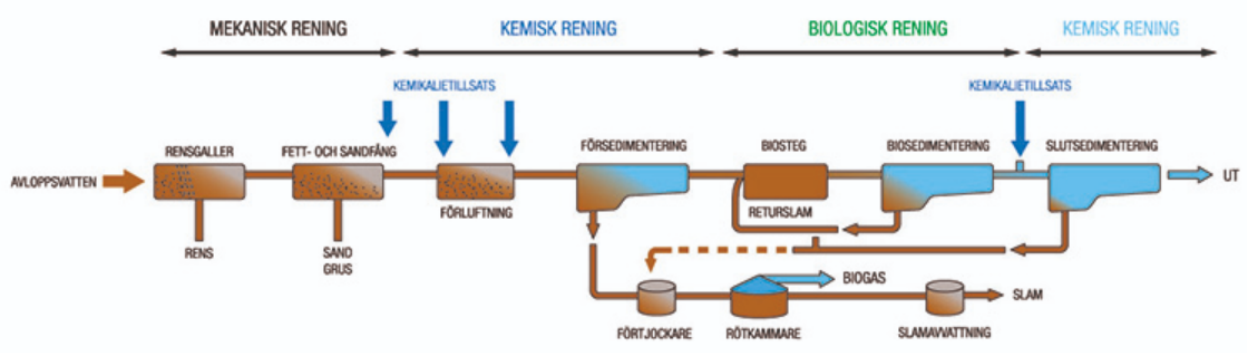
I Sverige finns som tidigare nämnts 1700 avloppsverk. Dessa är byggda för att ta hand om avloppsvatten från hushållens toaletter, bad, dusch, disk- och tvättmaskiner samt svensk industri. För att vattnet ska bli tillräckligt rent renas det i flera steg. Dessa ser olika ut i olika verk, men genomgående passerar vattnet genom någon form av mekanisk rening, biologisk rening och kemisk rening (VA SYD 2023).

Första steget i reningsprocessen är normalt sett grovreningen. Grovreningen kan till exempel bestå av silgaller som fångar upp det skräp som kommer i avloppet, samt sandfång som avlägsnar sand och grus från vattnet. Vattnet fortsätter därefter till nästa steg, försedimenteringen. I detta steg sjunker en stor del av de

partiklar som finns i vattnet ner till botten av de bassänger som försedimenteringen består av och filtreras bort som slam. Vidare fortsätter vattnet till nya bassänger där nästa steg, biologisk och kemisk rening sker. I detta steg tillsätts mikroorganismer som dels bryter ner organiska ämnen i vattnet, dels omvandlar kväve till kvävgas. Normalt tillsätts även någon fällningskemikalie, till exempel järnsulfat, som gör att den fosfor som finns i vattnet bildar fällningar och sjunker till botten med slammet. Efter den biologiska reningen fortsätter vattnet vidare till eftersedimentering. I eftersedimenteringen sjunker ytterligare partiklar som finns kvar i vattnet ner till botten i form av slam. Stora delar av slammet från eftersedimenteringen skickas ofta tillbaka till den biologiska reningen. Till sist tas de partiklar som fortfarande finns kvar i vattnet om hand i till exempel ett sandfilter. Även här tillsätts ibland järnsulfat (Käppala 2024).

Ett alternativ till sandfiltret i sista steget är den nya tekniken membranfiltrering. I denna ersätts sandfiltret av kassetter med långa rör av ett tunt membran som vattnet silas genom. Dessa sänks ner i ombyggda eftersedimenteringsbassänger, och filtrerar så pass noggrant att även mikroplaster och bakterier fastnar i membranet. Membrantrådarna rengörs regelbundet genom att luftbubblor släpps ut kring trådarna vilket får dessa att vibrera och därigenom skaka bort slammet som har fastnat. Denna teknik bedöms kunna dubblera kapaciteten på Henriksdals reningsverk i Stockholm, där denna nya teknik kommer att implementeras. Membrantekniken är mer yteffektiv, vilket gör att all rening kan flyttas in i berget vid Henriksdal utan att öka bassängvolymerna (Stockholm Vatten och Avfall 2021).

Slammet från reningsprocessen skickas på de flesta reningsverk vidare till en så kallad rötkammare, där den används för att producera biogas.



Figur 2: Processer i ett avloppsverk.

2.2 Automation

Detta arbete grundar sig till stor del på det som i dagens samhälle kallas för industriell eller avancerad automation. För att projektet ska vara möjligt att förstå kommer därför detta avsnitt att behandla vad automation är och hur det används i olika branscher.

I grunden kommer ordet automation från grekiskans *automatos* och betyder "på egen hand". Enkelt sagt innebär automation alltså att en maskin eller någon form av teknik utför ett arbete på egen hand. I verkligheten är automationssystem ofta otroligt komplexa, med stora datanätverk, stora mängder sensorer, visualisering och hantering av processinformation i realtid. För att en maskin eller en anläggning i ett automationssystem ska kunna fungera krävs någon form av styrning som kontrollerar vad olika delar av maskinen eller anläggningen ska göra. Detta kallas för styrsystem och kan i äldre eller mindre maskiner styras med så kallad relästyrning, där endast olika relän kontrollerar vad som ska göras.

I större anläggningar används däremot istället oftast en eller flera PLC:er för att kontrollera systemen. PLC står för *Programmable Logic Controller* och är en sorts dator som används för att ta emot och läsa signaler

från till exempel sensorer eller knappar och utifrån sin programmerade logik därefter utföra en uppgift, så som att starta en motor eller öppna en ventil. I ett vattenreningsverk kan antalet signaler variera mellan ungefär 500 signaler i ett litet verk, upp till kring 30000 signaler i ett stort reningsverk (Ramin et al. 2022).

Övergripande kan det sägas existera två olika typer av automation: automation av individuella processer samt automation av ett flöde (Hoffman-Walbeck 2022). Till den första kategorin hör till exempel automation av individuella maskiner eller robotar. Denna typ av automation fyller en viktig funktion i att minska den handpåläggning som krävs i en arbetsprocess, men bygger likväl på flera olika delar där någon form av stopp eller avbrott förekommer mellan varje steg. Den andra kategorin rör flöden där hela processen från start till slut i regel är automatiserad som ett flöde. Denna typ av automation kan för det mesta även den kräva någon form av handpåläggning, men oftast betydligt mindre (Hoffman-Walbeck 2022). Denna typ av flödesautomation är till exempel vanlig i VA-processer, och är därför den form av automation som är relevant för detta arbete.

Som nämnts ovan kan olika mycket handpåläggning krävas beroende på hur automationsprocessen är designad. Oavsett vilken av de två tidigare nämnda kategorierna en process tillhör kan processerna designas med olika krav på mänsklig inblandning. Tre olika nivåer kopplat till vilken inblandning som krävs kan sägas existera, *human-in-the-loop*, *human-on-the-loop* och *human-out-of-the-loop* (Myers et al. 2022). *Human-in-the-loop* syftar till system som programmeras och styrs med en tänkt hög mänsklig inblandning. Det kan till exempel röra sig om att en maskin gör en del av ett jobb, där det sedan krävs mänsklig inblandning för nästa steg. I dessa system samverkar alltså människa och maskin på ett planerat sätt för att utföra en process, ofta utifrån en tänkt turordning. *Human-on-the-loop* syftar till system som programmeras utifrån att övervakas av en operatör eller någon annan person som finns tillgänglig och kan ingripa om det skulle krävas av systemet, och endast i de fall där det är nödvändigt. I dessa system är alltså den mänskliga inblandningen måttlig, men likväl existerande. Den sista kategorin, *human-out-of-the-loop*, rör sig om system som i regel inte kräver mänsklig inblandning. Dessa system måste alltså vara självgående och robusta, för att kunna utföra ett tänkt arbete utan konstant övervakning eller mänsklig påverkan (Myers et al. 2022).

2.3 Simulering

Simulering är en kraftfull metod som används inom en mängd olika områden för att modellera och analysera system och processer. Genom att skapa virtuella representationer av verkliga fenomen möjliggör simulering förståelse, problemidentifiering och optimering utan att behöva genomföra dyra eller riskabla experiment i verkligheten. Simulering kan i grunden definieras som imitationen av driften av en verklig process eller system över tid (Goienetxea Uriarte 2019).

Simuleringsmetoder har utvecklats och förfinats under många år och har blivit en oumbärlig del av både forskning och industriell verksamhet. Historiskt sett har simulering använts inom fysik, kemi och andra naturvetenskapliga områden för att förutsäga och förstå komplexa fenomen. Med teknologins framsteg har simulering även blivit alltmer vanligt inom ingenjörsbranchen, medicin, ekonomi och många andra discipliner.

Det finns en mängd olika metoder för simulering, var och en med sina egna fördelar och tillämpningar. *Diskret händelsestyrd simulering*, *kontinuerlig simulering* och *agentbaserad simulering* är några exempel på vanliga metoder som används beroende på det specifika problemet och dess komplexitet (Goienetxea Uriarte 2019). Diskret händelsestyrd simulering används ofta för att modellera system där händelser inträffar vid specifika tidpunkter och därigenom orsakar en förändring i systemet, medan kontinuerlig simulering används för att analysera kontinuerliga processer över tid. Agentbaserad simulering fokuserar på att modellera individuella agenter och deras interaktioner för att studera hur deras beteende i komplexa system påverkar systemets utfall.

Inom industrin har simulering blivit ett populärt och kraftfullt verktyg för att optimera processer, förbättra produktkvalitet och minska kostnader. Genom att skapa virtuella modeller av produktionsanläggningar, logistiknätverk och andra komplexa system, så kallade digitala tvillingar, kan företag testa olika scenarier,

identifiera problem och optimera resursanvändning utan att behöva göra störande förändringar i den faktiska driften. Inom industrin tillämpas simulering för att förutsäga produktionens prestanda, utvärdera olika designalternativ och förbättra säkerheten och effektiviteten i olika processer (Ruiz Zúñiga 2020). Simulering används ofta i samband med att verkliga experiment anses för dyra eller omöjliga, och där analytiska lösningar är för komplicerade eller kostsamma att validera (Arjomandi Rad 2022).

2.4 Digital tvilling

Digitala tvillingar har på senare år framkommit som en banbrytande teknologi inom den industriella sektorn med potential att förändra hur företag planerar, utvecklar och hanterar sina produkter och processer. Konceptet digitala tvillingar innebär att skapa en virtuell modell eller kopia av en fysisk produkt, process eller system, som kontinuerligt uppdateras i realtid med data från den verkliga världen. Till skillnad från till exempel en så kallad digital skugga delas dock även data från den digitala tvillingen tillbaka till det fysiska systemet i realtid, snarare än att data endast går i en riktning (Herwig et al. 2021).

Olika teorier om ursprungskällan till uttrycket digital tvilling förekommer i olika artiklar, den mest genomgående förekomsten är dock att uttrycket från början föreslogs av Mike Shafto år 2010 och refererade då till virtuella kopior av utrustning som hörde till NASA. Vid den tidpunkten var digitala tvillingar baserade på datormodeller som kunde utvärdera och rekommendera förändringar till fysiska system för att optimera dessa. Sedan dess har konceptet genomgått ett flertal förändringar och utvecklingar. Den så kallade *fjärde industriella revolutionen* eller *Industrie 4.0* och alla de tekniska utvecklingar som följt med denna har varit en stor bidragande faktor till utvecklingen och populariseringen av digitala tvillingar. Genom framväxten av *IoT* och *Big Data* och den ökade möjligheten att både ta emot och skicka data i realtid till olika delar av maskiner eller processer som detta lett till, har en helt annan möjlighet till simulering i realtid uppkommit. Genom detta har digitala tvillingar i dess mer moderna form möjliggjorts (Lv, Fersman 2022).

Fördelarna som har framkommit med att använda digitala tvillingar som en del av en arbetsprocess är många. Genom att simulera och testa olika designalternativ i den digitala tvillingen kan företag till exempel accelerera produktutvecklingsprocessen och minska behovet av fysiska prototyper och testning. Digitala tvillingar möjliggör även kontinuerlig övervakning och analys av produktionsprocesser och utrustning i realtid. Detta gör det möjligt att identifiera och åtgärda problem innan de uppstår och optimera underhållsplaneringen för att minska driftstopp och produktionsbortfall (Lindblom, Samuelsson 2020). Genom att använda digitala tvillingar för att simulera och optimera produktionsprocesser kan företag även öka produktionshastigheten, förbättra kvaliteten och minska resursförbrukningen för att uppnå bättre produktionsprestanda och konkurrenskraft. Digitala tvillingar har även öppnat för nya affärsmodeller och tjänster, som till exempel tjänster baserade på prestanda eller prediktivt underhåll, där företag kan erbjuda sina kunder kontinuerlig övervakning och optimering av deras produkter och system.

Trots de potentiella fördelarna med digitala tvillingar finns det även utmaningar att övervinna, såsom dataintegration, datasäkerhet och komplexitet i att skapa och underhålla digitala tvillingar. Men med fortsatt teknologisk utveckling och innovation finns det stora möjligheter för företag att dra nytta av digitala tvillingar för att förbättra sin verksamhet och skapa konkurrensfördelar på marknaden.

2.5 AFRY

Som företag är AFRY involverade i hela näringskedjan inom VA (Vatten & Avlopp) industrin genom olika typer av konsultuppdrag och projekt mot flertalet kunder, alltifrån utredningar till byggnation och idrifttagning av anläggningar. Affärsområdet Advanced Automation inom divisionen Industrial & Digital Solutions, levererar framför allt projekt med el och automationslösningar till kunder som till exempel VA Syd, Sydvatten och Stockholm Vatten och Avfall med flera. AFRY erbjuder en mängd tjänster kopplade till VA-processer runt om i landet. Två av dessa tjänster är vattenrening och ledningsnät för VA system. Vattenrening innefattar deltjänster så som slamhantering, dagvattenhantering, kontrollprogram och processoptimering, medan ledningsnäten innefattar delar så som huvudledningar, distributionsnät,

pumpstationer och reservoarer. AFRY har utfört projekt och uppdrag inom området till flertalet kunder så som Sydvatten, Käppalaförbundet, BillerudKorsnäs, GE Healthcare och Stockholm Vatten och Avfall.

2.5.1 Sydvatten

AFRY genomför i samverkan med Sydvatten en uppgradering av alla automationssystem i de två vattenverken Ringsjöverket och Vombverket. Projektet genomförs i tre faser. Första fasen är kravutredning och systemval, vilket sedan följs upp av en samverkansentreprenad för en del av Vombverkets automationssystem som omfattar plattform och en processdel. I den tredje och sista fasen utförs ytterligare en samverkansentreprenad rörande resterande automationssystem i Vombverket samt Ringsjöverket.

2.5.2 Käppalaförbundet

AFRYs åtagande i samverkan med avloppsverket Käppalaförbundet bestod i att i två etapper migrera Käppala reningsverk från en i huvudsak ABB AC450-miljö till en AC800M-miljö. Båda systemen är ABB produkter. Uppgraderingen bestod av sju servrar samt sju klienter som konfigurerades och installerades i en ny process-domän. Projektet innefattade hårdvara, mjukvara, systemering, uppdatering av processbilder, installation, provning och idrifttagning. Omkoppling till den nya miljön kunde utföras med endast ett fåtal timmars stopp av processen per etapp.

2.5.3 BillerudKorsnäs

I samverkan med papperstillverkningen BillerudKorsnäs deltog AFRY i ett projekt för att ta fram ett reningskoncept för att minska utsläppen av fosfor för att kunna uppfylla framtida krav. AFRY utvecklade projektet i samverkan med kunden och ansvarade och deltog aktivt i projektets alla stadier. Projektet bestod av en konceptstudie, där flertalet olika teknik- och konceptalternativ utvärderades. Därefter gjordes en fördjupad förstudie av de bäst lämpade koncepten som tagits fram, innan ett förprojekt av det valda teknikkonceptet genomfördes. Därefter kunde projektet genomföras, innan det slutligen gjordes en processuppföljning.

2.5.4 GE Healthcare

GE Healthcare är ett företag i General Electric-koncernen, som bland annat tillverkar laboratorieinstrument och utrustning för medicinsk diagnostik. AFRY utvecklade tillsammans med GE Healthcare ett projekt för att ta fram koncept med avancerade tekniklösningar för att rena avloppsvatten från läkemedelsindustrin. Företaget ansvarade och deltog aktivt i alla steg längs vägen. Projektet bestod av en nulägesanalys som innefattande mätningar och framtagning av vatten- och materialbalanser. Därefter gjordes en förstudie av de bäst lämpade konceptalternativen. Projektet avslutades med pilotförsök av alternativa teknikkoncept.

2.5.5 Stockholm Vatten och Avfall

AFRY har genomfört och kommer vidare att genomföra ett flertal projekt för Stockholm Vatten och Avfall. Dessa projekt innefattar både automation men även elkraft. Tre av de större projekten rör Lovö vattenverk, Norsborg vattenreningsverk, samt Stockholms Framtida Avloppshantering.

2.5.5.1 Lovö vattenverk

På Lovö vattenverk genomförs ett projekt rörande byte av samtliga styrsystem. Det 60-70 tal PLC'er som tidigare har använts är uttjänta och går inte längre att reparera. AFRYs automationsansvariga arbetar bland annat med att utveckla systemstandard, dimensionering, programmering av nya PLC'er, konfigurering av operatörspaneler, samt överordnad kommunikation.

Parallellt genomförs även Projekt Elkraft Lovö på vattenverket. Projektet går ut på att verket ska öka sin kapacitet och framtidssäkra elkraftförsörjningen för att säkra en framtida ökning av

dricksvattenproduktionen i Storstockholmsområdet. Projektet består av att ersätta ett antal ställverk med tillhörande transformatorer. För att kunna klara av byte av ställverken under drift byggs nya ställverksbyggnader. De nya ställverken ska förses med PLC-system och operatörspaneler för styrning och övervakning. Dessa ska kommunicera överordnade SCADA-systemet. I detta projekt utför AFRY alla förstudier och utredningar kring byte av elkraftförsörjningen för att säkerställa framtida drift.

2.5.5.2 Norsborg vattenverk

På Norsborg vattenverk genomför AFRY en utredning kring att dela verket elkraftsmässigt i två delar, östra och västra verket. Projektet innefattar även en ny reservkraftsanläggning, med byte av elkraftsförsörjning för att säkerställa framtida drift. En utredning görs även kring både råvatten- och renvattenpumpar för den östra delen.

2.5.5.3 Stockholms Framtida Avloppshantering (SFA)

I samarbete med Stockholm Vatten och Avfall genomförs projektet Stockholms Framtida Avloppsrening. Bakgrunden till projektet är Stockholms ökande befolkning. Sedan 2003 har befolkningsmängden i länet ökat med 580 000 invånare, och ökningen väntas fortsätta, vilket ställer nya krav på reningen av Stockholms avloppsvatten. Projektet innefattar en ny avloppstunnel mellan Bromma och Henriksdal, om- och tillbyggnad av Henriksdalsverket med ny membranteknik, tillbyggnad av en ny förbehandlingsanläggning, Sicklaanläggningen, samt nedläggning av Bromma avloppsreningsverk. Projektet genomförs i 14 etapper.

De första fem etapperna innebar bland annat uppgradering av befintliga PCS 7-automationssystem, CPU-utbyte och redundansombyggnad av vissa processtationer. Även anpassning av operatörsbilder och uppgradering av datorhårdvara innefattades i de första etapperna. Vidare utvecklades en ny utvecklings- och kontrollmiljö och även en ny testdator som senare kommer att användas som utbildningsstation för verket. Dessa fem första etapper innebar även typkretsar för automationsskåp. Även ny programmering för MBR styrning av Biolinje 1 med tillhörande processystem, VVS, elkraft och belysning. Sist av de första fem etapperna var en ren installationsetapp med nya servrar, stationer och I/O-skåp.

Efterföljande etapper 6-11 är programmeringsetapper och applikationsarbeten för bilinjerna 2-7, rötkammare 1-7 och slamhanteringen med tillhörande processystem. Dessa etapper följs upp av etapperna 12-13 som kommer att vara applikationsetapper. I dessa kommer tillkommande processtationer, I/O-skåp för all ny process, VVS och elkraft på Sicklaanläggningen att innefattas. Även nya applikationer för samtliga funktioner med tillhörande processystem kommer att innefattas i dessa två etapper.

Etapp 14 kommer till sist att vara en rivningsetapp. I denna kommer rivningen av den gamla Sicklaanläggningen att hanteras. Även en uppdelning av det befintliga automationssystemet för Henriksdals reningsverk kommer att göras för att istället skapa två olika delsystem, Henriksdalsanläggningen samt Sicklaanläggningen.

På detta sätt kan arbetsprocessen för utbyggnad och uppdatering av ett automationssystem se ut. Mycket av arbetet består av tester och undersökningar för att kontrollera att alla delar som innefattas av systemen fungerar som de är tänkta. Med hjälp av utbyggda tester i simulerade miljöer så som den som skapas i detta arbete hade större delar av systemet kunnat testas innan det är tänkt att tas i bruk. Den nya Sicklaanläggningen som nämns ovan innefattar till exempel en motsvarande inloppsprocess som den som används i detta projekt.

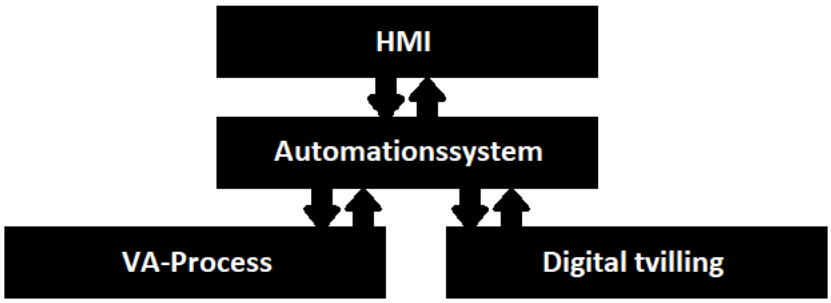
3 Implementation

3.1 Metod

I denna del av rapporten kommer sambandet mellan automationssystem, HMI, digital tvilling och process att förklaras mer genomgående. Även de ingående delarnas uppbyggnad kommer att studeras mer i detalj. Ett övergripande synsätt på sambandet mellan de olika delarna kan ses i figur 3. I centrum av figuren finns automationssystemet. Automationssystemet bygger på att olika sensorer och mätare registrerar och samlar information från den aktuella processen och utifrån den insamlade informationen styrs systemets pumpar, ventiler, motorer och andra aktuella objekt beroende på vilken påverkan som krävs på systemet.

För att en operatör, ingenjör eller annan person ska kunna övervaka och kontrollera systemet på ett effektivt sätt kopplas automationssystemet till ett så kallat *Human-Machine Interface* eller HMI. I HMI:t ges en grafisk representation av systemet och processen, baserat på den information som samlas in till automationssystemet. Från detta interface ges i regel även möjlighet till operatörer att påverka systemet genom till exempel olika knappar eller så kallade *Faceplates* där de inblandade objekten i systemet kan kontrolleras och påverkas. Även de alarm som finns programmerade i systemet brukar ges visuell representation i systemets HMI.

I de fall en digital tvilling finns och används är denna i regel också kopplad till den information om systemets objekt som finns i automationssystemet. Den digitala tvillingen byggs upp för att efterlikna den verkliga processen i så stor grad som möjligt. Den digitala tvillingen ger då möjlighet att i realtid se hur processen påverkas av olika förändringar i automationssystemet, men även hur automationssystemet påverkas av olika förändringar i processen. Till exempel kan de larm eller givare som ska ge utslag i olika situationer testas genom att den simulerade processen förändras för att motsvara de förhållanden som ska ge upphov till larm. Den digitala tvillingen bygger alltså på ett realtidsutbyte av information både från och till automationssystemet.



Figur 3: Samband mellan HMI, automationssystem, process och digital tvilling.

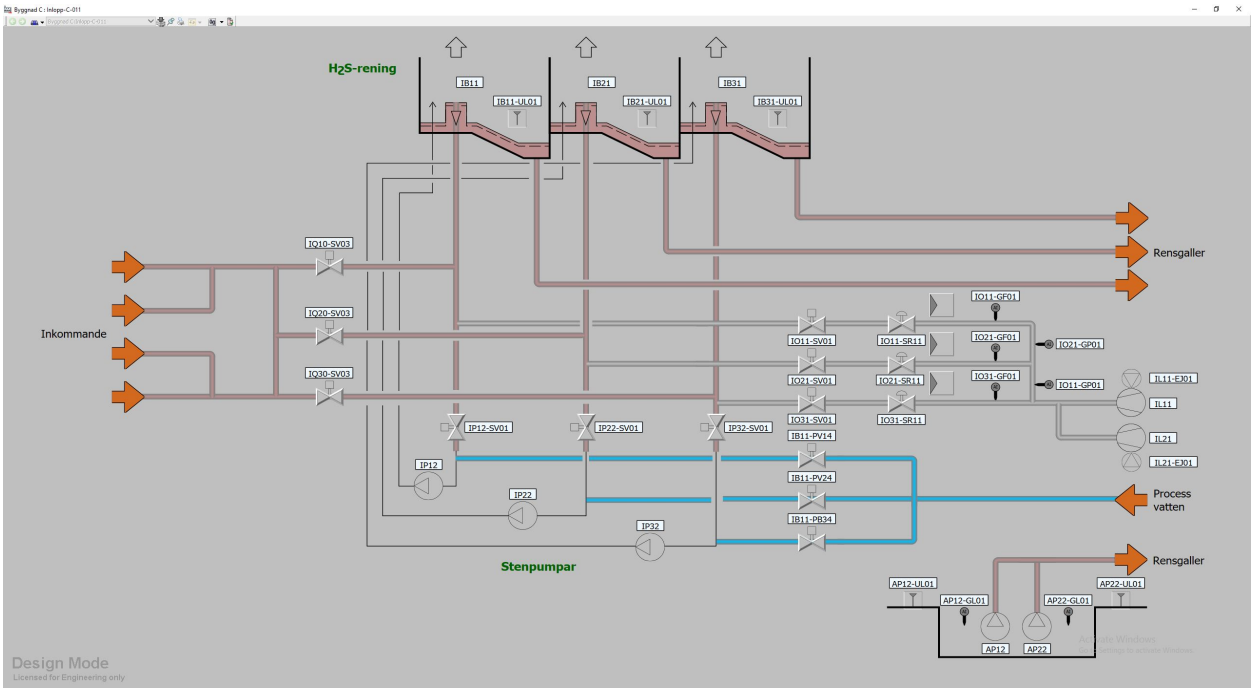
3.2 Arbetsprocess för automatisering

Arbetsprocessen för ett system i likhet med det som används i denna rapport är i regel uppdelad i tre huvudsakliga delar.

1. Processen studeras och planeras.
2. Automationssystemet skapas.
3. Simulering eller digital tvilling byggs upp.

1. Den första delen av arbetet består av att en processingenjör studerar den process som ska styras eller byggas upp. Processingenjören designar därefter hur den önskade processen ska se ut, vilka objekt som ska vara med och hur dessa ska bete sig. Processingenjören måste därför ha god kännedom om den process som ska konstrueras för att kunna designa den på ett fungerande sätt. När processen är uppbyggd på ett sätt som ingenjören tror ska fungera skriver denna en funktionsbeskrivning över systemet där systemets upplägg och funktion beskrivs på ett övergripande sätt.

Funktionsbeskrivningen innehåller även en lista över vilka objekt som finns med i systemet, hur många av respektive objekt som finns, information kring hur objekten är tänkta att fungera och hur de är uppbyggda, samt identifikation för varje objekt. Processingenjören skapar också en så kallad P&ID, ett process- och instrumentdiagram. I detta finns systemet uppritat med de ingående objekten placerade så som de är tänkta att struktureras i det verkliga systemet, med objektens namn eller identifikation utskrivet så att respektive objekt kan matchas med rätt objekt från funktionsbeskrivningen. För större system delas ofta processen upp i ett flertal delsystem, för vilka varsin P&ID görs till respektive del. Ett exempel på hur en P&ID kan se ut visas i figur 4.



Figur 4: Exempel på hur en P&ID för ett inloppssystem till ett avloppsreningsverk kan se ut.

2. Funktionsbeskrivningen och P&ID:en lämnas sedan till en automationsingenjör. Denna konstruerar och programmerar styrningen för systemet. Detta innefattar dels programmering av alla de olika inblandade objektens uppförande, dels hur objekten hänger samman och hur systemet som helhet ska bete sig. Programmeringen sker i någon form av automationssystem som till exempel ABB 800xA eller Siemens TIA Portal, beroende på vilket system som ska användas till processens styrning. Ofta väljs detta av den som står bakom processen, och lämnas till en automationsingenjör som har kunskap inom det önskade systemet.

För ingenjören består de ingående delarna som ska programmeras dels av det normala beteendet för alla objekt, dels hur objekten ska reagera på olika förändringar i systemet samt i vilka fall systemet ska larma och hur dessa larm ska fungera. Programmeringen sker ofta i flera olika nivåer, varav den lägsta i regel är den komponentnivå som är grunden för objektens olika funktioner. På denna nivå programmeras till

exempel även vilka olika IO-signaler som ska finnas till varje objekt och hur dessa ska vara kopplade. Nästa nivå är vanligtvis sambandet mellan de olika ingående delarna i systemet och hur objekten ska samspela och påverka varandra. Hur denna nivå konstrueras och programmeras är huvudsaken i hur processen kommer att fungera och styras.

Den sista nivån är att systemets HMI ska programmeras och grafiken till systemet ska skapas. På denna nivå skapas både grafiken för ingenjören som jobbar med systemet, men utifrån detta styrs även den framtida operatörens HMI. Här kontrolleras vilka delar som operatören ska kunna se och påverka, samt hur larm i systemet ska visas grafiskt. Denna del utgör enkelt utryck kopplingen mellan operatör och system. Alla dessa nivåer måste samverka och fungera på ett bra sätt för att hela systemet ska fungera. Processen är ofta iterativ, och sker ofta i samverkan med processingenjören för att systemets funktioner ska konstrueras på det tilltänkta sättet.

3. I de fall en simulering eller en digital tvilling ska konstrueras görs detta utifrån det uppbyggda automationssystemet samt den kännedom om den tilltänkta processen som finns. Även simuleringen utgår från den P&ID och funktionsbeskrivning som processingenjören har gjort. Från denna byggs en processmodell med de ingående objekten inlagda i den ordningen och på det sätt som de är beskrivna. Därefter skapas mallar, eller templates, för de olika typer av objekt som finns med i systemet, där objektens grundfunktion styrs samt vilka in- och utsignaler som ska gå att koppla till objektstypen. Hur dessa kan se ut visas i avsnitt *3.4.1*. Från dessa templates skapas sedan listor på de objekt som faktiskt ska ingå i systemet, med unika namn inlagda samt alla de specifika IO-adresser som ska kopplas till respektive objekt. Utifrån dessa listor, som med fördel kan exporteras till och importeras från Excel genereras sedan så kallad *hardware*, eller *HW*, till simuleringsprogrammet som i detta arbete och denna rapport alltså är SIMIT. Dessa objekt kopplas sedan ihop med respektive objekt i processmodellen genom att HW-objekten och processobjekten döps till motsvarande namn, med eller utan prefixet *HW_* beroende på vilken del av strukturen objektet kommer ifrån.

Beroende på vilken typ av koppling mellan automationssystem och simuleringsprogram som ska användas krävs olika typer av importer. Det vanligaste när system av olika tillverkare används är att en OPC-koppling används. Till simuleringsprogrammet importeras då också en lista på alla de OPC-adresser där de IO-signaler som används finns listade. Simuleringsprogrammet letar då via en OPC-server efter de IO-adresser som finns listade i systemet och kopplar sedan dessa till de objekt där de är inlagda i den genererade HW:n. Övriga objekt som påverkar processmodellen, till exempel så kallade trycknoder och andra objekt som fungerar för att konstruera slutna system anpassas därefter för att försöka efterlikna den verkliga processen i så stor utsträckning som möjligt.

Denna sistnämnda process är det som utgör grunden för arbetet som gjorts i detta projekt. Mer om denna arbetsprocess och hur det är gjort för detta specifika projekt visas senare i denna rapport i avsnitt *3.4*.

