CODEN:LUTEDX/(TEIE-3056)/1-57/(2016)

Industrial Electrical Engineering and Automation

Datahantering från PLC till SQL-databas

Fredrik Bergwall

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

Datahantering från PLC till SQL-databas

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Industriell elektroteknik och automation

Examensarbete:
Fredrik Bergwall

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

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

Tryckt i Sverige
Lunds universitet
Lund 2016

Sammanfattning

Detta examensarbete har utförts åt Mårtensson Consulting i Kristianstad. Arbetet syftade till att undersöka hur information från en PLC kan överföras till en SQL-databas för att sedan analyseras och presenteras. Arbetet gjordes med avsikt att implementeras hos en av Mårtensson Consultings kunder. Detta påverkade valet av bland annat PLC och programvara. Informationen som skulle loggas kom från en Clean In Place (CIP) process.

Examenarbetet utfördes genom att studera den befintliga koden för CIP-anläggningen och sedan kopplades PLCn upp mot en OPC-server. OPC-servern kopplades sedan upp mot en SQL-databas. Information från SQL-databasen presenterades sedan via ett webbgränssnitt. Resultatet är att det är möjligt att överföra information från en PLC till en SQL-databas. För att presentera informationen på ett lättförståeligt sätt för Mårtensson Consultings kund skapades ett webbgränssnitt.

Nyckelord: PLC, SQL-databas, OPC-server, CIP, webbgränssnitt

Abstract

This bachelor thesis was executed at Mårtensson Consulting in Kristianstad. The thesis aimed at investigating how information from a PLC can be transferred to an SQL-database. The project was done with the intention of implementing it at one of Mårtensson Consulting's clients. This influenced the choice of for instance PLC and software. The information that was to be logged came from a Clean In Place (CIP) process.

The execution of the thesis was done by studying the existing code for the CIP-facility and then connecting the PLC to an OPC-server. The OPC-server was then connected with an SQL-database. The information gathered from the SQL-database was presented via a web interface. The results are that it is possible to transfer information from a PLC to an SQL-database. In order to present the information in a comprehensive manner for Mårtensson Consulting's client a web interface was created.

Keywords: PLC, SQL-database, OPC-server, CIP, web interface

Förord

Detta examensarbete har gjorts som den sista delen av utbildningen till högskoleingenjör inom elektroteknik på Lunds tekniska högskola, under vårterminen 2016.

Jag vill tacka Mårtenssons Consulting för att de gav mig möjlighet att utföra examensarbetet hos dem och ett speciellt tack till Nils Mårtensson. Jag vill också tacka min handledare Christian Nyberg för råd och tips.

Till sist vill jag uttrycka min tacksamhet till min sambo Sara Nordkvist som stöttat och hjälpt mig under arbetets gång.

Terminologi

- Apache = Webbserver
- CIP = Clean In Place
- DCOM = Distributed Component Object Model
- DT = DATE_AND_TIME, en variabeltyp från Siemens för datum och tid
- HMI = Human Machine Interface
- IE = Industrial Ethernet
- IP = Internet Protocol
- OPC = Open Platform Communications
    - OPC DA = Data Access
    - OPC AE = Alarm & Events
    - OPC HAD = Historical Data Access
    - OPC UA = Unified Architecture
- PHP = PHP: Hypertext Preprocessor
- PLC = Programmable Logic Controller
- Profibus = Process Field Bus
- Sandbox = Ett sätt att köra otestad eller osäker programmeringskod i skyddad miljö på en PC
- SCADA = Supervisory Control And Data Acquisition
- SQL = Structured Query Language
- S7 = STEP 7, Siemens standard för automation, innehåller b.la kommunikation och kodning (1 Siemens 2012)
- TCP = Transmission Control Protocol
- TIME = Variabeltyp från Siemens som representerar tid

Innehållsförteckning

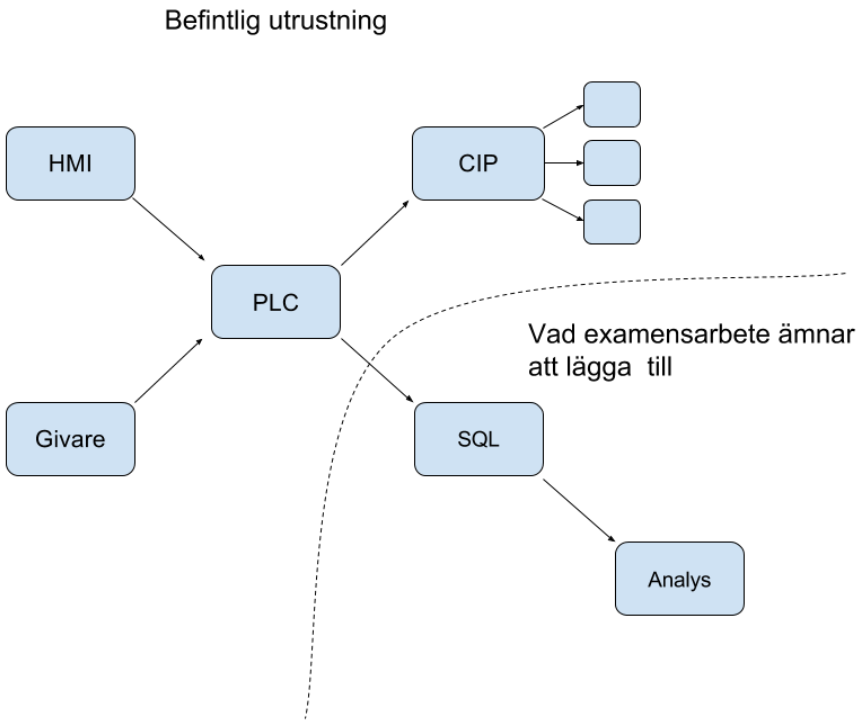
1 Inledning

Det inledande kapitlet i detta examensarbete kommer att innehålla bakgrund till arbetet, övergripande syfte och målformulering samt avgränsningar och resurser. Till sist kommer ett underkapitel innehållande källkritik att inkluderas.

1.1 Bakgrund

Examensarbetet har utförts åt Mårtensson Consulting AB i Kristianstad. Mårtensson Consulting grundades 1997 och har även kontor i Lund och Serbien och består av ett tjugotal anställda. Företaget är specialiserat inom process- och livsmedelsindustrin, där de arbetar med allt från PLC-programmering till support och felsökning.

Förtaget ville undersöka hur en befintlig Clean In Place (CIP) process hos en av deras kunder kunde förbättras, genom att skapa möjligheten att spara information från en CIP diskning. CIP är en metod för att rengöra insidan av bland annat rör och tankar, utan att behöva montera ner dem. Rengöringen sker med hjälp av vatten och lut. Förbättringen gjordes genom att skapa möjligheten att lagra och analysera information från ett antal olika givare ute i processen. Värden som önskades kunna sparas undan är till exempel temperatur, vattenförbrukning och tider. Utöver detta skulle också nödvändiga ändringar i kundens utrustning göras. Det här arbetet ämnar att förbättra Mårtensson Consultings kunders medvetenhet angående hur mycket vatten och tid det tar att utföra olika CIP rengöringsprogram, för diverse föremål i deras anläggning. Detta kommer att leda till en effektivare resurshantering. Figur 1.1.1 ger en enkel illustration över de olika komponenterna i CIP-processens styrsystem och vad examensarbetet har för avsikt att förbättra.



Figur 1.1.1 Är en enkel illustration över de olika delarna i CIP-processens styrsystem

1.2 Syfte

Det övergripande syftet är att effektivisera resurshanteringen för en av Mårtensson Consultings kunder. Detta görs genom att undersöka hur information från en PLC kan lagras i en SQL-databas och hur denna information sedan kan analyseras. Därefter ska ett CIP-system uppdateras för att information från CIP-systemet ska kunna överföras till en SQL-databas. Utöver detta är syftet också att redigera den befintliga koden så att det blir möjligt att spara undan information om vattenförbrukning och körtider för kundens anläggning. Examensarbetet kommer därför att öka medvetenheten kring effektiv resurshantering, i synnerhet av tid och vatten.