3.3 Automationssystem - struktur

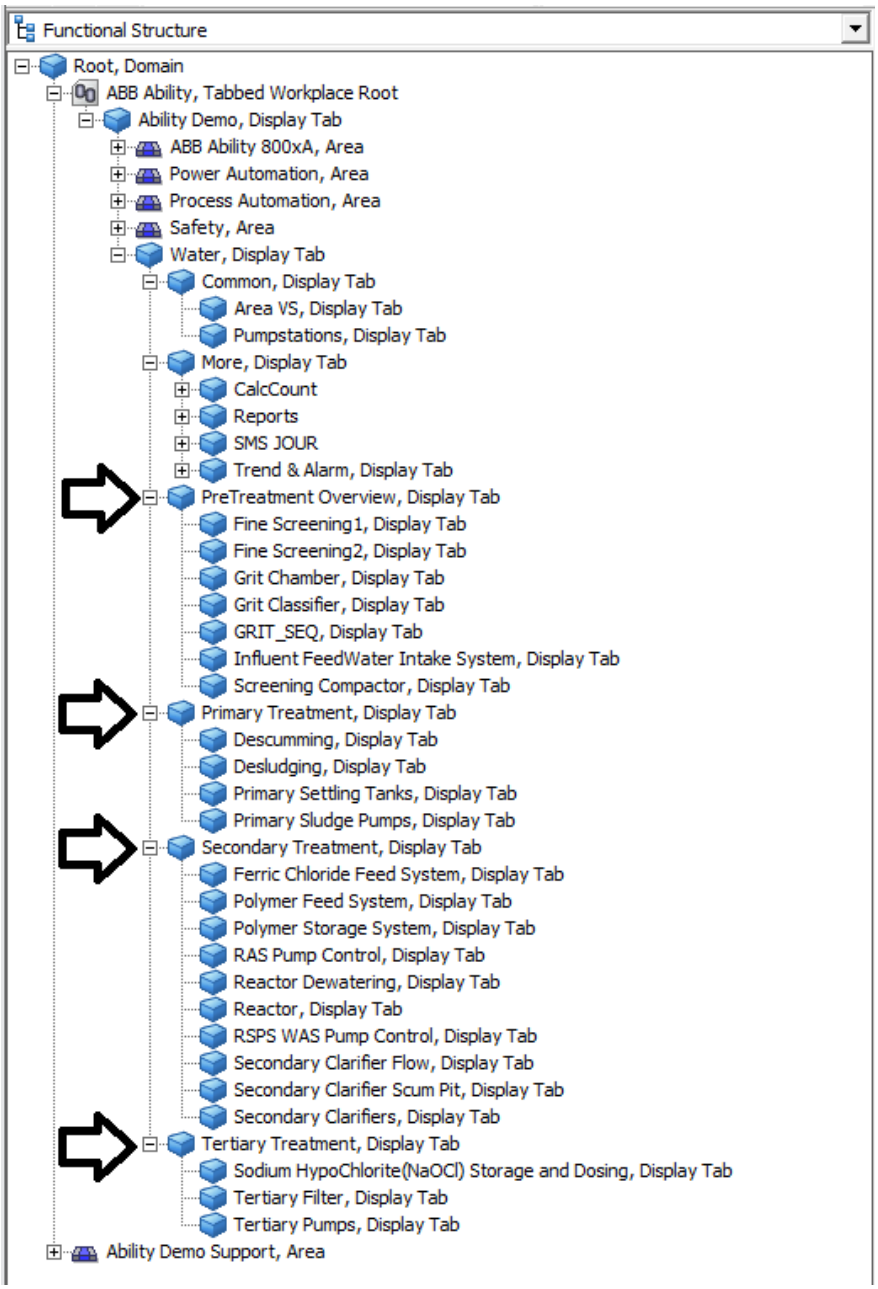
Automationssystem kan struktureras på många olika sätt och en mängd olika gränssnitt för uppbyggnad av dessa existerar. Som tidigare nämnts fokuserar detta arbete på system uppbyggda i ABB:s system 800xA. Ett exempel på hur ett sådant system kan vara uppbyggt visas i följande avsnitt.

3.3.1 ABB 800xA

Automationssystemet för detta projekt är som tidigare nämnts uppbyggt i ABB:s system 800xA. Innan det aktuella systemet studeras i mer detalj kommer ett exempelsystem från ABB att studeras för att visa på hur ett automationssystem kan vara uppbyggt i 800xA. Exempelsystemet är uppbyggt av flera olika delar som visar upp olika applikationer. Strukturen för projektet kan ses i projektets funktionella struktur enligt figur 5. Uppdelningen för den del av demoprojektet som rör VA-processer kan ses strukturerade under samlingsfliken *Water*. Under denna finns strukturen för ett tänkt avloppsreningsverk uppdelad enligt följande.

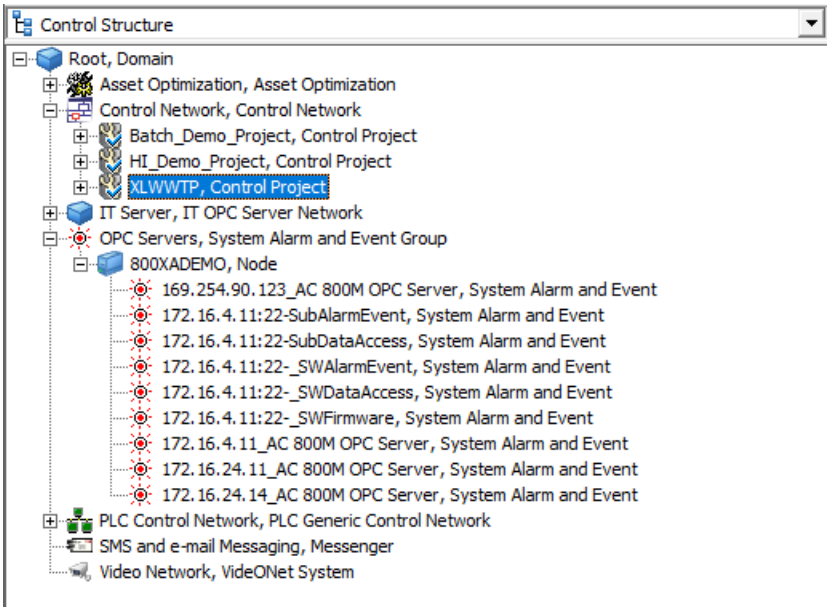
- *PreTreatment* - Förbehandling, med inlopp och grovrening.
- *Primary Treatment* - Primärbehandling, med sedimentering och slampumpar.
- *Secondary Treatment* - Sekundärbehandling, med kemisk och biologisk rening.
- *Tertiary Treatment* - Tertiärbehandling, med rening av mikroskopiska partiklar.

Under dessa flikar kan den grafiska designen för de ingående delarna ses, både som överblick där det finns en så kallad *Display Tab* för huvudfliken, men även de ingående delarna som till exempel *Influent FeedWater Intake System* som finns under fliken för *PreTreatment*.



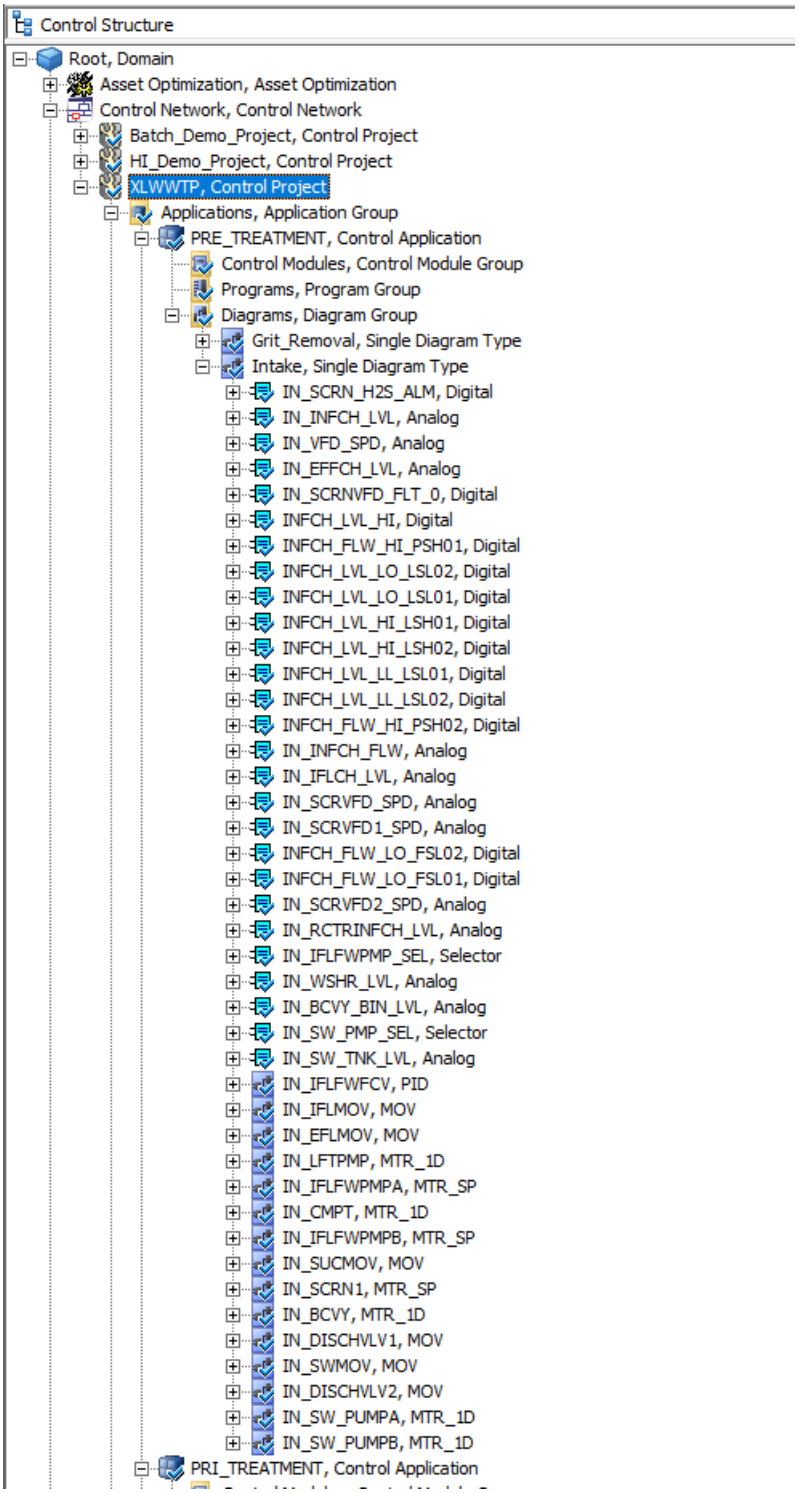
Figur 5: Funktionell struktur för systemet.

Om automationssystemets struktur ska studeras i mer detalj kan istället kontrollstrukturen väljas. I denna syns de bitar som ligger bakom styrningen av systemet. Även OPC servrarna går att hitta under denna del, vilket är relevant i de fall där en simulering ska göras, som till exempel för kopplingen mellan 800xA och SIMIT senare i projektet. Den del av demosystemet som är uppbyggt kring ett tilltänkt avloppsreningsverk ligger under *Control Network*-fliken och är döpt till *XLWWTP*, vilket kan ses i figur 6.

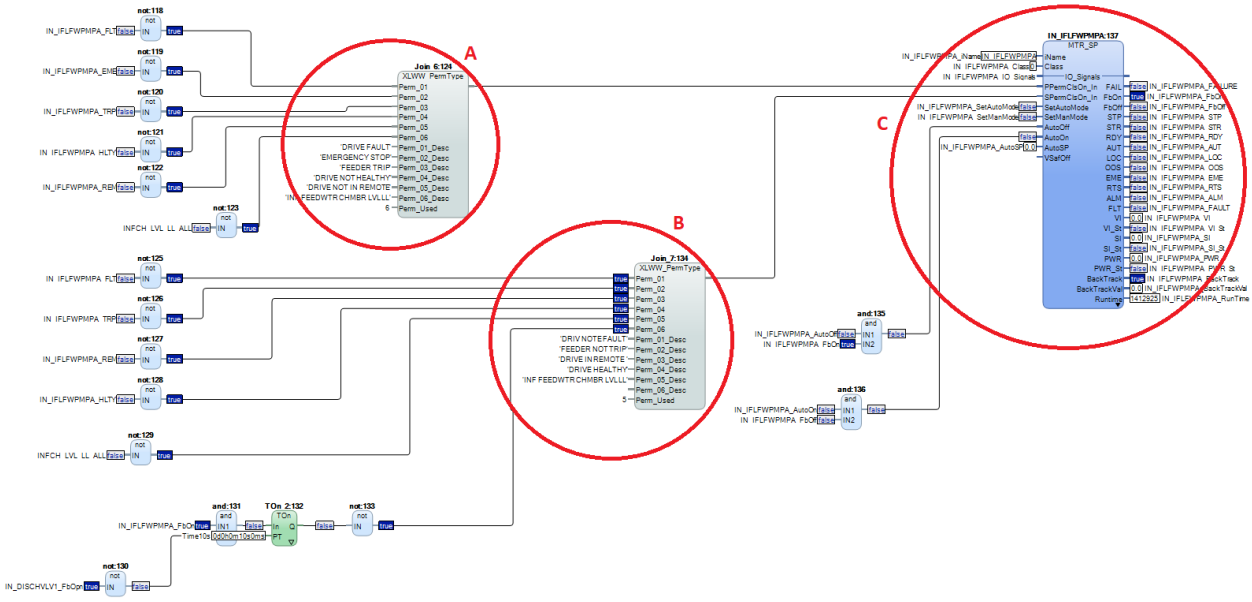


Figur 6: Kontrollstruktur för systemet.

Om denna flik expanderas kan strukturen för programmet studeras i större detalj. Även i kontrollstrukturen är projektet strukturerat enligt samma mönster som tidigare, där de olika ingående delarna *PreTreatment*, *Primary Treatment*, *Secondary Treatment* och *Tertiary Treatment* ligger som egna flikar under systemet. Vidare kan man under dessa flikar välja att gå djupare in i de ingående delarna, som till exempel *Intake* under fliken för *PreTreatment*, vilket kan ses i figur 7. Under denna kan de diagram som styrningens logik är upplagd efter hittas för alla de ingående objekten. Dessa diagram kan i sin tur expanderas för att studeras i mer detalj, vilket kan ses i figur 8.

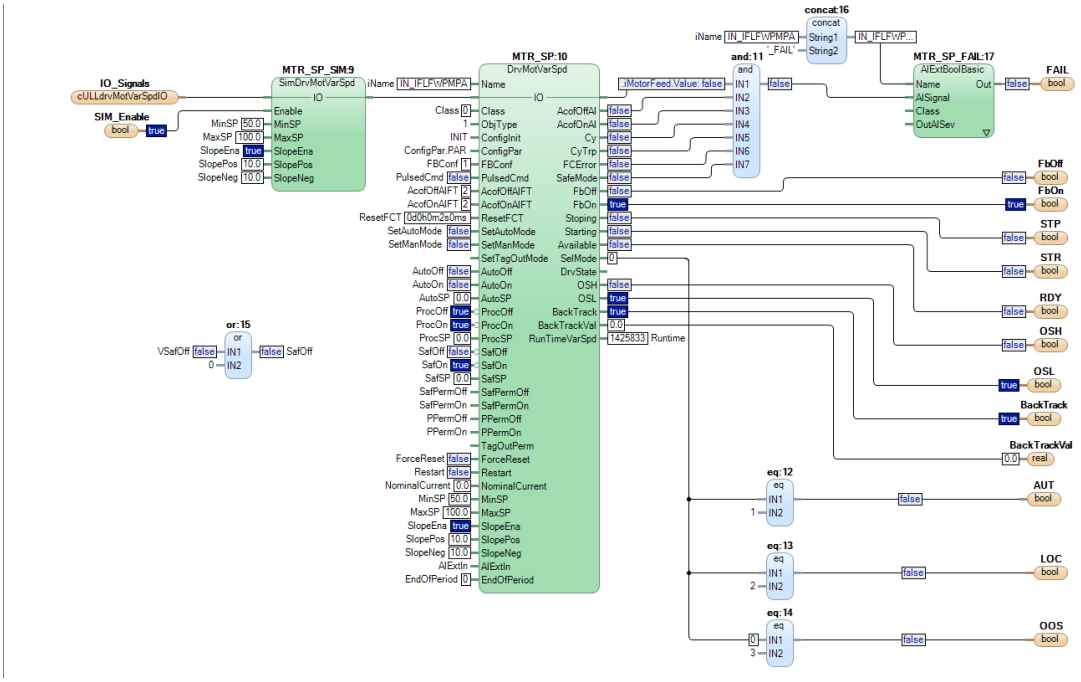


Figur 7: Expanderad kontrollstruktur för systemet. *PRE_ TREATMENT* kan ses expanderad med ingående objekt innan nästa flik *PRI_ TREATMENT* syns längst ner i figuren.



Figur 8: Diagram för en av pumparna i systemet (struktur). Blocken markerade med *A* och *B* hanterar signaler kopplade till andra objekt, medan blocket markerat med *C* hanterar pumpens interna logik. Denna kan ses expanderad i figur 9.

Från diagrammet som kontrollerar pumpens styrning kan en så kallad *Editor* öppnas där styrningen till motorn som i praktiken kontrollerar pumpen öppnas. I denna kan styrningen av motorn och därigenom pumpen ses med större noggrannhet. Ett sådant diagram för en av de motorer som styr en av pumparna i demosystemet kan ses i figur 9. I denna expanderade struktur kan även alla de signaler och variabler som finns i denna del studeras i sin helhet. Till exempel kan de ingående IO-signaler som rör det aktuella objektet expanderas och studeras, vilket även detta är relevant i de fall där en koppling mot till exempel SIMIT ska göras. Denna lista över signaler, med IO-signalerna expanderade, kan ses i figur 10.



Figur 9: Diagram för motorn som kontrollerar en av pumparna i systemet (struktur). Denna struktur är expanderad från C-blocket i figur 8.

Name	Current Value	Data Type	Variable	Start Attribute	Attributes	Direction	Initial Value	I/O Address	I/O Description	Access Variables	Description
iName	IN_IFLFWPMPA	string[20]	IN_IFLFWPMPA_iName			in	'NameM2'				Objet Name
Class	0	dint	IN_IFLFWPMPA_Class			in	1				EDIT Object Alarm class
IO_Signals		cULLdrvMotVarSpdIO	IN_IFLFWPMPA_IO_Signals			in_out					IN/OUT: IO datatype for object
iFbOff		BoolIO	IN_IFLFWPMPA_IO_Signals.iFbOff	retain							IN feedback for Off
Value	false	bool	IN_IFLFWPMPA_IO_Signals.iFbOff.Va	retain	displayvalue						Value in the application
IOValue	false	bool	IN_IFLFWPMPA_IO_Signals.iFbOff.IO\	retain							Value from I/O before forcing
Forced	true	bool	IN_IFLFWPMPA_IO_Signals.iFbOff.Fo	retain							Tells if the input is forced or not
Status	16#C0	dword	IN_IFLFWPMPA_IO_Signals.iFbOff.St	retain			16#00C0				Error status
iFbOn	true	BoolIO	IN_IFLFWPMPA_IO_Signals.iFbOn	retain							IN feedback for On
iLocalCtlMode	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iLocalCtlI								IN selection mode true=Local, false=Remote (DCS)
iMCCFeed	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iMCCFee								IN MCC operating voltage error
iMotorFeed	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iMotorFe								IN MCC error
iFCReady	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iFCReady								IN FC (drive) ready
iFCError	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iFCError								IN FC (drive) error
iCurrent	0.0	RealIO	IN_IFLFWPMPA_IO_Signals.iCurrent								IN Motor current
iFbMV	50.0	RealIO	IN_IFLFWPMPA_IO_Signals.iFbMV	retain							IN Actual motor speed feedback from field
iPanPOC	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iPanPOC								IN Local panel point of control active
iPanOff	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iPanOff								IN Local panel command to Off state
iPanOn	false	BoolIO	IN_IFLFWPMPA_IO_Signals.iPanOn								IN Local panel command to On state
iPanSP	0.0	RealIO	IN_IFLFWPMPA_IO_Signals.iPanSP								IN Local panel for speed reference
oCmdOff	false	BoolIO	IN_IFLFWPMPA_IO_Signals.oCmdOff								OUT On command to MCC
oCmdOn	true	BoolIO	IN_IFLFWPMPA_IO_Signals.oCmdOn								OUT Off command to MCC
oCmdSP	50.0	RealIO	IN_IFLFWPMPA_IO_Signals.oCmdSP								OUT Setpoint command to MCC
oResetFC	false	BoolIO	IN_IFLFWPMPA_IO_Signals.oResetFC								OUT Reset FC command to MCC

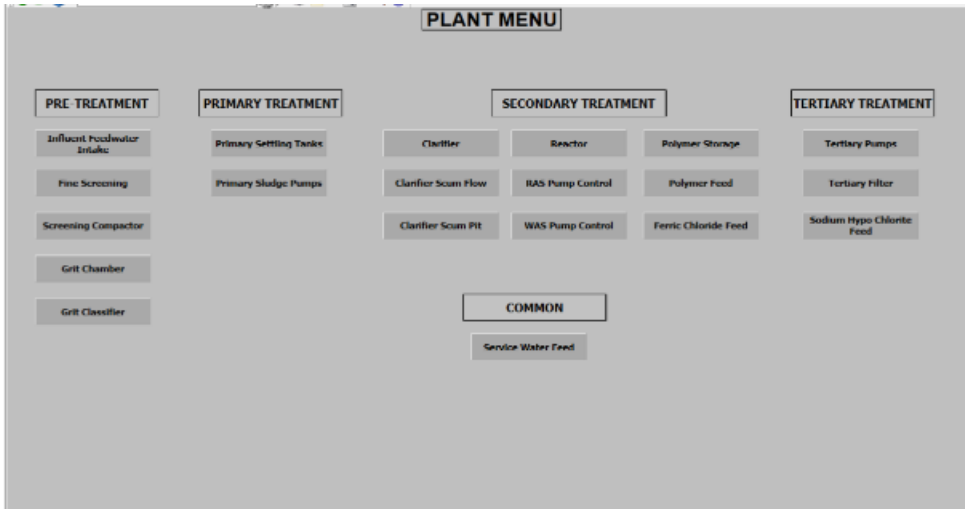
Figur 10: Signallista för motorn som kontrollerar en av pumparna i systemet.

På detta sätt är samtliga objekt i systemet uppbyggda, samt logiken som styr interaktionen mellan de ingående objekten. Dessa diagram kan variera i storlek eller komplexitet beroende på vilken typ av objekt som kontrolleras.

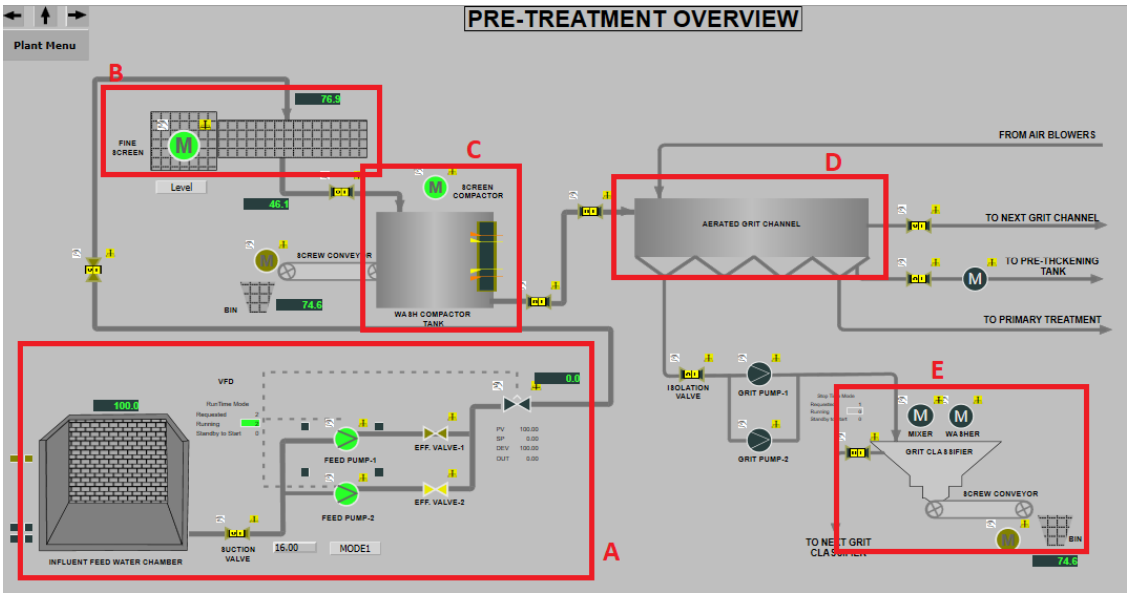
3.3.2 Automationssystem - processbeskrivning

Som tidigare nämnts är detta exempel ett automationssystem för styrningen av ett tilltänkt avloppsreningsverk. Övergripande består verket av en förbehandling, en primärbehandling, en

sekundärbehandling och en tertiärbehandling, vilket kan ses i figur 11. I det som i detta system kallas för förbehandling sker grovreningen genom det första intaget av vattnet in i systemet (Fig. 12, block A). Därefter går vattnet vidare till ett finare filter för ytterligare en silning (Fig. 12, block B). Vidare passerar sedan vattnet genom ett sandfilter (Fig. 12, block C) innan flödet delas upp, där majoriteten av flödet går vidare till primärbehandlingen (Fig. 12, block D). Skräp som har fastnat i de olika filtren sorteras bort från processen och sand och grus som har försvunnit skickas vidare till sortering (Fig. 12, block E). Ytterligare en del av vattnet skickas vidare till ännu ett sandfilter. Denna process visas i en överblick i figur 12.



Figur 11: Övergripande vy över verkets ingående delar.

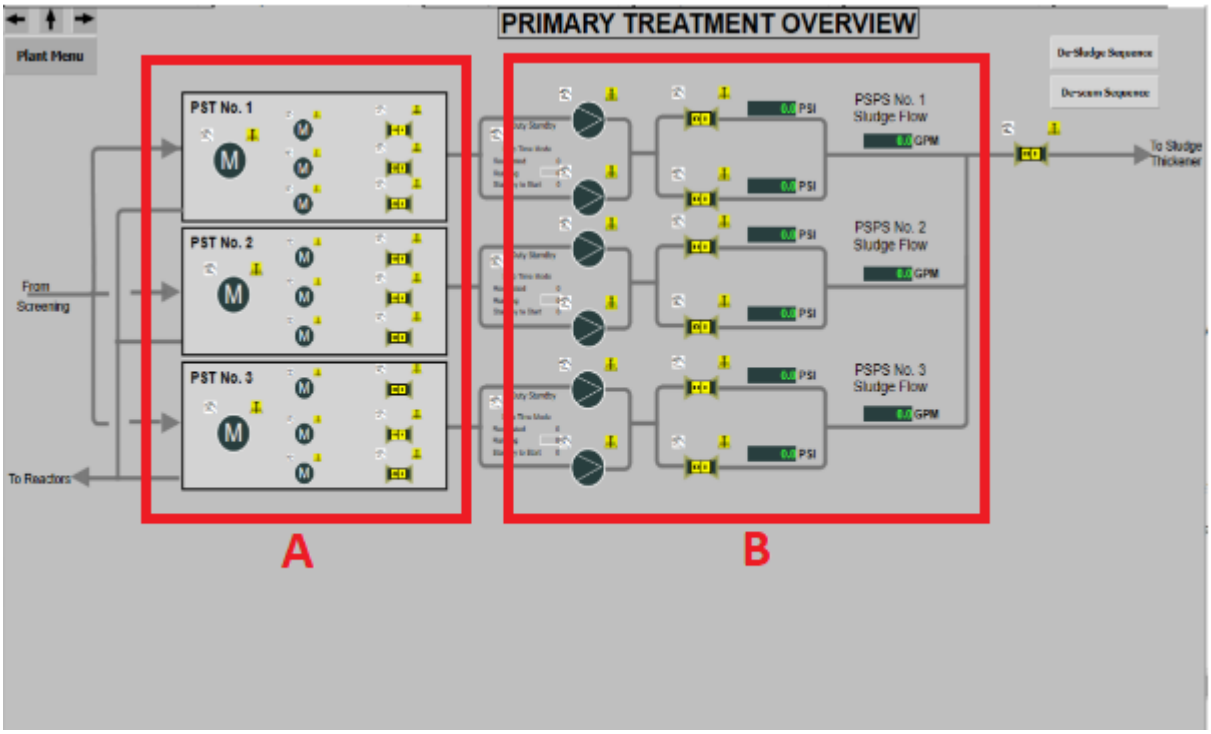


Figur 12: Ingående delar i förbehandlingen. Från bilden kan inloppet ses i block A, nästa filter i block B, *Wash Compactor* i block C, sandfiltret i block D och slutligen sorteringen av sten och grus i block E.

Vattnet passerar som tidigare nämnts vidare till primärbehandlingen. Denna del består av sedimenteringstankar (Fig. 13, block A) och slampumpar (Fig. 13, block B). Precis som i förbehandlingen fokuserar denna fas på att avlägsna grova partiklar och andra större föroreningar från vattnet innan det går vidare till mer avancerade reningstekniker.

Sedimenteringstankarna utgör en viktig del av primärbehandlingen vid ett avloppsreningsverk. Dessa tankar är utformade för att tvinga vattnet att sakta ner och möjliggöra avskiljning av tunga partiklar och sediment som finns i det inkommande vattnet. När vattnet strömmar in i sedimenteringstankarna minskar flödeshastigheten dramatiskt, vilket får partiklarna att sjunka till botten av tankarna genom gravitation, så kallad sedimentering. De avskilda partiklarna och sedimenten bildar ett slamlager på tankens botten, medan det renade vattnet fortsätter vidare till nästa steg i reningsprocessen.

Efter att sedimenteringstankarna har avlägsnat större partiklar och sediment från råvattnet eller avloppsvattnet krävs det att det samlade slammet transporteras vidare för vidare behandling. Till detta används så kallade slampumpar. Dessa pumpar är utformade för att suga upp och pumpa det avskilda slammet från botten av sedimenteringstankarna och transportera det till vidare behandling. En överblick över primärbehandlingen kan ses i figur 13. Från primärbehandlingen transporteras vattnet vidare till den så kallade sekundärbehandlingen.

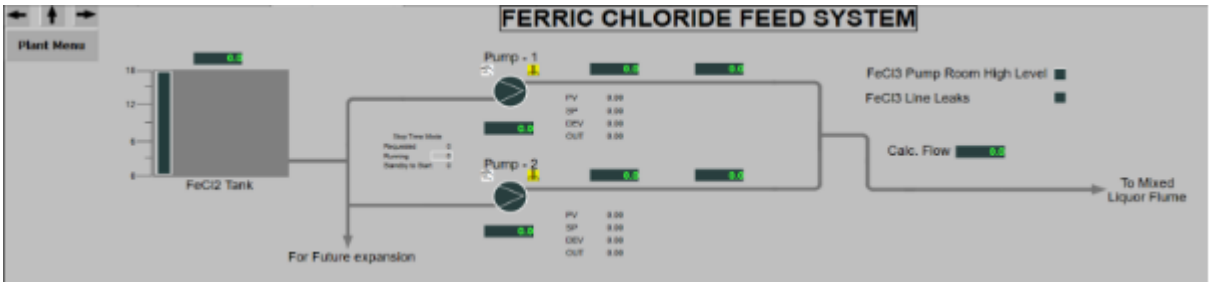


Figur 13: Överblick över primärbehandlingen.

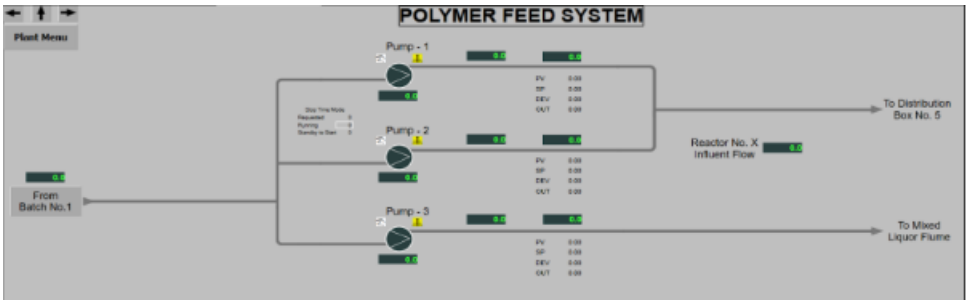
Sekundärbehandlingen utgör alltså som sagt nästa steg i det tilltänkta reningsverket och är tänkt att ytterligare rena vattnet från organiskt material och andra föroreningar som inte avlägsnades under primärbehandlingen. Denna fas omfattar flera komplexa processer och system som samverkar för att säkerställa att vattnet når önskad renhetsgrad innan det släpps ut i miljön. Några av dessa processer är följande.

Järnkloridmatningssystem och Polymermatningssystem:

Järnklorid och polymerer används ofta i sekundärbehandlingen för att underlätta avskiljning av fina partiklar och organiskt material från vattnet. Järnkloridmatningssystemet som kan ses i figur 14 injicerar järnklorid i vattnet för att underlätta fosfatavskiljning, medan polymermatningssystemet som kan ses i figur 15 tillför polymerer som bidrar till flockning av små partiklar för enklare avskiljning i efterföljande steg.



Figur 14: Sekundärbehandling, järnkloridmatningssystem.



Figur 15: Sekundärbehandling, polymermatningssystem.