1.3 Målformulering

Det här projektet har två mål.
Det första målet är att undersöka och hitta en lösning på hur information kan överföras från en PLC till en SQL-databas. Denna lösning ska vara ekonomisk och klara av att hantera många funktioner, till exempel att kunna rita upp diagram över vattenförbrukningen. Resultatet av undersökningen används sedan för att implementera en lösning hos företaget där deras CIP-system kopplas upp så att data kan sparas och analyseras.

Det andra målet är att kunna samla in och visa de insamlade värdena, som är:

- Vattenåtgång under rengöring
- Lutåtgång under rengöring
- Temperatur hos vatten och lut under olika faser av rengöring
- Tidsåtgången för en rengöring

Insamlingen av data ska ske till en SQL-databas. Informationen från PLCn ska sedan kunna analyseras, vilket görs automatiskt av programmet. Dessutom ska den kunna kontrollera vad varje steg i CIP förbrukar. Analysen ska kunna jämföra olika CIP-program: hur lång tid de tar och hur mycket vatten och lut som förbrukas respektive hur mycket som går att återanvända. Analysen ska dessutom visualiseras med diagram och tabeller. De ekonomiska aspekterna måste också beaktas under projektets gång.

1.4 Problemformulering

Det här examensarbetet har tre huvudsakliga problemställande frågor som besvaras i rapporten.

1. Hur kan informationen överföras mellan en PLC till en SQL-databas och hur gör man detta i praktiken på ett ekonomiskt försvarbart sätt?

2. Hur kan informationen från SQL-databasen presenteras och vilka lämpliga program kan användas för detta?

3. Hur gör man en emulering av systemet, med tilläggen i punkt 1 och 2, som sedan går att omvandla till ett fungerande system som kan implementeras?

1.5 Avgränsningar

Den nödvändiga apparaturen är redan installerad och givarna finns redan på plats, vilket betyder att det är en avgränsning i det här projektet. Systemet behöver endast fungera tillsammans med en Siemens SIMATIC ET 200S PLC. Databasen som används är av typen SQL.

1.6 Resurser

Den nödvändiga programvaran för examensarbetet, som program för PLC-programmering Siemens SIMATIC v5.5, MySQL och SCADA kommer Mårtensson Consulting att stå för. Litteratur för undersökningen är tillgänglig via Lunds Universitets interna tjänster.

1.7 Källkritik

De källor som används i examensarbetet har ofta varit från företag eller stiftelser som tillhandahåller produkterna och standarderna som nyttjats under arbetets gång. Det är förmånligt att använda information från tillverkarna då denna information kan förutsättas att vara korrekt. En nackdel med att använda information från samma företag och stiftelser som har utvecklat produkterna är att det är svårt att hitta negativ information då de vill framställa sina produkter så positivt som möjligt.

Andra källor som används är uppslagsverk, kurslitteratur och läroverktyg, som till exempel WC3school. Dessa resurser kan förutsättas vara objektiva. Viss kurslitteratur, så som ”Webbprogrammering med PHP” av Viktor Jonsson, är över ett decennium gamla och kanske inte därför innehåller de senaste funktionerna och uppdateringarna. Men eftersom information om hur grunderna fungerar är oförändrade så kunde den litteraturen användas ändå.

2 Teknisk bakgrund

I detta kapitel redogörs för olika begrepp, utrustning och programvara som behövdes för att genomföra examensarbetet.