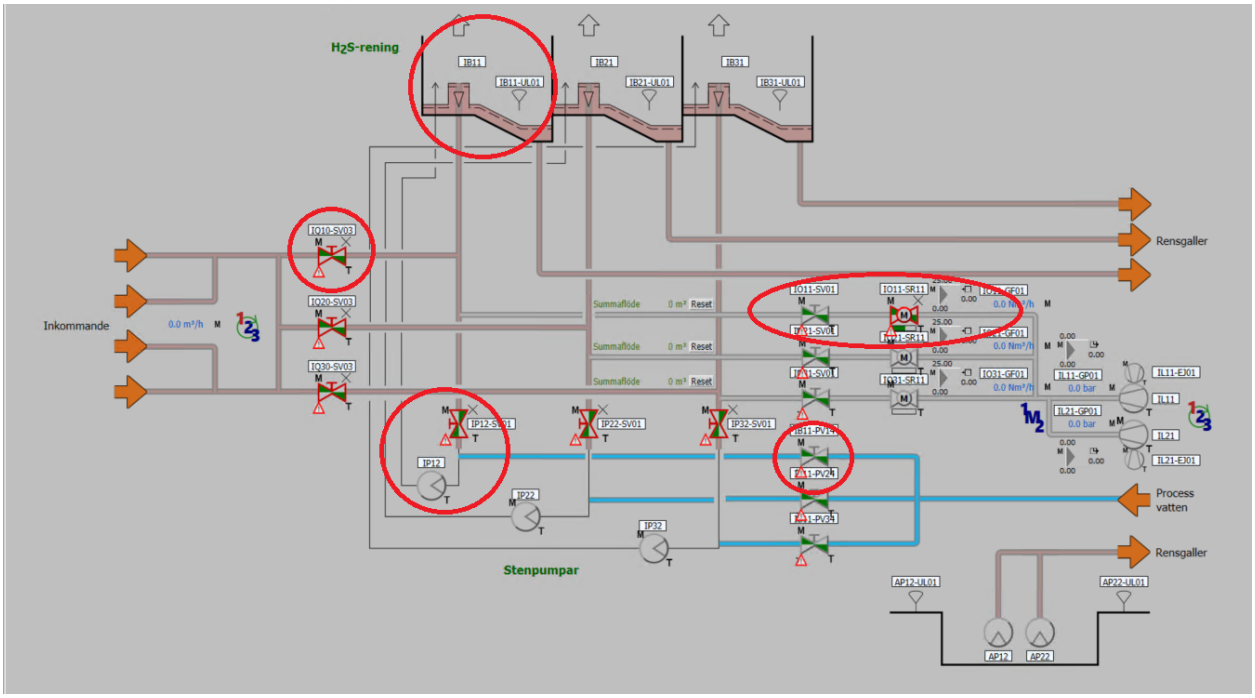
Processen fortsätter vidare till resterande steg. Verket i sin helhet kan studeras i bilaga A.

3.3.3 Utvald process

Den process som valts ut för att utgöra automationssystemet som senare i arbetet kommer att kopplas till den simulerade miljön är ett inloppssystem från ett verkligt avloppsreningsverk. Detta system motsvarar alltså ungefär *Feedwater Intake*-systemet från ABB:s exempel. Reningsverket i sig består av flera andra processer och anläggningar, men den utvalda processen finns i en separat del av verkets anläggningar och ser ut enligt följande.

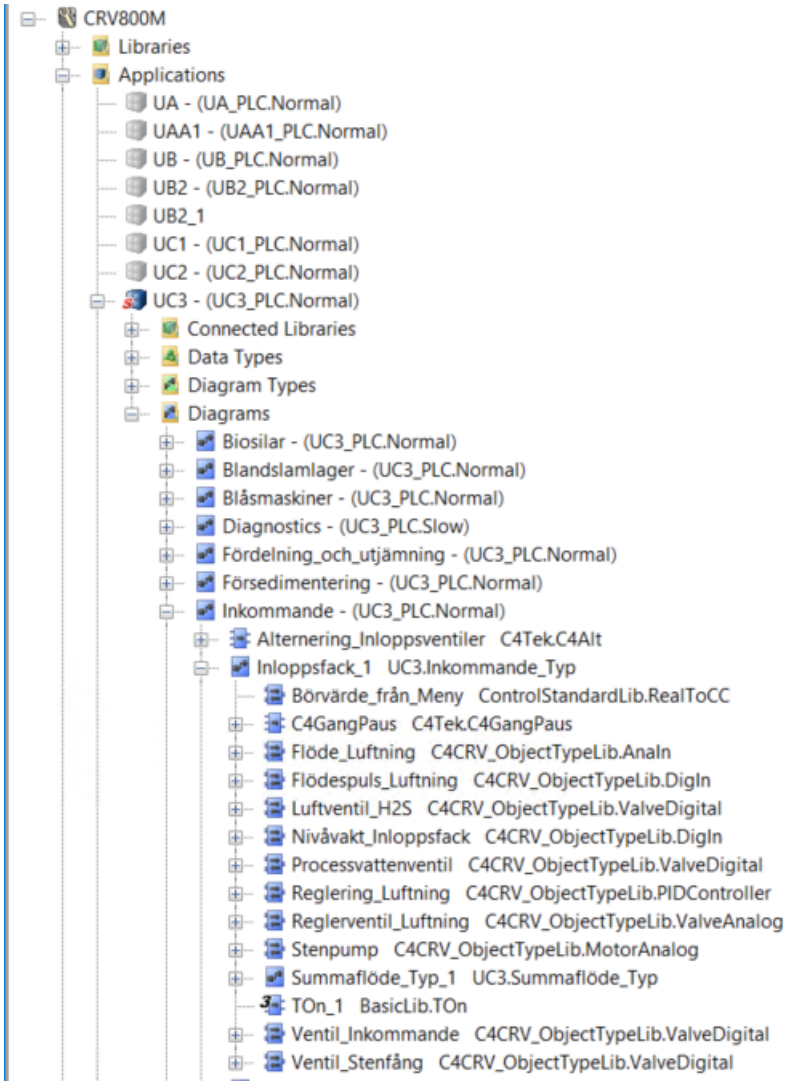
Avloppsvatten från nätet pumpas till reningsverket från två intagspumpstationer, via fyra separata ledningar. Alldeles före inloppet till reningsverket finns en kammare med automatiska ventiler, som styr och fördelar avloppsvattnet från de inkommande ledningarna till tre inloppsledningar, så att inte alla används vid lågflöden. På så sätt hålls flödeshastigheten uppe och därmed minskar risken för ansamling av sand och grus. Inloppsledningarna vänder uppåt i ett inloppstorn. I botten, där rören vinklar uppåt, finns ett ventilarrangemang som släpper uppfångad sten och grus från de tre ledningarna ner till varsin stenpump. Pumparna pumpar sedan gruset till toppen av tornet. På varje inloppsledning finns en ring av luftmunstycken, där luft blåses in från blåsmaskiner som står i tornets bottenvåning. Syftet är att driva av vätesulfid (H_2S, ”svavelväte”) från vattnet. I toppen av tornet finns kammare, en för varje ledning.

Avloppsvattnet flödar som en fontän genom rörens mynningar, vilket ytterligare bidrar till avdrivningen. Ovanför kammaren finns ytterligare en våning med utrustning för utsug av luften och katalytisk förbränning av vätesulfiden därifrån. Ventilation och förbränning beskrivs inte i det här dokumentet. Från inloppskammaren i tornets övre del flödar vattnet vidare in till gallerstationen. I normal drift bildar inloppet ett vattenlås som håller den vätesulfidhaltiga luften på torn-sidan. En översikt över inloppsystemet kan ses i figur 16.



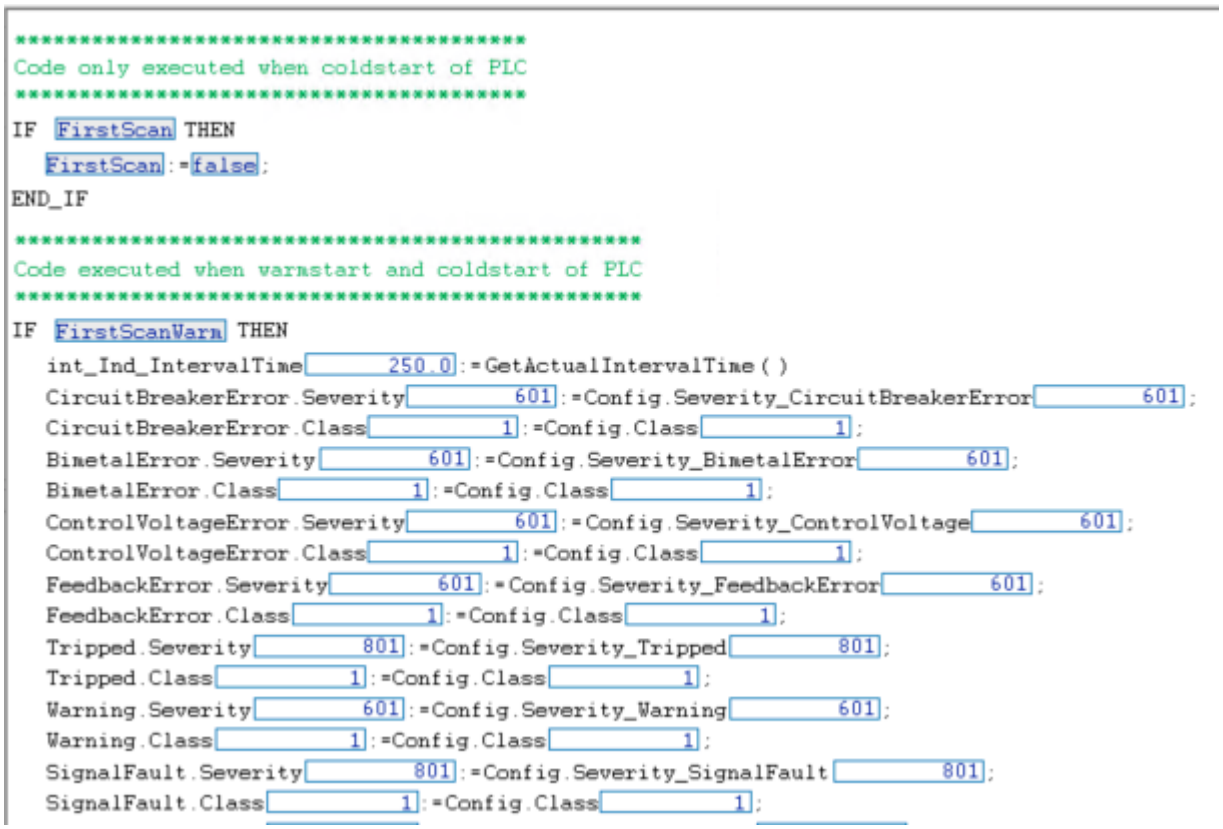
Figur 16: Det utvalda inloppssystemet som det ser ut i ABB 800xA. Markerat är de objekt som sorterar under inloppsfack 1.

De objekt som innefattas i systemet är strukturellt fördelade i tre olika inloppsfack. Dessa fack ser likadana ut och innefattar en luftventil, en processvattenventil, en reglerventil för luftning, en stenpump, en ventil för inkommande vatten, en ventil kopplad till stenfånget samt ett antal digitala eller analoga signaler. Strukturen för inloppsfack 1 kan ses i figur 17. En anledning till denna struktur är att inloppsfack 1 och 2 styrs av en annan PLC än vad inloppsfack 3 gör. Hela verket är upplagt på detta sätt, vilket är tänkt att skapa en viss redundans i systemet, även om inte total redundans kan uppnås. I strukturen kan även objekttyperna för de respektive objekten utläsas, till exempel kan stenpumpen ses vara av objekttypen MotorAnalog. En template för varje sådan typ kommer senare att behöva skapas i SIMIT för att styra beteendet av de ingående objekten. Denna uppbyggnad kan ses i bilaga B.



Figur 17: Strukturen för det utvalda inloppssystemet. Objekten som sorterar under inloppsfack 1 kan ses markerade i figur 16.

Om ett objekt från strukturen öppnas och studeras med större noggrannhet, till exempel den tidigare nämnda stenpumpen, syns den kod som styr objektet. Ett sådant exempel kan ses i figur 18. Koden är i detta fall gjord i strukturerad text, och den del av koden som syns i figuren styr vad som ska ske i samband med en kallstart av den aktuella PLC:n.



```
****************************************
Code only executed when coldstart of PLC
****************************************
IF  FirstScan THEN
   FirstScan:=false;
END_IF
************************************************
Code executed when varmstart and coldstart of PLC
************************************************
IF  FirstScanWarm THEN
   int_Ind_IntervalTime      250.0:=GetActualIntervalTime()
   CircuitBreakerError.Severity      601:=Config.Severity_CircuitBreakerError      601;
   CircuitBreakerError.Class      1:=Config.Class      1;
   BimetalError.Severity      601:=Config.Severity_BimetalError      601;
   BimetalError.Class      1:=Config.Class      1;
   ControlVoltageError.Severity      601:=Config.Severity_ControlVoltage      601;
   ControlVoltageError.Class      1:=Config.Class      1;
   FeedbackError.Severity      601:=Config.Severity_FeedbackError      601;
   FeedbackError.Class      1:=Config.Class      1;
   Tripped.Severity      801:=Config.Severity_Tripped      801;
   Tripped.Class      1:=Config.Class      1;
   Warning.Severity      601:=Config.Severity_Warning      601;
   Warning.Class      1:=Config.Class      1;
   SignalFault.Severity      801:=Config.Severity_SignalFault      801;
   SignalFault.Class      1:=Config.Class      1;
```

Figur 18: Delar av koden för ett objekt, en stenpump, i det utvalda inloppssystemet.

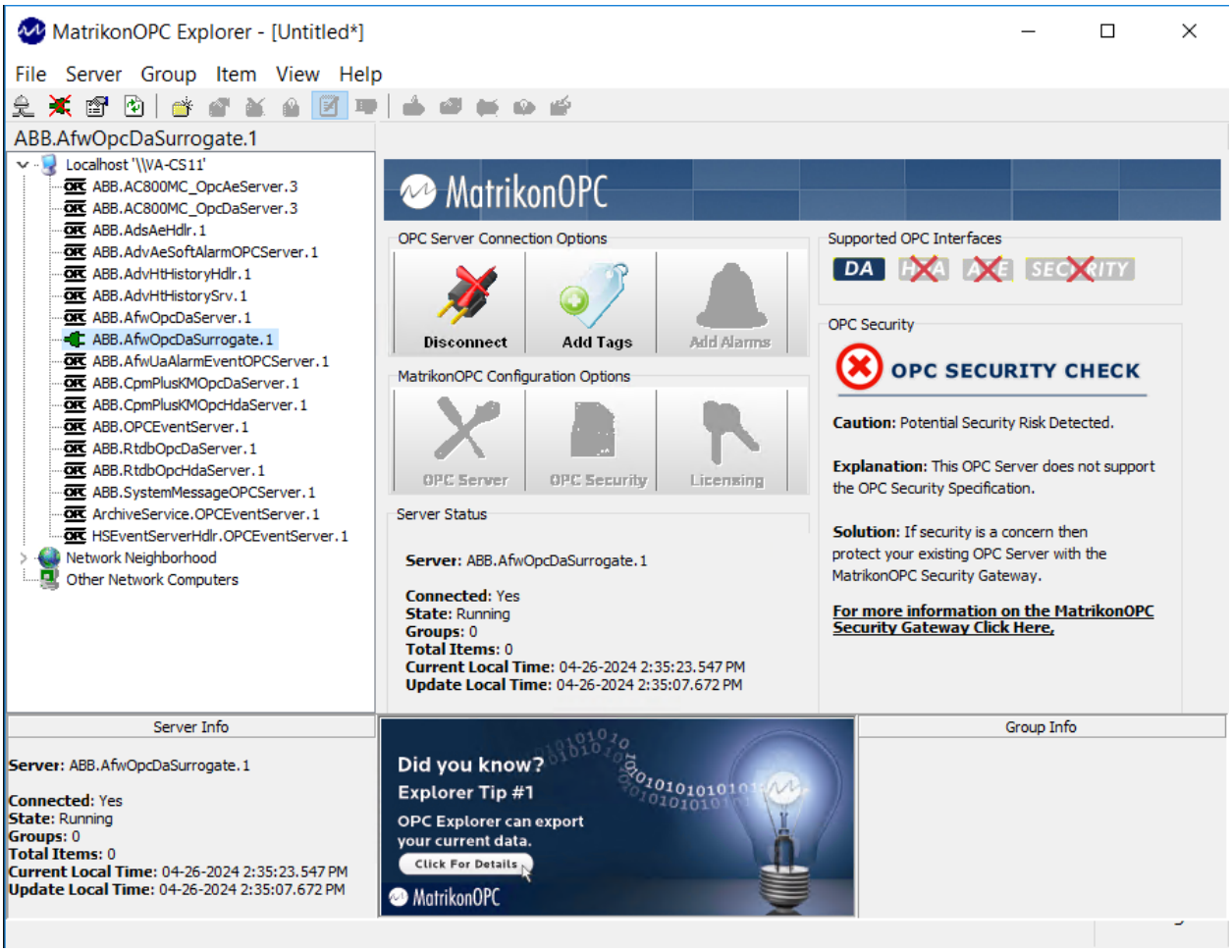
Vidare när objektet öppnas kan de olika IO-signaler som är kopplade till objektet ses i en lista överst i fönstret. Denna lista för en av stenpumparna kan ses i figur 19. Som kan ses i listan under fliken IO finns en stor mängd både IN-, och OUT-signaler uppskrivna, dock är endast ett mindre antal av dessa kopplade till faktiska signaler på IO-kortet. Vilka av dessa det är kan ses utifrån att dessa har IO-adresser i fältet näst längst ut till höger i figuren. Dessa signaler, de med faktisk IO-koppling, är det som kommer att behöva kopplas till SIMIT-objekt och hanteras genom OPC-kopplingen.

Name	Current Value	Data Type	Variable	Start Attribute	Attributes	Direction	Initial Value	I/O Address	I/O Description
Name	IP12	string[30]	stenpump_name			in			
Description	Stenpump inloppsfack 1	string[50]	stenpump_desc			in	DEFAULT		
IN	0.0	ControlConnection	<Stenpump.IN>			in	DEFAULT		
IO		IO_MotorAnalog_Type	stenpump_io			in_out	DEFAULT		
IN_Bimetal	false	BoolIO	stenpump_io.IN_Bime		hidden			UC3_PLC.1.:	IP12 Larm överh
IN_CircuitBreaker	false	BoolIO	stenpump_io.IN_Circu		hidden			UC3_PLC.1.:	IP12 Säkerhetsb
IN_Current	0.0	RealIO	stenpump_io.IN_Curr						
IN_FreqconvError	false	BoolIO	stenpump_io.IN_Freq		hidden			UC3_PLC.1.:	IP12 Summalarn
IN_Local	false	BoolIO	stenpump_io.IN_Loca		hidden				
IN_LocalBackwardSelected	false	BoolIO	stenpump_io.IN_Loca						
IN_LocalForwardSelected	false	BoolIO	stenpump_io.IN_Loca						
IN_LocalStartNormal	false	BoolIO	stenpump_io.IN_Loca		hidden				
IN_LocalStartSlow	false	BoolIO	stenpump_io.IN_Loca		hidden				
IN_Power	0.0	RealIO	stenpump_io.IN_Pow					UC3_PLC.1.:	IP12 Effekt
IN_ReadyToRun	false	BoolIO	stenpump_io.IN_Reac		hidden				
IN_Remote	false	BoolIO	stenpump_io.IN_Rem		hidden				
IN_RunningBackward	false	BoolIO	stenpump_io.IN_Runr						
IN_RunningForward	false	BoolIO	stenpump_io.IN_Runr					UC3_PLC.1.:	IP12 Driftsignal s
IN_Speed	0.0	RealIO	stenpump_io.IN_Spe						
IN_Stopped	false	BoolIO	stenpump_io.IN_Stop						
IN_SwitchgearFault	false	BoolIO	stenpump_io.IN_Swit		hidden			UC3_PLC.1.:	IP12 driftsvar mo
IN_Temp1	0.0	RealIO	stenpump_io.IN_Temp						
IN_Temp2	0.0	RealIO	stenpump_io.IN_Temp						
IN_Temp3	0.0	RealIO	stenpump_io.IN_Temp						
IN_TorqueFault	false	BoolIO	stenpump_io.IN_Torq		hidden				
IN_Tripped	false	BoolIO	stenpump_io.IN_Tripp		hidden				
IN_UnitStatus	0	dint	stenpump_io.IN_UnitS	retain	hidden		0		
IN_Warning	false	BoolIO	stenpump_io.IN_Warr		hidden				
OUT_LocalLamp	false	BoolIO	stenpump_io.OUT_Lc		hidden				
OUT_OrderSpeed	0.0	RealIO	stenpump_io.OUT_Or		hidden			UC3_PLC.1.:	IP12 Stenpump
OUT_OrderStartBackward	false	BoolIO	stenpump_io.OUT_Or		hidden				
OUT_OrderStartForward	false	BoolIO	stenpump_io.OUT_Or		hidden			UC3_PLC.1.:	IP12 Startsignal
OUT_OrderStop	false	BoolIO	stenpump_io.OUT_Or		hidden				
OUT_Reset	false	BoolIO	stenpump_io.OUT_Re		hidden				
SO		SO_MotorAnalog_Type	stenpump_so			in_out			
BUS		BUS_MotorAnalog_Type	<Stenpump.BUS>			in_out	DEFAULT		
Config		Config_MotorAnalog_Type	<Stenpump.Config>			in	DEFAULT		

Figur 19: IO-signaler för ett objekt, en stenpump, i det utvalda inloppssystemet.

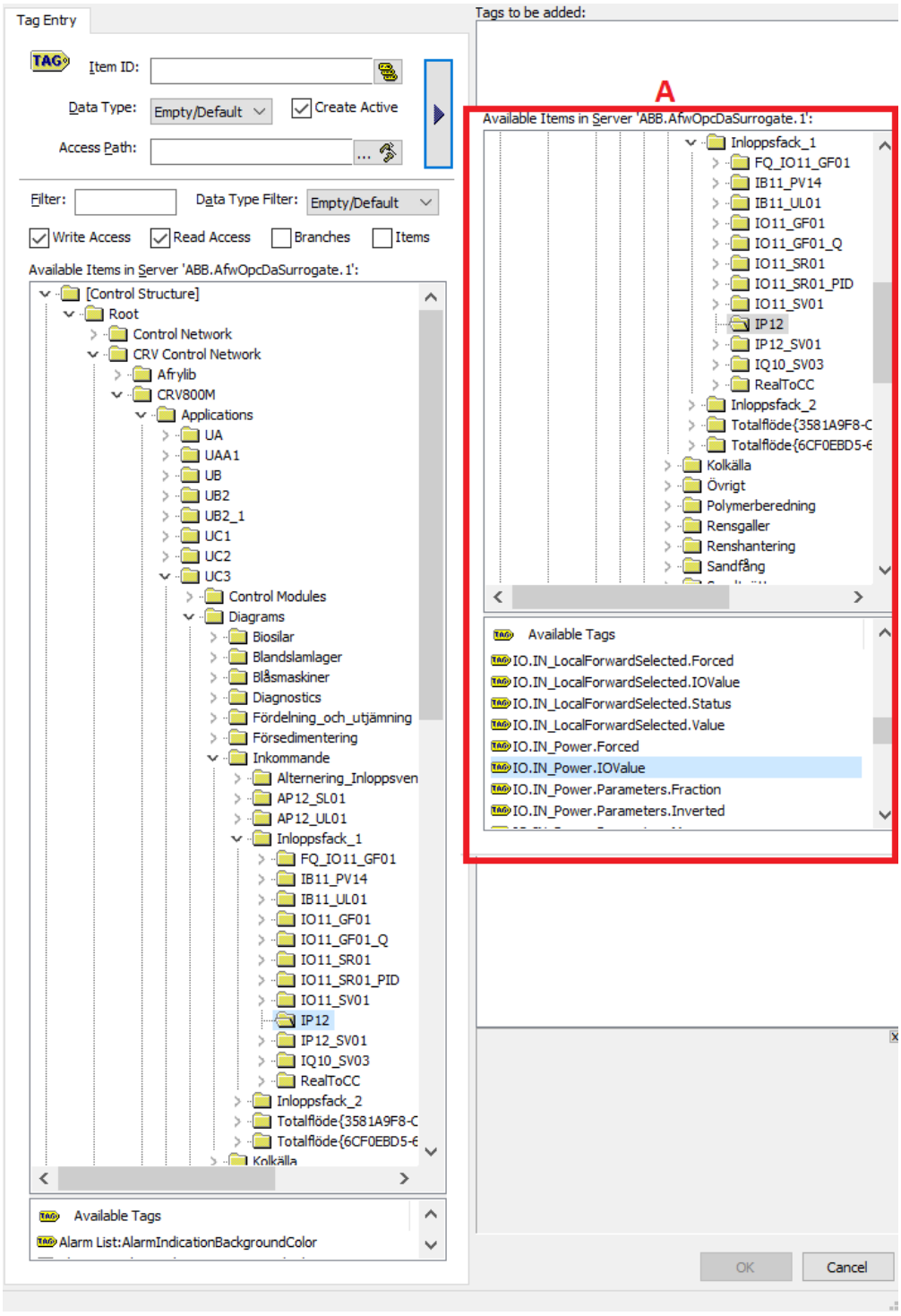
3.3.4 Integration med simulering

Utifrån det uppbyggda automationssystemet byggs som tidigare beskrivits en processmodell i SIMIT, de ingående komponenterna genereras och kopplas till de aktuella IO-signaler som krävs och en OPC-lista skapas med alla dessa signaler. Det är på detta sätt simuleringen integreras med automationssystemet. För att OPC-kopplingen ska fungera används en klient i ena änden som kopplas mot en server i andra änden där klienten kopplar de listade IO-signalerna mot motsvarande signaler som kan hittas på OPC-servern. För system som använder ABB 800xA kopplas OPC-klienten mot ABB:s *DaSurrogate*-server, vilken kan ses i figur 20. För att identifiera de IO-adresser som ska användas kan en *OPC Explorer* så som *Matrikon* användas. Från denna kan en koppling mot rätt OPC-server upprättas och strukturen gås igenom.



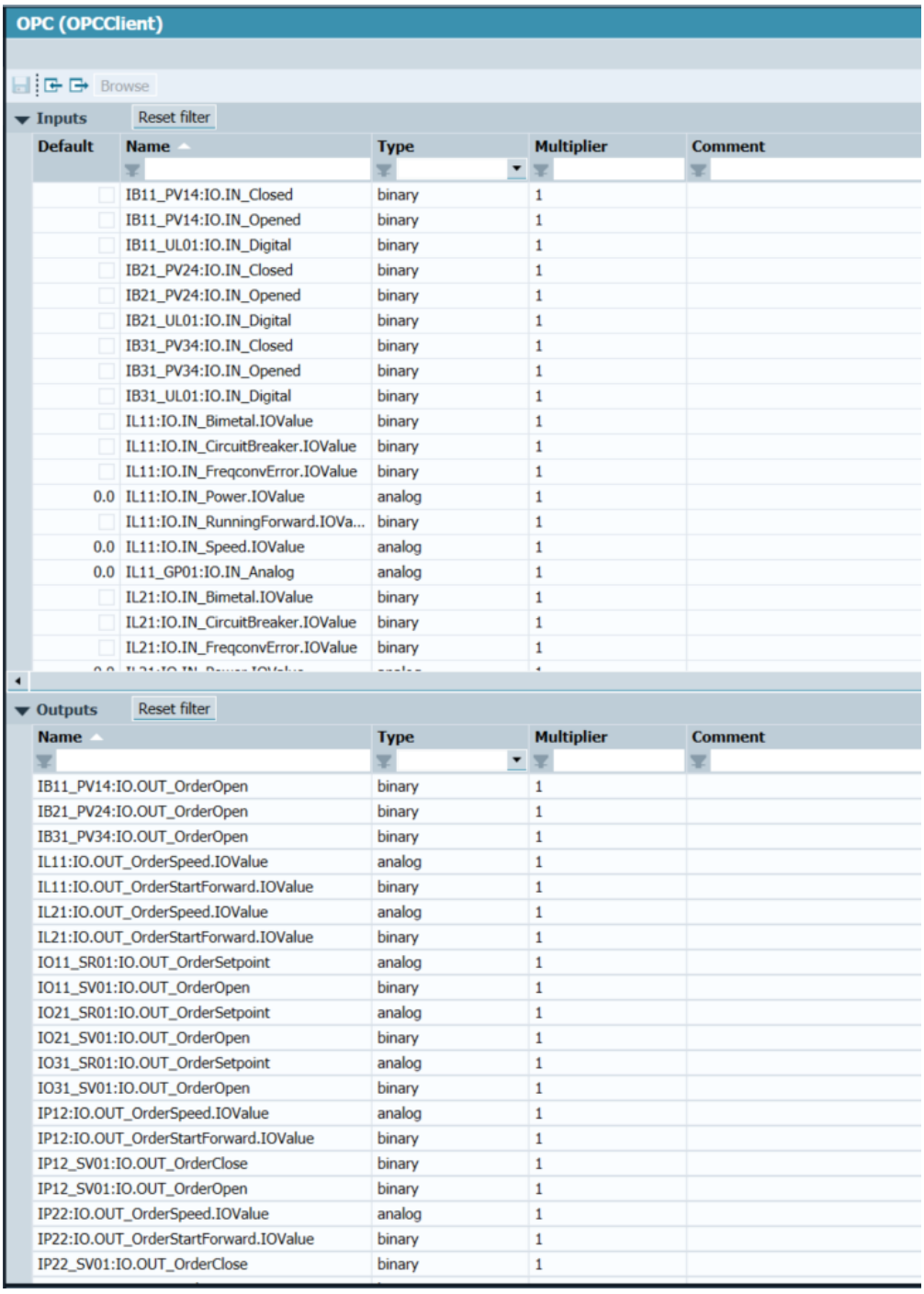
Figur 20: MatrikonOPC Explorer används för att identifiera OPC-adresserna.

Strukturen för *Surrugate*-servern för det aktuella projektet kan ses i figur 21. Här har strukturen expanderats under *Control Network* och vidare ner under de olika PLC:er som kontrollerar hela systemet. Under PLC:n *UC3* kan inloppsfack 1 och 2 hittas, och om dessa vidare expanderas kan de objekt som tillhör respektive del identifieras. I figuren är en av stenpumparna, *IP12*, markerad. För det markerade objektet kan sedan de olika taggar som finns tillgängliga för objektet ses i en lista. En del av denna lista för den markerade pumpen kan ses inklippt i figur 21. På detta sätt kan de IO-signaler som behövs för att upprätta OPC-kopplingen identifieras och läggas in i den lista som SIMIT utgår ifrån.



Figur 21: Struktur för att hitta rätt OPC-taggar. I det inklippta block A kan taggen som används för IO-signalen IN_Power ses.

Som tidigare nämnts listas alltså alla de IO-signaler som ska användas till systemet under SIMIT:s OPC-klient. Signalerna delas upp i vilka som är ut- eller insignaler, och specificeras avseende vilken typ av signal det är samt vilket normalstadie signalen ska starta i. Därefter väljs vilken OPC-server SIMIT ska leta i för att hitta de listade IO-signalerna. En del av listan för det aktuella projektet kan ses i figur 22.



OPC (OPCClient)

Browse

Inputs — Reset filter

Default	Name	Type	Multiplier	Comment
	IB11_PV14:IO.IN_Closed	binary	1	
	IB11_PV14:IO.IN_Opened	binary	1	
	IB11_UL01:IO.IN_Digital	binary	1	
	IB21_PV24:IO.IN_Closed	binary	1	
	IB21_PV24:IO.IN_Opened	binary	1	
	IB21_UL01:IO.IN_Digital	binary	1	
	IB31_PV34:IO.IN_Closed	binary	1	
	IB31_PV34:IO.IN_Opened	binary	1	
	IB31_UL01:IO.IN_Digital	binary	1	
	IL11:IO.IN_Bimetal.IOValue	binary	1	
	IL11:IO.IN_CircuitBreaker.IOValue	binary	1	
	IL11:IO.IN_FreqconvError.IOValue	binary	1	
0.0	IL11:IO.IN_Power.IOValue	analog	1	
	IL11:IO.IN_RunningForward.IOVa...	binary	1	
0.0	IL11:IO.IN_Speed.IOValue	analog	1	
0.0	IL11_GP01:IO.IN_Analog	analog	1	
	IL21:IO.IN_Bimetal.IOValue	binary	1	
	IL21:IO.IN_CircuitBreaker.IOValue	binary	1	
	IL21:IO.IN_FreqconvError.IOValue	binary	1	

Outputs — Reset filter

Name	Type	Multiplier	Comment
IB11_PV14:IO.OUT_OrderOpen	binary	1	
IB21_PV24:IO.OUT_OrderOpen	binary	1	
IB31_PV34:IO.OUT_OrderOpen	binary	1	
IL11:IO.OUT_OrderSpeed.IOValue	analog	1	
IL11:IO.OUT_OrderStartForward.IOValue	binary	1	
IL21:IO.OUT_OrderSpeed.IOValue	analog	1	
IL21:IO.OUT_OrderStartForward.IOValue	binary	1	
IO11_SR01:IO.OUT_OrderSetpoint	analog	1	
IO11_SV01:IO.OUT_OrderOpen	binary	1	
IO21_SR01:IO.OUT_OrderSetpoint	analog	1	
IO21_SV01:IO.OUT_OrderOpen	binary	1	
IO31_SR01:IO.OUT_OrderSetpoint	analog	1	
IO31_SV01:IO.OUT_OrderOpen	binary	1	
IP12:IO.OUT_OrderSpeed.IOValue	analog	1	
IP12:IO.OUT_OrderStartForward.IOValue	binary	1	
IP12_SV01:IO.OUT_OrderClose	binary	1	
IP12_SV01:IO.OUT_OrderOpen	binary	1	
IP22:IO.OUT_OrderSpeed.IOValue	analog	1	
IP22:IO.OUT_OrderStartForward.IOValue	binary	1	
IP22_SV01:IO.OUT_OrderClose	binary	1	

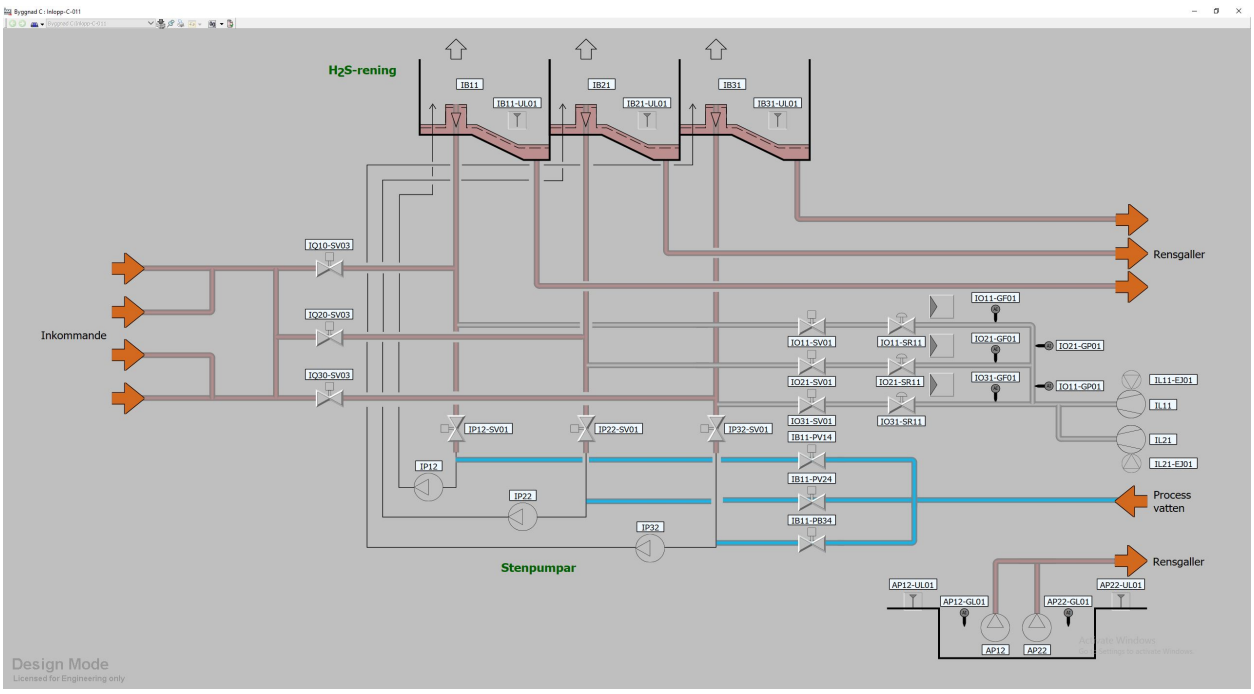
Figur 22: Lista över de IO-signaler som är kopplade till de genererade objekten.

3.4 Simulering

Som tidigare nämnts grundar sig projektet i ett automationssystem skapat av AFRY till ett verkligt avloppsreningsverk. Från detta system har en del av processen valts ut för att utgöra underlaget till simuleringen, nämligen inloppet till verket. Utifrån denna del byggs en processmodell upp, varefter hårdvara för samtliga komponenter genereras och kopplas till de IO-signaler som hänger ihop med respektive komponent. Därefter skrivs alla IO-signaler in i OPC-klienten som kopplas mot automationssystemets OPC-server enligt vad som tidigare nämnts i avsnitt *3.3.3*. Mer om simuleringsmodellens uppbyggnad i detalj kommer nu att visas i följande avsnitt.

3.4.1 SIMIT

Som tidigare nämnts använder detta projekt Siemens simuleringsverktyg SIMIT för att bygga upp den simulering som ska göras. När en simuleringsmodell ska byggas upp är första steget att skapa en processmodell som i så stor utsträckning som möjligt designas för att efterlikna den verkliga processen. För att skapa denna är det lättast att utgå från den P&ID som i regel är skapad för systemet. Utifrån denna samt systemets funktionsbeskrivning kan en modell skapas där de ingående objekten placeras ut i den följd de är tänkta att arbeta i. P&ID:n för detta projekts aktuella system kan ses i figur 23.



Figur 23: P&ID för det aktuella inloppssystemet skapad i 800xA.

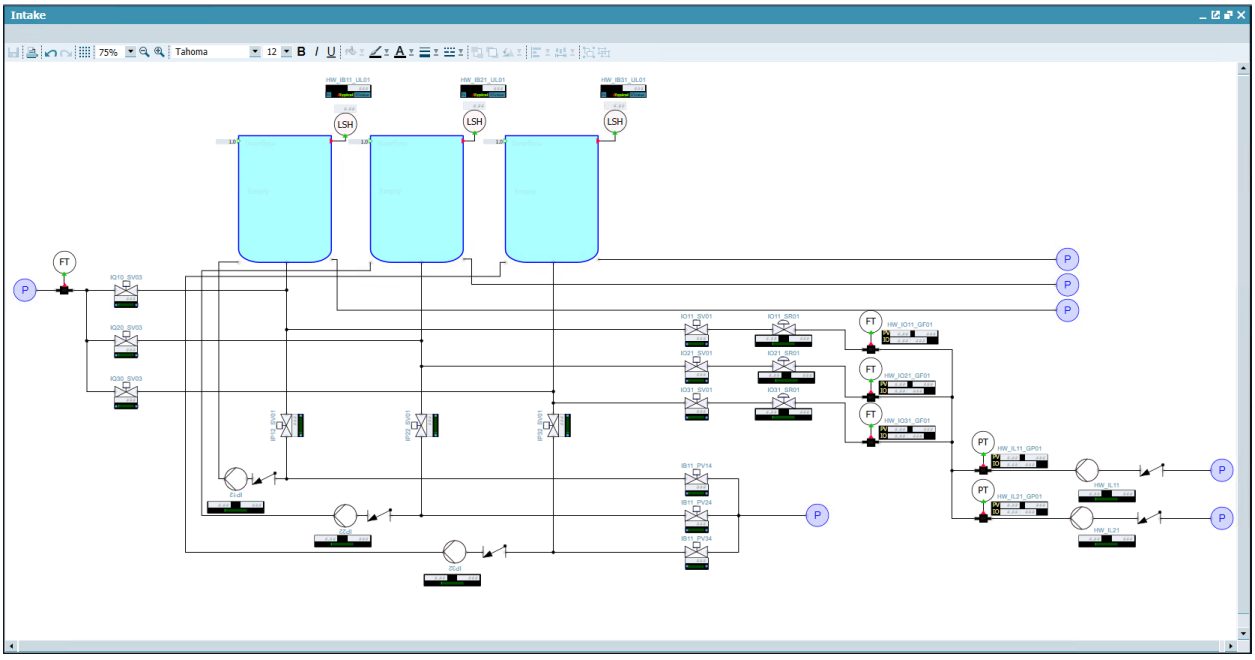
Utifrån denna skapas sedan processmodellen. När processmodellen designas i SIMIT är första steget att placera ut samtliga ingående objekt. Samtliga kan ses i tabell 1. I det aktuella fallet börjar processen till vänster i bilden, vilket gör att det första som placeras in i processmodellen är de tre inloppsventilerna. Vidare infogas sedan automatventilerna, följt av stenpumparna. Därefter infogas spolventilerna, samt de följande automatventilerna. Samtliga av dessa ventiler utgår från samma grundkomponent från AFRY:s RDT Runtime bibliotek, en modifierad variant av SIMIT:s Flownet ventil, där komponenternas parametrar är anpassade för att passa motsvararande template som kommer att visas senare i rapporten. Dessa

ventiler är kopplade till digitala signaler, och är antingen öppna eller stängda. Även pumparna som hittills är placerade i systemet är från AFRY:s RDT Runtime bibliotek och är en modifierad variant av en pump från SIMIT:s Flownet bibliotek. Denna komponent är anpassad för att motsvara den template som bygger upp *MotorAnalog*-objekten senare i detta avsnitt.

I nästa steg infogas de tre reglerventilerna. För dessa ventiler ska öppningsgraden kunna kontrolleras, vilket alltså kräver en analog signal. Till dessa används därför Runtime komponenten *ControlValvePneumatic* som kan styras på detta sätt. Nästa steg i processen är de tre flödesgivarna, följt av de två tryckgivarna som kopplas till AFRYs *Transmitter*-komponent. Denna skickar en digital signal när tanken passerar en förutbestämd nivå. De två tryckgivarna följs sedan av de två pumparna som motsvarar systemets blåsgivare. Överst i systemet läggs de tre tankarna kopplat till respektive nivågivare. Nivågivarna är kopplade till digitala signaler som motsvarar RDT templaten *DigitalInput*. För att simuleringen ska fungera kan inga så kallade öppna grenar finnas, utan alla inlopp och utlopp i systemet måste kopplas till en så kallad *PNode*, eller trycknod. Denna representerar det tryck som normalt sätt hade funnits i systemet om detta varit kopplad till en annan del av processen. Processmodellen kan ses i figur 24.

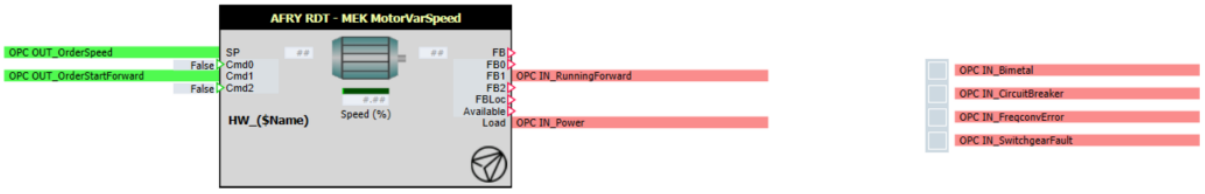
Komponenttyp	**Komponentnamn**
Inloppsventil	IQ10_SV03, IQ20_SV03, IQ30_SV03
Automatventil	IP12_SV01, IP22_SV01, IP32_SV01 IO11_SV01, IO21_SV01, IO31_SV01
Stenpump	IP12, IP22, IP32
Spolventil	IB11_PV14, IB11_PV24, IB11_PV34
Reglerventil	IO11_SR01, IO21_SR01, IO31_SR01
Flödesgivare	IO11_GF01, IO21_GF01, IO31_GF01
Tryckgivare	IL11_GP01, IL21_GP01
Blåsgivare	IL11, IL21
Tank	IB11, IB21, IB31
Nivågivare	IB11_UL01, IB21_UL01, IB31_UL01

Tabell 1: Tabell över de ingående objekten i processmodellen.



Figur 24: Processmodell för det aktuella inloppssystemet i SIMIT.

För att de simulerade objekten ska motsvara de verkliga objekten måste deras beteende programmeras och styras. De objekt som används till detta projekt grundar sig i templates som i stort sett är färdigskapade av AFRY:s RDT-avdelning och samlade i AFRY:s RDT Runtime bibliotek. Dessa templates anpassas sedan för att passa de objekt som ska skapas i processmodellen. Till de templates som används kopplas sedan de inputs och outputs som är kopplade till objektet och som kommer att vara grunden för OPC-kopplingen. Den template som används för komponenttypen MotorAnalog kan ses i figur 25.



Figur 25: Template för komponenttypen MotorAnalog.

Från figuren kan ses att två olika *OUT* signaler är kopplade mot ingångarna *SP* och *Cmd1* på vänster sida av templaten. Ingången *SP* hanterar en analog signal och är därför kopplad mot IO-signalen *OrderSpeed*. På motsvarande sätt hanterar ingången *Cmd1* en *Boolean* och är därför kopplad mot signalen *OrderStartForward* som endast kan vara *TRUE* eller *FALSE*. Till utgångarna på högra sidan kopplas de insignaler som skickas från den simulerade komponenten till styrningen, som i detta fall är kopplade mot *FB1* och *Load*. *FB1* hanterar signaltypen *Bool* och är därför kopplad mot IO-signalen *RunningForward*

som även den endast kan vara *TRUE* eller *FALSE*, medan *Load* hanterar en analog signal och därför är kopplad mot *Power*. Längst till höger i bilden ses de signaler som inte är direkt kopplade in eller ut ur komponenten, i detta fall olika larmsignaler. Dessa kopplas istället mot switchar som kan hanteras separat, och matchas sedan mot motsvarande IO-signaler i OPC-kopplingen.

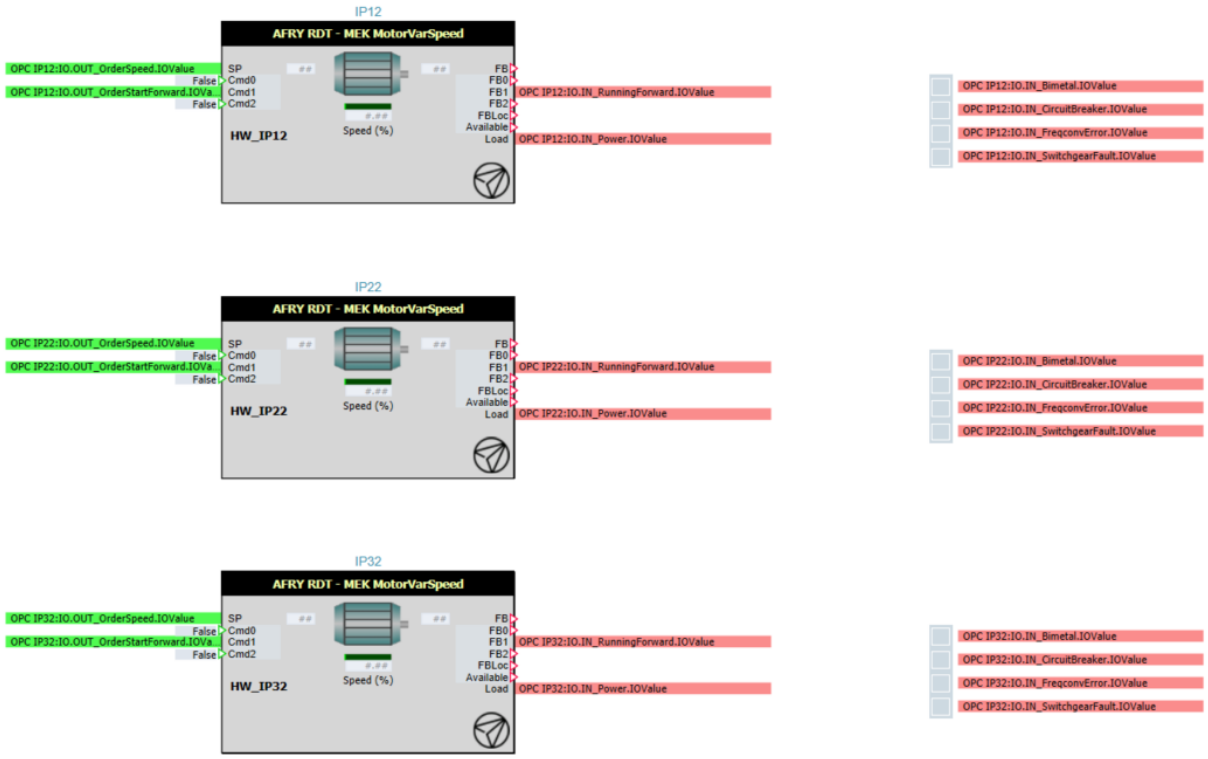
HIERARCHY	TEMPLATE	CHART	OPC	IN_Bimetal	IN_CircuitBreaker	IN_FreqconvError	IN_Power
Generated Hardware	MotorAnalog	Motor	OPC	IP12:IO.IN_Bimetal.IOValue	IP12:IO.IN_CircuitBreaker.IOValue	IP12:IO.IN_FreqconvError.IOValue	IP12:IO.IN_Power.IOValue
Generated Hardware	MotorAnalog	Motor	OPC	IP22:IO.IN_Bimetal.IOValue	IP22:IO.IN_CircuitBreaker.IOValue	IP22:IO.IN_FreqconvError.IOValue	IP22:IO.IN_Power.IOValue
Generated Hardware	MotorAnalog	Motor	OPC	IP32:IO.IN_Bimetal.IOValue	IP32:IO.IN_CircuitBreaker.IOValue	IP32:IO.IN_FreqconvError.IOValue	IP32:IO.IN_Power.IOValue

IN_RunningForward	IN_SwitchgearFault	Name	OUT_OrderSpeed	OUT_OrderStartForward
IP12:IO.IN_RunningForward.IOValue	IP12:IO.IN_SwitchgearFault.IOValue	IP12	IP12:IO.OUT_OrderSpeed.IOValue	IP12:IO.OUT_OrderStartForward.IOValue
IP22:IO.IN_RunningForward.IOValue	IP22:IO.IN_SwitchgearFault.IOValue	IP22	IP22:IO.OUT_OrderSpeed.IOValue	IP22:IO.OUT_OrderStartForward.IOValue
IP32:IO.IN_RunningForward.IOValue	IP32:IO.IN_SwitchgearFault.IOValue	IP32	IP32:IO.OUT_OrderSpeed.IOValue	IP32:IO.OUT_OrderStartForward.IOValue

Figur 26: Motvarande rader för MotorAnalog i Excel.

Templaten i figur 25 exporteras sedan till Excel där programmet genererar de kolumner som behöver fyllas i för att kunna använda templaten för att generera de simulerade objekten i ett senare skeda. Dessa rader fylls sedan i enligt exemplet i figur 26. På respektive plats skrivs namnet på den OPC-tagg som senare kommer användas för att koppla SIMIT mot styrningen i 800xA. Taggarnas namn har hittats och kontrollerats via OPC-utforskaren Matrikon enligt vad som visats tidigare i rapporten. Samtliga templates uppbyggnad redovisas i bilaga B.

Utifrån dessa templates och motsvarande rader i Excel kan sedan de objekt som utgör den faktiska modellen genereras. Detta görs genom den inbyggda SIMIT funktionen *Instantiate templates*, som importerar en fil från Excel och genom denna genererar de objekt som skrivits in i filen. I filen skrivs var i hierarkin komponenterna ska genereras, vilken template de ska utgå ifrån, och vilka parametrar som ska skapas. Dessa parametrar läses sedan in till varje komponent. Varje rad i Excelfilen genererar ett objekt. Då det i filen anges vilket så kallat *Chart* komponenterna ska skapas i kan samtliga komponenter egentligen genereras från samma fil. I detta projekt har dock varje template gjorts till en separat fil som sedan har använts till att generera respektive komponenter. I figur 27 kan de genererade objekten av typen *MotorAnalog* ses. Dessa objekt motsvarar stenpumparna *IP12*, *IP22* och *IP32* från processmodellen. I figuren kan de IO-adresser som är kopplade till de specifika signalerna ses. En fullständig genomgång av de genererade objekt som ingår i modellen redovisas i bilaga C.



Figur 27: Genererade objekt av typen MotorAnalog.

Samtliga IO-signaler från dessa genererade objekt sammanställs sedan i en lista. Även detta görs med fördel via Excel, även om det kan göras direkt i OPC-listan i SIMIT. I listan anges främst den verkliga OPC-adressen för varje signal samt om signalen är en in- eller utsignal. En del av denna lista kan ses i figur 28.

OPC (OPCClient)

Browse

Inputs Reset filter

Default	Name	Type	Multiplier	Comment
	IB11_PV14:IO.IN_Closed	binary	1	
	IB11_PV14:IO.IN_Opened	binary	1	
	IB11_UL01:IO.IN_Digital	binary	1	
	IB21_PV24:IO.IN_Closed	binary	1	
	IB21_PV24:IO.IN_Opened	binary	1	

Figur 28: Lista över de IO-signaler som är kopplade till de genererade objekten.

För att de genererade objekten ska kopplas ihop med objekten i processmodellen måste de så kallade *hardware*-objekten kopplas ihop till processobjekten. Detta görs genom att en kopplad *faceplate*, alltså en grafisk representation av objektets styrning, dras ut från den genererade hårdvaran, som då automatiskt länkar till den hårdvara den kommer ifrån. Detta görs för alla objekt i processmodellen. Därefter kopplas processobjekten till hårdvaran genom att namnen matchas, med tillägget av prefixet *HW_* på hårdvaruobjekten. På detta sätt utför SIMIT automatiskt kopplingen mellan objekten genom namnmatchning. För att detta ska fungera krävs dock att inga objekt delar namn, utan namnen för alla objekt måste vara unika.

3.5 Avgränsningar

Projektet har inte varit utan svårigheter och problem, vilket har lett till att en hel del avgränsningar har fått göras i samband med arbetet. De stora avgränsningarna har gjorts kopplat till de tidsmässiga begränsningar som har funnits, men även en del tekniska begränsningar har funnits. Ytterligare begränsningar har gjorts, orsakade av tekniska problem som i sin tur har lett till minskad tillgänglig tid för projektet. Några av dessa begränsningar presenteras i följande avsnitt, fördelat på tekniska- samt tidsmässiga begränsningar.

3.5.1 Tekniska begränsningar

Flera tekniska begränsningar har uppkommit längs vägen i projektet, inte minst kopplat till automationssystemet. Projektet var nämligen tänkt att egentligen byggas kring det demosystem från ABB som presenterades tidigare i rapporten, och en stor del av arbetet skedde alltså kopplat till denna miljö. När denna miljö studerades i samarbete med AFRY:s RDT avdelning upptäcktes bland annat att miljön hade en struktur som gjorde det praktiska arbetet svårt, då objekten var skapade på ett komplicerat sätt utifrån diagram. Något som inte samma erfarenhet fanns kring, jämfört med andra metoder. När miljön studerades upptäcktes även att objekten i projektet tillhörde ett okänt bibliotek och alltså inte de standardkomponenter som vanligtvis används. Detta skulle med andra ord leda till att de Runtime-komponenter som var tänkta att kunna utnyttjas inte längre skulle fungera för projektet. Istället skulle varje komponent ha behövts byggas upp från grunden, vilket skulle vara både komplicerat och tidskrävande. En ytterligare svårighet kring att bygga dessa komponenter, samt en processmodell, var bristen på funktionsbeskrivning och en riktig P&ID. I brist på P&ID användes systemets HMI för att bygga upp processmodellen för att efterlikna systemet i så stor utsträckning som möjligt. Utan kopplingen mellan P&ID och funktionsbeskrivning var dock uppbyggnaden av komponenterna svår att göra på ett korrekt sätt. Komponenterna började därför att skapas utifrån den kunskapen som fanns tillgänglig kring hur systemet skulle kunna fungera.

I samband med att detta arbete pågick upptäcktes dock nästa problem. När OPC-kopplingen till systemet skulle testas för att identifiera de korrekta IO-adresserna, upptäcktes att strukturen av någon anledning inte gick att expandera för att hitta rätt adresser. Flera försök till lösningar gjordes, men inget lyckades. När systemet kopierades och överfördes från ABB hade alltså någon del av projektet blivit korrupt, vilket gjorde att systemet inte gick att expandera i OPC-servern. Utan möjlighet till OPC-kopplingen fanns alltså ingen möjlighet till att koppla simuleringen till automationssystemet. I ett försök att lösa problemet försöktes en ny kopia av systemet göras, genom att flytta endast de delar som var relevanta för detta projekt, och hoppas att den korrupta delen av systemet inte låg i denna del. När de relevanta delarna överfördes försvann vissa delar av problemet, då strukturen faktiskt kunde expanderas genom OPC-utforskaren. Men med denna nya överföring uppkom istället ett nytt problem. Vissa delar av de objekt som fanns i automationssystemet, bland annat all grafisk representation, visade sig ligga i ett låst bibliotek som inte gick att överföra. Detta bibliotek krävde nämligen en utvidgning av 800xA miljön som normalt sett inte används och därför inte fanns i den nya miljön. Ett flertal kunniga och erfarna personer försökte att lösa även detta problem utan framgång.

Utifrån ovanstående beslutades därför att en annan miljö skulle användas, vilket gjorde att skiftet till den nya miljön som sedan har utgjort grunden för arbetet utfördes.

3.5.2 Tidsbegränsningar

Utifrån föregående nämnda problem kan en övergång göras till de tidsmässiga begränsningar som har uppkommit. Genomgående i projektet har många olika bitar tagit lång tid att få på plats, vilket har påverkat den tillgängliga tiden. Till exempel dröjde det innan demosystemet från ABB levererades, vilket givetvis fördröjde arbetet. Även alla de tidigare nämnda tekniska problemen har utöver deras rent tekniska påverkan också fördröjt arbetet. Stora mängder tid gick under projektet åt till att försöka lösa de problem som uppkom med de olika systemen, snarare än arbete som faktiskt tog projektet framåt.

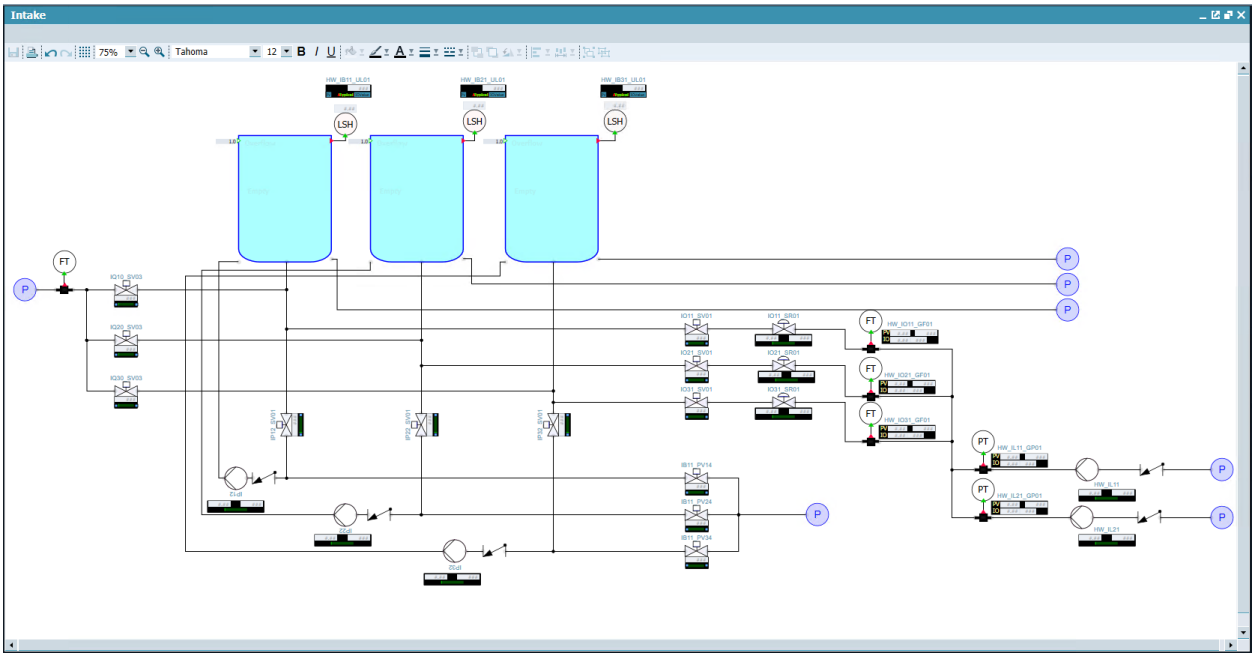
Störst påverkan på arbetet hade dock självklart skiftet till en ny miljö. Skiftet var nödvändigt för att arbetet skulle kunna fortsätta, men ett sådant sent skifte har såklart en stor påverkan även ur en tidsaspekt. Stora delar av det arbete som påbörjats kring simuleringsmodellen fick göras på nytt för det nya systemet och även om arbetet kunde fortgå så blev den önskvärda tidsplanen oundvikligt förskjuten. Utifrån detta fick den del ur den nya miljön som valdes som grund till simuleringen begränsas till en av de mindre delarna av systemet, för att arbetet skulle vara möjligt att utföra på den begränsade tid som återstod av projektet.

4 Resultat

Trots alla de svårigheter som har uppkommit och alla de begränsningar som har gjorts har arbetet i slutändan lett till ett resultat. Simuleringsmodellen är uppbyggd i SIMIT, med både processmodell, uppbyggda komponenter, genererad hårdvara och IO-lista. Automationsmiljön fungerar och går att köra och OPC-kopplingen mellan dessa båda system är även den fungerande. Det framarbetade resultatet presenteras i följande avsnitt.

4.1 Simulering

Som tidigare nämnts består simuleringsmodellen av flera bakomliggande system som kontrollerar hur de involverade objekten ska bete sig och vilka IO-signaler som ska styra respektive objekt. När simuleringen körs är dock fokus på processmodellen, som i grunden är tänkt att representera den verkliga processen. När simuleringen är offline är färgen på omgivningens tema blått, enligt vad som kan ses i figur 29. I detta läge kan objekten redigeras direkt i modellen och ingen koppling till OPC-servern är igång.

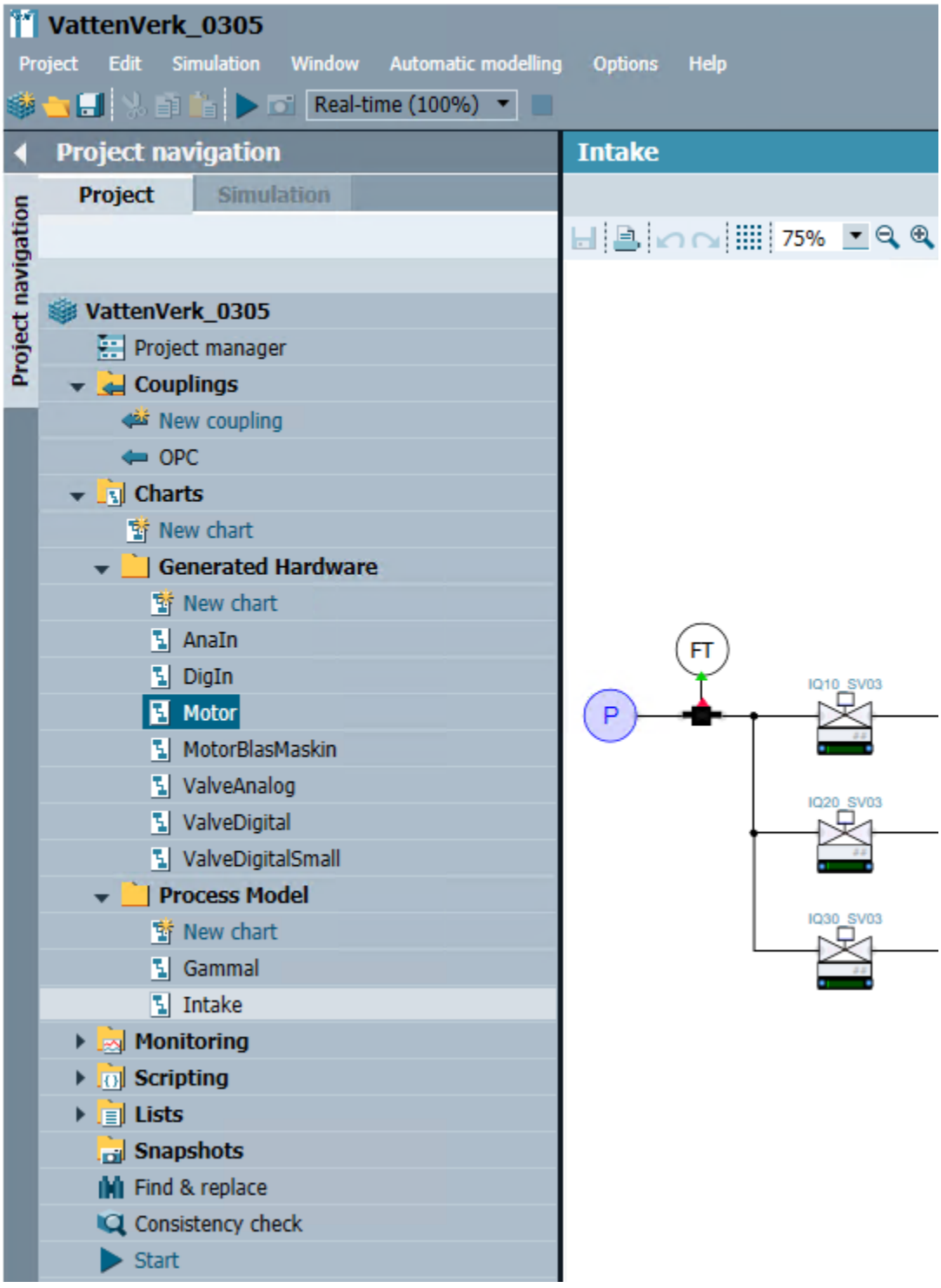


Figur 29: Processmodellen i SIMIT i offline-läge. Den blåa färgen representerar att systemet är offline.