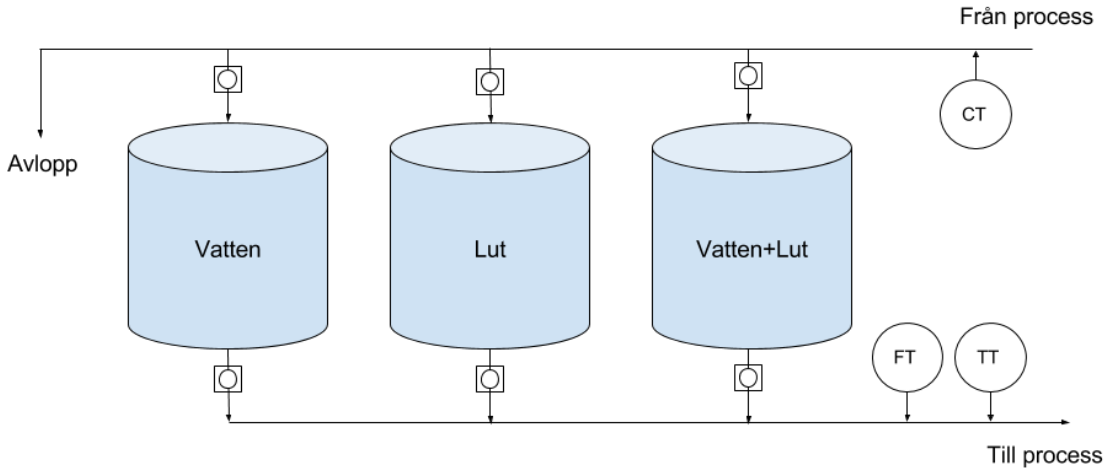
2.1 CIP

CIP är benämningen för en generell metod för rengöring av insidan av rör, behållare, tankar, filter med mera. Det som karakteriserar CIP är att rengöringen sker utan att utrustningen behövs monteras ner. Denna metod används i stor utsträckning av livsmedelsindustrin idag, eftersom den är mycket snabbare än att behöva montera ner och städa utrustningen manuellt (Process Worldwide 2016).

Den aktuella anläggningens CIP-process består av tre stycken tankar som innehåller: rent vatten, lut och en blandning av lut och vatten. Detta upplägg är vanligt förekommande inom livsmedelsindustrin men för till exempel kemisk industri används ett annat tillvägagångssätt. När en del i anläggningen, till exempel en tank, behöver diskas kopplas CIP systemet in. Genom att mäta konduktivitet i vätskan på vägen mot avloppet kan man avgöra om vattnet, luten eller blandningen av dem är tillräckligt rent för att återföras till respektive tank. I anläggningen används lut bestående av vatten (ca 98 %) och natriumkarbonat (ca 2 %). Diskningen sker efter följande rutin:

1. Diskningen av objektet för sköljs med blandningen av lut och vatten
2. Systemet sköljs med upphettat vatten
3. Systemet sköljs med lut
4. Systemet sköljs rent från lut med vatten

Denna process är tids- och energikrävande, därför är det bra att jämföra olika CIP-program med varandra. Figur 2.1.1 visar en enkel illustration av hur anläggningen ser ut.



Figur 2.1.1 Illustration av aktuell anläggnings CIP process. CT = konduktivitetsgivare, FT= flödesgivare och TT = temperaturgivare.