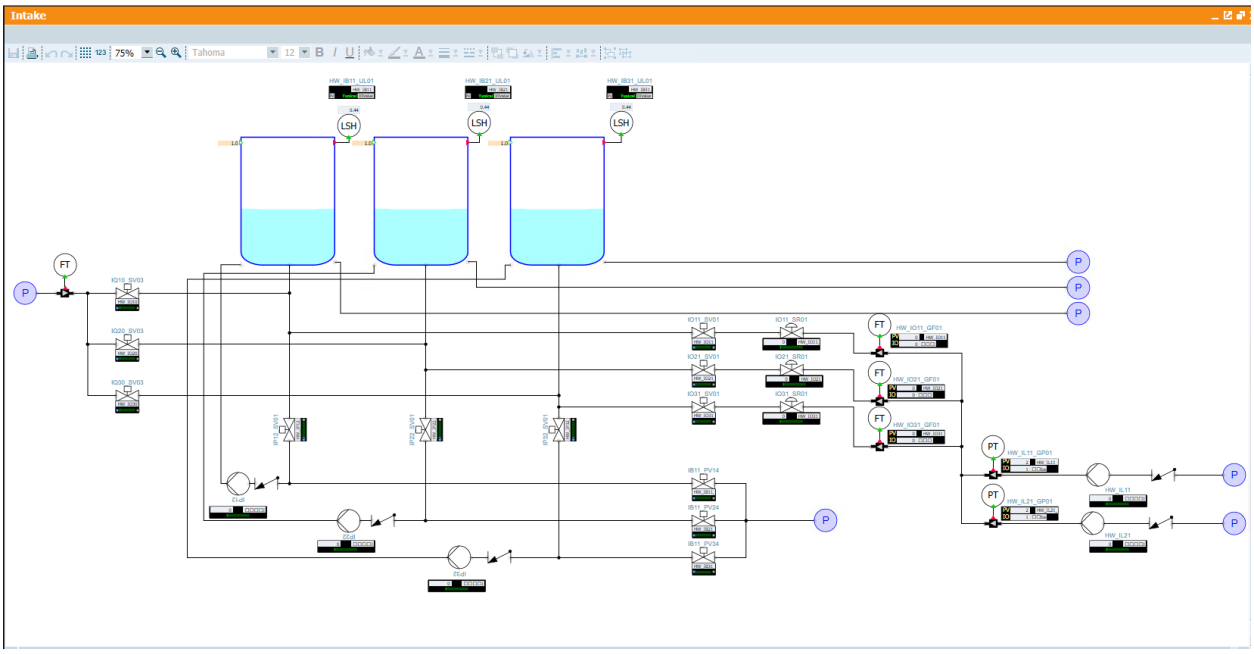
Simuleringen startas genom att antingen trycka på start-knappen i den övre menyn eller från listan till vänster i programmet, enligt vad som kan ses i figur 30. SIMIT gör då en så kallad *Consistency check* för att till exempel säkerställa att inga öppna grenar eller andra problem finns i modellen som hade kunnat hindra den från att köras. Även alla namn på de olika objekten kontrolleras för att säkerställa att alla objekt har unika namn, samt att alla hårdvaruobjekt dels är rätt hänvisade i de fall de är kopplade mot processobjekt, dels att de IO-signaler som är kopplade motsvarar den signaltyp som hanteras av motsvarande koppling. Om OPC-fliken är aktiverad kontrolleras även denna, för att säkerställa att namnen i IO-listan stämmer överens med de IO-signaler som angetts i objekten. Allt detta kontrolleras innan simuleringen startar.

Vissa fel, som till exempel namn som inte matchar eller signaler som inte uppenbart används, ger varningar men hindrar inte simuleringen från att köras. Däremot större fel, som öppna grenar i modellen, hindrar

simuleringen från att starta överhuvudtaget. När simuleringen är online skiftar färgen på det omgivande temat från blå till orange, för att visa att programmet körs. Detta kan ses i figur 31.

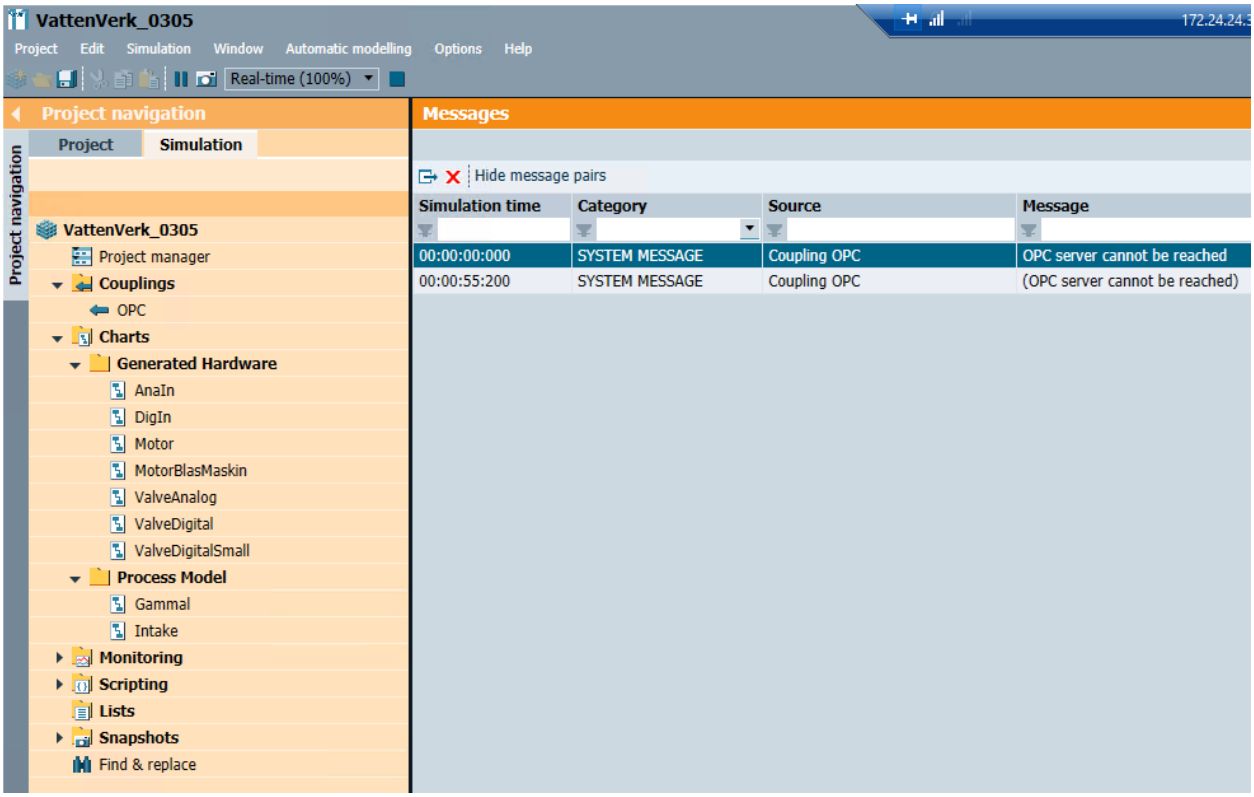


Figur 30: Simuleringen startas genom att trycka på någon av start-knapparna.



Figur 31: Processmodellen i SIMIT i online-läge, vilket kan ses av den orangea färgen.

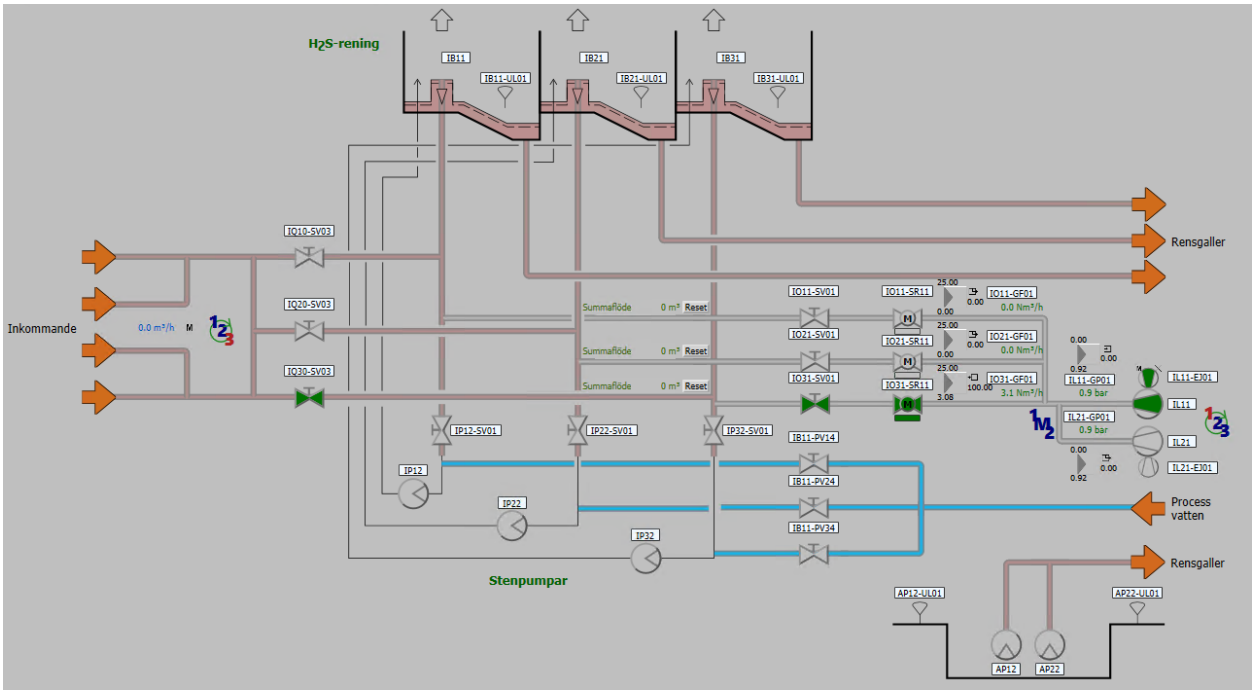
Om OPC-kopplingen är aktiverad, det vill säga om simuleringen är tänkt att kopplas mot automationssystemet, försöker SIMIT nå OPC-servern direkt när simuleringen startas. Medan *Consistency checken* körs och innan SIMIT har fått kontakt med OPC-servern varnar programmet för att OPC-servern inte kan nås. När SIMIT sedan når OPC-servern och upprättar kopplingen till automationssystemet ersätts det föregående meddelandet av ett motsvarande meddelande satt i parentes, vilket betyder att problemet är åtgärdat. Ett exempel på detta kan ses i figur 32.



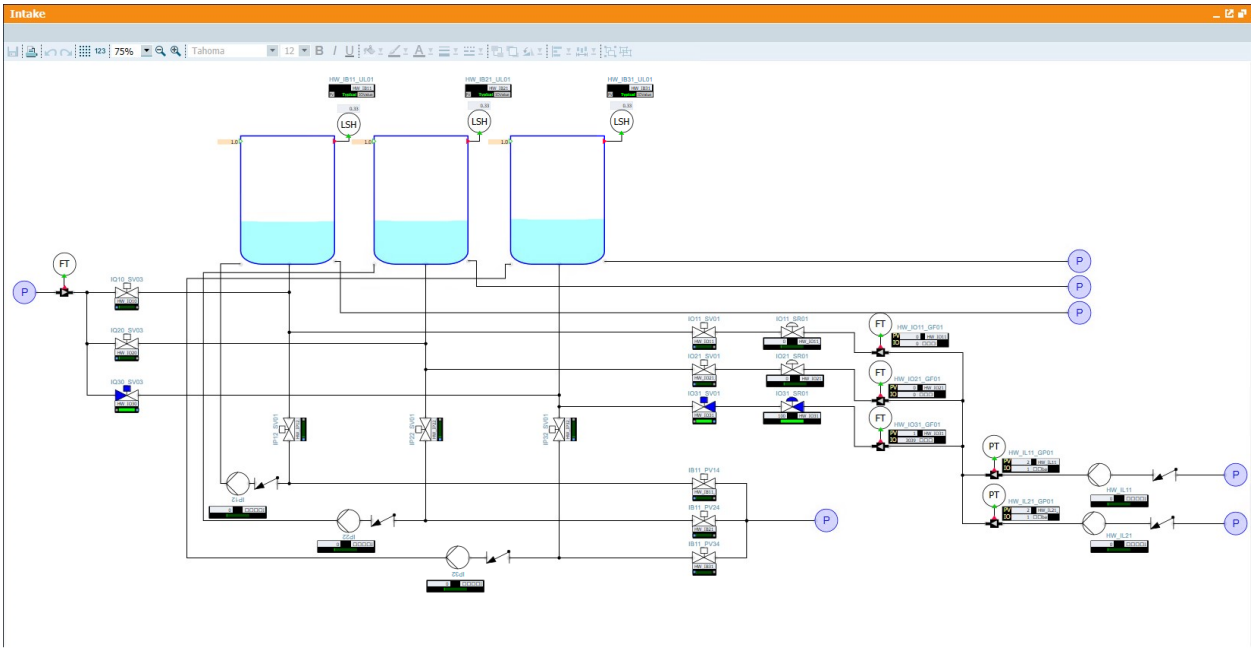
Figur 32: Innan OPC-kopplingen har upprättats varnar SIMIT för att OPC-servern inte kan nås.

När OPC-kopplingen är igång kan simuleringen köras på riktigt. Från automationssystemet kan systemet startas upp.

Det avloppsreningsverk som automationssystemet är tänkt att styra är programmerat för att alltid ha en inloppsventil öppen. När inloppsventilen är öppen ska även motsvarande luftningsventiler i samma linje vara öppna. Hur detta ser ut i 800xA-miljön kan ses i figur 33. I detta fall är inloppsventil tre med motsvarande linje öppen, vilket kan ses av de grönmarkerade objekten i denna linje. Vilken av ventilerna som är öppen i utgångsläget går att styra genom att ändra prioriteringsordningen för ventilerna. Detta kan göras direkt i HMI:t genom att trycka på symbolen med siffrorna 1, 2 och 3 till vänster om ventilerna. Motsvarande linje öppnas då även i simuleringsmodellen. Detta styrs av att IO-värdet på respektive objekts IO-signal *Opened* eller *Closed* skrivs till *true* respektive *false* i automationssystemet, vilket därefter via OPC-kopplingen skickar motsvarande signal till simuleringen. I de fall där objekten ska ha en öppningsgrad eller hastighet skickas detta som en analog symbol, som även den då ger ett värde till motsvarande IO-signal. Den öppna linjen i SIMIT kan ses i figur 34.



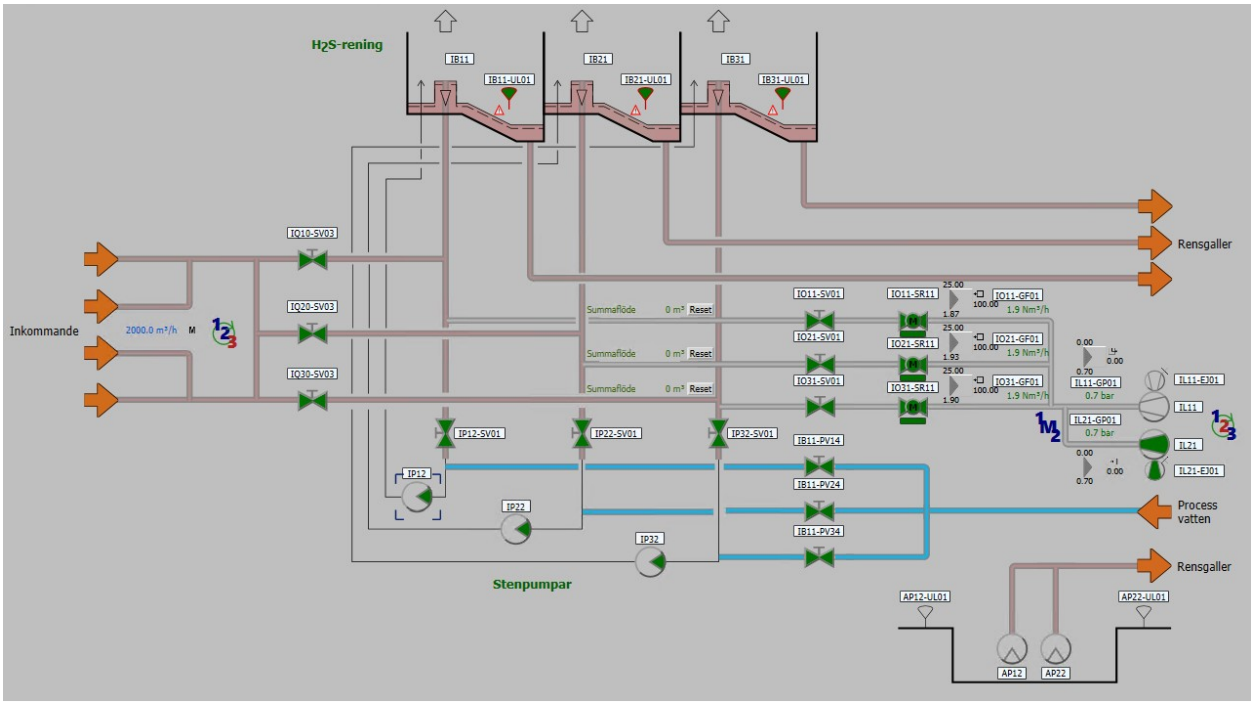
Figur 33: En öppen linje i automationssystemet i 800xA-miljön kan ses av de grön-markerade objekten.



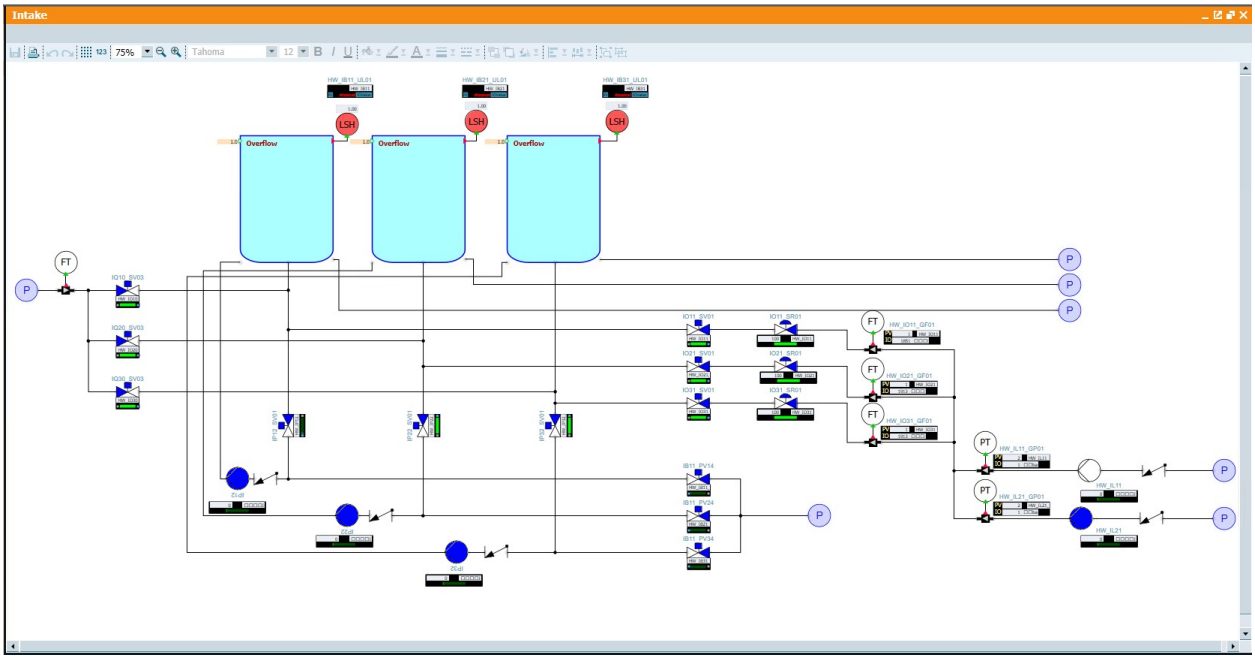
Figur 34: Motsvarande öppen linje i simuleringsmodellen i SIMIT kan ses av de blåa objekten.

Från automationssystemet kan totalflödet in i systemet manuellt ändras. Inloppsventilerna är programmerade för att starta i sekvens beroende på totalflödets storlek, där ett flöde under 1000 m^3/h endast använder en av inloppsventilerna. Om flödet istället är mellan 1000-2000 m^3/h öppnas ytterligare en av ventilerna, och om flödet är över 2000 m^3/h öppnas alla tre. I figur 35 har flödet ställts upp till 2000 m^3/h, vilket alltså resulterar i att alla linjer är öppna. Motsvarande kan då ses i processen i SIMIT, vilket kan ses i figur 36. Utifrån de flöden genom ventilerna som är inlagda i processmodellen fylls då tankarna på i takt med att simuleringen körs. I figuren kan ses att simuleringen har körts under en period, vilket har resulterat i att tankarna har nått upp till nivågivaren, som därför har larmat till systemet att tankarna håller på att fyllas. Detta kan i SIMIT ses genom att dessa mätpunkter är röda istället för vita. I automationssystemets HMI kan det ses genom att indikationen för nivågivarna har fyllts i med grön färg med en röd kant, samt en markerad varning bredvid ikonen.

I automationssystemet kan då flödet ställas ner, vilket i simuleringen leder till att nivån i tankarna minskar. Från HMI:t kan då larmet i sin tur markeras som åtgärdat, vilket gör att nivågivarna återställs till sitt ursprungsläge.

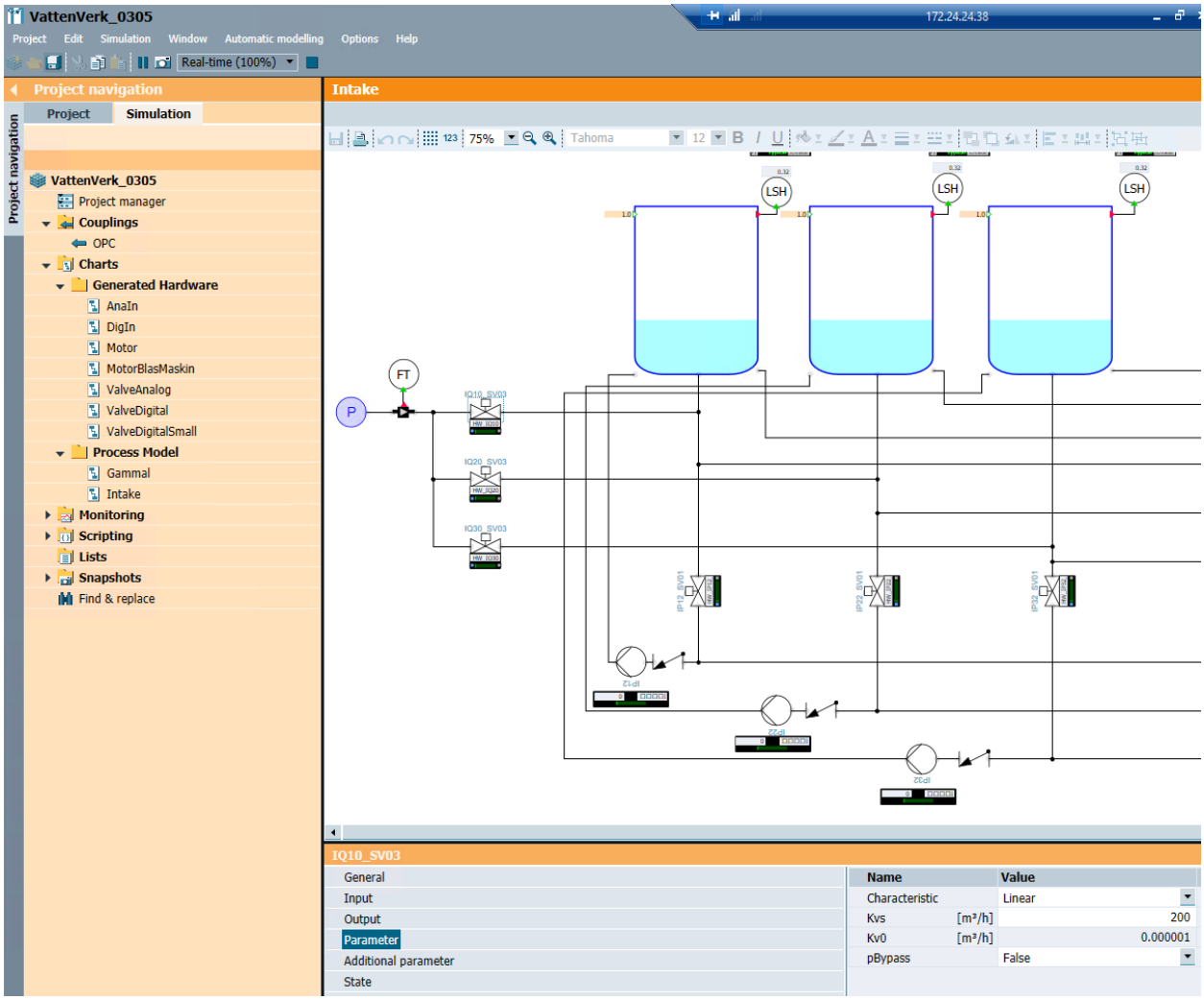


Figur 35: Alla linjer öppna i automationssystemet med flödet 2000 m^3/h.

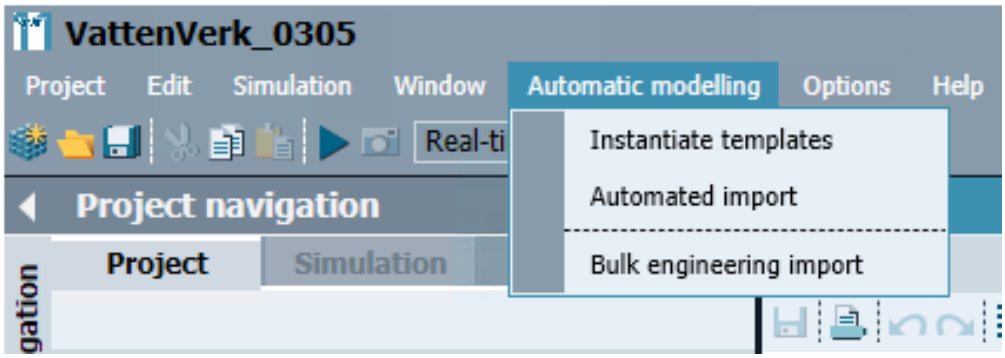


Figur 36: Alla linjer öppna i SIMIT med flödet 2000 m^3/h.

Om ändringar ska utföras i simuleringsmodellen medan simuleringen körs kan detta göras på två sätt beroende på vilken ändring som vill åstadkommas. Vissa parametrar för de objekt som är inlagda i processmodellen går att ändra genom att högerklicka på objektet i fråga och direkt ändra den önskade parametern i menyn i botten av fönstret. Detta alternativ kan ses i figur 37. Dessa ändringar körs då direkt, men sparas inte efter att simuleringen har stängts av. I de fall ändringarna behöver sparas i efterhand kan antingen samma ändringar göras manuellt för varje objekt i offline läget, alternativt kan en så kallad *Bulk engineering import* göras. I detta fall listas alla de ändringar som gjordes medan simuleringen kördes, varpå de som önskas sparas kan markeras och importeras till modellen. I fall där många ändringar gjorts kan detta fylla en stor funktion. Detta alternativ kan ses i figur 38.

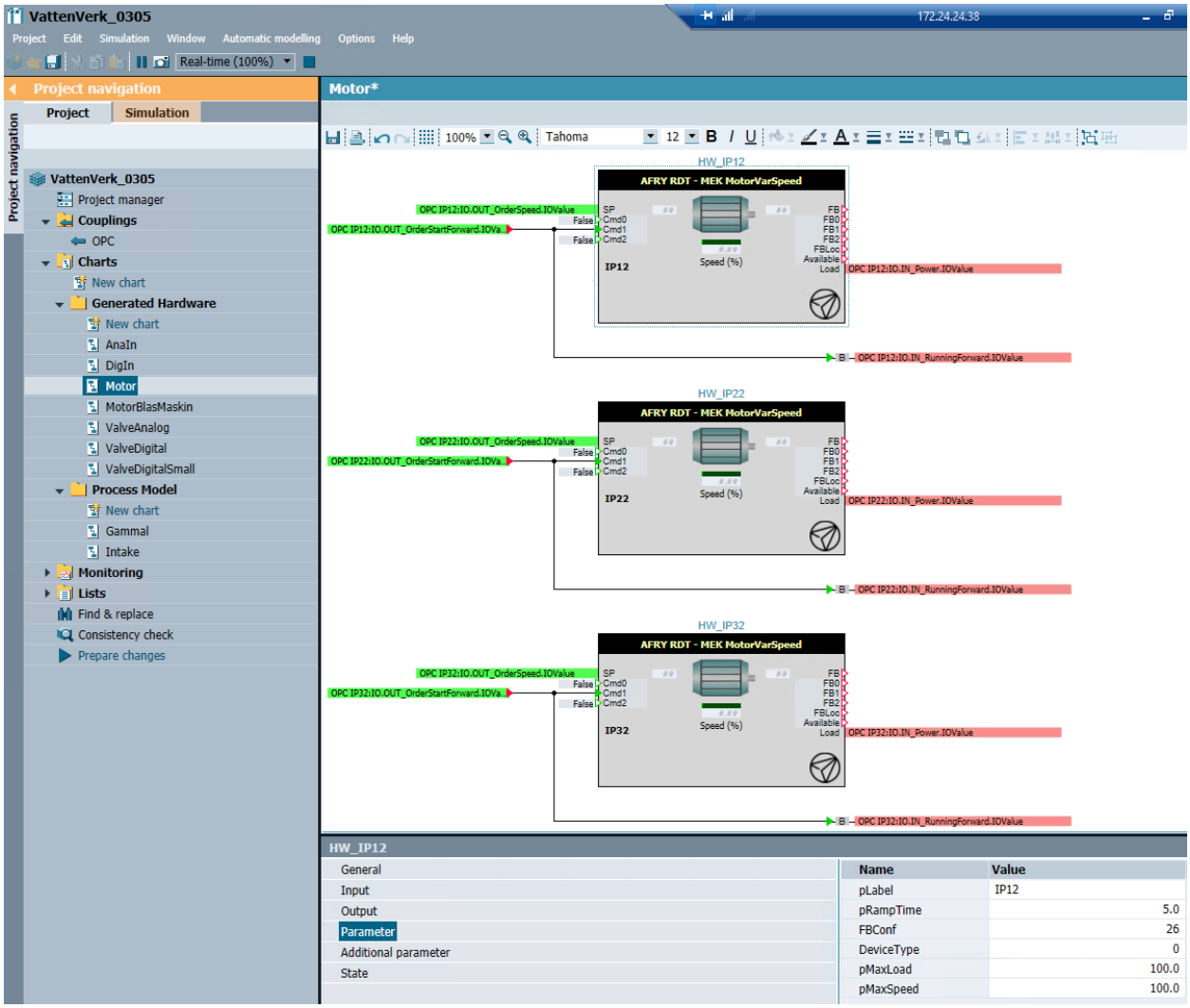


Figur 37: Ändringar i objekt i processmodellen kan göras online.

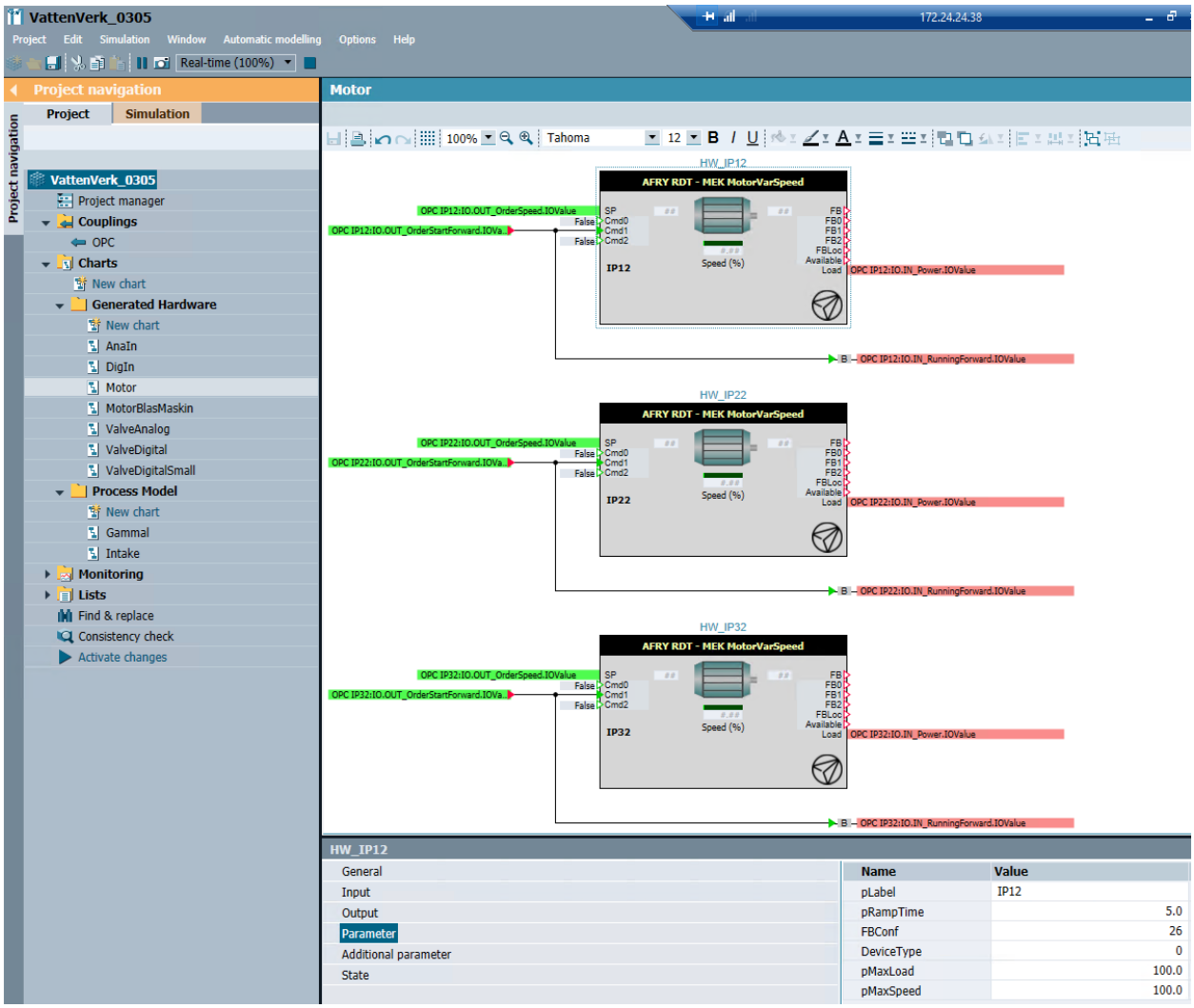


Figur 38: Om ändringar ska läsas in efter avslutad simulering kan *Bulk engineering import* användas.

Om större ändringar ska göras eller ändringar måste göras i hårdvaruobjekten kan dessa öppnas parallellt. Det *chart* som innehåller de aktuella objekten öppnas då i ett nytt fönster i simuleringen, och genom att högerklicka i fönstret kan alternativet *Edit offline* väljas. Detta *chart* öppnas då offline samtidigt som simuleringen fortfarande körs. Detta kan ses i figur 39. I detta fönster kan de önskade ändringarna då genomföras. När ändringarna är utförda och önskas implementeras i simuleringen används alternativet *Prepare changes* som kan ses i menyn till vänster. Ändringarna förbereds då av programmet för att kunna implementeras i systemet utan avbrott i simuleringen. När ändringarna är förberedda ändras menyn till att istället ge alternativet *Activate changes* i menyn till vänster. Detta kan ses i figur 40.

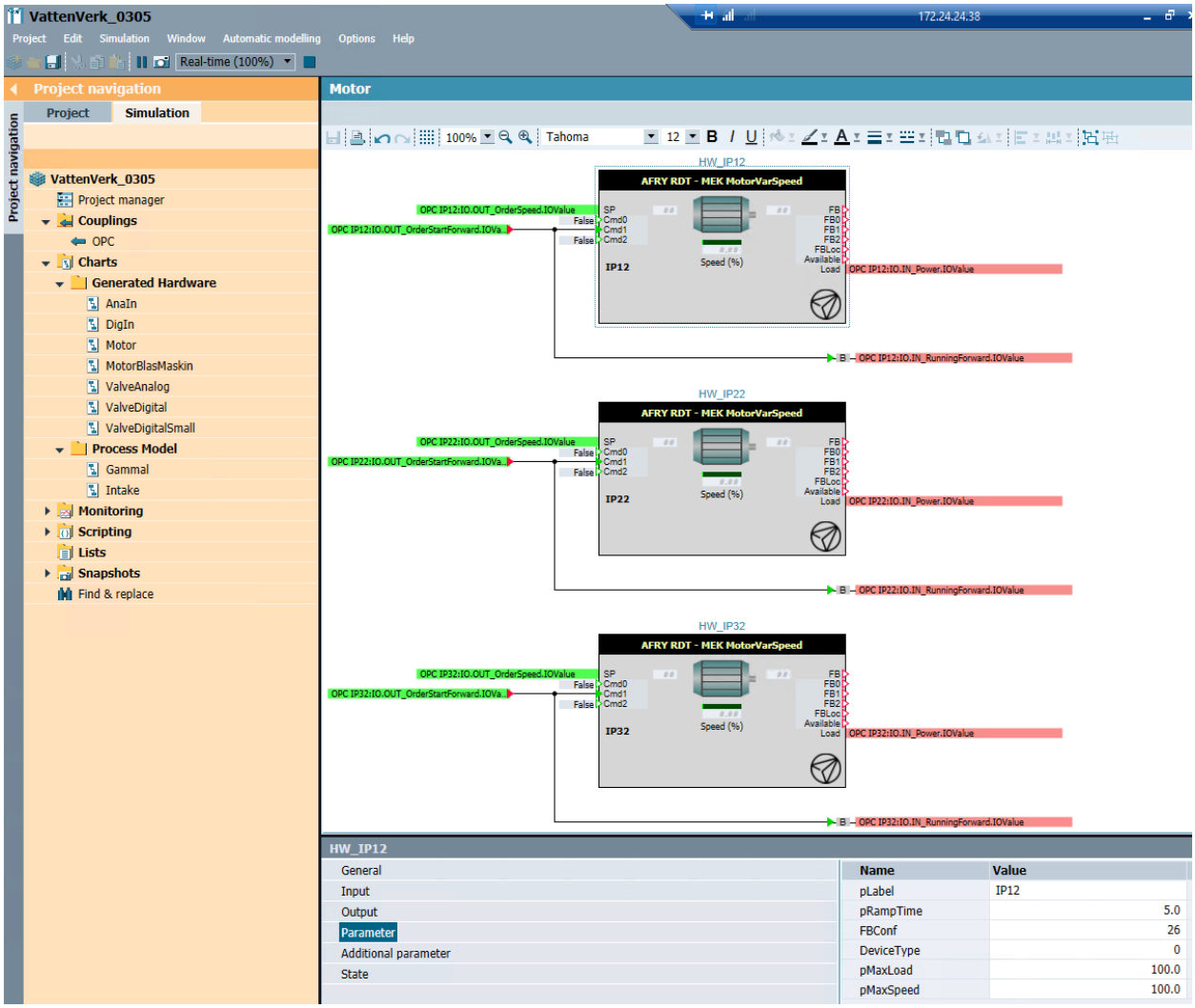


Figur 39: Alternativet *Prepare changes* kan ses i menyn till vänster.



Figur 40: Alternativet *Activate changes* kan ses i menyn till vänster.

När ändringarna är genomförda skiftar menyn i vänsterkanten till orange, vilket visar att systemet återigen är online i sin helhet. Detta kan ses i figur 41. Det diagram som har använts för att göra ändringarna kan då stängas om så önskas. Fördelen med denna metod är att större ändringar kan göras utan risk att simuleringen avbryts på grund av något fel, då SIMIT kontrollerar de önskade ändringarna innan dessa implementeras.



Figur 41: De utförda ändringarna är online.

Med detta resultat kan alltså en relativt enkel simulering köras. Simuleringen startas och kopplas upp mot OPC-servern från SIMIT, varpå simuleringen kan köras. I detta fall körs därefter automationssystemet från HMI:t i 800xA genom att påverka flödet in i systemet. I grunden körs systemet i automatläge, vilket anges i funktionsbeskrivningen, men kan även kontrolleras manuellt objekt för objekt. Ändringar i systemet kan därefter göras antingen från automationssystemet eller från simuleringsmodellen, vilket i sin tur påverkar processen.

5 Diskussion

I föregående avsnitt presenterades det resultat som har arbetats fram under projektets gång. En koppling mellan automationsmiljön i ABB 800xA och simuleringsmodellen i SIMIT har kunnat åstadkommas med hjälp av en OPC-koppling. På detta sätt har en simulering genomförts. Simuleringen är uppbyggd kring en processmodell i SIMIT, samt hårdvaruobjekt som byggts upp för att motsvara de verkliga objekten. Denna är kopplad till ett 800xA system som är gjort för att styra inloppet till ett avloppsreningsverk, som är grunden i arbetet.

Simuleringen får anses som delvis lyckad, om än inte särskilt utbyggd. Anledningen till den något småskaliga storleken på simuleringen är främst den sena ändringen i automationsmiljö, som skapade stora tidsbegränsningar på arbetet, trots att ändringen i slutändan var oundviklig. I följande avsnitt kommer det uppnådda resultatet att diskuteras med fokus på områdena tekniska implikationer, lärandemiljö, samt hållbarhetsperpektiv.

5.1 Tekniska implikationer

Med en utbyggd och fungerande simulering kan ett antal tekniska faktorer kring automationsarbete med VA-processer påverkas. Den främsta funktionen som skulle vara önskvärd att åstadkomma är en utbyggd möjlighet att testa ett automationssystem innan driftsättningen är tänkt att ske. Om ökad testning kan ske utan påverkan på det verkliga systemet minskar risken för kostsamma fel, störningar, eller avbrott i processen. Att kunna testa systemets funktion på ett så verklighetstroget sätt som möjligt skulle kunna effektivisera arbetet som sedan sker på plats vid anläggningen. Genom att ha testat såväl att alla grundfunktioner i systemet fungerar, som hur systemet beter sig i samband med olika scenarier, kan förändringar göras redan innan dessa testas på plats. Även olika larm kan testas antingen genom att anpassa simuleringen till ett scenario som bör ge larm, eller att helt enkelt tvinga simuleringen att skicka en felsignal till automationssystemet och iaktta om den aktuella larmsignalen ger det utslag som är tänkt.

De stora svårigheterna för att en så pass extensiv testning ska kunna ske är att det simulerade systemet måste efterlikna det verkliga systemet i väldigt stor utsträckning. Detta går i hög grad att utföra för en hel del system, men ännu så länge finns en del begränsningar i SIMIT som påverkar vad som går att göra eller inte. Två av de stora begränsningarna som påverkar vilka skillnader som uppkommer mellan ett verkligt system och en simulering är att SIMIT i nuläget inte tar hänsyn till längden på tänkta rör, samt att ett system i SIMIT inte kan blanda luft och vatten i samma system. I SIMIT görs kopplingen mellan olika objekt genom att helt enkelt dra streck mellan de aktuella objekten för att visa hur ett tänkt flöde ska förflytta sig genom systemet, vilket kan ses från de tidigare bilderna över processmodellen från projektet. Denna koppling ses då av systemet som omedelbar. Fördröjning av processen kan läggas in i de processobjekt som används, men inte i rören i sig. Detta skapar alltså en stor skillnad mot den verkliga processen då längd och tjocklek på rören givetvis spelar stor roll för hastigheten på systemet. I den simulerade miljön kommer en förändring vid något objekt i systemet omedelbart att påverka nästföljande objekt. Vidare kommer det att fortsätta så genom systemet, vilket inte nödvändigtvis är fallet i verkligheten. En förändring i ett verkligt objekt, så som en ventil, kan dröja innan det påverkar exempelvis en efterföljande tank, beroende på hur långt det är mellan objekten. Påverkan i objekten i sig bör inte vara stor, då systemet i sig ändå bör bete sig på det tänkta sättet, om än något snabbare.

En av de andra svårigheterna är som tidigare nämnts att system i SIMIT inte kan blanda luft och vätska i samma system. Båda dessa medium går att hantera i en simuleringsmodell, men i nuläget går dessa alltså inte att koppla samman. Det får påverkan på system som till exempel den inloppsprocess som har utgjort automationsgrunden till detta arbete. I systemet finns nämligen blåsmaskiner som används för att blåsa luft genom systemet för att avlägsna vätesulfid ur det inkommande vattnet. För dessa blåsmaskiner finns alltså ingen möjlighet att hantera som att de blåser luft i systemet, utan måste istället försöka efterliknas på något annat sätt. För att efterlikna de tryck som ska uppkomma på olika ställen i systemet samt flödets riktning genom systemet kan istället pumpar som hanterar vatten användas. Dessa anpassas då för att

efterlikna att de blåser luft i så stor utsträckning som möjligt, men en skillnad är ofrånkomlig.

5.2 Lärandemiljö

Som redan nämnts en del kan en fullt utbyggd simulering fungera på ett bra sätt för att testa automationssystem innan dessa tas i drift. En utbyggd simulering kan dock även användas som lärandemiljö för de operatörer som sedan jobbar med systemet, även efter att det verkliga systemet är driftsatt. På detta sätt kan operatörer utbildas och testa olika scenarion som kan uppkomma utan att påverka det verkliga systemet. Eftersom alla objekt går att påverka och styra är det ett bra sätt för en operatör att kunna lära känna processen som denne är tänkt att jobba med. På detta sätt går olika troliga scenarion att iscensätta för att operatören ska kunna se hur dessa påverkar systemet och vad som eventuellt behöver åtgärdas. På samma sätt kan olika larmsignaler testas vilket ger operatören en möjlighet att se hur dessa ser ut och beter sig på förhand, och kan då på ett effektivt sätt lära sig att känna igen och hantera dessa.

5.3 Hållbarhetsperspektiv

En annan aspekt som kan påverkas med hjälp av en utökad möjlighet till simulering av automationsprocesser är hållbarheten. Med hjälp av utbyggda simuleringar kan system ändras och testas på ett betydligt enklare sätt än om testning ska ske i samband med en verklig process. På detta sätt kan ett utökat arbete för att optimera de aktuella processerna göras, då fler varianter kan testas och olika variationers påverkan kan studeras. Genom att optimera processer samt minska risken för fel eller andra situationer som kan påverka driften kan dessa processer optimeras ur ett hållbarhetsperspektiv. Detta kan till exempel göras genom att tillförsel av olika kemikalier minimeras för att minska åtgången av dessa, eller att processen optimeras för att ha så låg energiåtgång som möjligt. Detta kan till exempel göras mätbart genom så kallad *benchmarking* av data så som bland annat kvalitetsmått för avloppsvatten, index för driftskostnad, energivillkor för pumpning och luftning, kostnad för extern koltillsats, biogasproduktion och mått på slamproduktion (Vrecko, et al. 2014).

Om denna typ av optimering ska kunna göras måste dock simuleringen stämma väldigt bra överens med det verkliga systemet. Om för stora skillnader finns mellan det verkliga och det simulerade systemet kommer eventuella förändringar och optimeringar inte att kunna ses som relevanta på samma sätt, utan att först göra motsvarande undersökningar på det verkliga systemet. Med andra ord försvinner i stort fördelen med simuleringen. Även om idéer fortfarande kan testas i större utsträckning är fördelen betydligt större ju mer relevant information som kan dras från den simulerade miljön. En sådan utbyggd simulering bör vara möjlig att göra för de flesta processer, men ställer stora krav på den som ska bygga upp simuleringen. Stor kännedom om den verkliga processen krävs och ett stort arbete för att bygga upp systemet på ett sätt som i så stor detalj som möjligt efterliknar verkligheten.

5.4 Kravbild

Som tidigare nämnts finns en del krav på en process och ett automationssystem om dessa ska gå att simulera på ett rimligt sätt med hjälp av SIMIT. Det allra mest grundläggande kravet är att den process som ska simuleras i regel måste vara objektorienterad, det vill säga designad utifrån fokus på de ingående objekten i processen. Då simuleringen byggs upp på detta sätt blir möjligheten att åstadkomma en verklighetstrogen simulering betydligt större om både processen och automationsystemet är uppbyggda på samma sätt. Processer som inte är uppbyggda kring de ingående komponenterna kan bli svåra att simulera på ett verklighetstroget sätt, då skillnaden i hur systemet byggs upp växer.

Utöver uppbyggnaden av systemet finns en del andra krav eller faktorer som påverkar hur verklighetstrogen en simulerad modell kan göras. Simuleringen grundar sig i stort på systemets P&ID, funktionsbeskrivning och fysisk modell, vilket med andra ord innebär att dessa måste finnas att tillgå på ett enkelt sätt. En tydlig bild över systemet behöver kunna skapas för att en simulering ska kunna byggas upp. Ju mer information som finns att tillgå i funktionsbeskrivningen, desto mer kan de simulerade objekten designas

för att efterlikna sina verkliga motsvarigheter. Detta ställer såklart en del krav på den process- eller automationsingenjör som skriver funktionsbeskrivningen då dokumentationen eventuellt blir mer omständlig. Förhoppningen är dock givetvis att fördelarna med den utökade möjligheten till simulering väger upp eventuellt merarbete kring systemets dokumentation.

6 Slutsatser och framtida arbete

Från arbetet kan självklart ett antal slutsatser dras. Först och främst får det sägas att en simulering har gått att uppnå, även om omfattningen är begränsad. Utifrån den definition som tidigare presenterats, det vill säga att skillnaden mellan en digital skugga och en digital tvilling är att data delas både till och från den simulerade modellen (Herwig, et al. 2021) kan det till och med sägas att det i viss mån har skapats en enkel digital tvilling. Modellen är dock inte särskilt utbyggd utan är begränsad till en liten del av ett större avloppsreningsverk. Detta beror till stor del på de svårigheter som har uppkommit under arbetets gång. Även baserat på dessa svårigheter kan en del slutsatser dras. Med start i arbetets början kan slutsatsen dras att om en simulering som efterliknar verkligheten ska kunna göras krävs en tydlig P&ID med alla tilltänkta objekt inritade och namngivna för att undvika förvirring kring vilka objekt som finns var i processen. Namngivningen tjänar även syftet att P&ID:en på ett effektivt sätt kan kopplas ihop med funktionsbeskrivningen, och därigenom även simuleringen. På detta spår kan även slutsatsen dras att en noggrann och utbyggd funktionsbeskrivning krävs för att simuleringen ska kunna efterlikna den verkliga processen och egenskaperna hos de involverade objekten. Utan en riktig funktionsbeskrivning finns inte tillräckligt med kunskap tillgänglig för att en verklighetstrogen simulering ska kunna byggas upp.

Vidare slutsatser kan dras av detta projekt. Den övergripande målsättningen för detta arbete var att implementera en simuleringslösning för ett avloppsreningsverk samt koppla ihop simuleringen med en befintlig automationslösning. Detta får anses som möjligt och utfört. Även om det som tidigare nämnts endast har skapats en modell av en del av avloppsreningsverket har det visats att denna typ av lösning är möjlig. Med mer tillgänglig tid hade simuleringen kunnat byggas ut för att innefatta fler delar av verket. Vidare var det tänkt att det i simuleringen skulle vara möjligt att göra anpassningar i processen med hjälp av SIMIT, där det även skulle finns grafisk presentation av simuleringen, och påverkan skulle visas i operatörsbilder till automationslösningen. Även detta får på samma sätt anses som utfört och fungerande, om än begränsat. Den koppling mellan existerande automationssystem och simuleringsmodell som skulle undersökas kunde utföras med hjälp av en OPC-koppling. I OPC-kopplingen utgjorde i detta fall automationssystemet i ABB 800xA OPC-servern och simuleringsmodellen i sig utgjorde OPC-klienten. För att detta ska gå att utföra måste relevanta IO-adresser gå att hitta, till exempel med hjälp av en OPC-utforskare så som *Matrikon*.

Möjligt framtida arbete kring detta projekt kan till exempel utgöras av att bygga ut simuleringen till att innefatta även de andra delarna av avloppsreningsverket. Detta är ett arbete som kan bygga på samma grund som hittills, och därigenom expanderas. På en högre nivå existerar även framtida arbete för att utveckla SIMIT som program så väl som AFRY:s RDT bibliotek. För att simuleringar i SIMIT ska kunna byggas upp för att i ännu högre grad efterlikna verkligheten bör de två tidigare nämnda svårigheterna lösas. Med andra ord bör hänsyn kunna tas till längd och dimension på de rör som kopplar samman systemen, samt blandningar av luft och vatten bör kunna hanteras i samma system.

7 Referenser

ABB 2015. *Användning – System 800xA.*
https://library.e.abb.com/public/6884a5fc0f5a428dacfec7f33da44862/3BSE036904-600_-_sv_System_800xA_Operations_6_0.pdf. (2024-06-13).

Arjomandi Rad, M. 2022. *Data-driven and real-time prediction models for iterative and simulation-driven design processes.* Jönköping University, Tekniska Högskolan, JTH, Industriell produktutveckling, produktion och design, Konstruktion och produktutveckling.
https://hj.diva-portal.org/smash/get/diva2:1661207/FULLTEXT01.pdf. (2024-06-13).

Gernaey, K.V., Jeppsson, U., Vanrolleghem, P.A. & Copp, J.B. 2014. *Benchmarking of Control Strategies for Wastewater Treatment Plants.* IWA Publishing. London, UK.
https://directory-doabooks-org.ludwig.lub.lu.se/handle/20.500.12854/38129. (2024-06-13).

Goienetxea Uriarte, A. 2019. *Bringing Together Lean, Simulation and Optimization: Defining a framework to support decision-making in system design and improvement.* Högskolan i Skövde, Institutionen för ingenjörsvetenskap, Forskningscentrum för Virtuella system.(Produktion och automatiseringsteknik, Production and Automation Engineering).
https://his.diva-portal.org/smash/get/diva2:1334388/FULLTEXT01.pdf. (2024-06-13).

Herwig, C., Pörtner, R., & Möller, J. (Eds.) 2021. *Digital Twins.*
https://link-springer-com.ludwig.lub.lu.se/content/pdf/10.1007/978-3-030-71660-8.pdf (2024-06-13).

Hoffmann-Walbeck, T. 2022. *Workflow Automation.*
https://link-springer-com.ludwig.lub.lu.se/content/pdf/10.1007/978-3-030-84782-1.pdf (2024-06-13).

Käppala 2024. *Reningsprocessen.* https://www.kappala.se/vad-vi-gor/avloppsrening/reningsprocessen/ (2024-02-08).

Lindblom, E., Samuelsson, O. 2020. *Virtuell driftsättning av styrsystem på reningsverk.* IVL Svenska Miljöinstitutet. https://ivl.diva-portal.org/smash/get/diva2:1552226/FULLTEXT01.pdf. (2024-06-13).

Lv, Z., & Fersman, E. (Eds.) 2022. *Digital Twins.*
https://link-springer-com.ludwig.lub.lu.se/content/pdf/10.1007/978-3-031-11401-4.pdf (2024-06-13).

Myers, M., Brace, C., & Carden, L. (Eds.) 2022. *Intelligent Automation: Bridging the Gap between Business and Academia* (1^{st} ed.). Chapman and Hall/CRC.
https://doi-org.ludwig.lub.lu.se/10.1201/9781003276128 (2024-06-13).

NCC 2024. *Hur kan Sveriges VA-system moderniseras?*
https://www.ncc.se/siteassets/vart-erbjudande/infrastruktur/va-dagvatten/ncc_va_rapport.pdf (2024-02-07).

Ramin, P., Flores-Alsina, X., Topalian, S.O.N., Jeppsson, U. & Gernaey, K. 2022. *Fault detection in a benchmark simulation model for wastewater treatment plants.* Computer Aided Chemical Engineering 49: s. 1363-1368. https://doi.org/10.1016/B978-0-323-85159-6.50227-X.
(https://www.sciencedirect.com/science/article/pii/B978032385159650227X). (2024-06-13).

Ruiz Zúñiga, E. 2020. *Facility layout design with simulation-based optimization: A holistic methodology including process, flow, and logistics requirements in manufacturing.* Högskolan i Skövde, Institutionen för ingenjörsvetenskap, Forskningsmiljön Virtuell produkt- och produktionsutveckling.(Produktion och Automatiseringsteknik, Production and Automation Engineering).
https://his.diva-portal.org/smash/get/diva2:1507434/FULLTEXT02.pdf. (2024-06-13).

Siemens 2013. *SIMIT 7 Profinet IO Gateway.* SIMIT_PROFINET_IO_Gateway_e.pdf (siemens.com).

Siemens 2019. *Engineering Efficiency in the Interaction of PCS 7, PAA and SIMIT. Engineering Efficiency in the Interaction of PCS 7, PAA and SIMIT* (siemens.com).

Siemens 2022. *SIMIT Simulation Platform (V11) Operating Manual.*

Stockholm Vatten och Avfall 2021. *Stockholms framtida avloppsrening.*
https://www.stockholmvattenochavfall.se/projekt/stockholms-framtida-avloppsrening/ (2024-02-08).

Svenskt Vatten 2016. *Produktion av dricksvatten.*
https://www.svensktvatten.se/fakta-om-vatten/dricksvattenfakta/produktion-av-dricksvatten/
(2024-02-08).

Svenskt Vatten 2019. *Vattnets Kretslopp.*
https://www.svensktvatten.se/fakta-om-vatten/vattnets-kretslopp/ (2024-02-07).

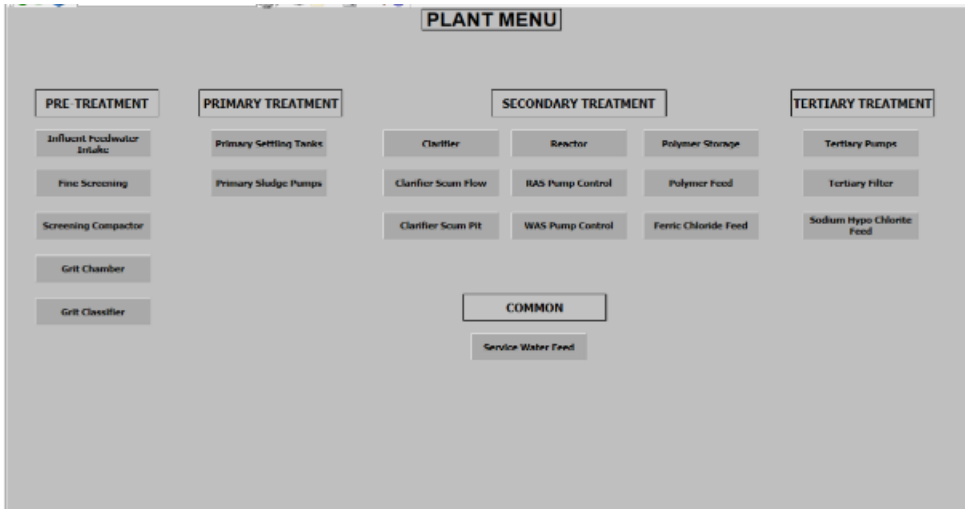
Sveriges Radio 2019. *Vattenbristen oroar – historiskt låga nivåer.* https://sverigesradio.se/artikel/7360659
(2024-02-07).

VA SYD 2023. *Våra avloppsreningsverk.*
https://www.vasyd.se/Artiklar/Avlopp-och-rening/Våra-avloppsreningsverk (2024-02-08).

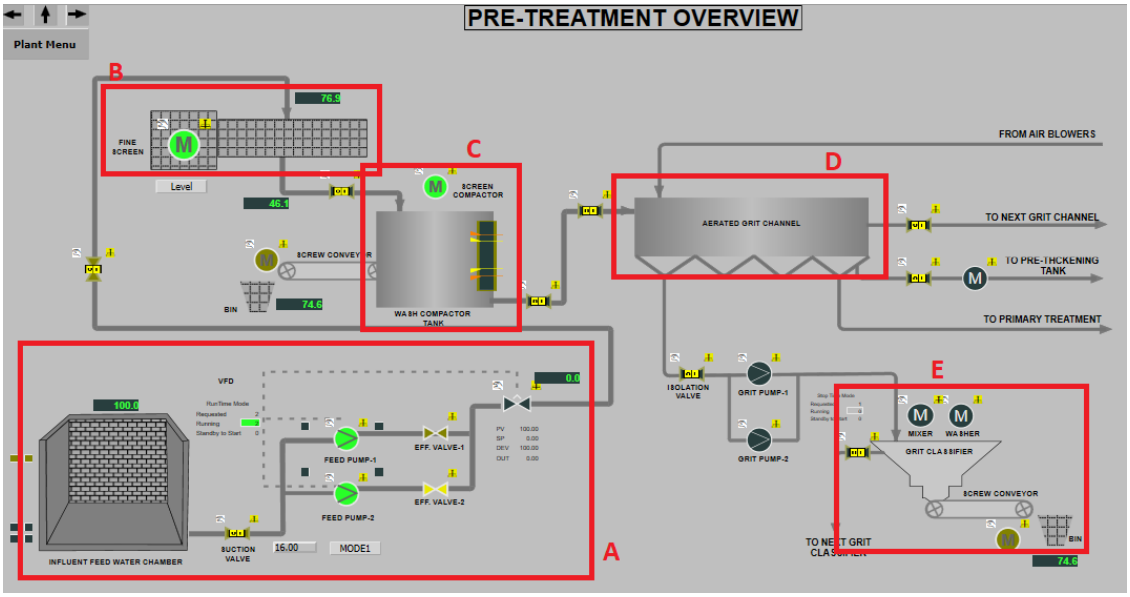
Bilagor

A. Demosystem

I denna bilaga presenteras demosystemets avloppsreningsverk i sin helhet, vilket leder till viss överlappning med tidigare avsnitt. Övergripande består verket av en förbehandling, en primärbehandling, en sekundärbehandling och en tertiärbehandling, vilket kan ses i figur 42. I det som i detta system kallas för förbehandling sker grovreningen genom det första intaget av vattnet in i systemet (Fig. 43, block A). Därefter går vattnet vidare till ett finare filter för ytterligare en silning (Fig. 43, block B). Vidare passerar sedan vattnet genom ett sandfilter (Fig. 43, block C) innan flödet delas upp, där majoriteten av flödet går vidare till primärbehandlingen (Fig. 43, block D). Skräp som har fastnat i de olika filtren sorteras bort under processen och sand och grus som har försvunnit skickas vidare till sortering (Fig. 43, block E). Ytterligare en del av vattnet skickas vidare till ännu ett sandfilter. Denna process kan ses i en överblick i figur 43.



Figur 42: Övergripande vy över verkets ingående delar.

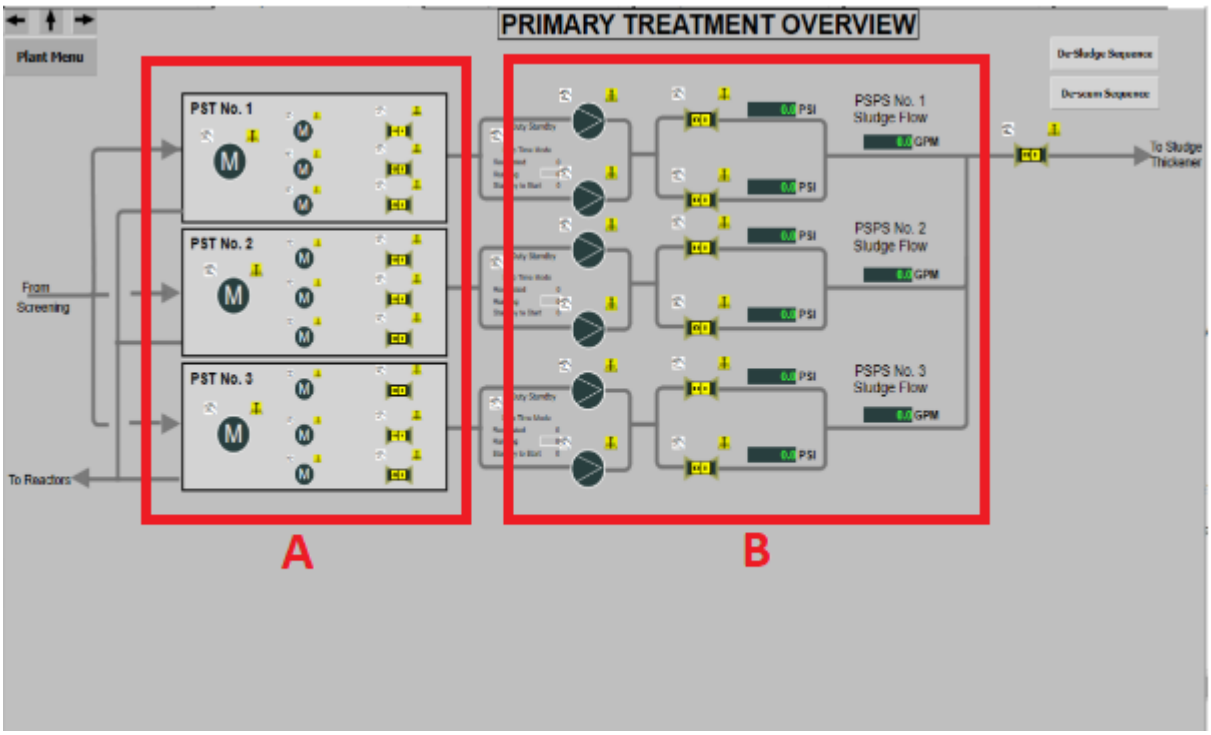


Figur 43: Ingående delar i förbehandlingen. Från bilden kan inloppet ses i block A, nästa filter i block B, *Wash Compactor* i block C, sandfiltret i block D och slutligen sorteringen av sten och grus i block E.

Vattnet passerar som tidigare nämnts vidare till primärbehandlingen. Denna del består av sedimenteringstankar (Fig. 44, block A) och slampumpar (Fig. 44, block B). Precis som i förbehandlingen fokuserar denna fas på att avlägsna grova partiklar och andra större föroreningar från vattnet innan det går vidare till mer avancerade reningstekniker.

Sedimenteringstankarna utgör en viktig del av primärbehandlingen vid ett avloppsreningsverk. Dessa tankar är utformade för att tvinga vattnet att sakta ner och möjliggöra avskiljning av tunga partiklar och sediment som finns i det inkommande vattnet. När vattnet strömmar in i sedimenteringstankarna minskar flödeshastigheten dramatiskt, vilket får partiklarna att sjunka till botten av tankarna genom gravitation, så kallad sedimentering. De avskilda partiklarna och sedimenten bildar ett slamlager på tankens botten, medan det renade vattnet fortsätter vidare till nästa steg i reningsprocessen.