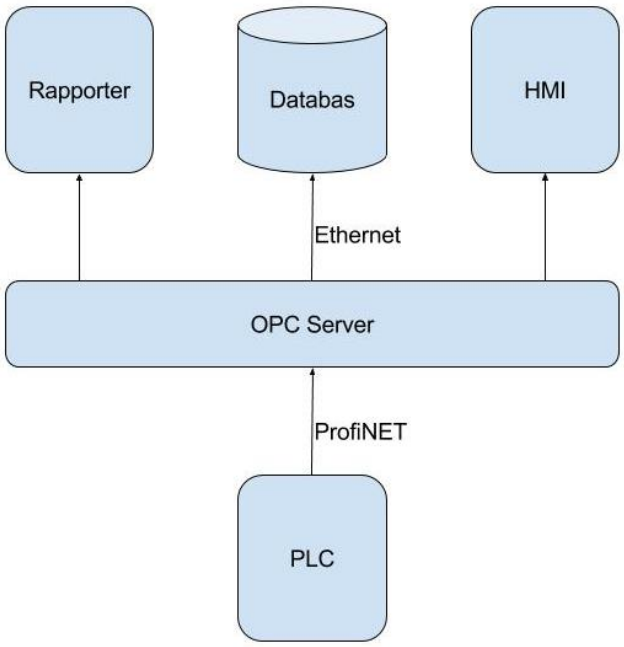
2.1.1 Givare

I CIP-anläggningen sitter följande typer av givare: En flödesgivare som både ger ut det momentana flödet som ett flyttal med storheten liter i timmen, samt en puls för varje liter som passerar igenom den. Flödesgivaren är placerad efter tankarna. Temperaturgivare som ger den aktuella temperaturen som ett flyttal, är placerad efter tankarna, och en konduktivitetsgivare som mäter konduktiviteten på vätskan och är placerad innan tankarna på väg mot avloppet.

2.2 OPC-server

Enligt OPC Foundation, utvecklarna och förvaltarna av standarden Open Platform Communication (OPC), är OPC en standard för ”säker och pålitligt utbyte av data i industriell automation.” (egen översättning, OPC Foundation 2016). Företag som till exempel Microsoft, Siemens och Schneider är medlemmar i OPC Foundation och är drivande för utvecklingen av standarden. Utöver detta är så är OPC ”plattformsoberoende och säkerställer ett kontinuerligt flöde av information mellan enheter från flera leverantörer.” (egen översättning, OPC Foundation 2016). Standarden är en vidareutveckling från Microsoft DCOM som togs fram för Windowsplattformen under 1990-talet. Idag är DCOM-standarden på tillbakagång då ny industri använder OPC-standarden istället. I detta arbete har understandarden OPC Data Access (DA) används. De första standarderna som släpptes var OPC DA, OPC Alarm and Event (AE) och OPC Historical Data Access (HDA). Dessa kom ut 1996 och brukar numera kallas OPC Classic. Syftet med OPC-standarderna var att samla ihop olika PLC-kommunikationsprotokoll, som till exempel Modbus och Profibus (Process Field Bus) till ett standardgränssnitt. Målet med att gör detta var det skulle bli möjligt för olika HMI och SCADA att kommunicera med PLCn genom att använda OPC som en mellanhand för PLC och HMI. Idén var att HMI och SCADA skulle kunna kommunicera genom OPC som

sedan översatte informationen till de kommunikationsprotokoll som PLCn använde, se figur 2.2.1 för en illustration av detta.



Figur 2.2.1 visar hur OPC kan användas för att kunna kommunicera med olika protokoll.

Sammanfattningsvis används OPC för att hantera kommunikation mellan industriautomation, till exempel PLC, och vanlig mjukvara, till exempel till ett HMI eller PC-program. Detta kräver att industrikomponenten, ofta en PLC, är utrustad med och har stöd för kommunikation med Industrial Ethernet (IE) eller profibus. Nedan följer en kort beskrivning av vad de olika understandarderna används till i OPC Classic:

- OPC DA beskriver hur OPC ska överföra data, som till exempel information om tid, mängd och kvalitet. För att OPC DA ska hitta och skicka data måste adressen där datan finns först definieras, det vill säga att storlek och datatyp anges.
- OPC AE beskriver hur OPC ska hantera larm och händelsebaserade meddelanden och information, samt för att hantera och ändra statusinformation.
- OPC HDA beskriver hur OPC kan användas för att lagra data från PLC till en databas på OPC-servern.

Den nyaste standaren heter OPC Unified Architecture (UA) vars första stabila version släpptes 2008. En av skillnaderna mellan OPC UA och OPC Classic är att alla understandarderna är samlade under en standard. Fördelen med att slå ihop standarderna är att en OPC-server som har stöd för OPC UA också har stöd för att använda samtliga funktioner. Detta till skillnad från idag där en

OPC-server inte behöver stödja hela OPC Classic utan endast några understandarder. OPC UA stödjer också kommunikation med flera operativsystem till skillnad från OPC Classic som byggde på Microsofts teknik och därav hade begränsat stöd för andra plattformar än Microsofts. Alla standarder finns definierade hos OPC Foundation (3 OPC Foundation 2016).