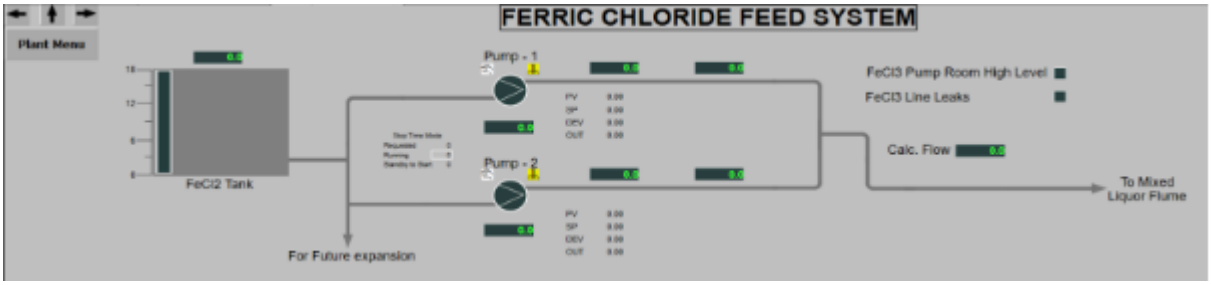
Efter att sedimenteringstankarna har avlägsnat större partiklar och sediment från råvattnet eller avloppsvattnet krävs det att det samlade slammet transporteras vidare för vidare behandling. Till detta används så kallade slampumpar. Dessa pumpar är utformade för att suga upp och pumpa det avskilda slammet från botten av sedimenteringstankarna och transportera det till vidare behandling. En överblick över primärbehandlingen kan ses i figur 44. Från primärbehandlingen transporteras vattnet vidare till den så kallade sekundärbehandlingen.



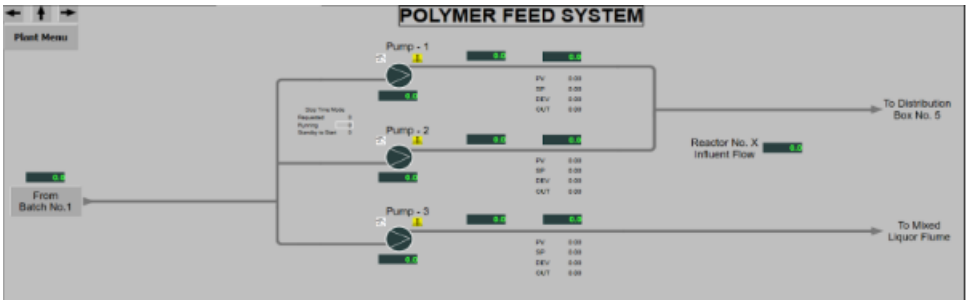
Figur 44: Överblick över primärbehandlingen.

Sekundärbehandlingen utgör nästa steg i det tilltänkta reningsverket och är tänkt att ytterligare rena vattnet från organiskt material och andra föroreningar som inte avlägsnades vid primärbehandlingen. Denna fas omfattar flera komplexa processer och system som samverkar för att säkerställa att vattnet når önskad renhetsgrad innan det släpps ut i miljön eller distribueras som dricksvatten. Några av dessa processer är följande.

Järnkloridmatningssystem och Polymermatningssystem:
Järnklorid och polymerer används ofta i sekundärbehandlingen för att underlätta avskiljning av fina partiklar och organiskt material från vattnet. Järnkloridmatningssystemet som kan ses i figur 45 injicerar järnklorid i vattnet för att underlätta fosfatavskiljning, medan polymermatningssystemet som kan ses i figur 46 tillför polymerer som bidrar till flockning av små partiklar för enklare avskiljning i efterföljande steg.

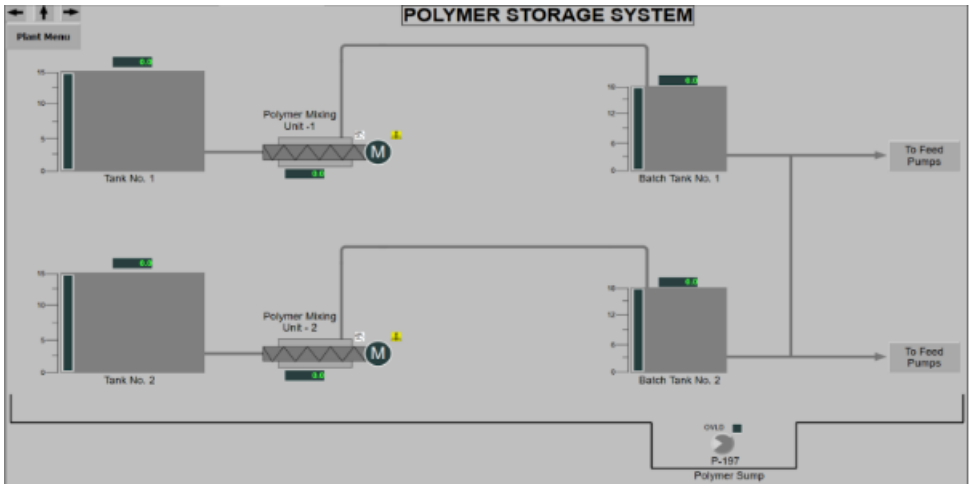


Figur 45: Sekundärbehandling, järnkloridmatningssystem.



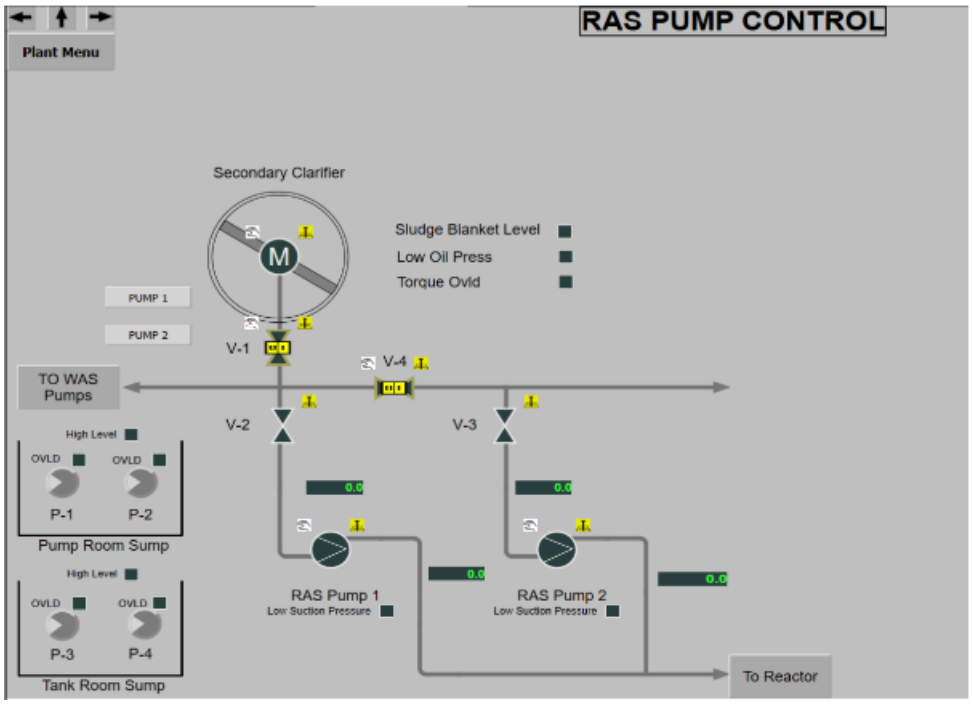
Figur 46: Sekundärbehandling, polymermatningssystem.

Polymerlagringssystem:
Polymerlagringssystemet fungerar som en lagringsplats för polymerer innan de används i behandlingsprocessen. Det är viktigt att polymererna förvaras på rätt sätt för att säkerställa deras effektivitet när de används för att förbättra avvattningen och avskiljningen av partiklar i vattnet. Detta system kan ses i figur 47.

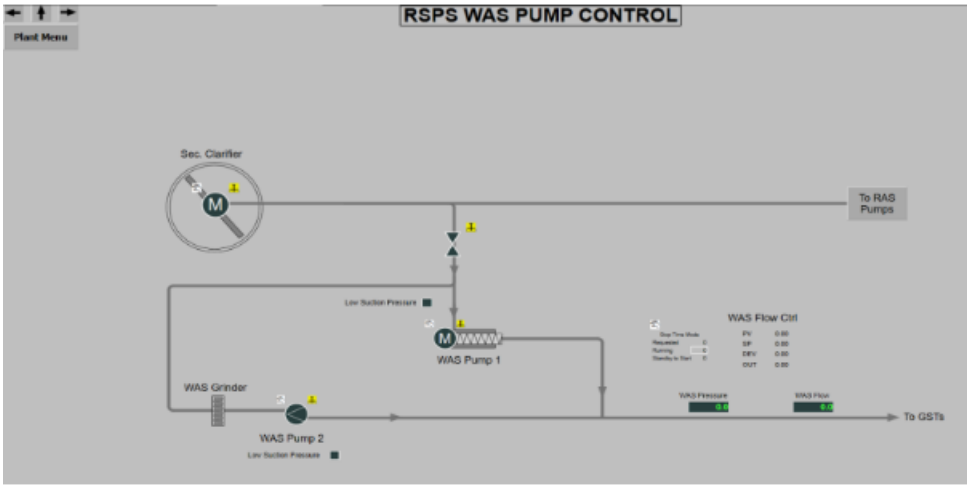


Figur 47: Sekundärbehandling, polymerlagringssystem.

RAS Pumpkontroll och RSPS WAS Pumpkontroll:
RAS (Return-Activated Sludge) och RSPS (Return Sludge Pump Station) WAS (Waste-Activated Sludge) är system som används för att återföra aktivt slam till systemet för ytterligare behandling och för att pumpa ut överskottsslam från processen. RAS pumpkontroll-systemet kan ses i figur 48 och RSPS WAS pumpkontroll-systemet kan ses i figur 49.



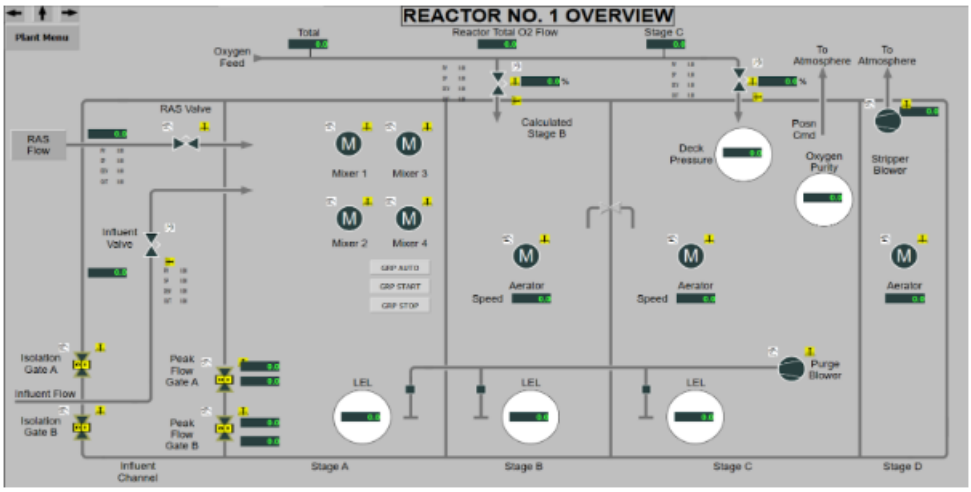
Figur 48: Sekundärbehandling, RAS pumpkontroll.



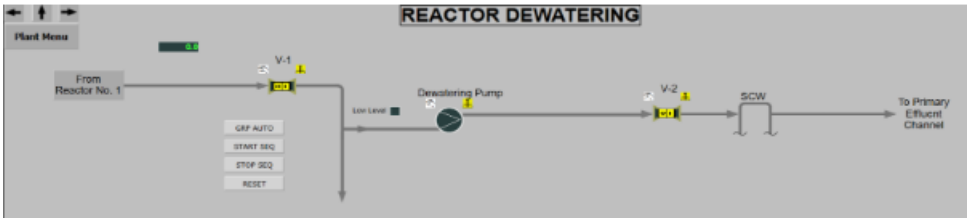
Figur 49: Sekundärbehandling, RSPS WAS pumpkontroll.

Reaktor och Reaktoravvattning:
Den så kallade reaktorn är en central del av sekundärbehandlingen där biologisk nedbrytning av organiskt material sker med hjälp av mikroorganismer. Reaktoravvattningssystemet ansvarar för att avlägsna överskottsslam från reaktorn för att bibehålla en optimal balans mellan slamproduktion och

reningseffektivitet. En överblick över reaktorn kan ses i figur 50 och systemet för reaktoravvattningen kan ses i figur 51.

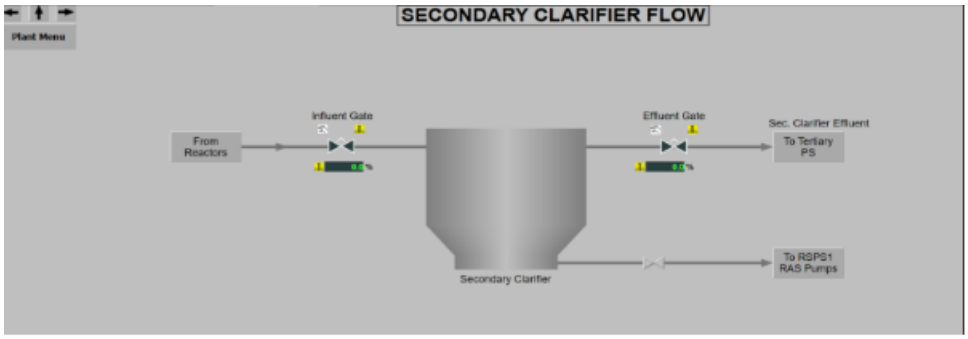


Figur 50: Sekundärbehandling, överblick över en av systemets reaktorer.

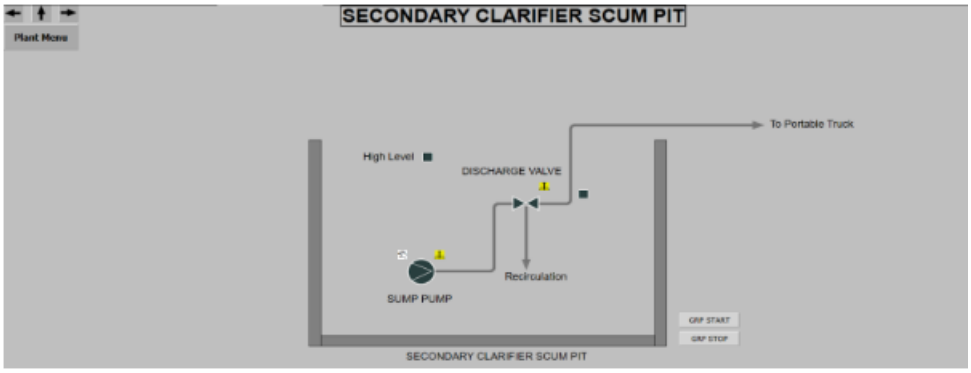


Figur 51: Sekundärbehandling, reaktoravvattning.

Clarifier Flow, Clarifier Scum Pit och Clarifiers:
Clarifier Flow-systemet reglerar flödet av vatten genom så kallade *Clarifiers*, vilket är stora bassänger där avskiljning av fast material från vattnet återigen sker genom sedimentering. Clarifier Scum Pit fångar upp flytande fetter och oljor som avskilts från vattnet. Dessa system kan ses i figur 52 och 53.



Figur 52: Sekundärbehandling, clarifier flow.



Figur 53: Sekundärbehandling, clarifier scum pit.

Sekundärbehandlingen vid ett avloppsreningsverk omfattar alltså flera viktiga processer och system som samverkar för att ytterligare rena vattnet från organiskt material och andra föroreningar. Genom att optimera järnkloridmatning, polymermatning, reaktoravvattning och *clarifierns* funktion säkerställs att avloppsreningsverket producerar rent och säkert vatten som uppfyller de nödvändiga kvalitetsstandarderna för distribution eller utsläpp. Effektiv drift och övervakning av dessa system är avgörande för att säkerställa en effektiv och hållbar vattenrening för samhället och miljön och styrningen av dessa är alltså central. Från sekundärbehandlingen transporteras vattnet vidare till tertiärbehandlingen.

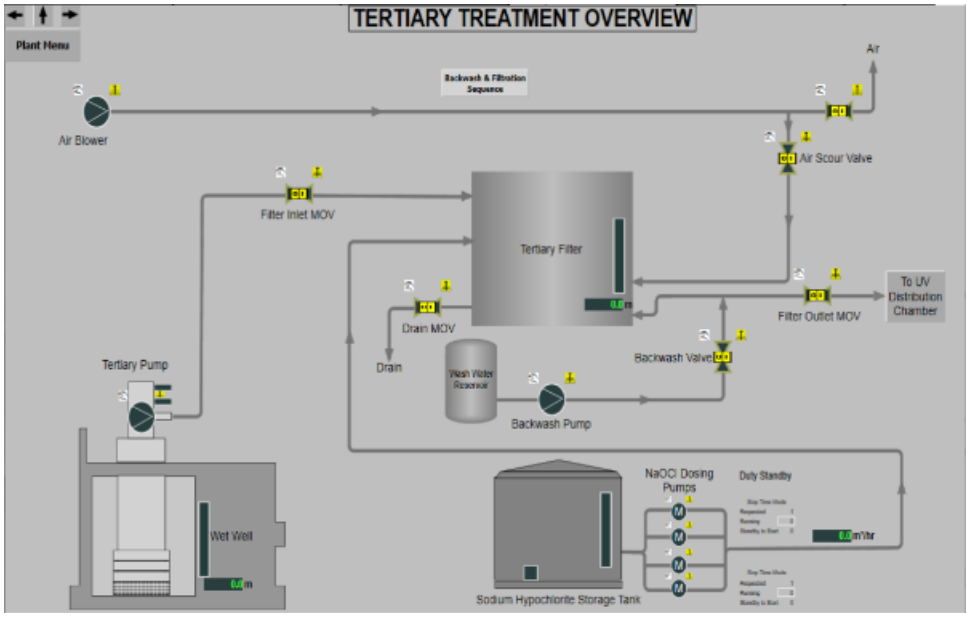
I detta tilltänkta avloppsreningsverk som utgör demoprojektet är tertiärbehandlingen den sista fasen i reningsprocessen och syftar till att ytterligare förbättra vattnets kvalitet genom att avlägsna mikroskopiska partiklar, organiska föroreningar och eventuella kvarvarande oönskade organismer för att säkerställa att det renade vattnet uppfyller höga kvalitetsstandarder. I det aktuella fallet består tertiärbehandlingen av följande processer.

Lagring och Dosering av Natriumhypoklorit:
Natriumhypoklorit är ett vanligt desinfektionsmedel som används för att döda patogener och bakterier som kan finnas kvar i vattnet efter tidigare rening. I tertiärbehandlingen lagras natriumhypoklorit i säkra och kontrollerade behållare och doseras sedan i vattnet med en noggrant beräknad hastighet för att säkerställa en effektiv desinfektion.

Tertiärfilter:
Tertiärfiltreringen används för att avlägsna små partiklar och resterande föroreningar som kan ha överlevt de tidigare reningsstegen. Tertiärfilter är vanligtvis avsedda för finfiltrering och kan vara utformade med olika material och tekniker för att effektivt avskilja även de minsta föroreningarna från vattnet.

Tertiärpumpar:
Tertiärpumparna används för att transportera vattnet genom tertiärfiltreringssystemet och genom desinfektionsprocessen. Dessa pumpar är utformade för att hantera det rena vattnet med hög effektivitet och noggrannhet för att upprätthålla önskad hastighet genom systemet.

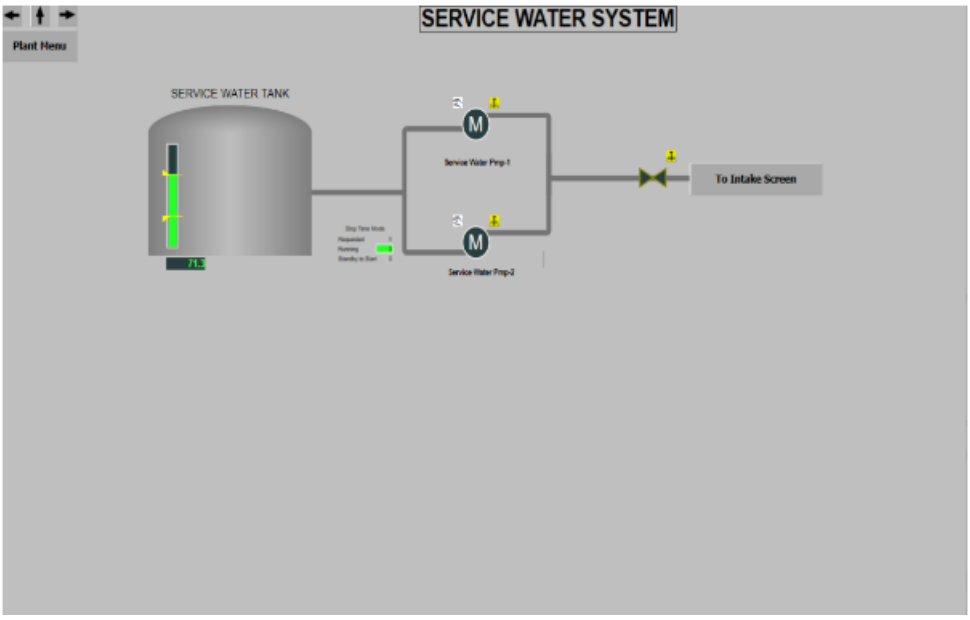
Tertiärbehandlingen vid ett avloppsreningsverk spelar alltså en stor roll för att säkerställa att det renade vattnet uppfyller de krav som ställs på slutprodukten i verket. Genom lagring och dosering av natriumhypoklorit, användning av tertiärfilter och tertiärpumpar säkerställs att eventuella återstående föroreningar avlägsnas och att vattnet desinficeras effektivt innan det släpps ut för användning eller distribution. En överblick över tertiärbehandlingen kan ses i figur 54.



Figur 54: Överblick över tertiärbehandlingen.

Ett system som används i alla delar av verket i demosystemet är servicevattensystemet. Servicevattensystemet vid det tänkta avloppsreningsverket bidrar genom att tillhandahålla det vatten som krävs för att driva och underhålla de olika processer och system som är nödvändiga för att säkerställa en effektiv och kontinuerlig vattenrening. Det servicevatten som levereras används för en rad olika ändamål, inklusive kylning av maskiner, spolning av system, blandning av kemikalier och förberedelse av rengöringslösningar.

Servicevattensystemet tar emot vatten från källor som är separata från det vatten som ska renas. Det kan vara kommunalt vatten eller vatten från en separat brunn eller reservoar. Detta vatten passerar genom en serie filter och behandlingssteg för att säkerställa att det är rent och lämpligt för användning inom reningsprocesserna. Efter att ha blivit behandlat distribueras servicevattnet genom ett nätverk av rörledningar och ventiler till olika delar av avloppsreningsverket. Detta inkluderar tillförsel till pumpstationer, kemikalieförvaringstankar, kylsystem, tvättstationer och andra anläggningar där det behövs för drift och underhåll. Servicevattensystemet övervakas kontinuerligt för att säkerställa att tillräckliga mängder vatten levereras till de olika delarna av avloppsreningsverket. Nivågivare, tryckmätare och andra sensorer används för att övervaka systemets prestanda och identifiera eventuella problem eller avvikelser som kan kräva åtgärd. Regelbundet underhåll av servicevattensystemet är avgörande för att säkerställa dess pålitlighet och effektivitet. Detta kan inkludera rengöring av filter, byte av ventiler och rörledningar, samt kalibrering av övervakningsutrustning. Genom att upprätthålla en väl fungerande servicevattenförsörjning kan avloppsreningsverket fortsätta att fungera smidigt och effektivt. Servicevattensystemet i det aktuella systemet kan ses i figur 55.

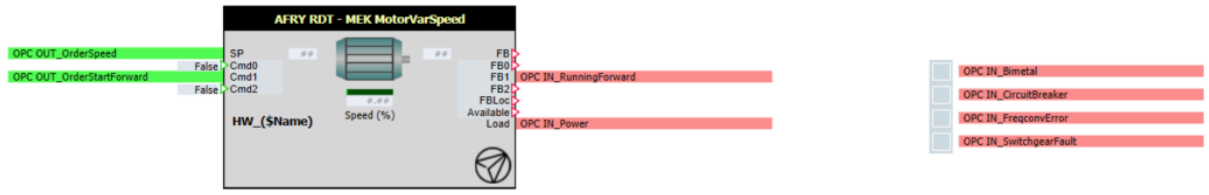


Figur 55: Servicevattensystem.

På detta sätt kan ett system för ett avloppsreningsverk vara uppbyggt i ABB 800xA. De ingående komponenterna och sambandet mellan dessa är i detta exempelsystem uppbyggda utifrån diagram som visar hur systemet ska bete sig. Denna typ av programmering går att göra på ett antal olika sätt, där diagram är ett av dessa. Ett annat, ofta mer vanligt sätt, är att programmera komponenterna och systemet i så kallad strukturerad text. Detta sätt är ofta lättare att förstå för en utomstående som studerar systemet då det textbaserade upplägget ofta ger en tydligare bild över ingående funktioner. Systemets HMI för de olika delarna är designat av ABB som ett exempel. Då ett systems HMI ofta programmeras och ritas av de inblandade ingenjörerna kan dessa se olika ut beroende på vilket projekt det är, men något i stil med ABBs exempel är vanligt.

B. Uppbyggnad av templates

För att de simulerade objekten ska uppföra sig på det sätt som de är tänkt och motsvara de verkliga objekten måste deras beteende programmeras och styras. De objekt som används till detta projekt grundar sig i templates som i stort sett är färdigskapade av AFRY:s RDT-avdelning och samlade i AFRY:s RDT Runtime bibliotek. Dessa templates anpassas sedan för att passa de objekt som ska skapas i processmodellen. Till de templates som används kopplas sedan de inputs och outputs som är kopplade till objektet och som kommer att vara grunden för OPC-kopplingen. Den template som används för komponenttypen MotorAnalog kan ses i figur 56.



Figur 56: Template för komponenttypen MotorAnalog.

Från figuren kan ses att två olika *OUT* signaler är kopplade mot ingångarna *SP* och *Cmd1* på vänster sida av templaten. Ingången *SP* hanterar en analog signal och är därför kopplad mot IO-signalen *OrderSpeed*. På motsvarande sätt hanterar ingången *Cmd1* en *Boolean* och är därför kopplad mot signalen *OrderStartForward* som endast kan vara *TRUE* eller *FALSE*. På utgångarna på högra sidan kopplas de insignaler som skickas från den simulerade komponenten till styrningen, som i detta fall är kopplade mot *FB1* och *Load*. *FB1* hanterar signaltypen *Bool* och är därför kopplad mot IO-signalen *RunningForward* som även den endast kan vara *TRUE* eller *FALSE*, medan *Load* hanterar en analog signal och därför är kopplad mot *Power*. Längst till höger i bilden ses de signaler som inte är direkt kopplade in eller ut ur komponenten, i detta fall olika larmsignaler. Dessa kopplas istället mot switchar som kan hanteras separat, och matchas sedan mot motsvarande IO-signaler i OPC-kopplingen.

HIERARCHY	TEMPLATE	CHART	OPC	IN_Bimetal	IN_CircuitBreaker	IN_FreqconvError	IN_Power
Generated Hardware	MotorAnalog	Motor	OPC	IP12:IO.IN_Bimetal.IOValue	IP12:IO.IN_CircuitBreaker.IOValue	IP12:IO.IN_FreqconvError.IOValue	IP12:IO.IN_Power.IOValue
Generated Hardware	MotorAnalog	Motor	OPC	IP22:IO.IN_Bimetal.IOValue	IP22:IO.IN_CircuitBreaker.IOValue	IP22:IO.IN_FreqconvError.IOValue	IP22:IO.IN_Power.IOValue
Generated Hardware	MotorAnalog	Motor	OPC	IP32:IO.IN_Bimetal.IOValue	IP32:IO.IN_CircuitBreaker.IOValue	IP32:IO.IN_FreqconvError.IOValue	IP32:IO.IN_Power.IOValue

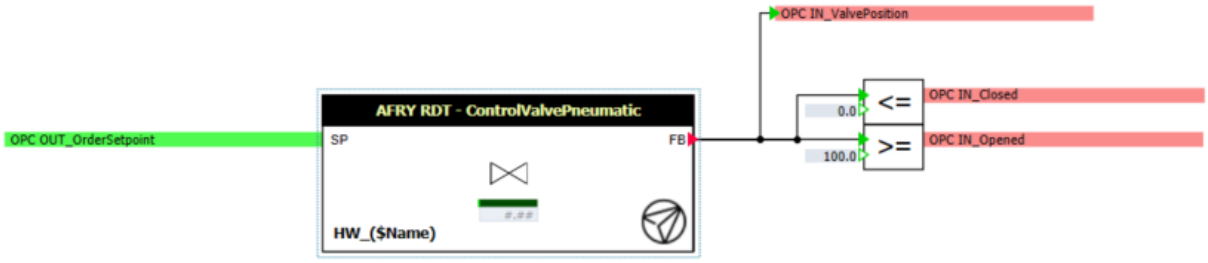
IN_RunningForward	IN_SwitchgearFault	Name	OUT_OrderSpeed	OUT_OrderStartForward
IP12:IO.IN_RunningForward.IOValue	IP12:IO.IN_SwitchgearFault.IOValue	IP12	IP12:IO.OUT_OrderSpeed.IOValue	IP12:IO.OUT_OrderStartForward.IOValue
IP22:IO.IN_RunningForward.IOValue	IP22:IO.IN_SwitchgearFault.IOValue	IP22	IP22:IO.OUT_OrderSpeed.IOValue	IP22:IO.OUT_OrderStartForward.IOValue
IP32:IO.IN_RunningForward.IOValue	IP32:IO.IN_SwitchgearFault.IOValue	IP32	IP32:IO.OUT_OrderSpeed.IOValue	IP32:IO.OUT_OrderStartForward.IOValue

Figur 57: Motvarande rader för MotorAnalog i Excel.

Templaten i figur 56 exporteras sedan till Excel där programmet genererar de kolumner som behöver fyllas i för att kunna använda templaten för att generera de simulerade objekten i ett senare skeda. Dessa rader fylls sedan i enligt exemplet i figur 57. På respektive plats fylls namnet i på den OPC-tagg som senare kommer användas för att koppla SIMIT mot styrningen i 800xA. Taggarnas namn har hittats och kontrollerats via Matrikon enligt vad som visats tidigare i rapporten.

I figur 58 kan templaten för komponenttypen ValveAnalog ses. IO-signalen *OrderSetpoint* är kopplad mot ingången *SP* på vänstersidan av templaten som hanterar en analog signal. På höger sida har templaten

modifierats något, då endast en feedbackutgång existerar. *FB* kopplingen hanterar en analog signal och kan därför hantera IO-signalen *ValvePosition* med en direkt koppling. För att hantera IO-signalerna *Opened* och *Closed* skapas en typ av switch som kan översätta de binära signalerna till den analoga kopplingen. Denna skapas med hjälp av jämförelseelement från SIMIT:s standardbibliotek och jämför hur den analoga signalen är jämfört med de gränsvärden som har angetts. Denna template används på samma sätt som tidigare för att sedan generera Excelraderna som kan ses i figur 59.

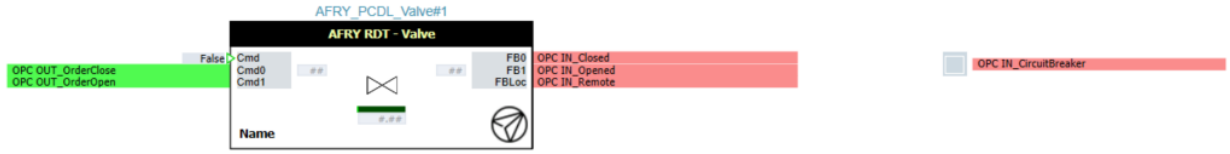


Figur 58: Template för komponenttypen ValveAnalog.

HIERARCHY	TEMPLATE	CHART	OPC	IN_Closed	IN_Opened	IN_ValvePosition	HW_($Name)	OUT_OrderSetpoint
	ValveAnalog			IO11_SR01:IO.IN_Closed	IO11_SR01:IO.IN_Opened	IO11_SR01:IO.IN_ValvePosition	IO11_SR01	IO11_SR01:IO.OUT_OrderSetpoint
	ValveAnalog			IO21_SR01:IO.IN_Closed	IO21_SR01:IO.IN_Opened	IO21_SR01:IO.IN_ValvePosition	IO21_SR01	IO21_SR01:IO.OUT_OrderSetpoint
	ValveAnalog			IO31_SR01:IO.IN_Closed	IO31_SR01:IO.IN_Opened	IO31_SR01:IO.IN_ValvePosition	IO31_SR01	IO31_SR01:IO.OUT_OrderSetpoint

Figur 59: Motvarande rader för ValveAnalog i Excel.

Templaten för komponenttypen ValveDigital kan ses i figur 60. De binära IO-signalerna *OrderClose* och *OrderOpen* är kopplade mot ingångarna *Cmd0* och *Cmd1* som båda två hanterar signaltypen *Boolean*. På motsvarande sätt är IO-signalerna *Closed*, *Opened* och *Remote* kopplade mot utgångarna *FB0*, *FB1* och *FBLoc* som alla hanterar signaltypen *Boolean*. Ytterligare en signal är kopplad mot en switch som hanterar den larmsignal som är kopplad mot IOn. Raderna för motsvarande Excel kan ses i figur 61.

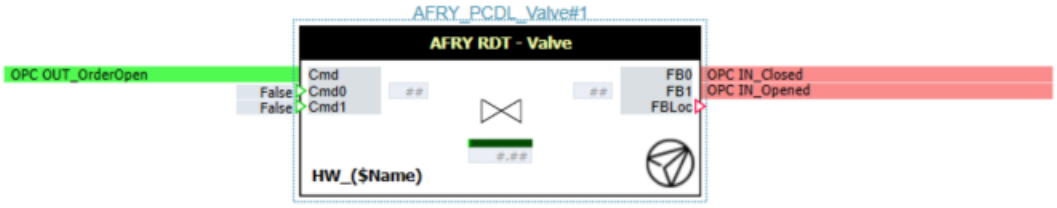


Figur 60: Template för komponenttypen ValveDigital.

HIERARCHY	TEMPLATE	CHART	OPC	IN_CircuitBreaker	IN_Closed	IN_Opened	IN_Remote	HW_($Name)	OUT_OrderClose	OUT_OrderOpen
	ValveDigital			IP12_SV01:IO.IN_CircuitBreaker	IP12_SV01:IO.IN_Closed	IP12_SV01:IO.IN_Opened	IP12_SV01:IO.IN_Remote	IP12_SV01	IP12_SV01:IO.OUT_OrderClose	IP12_SV01:IO.OUT_OrderOpen
	ValveDigital			IQ10_SV03:IO.IN_CircuitBreaker	IQ10_SV03:IO.IN_Closed	IQ10_SV03:IO.IN_Opened	IQ10_SV03:IO.IN_Remote	IQ10_SV03	IQ10_SV03:IO.OUT_OrderClose	IQ10_SV03:IO.OUT_OrderOpen
	ValveDigital			IP22_SV01:IO.IN_CircuitBreaker	IP22_SV01:IO.IN_Closed	IP22_SV01:IO.IN_Opened	IP22_SV01:IO.IN_Remote	IP22_SV01	IP22_SV01:IO.OUT_OrderClose	IP22_SV01:IO.OUT_OrderOpen
	ValveDigital			IQ20_SV03:IO.IN_CircuitBreaker	IQ20_SV03:IO.IN_Closed	IQ20_SV03:IO.IN_Opened	IQ20_SV03:IO.IN_Remote	IQ20_SV03	IQ20_SV03:IO.OUT_OrderClose	IQ20_SV03:IO.OUT_OrderOpen
	ValveDigital			IP32_SV01:IO.IN_CircuitBreaker	IP32_SV01:IO.IN_Closed	IP32_SV01:IO.IN_Opened	IP32_SV01:IO.IN_Remote	IP32_SV01	IP32_SV01:IO.OUT_OrderClose	IP32_SV01:IO.OUT_OrderOpen
	ValveDigital			IQ30_SV03:IO.IN_CircuitBreaker	IQ30_SV03:IO.IN_Closed	IQ30_SV03:IO.IN_Opened	IQ30_SV03:IO.IN_Remote	IQ30_SV03	IQ30_SV03:IO.OUT_OrderClose	IQ30_SV03:IO.OUT_OrderOpen

Figur 61: Motvarande rader för ValveDigital i Excel.

Templaten för komponenttypen ValveDigitalSmall kan ses i figur 62. Denna template är upplagd på samma sätt som den för den större varianten ValveDigital, men med färre kopplade IO-signaler. Denna komponent har endast en signal kopplad mot ingångarna på vänstersidan, vilket gör att ingången *Cmd* används. På andra sidan är signalerna *Closed* och *Opened* kopplade precis som tidigare. För denna komponent används inte de tidigare signalerna *Remote* och *CircuitBreaker* och behöver därför inte tas med. De aktuella raderna i Excel kan ses i figur 63.

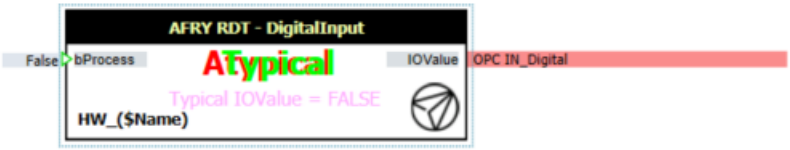


Figur 62: Template för den mindre komponenttypen ValveDigital.

HIERARCHY	TEMPLATE	CHART	OPC	($FBConf)	IN_Closed	IN_Opened	HW_($Name)	OUT_OrderOpen
	ValveDigitalSmall			11	IO11_SV01:IO.IN_Closed	IO11_SV01:IO.IN_Opened	IO11_SV01	IO11_SV01:IO.OUT_OrderOpen
	ValveDigitalSmall			11	IO21_SV01:IO.IN_Closed	IO21_SV01:IO.IN_Opened	IO21_SV01	IO21_SV01:IO.OUT_OrderOpen
	ValveDigitalSmall			11	IO31_SV01:IO.IN_Closed	IO31_SV01:IO.IN_Opened	IO31_SV01	IO31_SV01:IO.OUT_OrderOpen
	ValveDigitalSmall			11	IB11_PV14:IO.IN_Closed	IB11_PV14:IO.IN_Opened	IB11_PV14	IB11_PV14:IO.OUT_OrderOpen
	ValveDigitalSmall			11	IB21_PV24:IO.IN_Closed	IB21_PV24:IO.IN_Opened	IB21_PV24	IB21_PV24:IO.OUT_OrderOpen
	ValveDigitalSmall			11	IB31_PV34:IO.IN_Closed	IB31_PV34:IO.IN_Opened	IB31_PV34	IB31_PV34:IO.OUT_OrderOpen

Figur 63: Motvarande rader för ValveDigital i Excel.

En mindre template kan ses i figur 64, vilken hanterar komponenttypen DigIn. Denna template har bara en IO-signal kopplad, *Digital* som är kopplad mot utgången *IOValue* som hanterar signaler av signaltypen *Boolean*. Denna komponent används för att hantera digitala signaler. Motsvarande rader i Excel kan ses i figur 65.

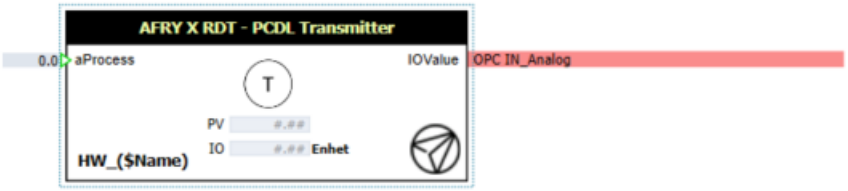


Figur 64: Template för komponenttypen DigIn.

HIERARCHY	TEMPLATE	CHART	OPC	HW_($Name)	IN_Digital
	DigIn			IB11_UL01	IB11_UL01:IO.IN_Digital
	DigIn			IB21_UL01	IB21_UL01:IO.IN_Digital
	DigIn			IB31_UL01	IB31_UL01:IO.IN_Digital

Figur 65: Motvarande rader för DigIn i Excel.

Ytterligare en mindre template kan ses i figur 66, vilken hanterar komponenttypen AnaIn. Denna bygger på en template för en transmitter och har även den bara en IO-signal kopplad, *Analog* som är kopplad mot utgången *IOValue* som hanterar signaler av signaltypen *Analog*. Denna komponent används för att hantera analoga signaler. Motsvarande rader i Excel kan ses i figur 67. I denna figur kan även de angivna min- och maxvärdena för de komponenter som kommer att genereras ses.

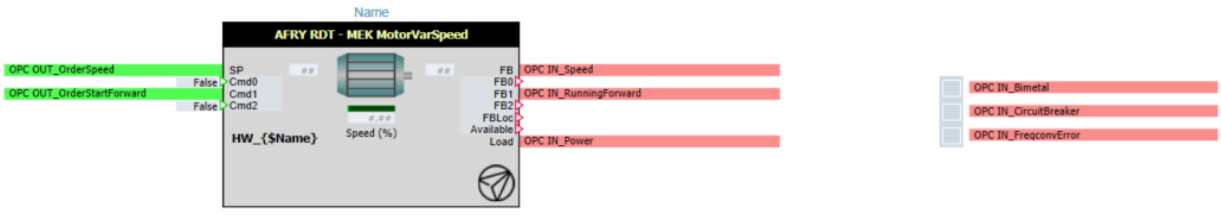


Figur 66: Template för komponenttypen AnaIn.

HIERARCHY	TEMPLATE	CHART	OPC	Enhet	HW_($Name)	IN_Analog	Max	Min
	AnaIn			1000 l/h	IO11_GF01	IO11_GF01:IO.IN_Analog	0	200000
	AnaIn			1000 l/h	IO21_GF01	IO21_GF01:IO.IN_Analog	0	200000
	AnaIn			1000 l/h	IO31_GF01	IO31_GF01:IO.IN_Analog	0	200000

Figur 67: Motvarande rader för AnaIn i Excel.

En mer utbyggd template kan ses i figur 68. Denna template är byggd för komponenttypen MotorBlasMaskin, som kommer att ha två objekt i processmodellen. Templaten har två IO-signaler kopplade på vänstersidan, den analoga signalen *OrderSpeed* som är kopplad mot ingången *SP* samt den binära signalen *OrderStartForward* som är kopplad till ingången *Cmd1*. På högersidan finns tre insignaler kopplade, de analoga signalerna *Speed* och *Power* som är kopplade mot *FB* respektive *Load* som båda två hanterar analoga signaler. Även den binära signalen *RunningForward* är kopplad mot utgången *FB1*, som hanterar signaler av typen *Boolean*. Vidare finns ytterligare tre binära signaler kopplade mot brytare i högra delen av templaten, *Bimetal*, *CircuitBreaker* och *FreqconvError* som hanterar de olika larmsignaler som är kopplade till denna typ av objekt. Motsvarande rader i Excel kan ses i figur 69.



Figur 68: Template för komponenttypen MotorBlasMaskin.

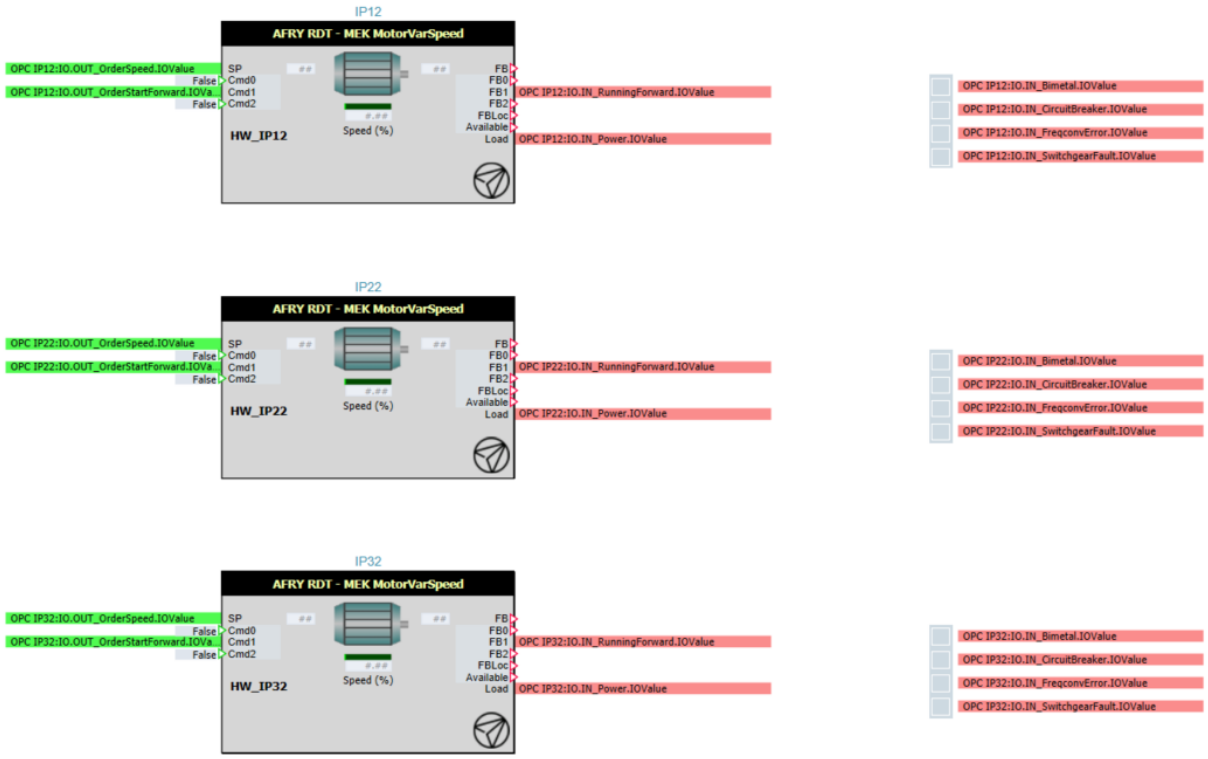
HIERARCHY	TEMPLATE	CHART	OPC	IN_Bimetal	IN_CircuitBreaker	IN_FreqconvError
Generated Hardware	MotorBlasMaskin	MotorBlasMaskin	OPC	IL11:IO.IN_Bimetal.IOValue	IL11:IO.IN_CircuitBreaker.IOValue	IL11:IO.IN_FreqconvError.IOValue
Generated Hardware	MotorBlasMaskin	MotorBlasMaskin	OPC	IL21:IO.IN_Bimetal.IOValue	IL21:IO.IN_CircuitBreaker.IOValue	IL21:IO.IN_FreqconvError.IOValue

IN_Power	IN_RunningForward	IN_Speed	Name	OUT_OrderSpeed	OUT_OrderStartForward
IL11:IO.IN_Power.IOValue	IL11:IO.IN_RunningForward.IOValue	IL11:IO.IN_Speed.IOValue	IL11	IL11:IO.OUT_OrderSpeed.IOValue	IL11:IO.OUT_OrderStartForward.IOValue
IL21:IO.IN_Power.IOValue	IL21:IO.IN_RunningForward.IOValue	IL21:IO.IN_Speed.IOValue	IL21	IL21:IO.OUT_OrderSpeed.IOValue	IL21:IO.OUT_OrderStartForward.IOValue

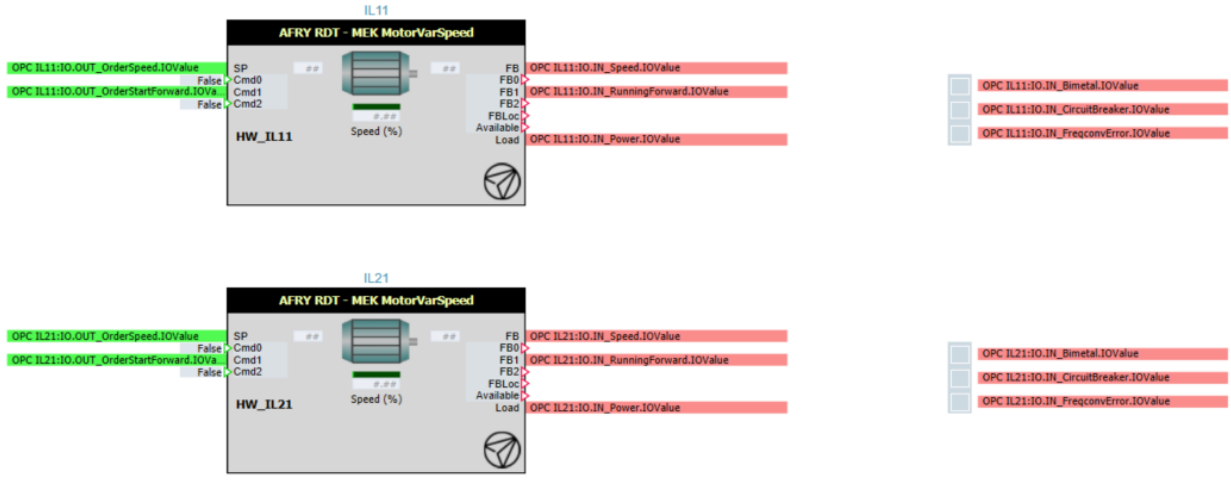
Figur 69: Motvarande rader för MotorBlasMaskin i Excel.

C. Genererade objekt

Utifrån de templates och motsvarande rader i Excel som visats i föregående bilaga kan sedan de objekt som utgör den faktiska modellen genereras. Detta görs genom den inbyggda SIMIT funktionen *Instantiate templates*, som importerar en fil från Excel och genom denna genererar de objekt som skrivits in i filen. I filen skrivs var i hierarkin komponenterna ska genereras, vilken template de ska utgå ifrån, och vilka parametrar som ska skapas. Dessa parametrar läses sedan in till varje komponent. Varje rad i Excelfilen genererar ett objekt. Då det i filen anges vilket så kallat *Chart* komponenterna ska skapas i kan samtliga komponenter egentligen genereras från samma fil. I detta projekt har dock varje template gjorts till en separat fil som sedan har använts till att generera respektive komponenter. I figur 70 kan de genererade objekten av typen *MotorAnalog* ses. Dessa objekt motsvarar stenpumparna *IP12*, *IP22* och *IP32* från processmodellen. I figuren kan de IO-adresser som är kopplade till de specifika signalerna ses. Motsvarande genererade objekt av typen *MotorBlasMaskin* kan ses i figur 71. I denna genereras objekten *IL11* och *IL21* som motsvarar blåsmaskinerna i processmodellen.



Figur 70: Genererade objekt av typen MotorAnalog.



Figur 71: Genererade objekt av typen MotorBlasMaskin.

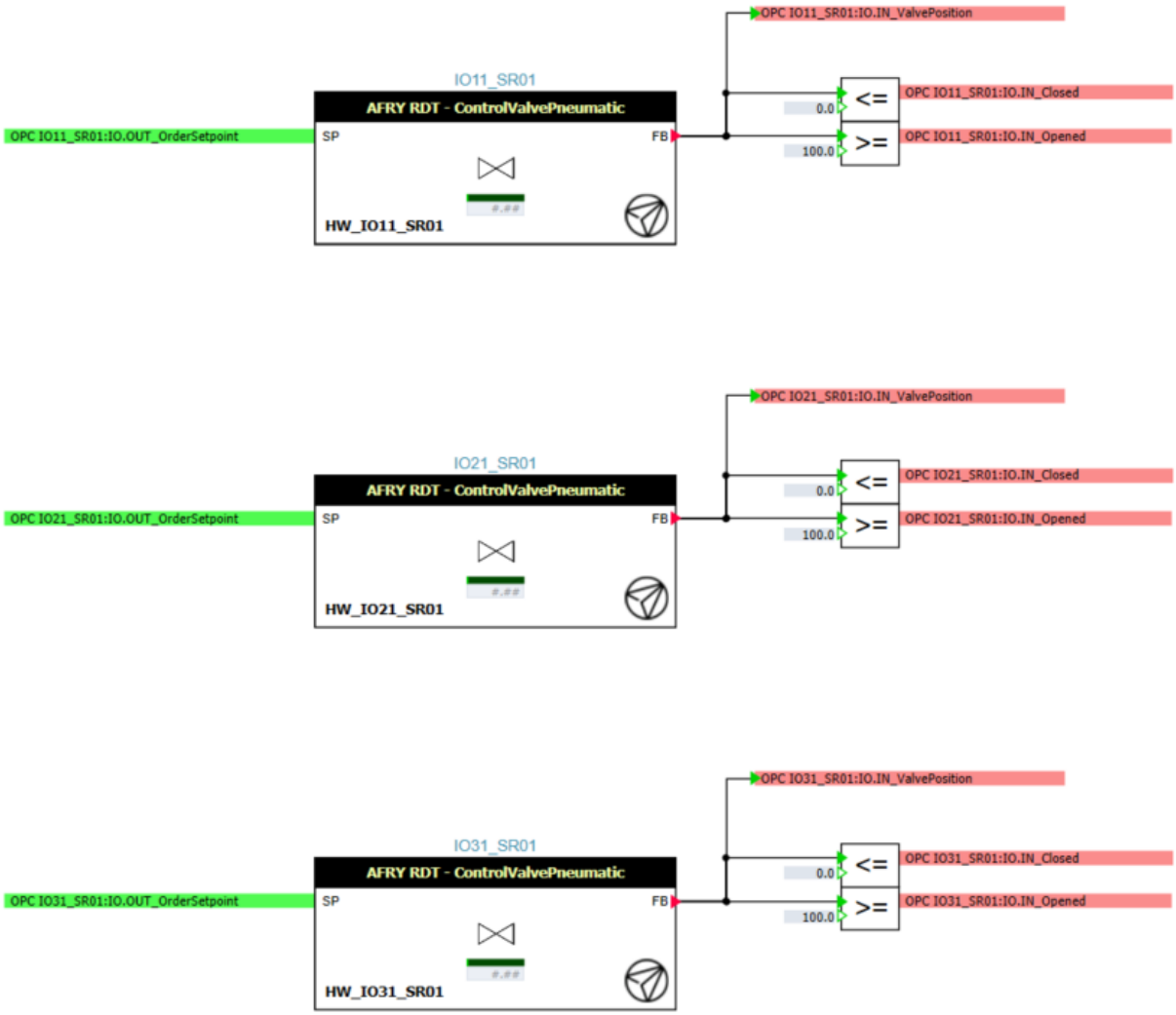
I nästa figur, figur 72, kan de genererade objekten av typen *ValveDigital* ses. Jämfört med de två tidigare templaten kan det ses att denna komponenttyp har ett antal fler objekt. Det är i dessa fall den stora vinningen med detta tillvägagångssätt kommer fram. Vid endast ett fåtal objekt skulle dessa kunna skapas varje objekt för sig, men när antalet objekt ökar skulle detta jobb bli tidskrävande. Även nu är skapandet av objekt en av de mest tidskrävande processerna rörande simuleringsdelen av projektet, men då objekten kan genereras från listorna i Excel minskar tiden avsevärt. I samband med ännu större projekt kan dessa genererade objekt vara väldigt många fler än vad som är aktuellt i detta projekt och i de fallen är vinningen ännu större. De objekt som genereras i denna lista motsvarar inloppsventilerna *IQ10_SV03*, *IQ20_SV03* och *IQ30_SV03*, samt automatventilerna *IP12_SV01*, *IP22_SV01* och *IP32_SV01*.

Figur 72: Genererade objekt av typen ValveDigital.

Även de genererade objekten av typen *ValveDigitalSmall* är flera till antalet. Denna komponenttyp har relativt sett färre signaler jämfört med den större varianten *ValveDigital* och skulle därför återigen ha kunnat genereras med varje objekt för sig. Med hjälp av templatens användning ges dock en annan möjlighet att överblicka de parametrar som anges och fylls i listan i Excel. På detta sätt genereras rätt parametrar med rätt IO-adresser. Objekten som genereras av denna typ är spolventilerna *IB11_PV14*, *IB11_PV24* och *IB11_PV34*, samt automatventilerna *IO11_SV01*, *IO21_SV01* och *IO31_SV01*. Dessa objekt kan ses i figur 73.

Figur 73: Genererade objekt av typen ValveDigitalSmall.

I figur 74 kan objekten av typen *ValveAnalog* ses. Dessa objekt motsvarar de tre reglerventilerna *IO11_SR01*, *IO21_SR01* och *IO31_SR01* från processmodellen. Dessa är som tidigare nämnts byggda kring RDT komponenten *ControlValvePneumatic*, som i grunden är byggt för att representera en luftstyrd reglerventil. I detta fall spelar typen av ventil egentligen ingen roll, ventilen hade med andra ord lika gärna kunnat vara en elektrisk ventil, då det som programmeras i detta projekt endast styr öppningsgraden på ventilen via IO-signalerna.

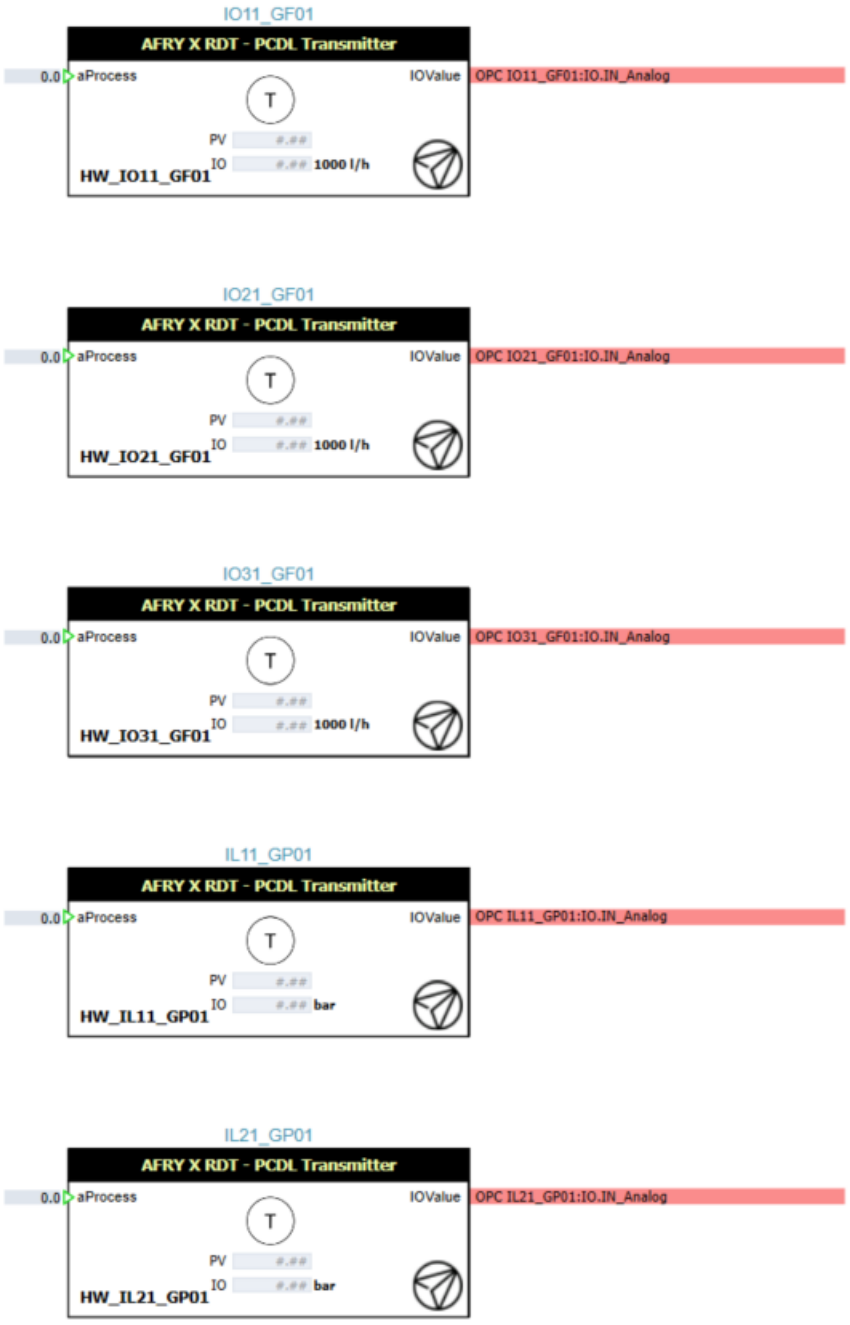


Figur 74: Genererade objekt av typen ValveAnalog.

De tre *DigIn* objekten som genererats kan ses i figur 75. Dessa är kopplade till nivågivarna *IB11_ UL01*, *IB21_ UL01* och *IB31_ UL01* i de tre tankarna *IB11*, *IB21* och *IB31*. Nivågivarna är kopplade till det överflödeslarm som signalerar att nivån i tankarna har passerat en angiven nivå. I det verkliga verket reglerar nivågivarna hur mycket vatten som pumpas in till verket från de externa pumpstationerna. Då pumpstationerna är externa, och alltså inte är programmerade i samma 800xA miljö går denna koppling dock inte att göra i detta projekt via de IO-signaler som finns tillgängliga. Totalflödet in i systemet styrs istället manuellt och påverkar därigenom hur processen beter sig. Om pumpstationerna hade varit direkt kopplade till samma miljö som styrningen av verket hade denna rimligtvis också kunnat simulerats ihop med det övriga inloppet.

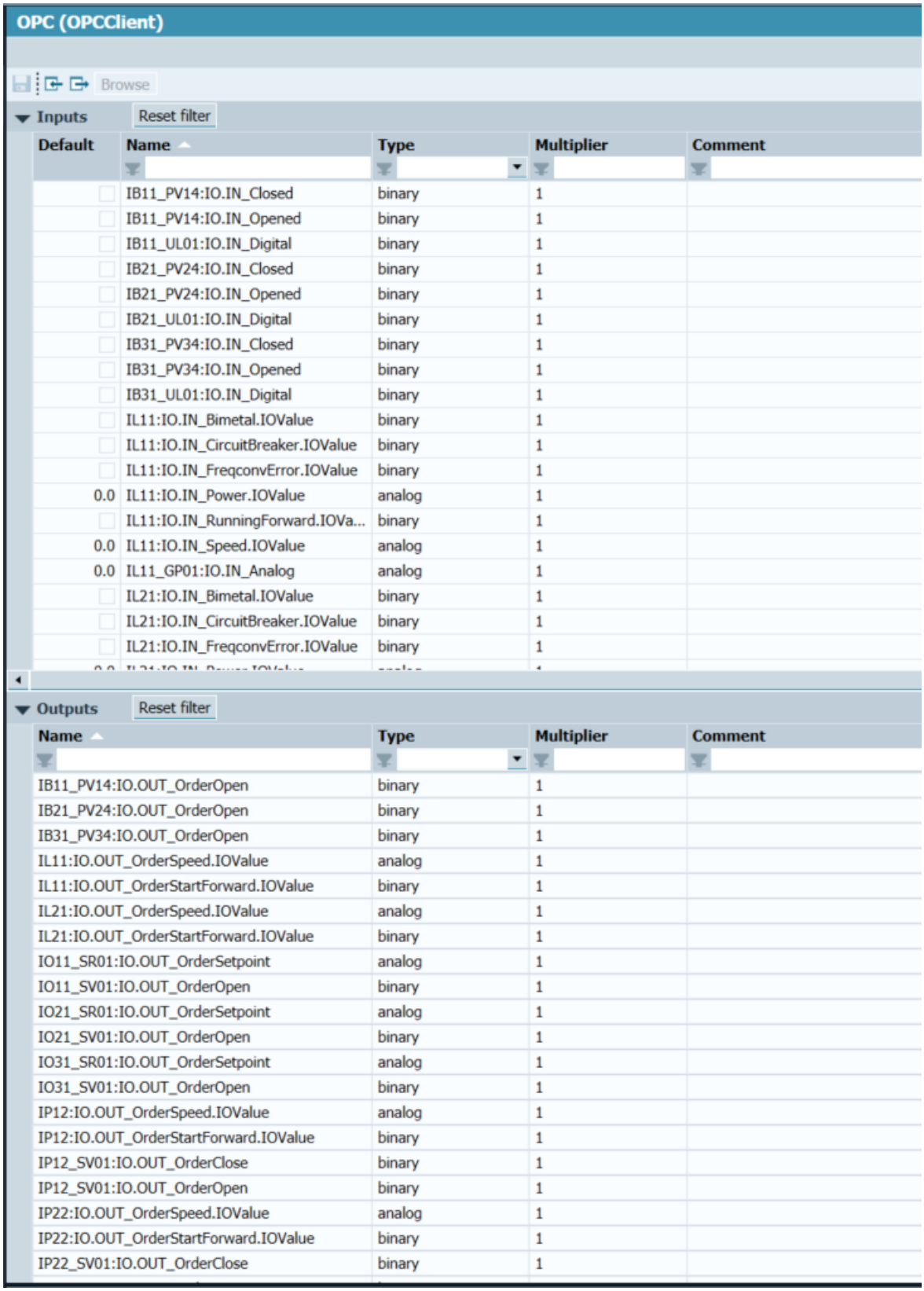
Figur 75: Genererade objekt av typen DigIn.

Till skillnad från de digitala signalerna från nivågivarna som hanteras av *DigIn* objekten, som endast ger signalerna *true* eller *false*, måste både flödesgivarna och tryckgivarna ange analoga värden. Dessa genereras därför som objekt av typen *AnaIn*, som bygger på Runtime komponenten *PCDL Transmitter*. De genererade objekten *IO11_GF01*, *IO21_GF01* och *IO31_GF01* motsvarar de tre flödesgivarna i processmodellen, och anger flödet genom de angivna punkterna i systemet i enheten Nm^3/h. De två tryckgivarna *IL11_GP01* och *IL21_GP01* anger trycket i systemet vid de två mätpunkterna i enheten bar. De genererade objekten av denna komponenttyp, *AnaIn*, kan ses i figur 76.



Figur 76: Genererade objekt av typen AnaIn.

Samtliga IO-signaler från alla dessa genererade objekt sammanställs sedan i en lista. Även detta görs med fördel via Excel, även om det kan göras direkt i OPC listan i SIMIT. I listan anges främst den verkliga OPC-adressen för varje signal samt om signalen är en in- eller utsignal. En del av denna lista kan ses i figur 77.



Figur 77: Lista över de IO-signaler som är kopplade till de genererade objekten.

För att de genererade objekten ska kopplas ihop med objekten i processmodellen måste de så kallade *hardware*-objekten kopplas ihop till processobjekten. Detta görs genom att en kopplad *faceplate* dras ut från den genererade hårdvaran, som då automatiskt länkar till den hårdvara den kommer ifrån. Detta görs för alla objekt i processmodellen. Därefter kopplas processobjekten till hårdvaran genom att namnen matchas, med tillägget av prefixet *HW_* på hårdvaruobjekten. På detta sätt utför SIMIT automatiskt kopplingen mellan objekten genom namnmatchning. För att detta ska fungera krävs dock att inga objekt delar namn, utan namnen för alla objekt måste vara unika.