2.3 Kommunikationsprotokoll

Nedan följer en kort beskrivning av ett fåtal kommunikationsprotokoll som kan används för kommunikation mellan PLC, givare och HMI.

2.3.1 Industrial Ethernet

IE bygger på det klassiska Ethernetprotokollet som är vanligt förekommande i LAN-nätverk. Skillnaden mellan IE och Ethernet är att IE har förstärkta kontakter och mycket högre krav på kommunikationsutrustningen att kunna hantera höga temperaturer, fukt och vibrationer. Detta eftersom tekniken används i industrin där det ofta krävs att utrustningen är stöttålig och hållbar. Det finns ingen skillnad mellan IE och Ethernet gällande hur information överförs mellan två enheter. Det vill säga IE bygger kommunikationsmässigt helt på Ethernetstandarden, som finns definierad hos IEEE (IEEE 2016). IE är i sig själv inte ett komplett kommunikationsprotokoll utan ligger i länklagret och måste användas med andra protokoll för kunna skicka över information. IE används främst för att kommunicera mellan olika enheter, till exempel mellan två PLCer. IE används däremot inte för att kommunicera mellan PLC och givare. Nedan följer en kort lista med för- och nackdelar med IE:

+ Möjligheten att använda standard Ethernet utrustning, som till exempel kablar och routrar finns
+ Genom att använda optisk fiber försvinner risken för elektromagnetiska störningar
+ Har lång räckvidd
+ Högre överföringshastigheter än de andra protokollen som nämns i rapporten
- Kan vara svårt att integrera med befintliga protokoll
- Onödigt stora paket, minsta storleken på ett Ethernet paket är 64 bytes medan kommunikation mellan industrikomponenter kan vara att skicka boolean

2.3.2 Modbus

Modbus togs fram under 1970-talet av Schneider Electrics, dåvarande Modicon. Idag ägs och utvecklas standarden av Modbus Organization. Modbus protokollet ligger på applikationsnivå och använder bland annat TCP/IP eller Modbus Plus (som i sin tur använder standarden EIA-232) för att skicka data över nätverk. Modbus har fördelen att det är en gratislicens för att använda standarden. Det är dessutom lätt att koppla ihop olika

Modbusstrukturer med varandra, till exempel att få en enhet som använder TCP/IP att kommunicera med en som använder Modbus Plus (Modbus Organization 2012).

2.3.3 Profibus

Profibus ett kommunikationsprotokoll som används framförallt av Siemens och togs fram under 1980-talet av det tyska utbildningsdepartementet i samarbete med ett stort antal företag. Profibus är definierat enligt standarden IEC 61158 (2 Siemens 2012).

2.3.4 ProfiNET

ProfiNET är också ett kommunikationsprotokoll som är framtaget av samma grupp som utvecklade Profibus. Till skillnad från Profibus använder sig ProfiNET av IE-standarden för sin kommunikation (2 Siemens 2014).

2.4 Siemens SIMATIC

Siemens SIMATIC är grundmjukvaruprogrampaket för programmering av och kommunikation med alla Siemens egna produkter. Det finns många tillägg och ändringar som går att göra beroende på vilket typ av Siemensprodukter som används. SIMATIC är det äldre av två olika programstrukturer, Siemens nyaste program heter ”Totally Integrated Automation (TIA) portal”.

2.4.1 Siemens SIMATIC ET 200S PLC

SIMATIC ET 200S är en serie industridatorer som används för styra och kontrollera olika industrimiljöer. Det som skiljer de olika modellerna åt är vilken processor och hur mycket RAM-minne de har. Denna PLC serie bygger på ett modulkoncept där moduler för bland annat in- och utgångar kopplas in till PLCn efter behov (4 Siemens 2016). Den har stöd för kommunikation via både ProfiNET (IE) och Profibus.

2.4.2 SIMATIC manager

Detta är grundprogrammet. Här specificerar man vilken typ av PLC, eventuella moduler och annan utrustning som ska användas. En hårdvaruinställning görs, där bland annat typ av processor och moduler för kommunikation väljs. Det är också i detta program som kodning av programvaran sker (1 Siemens 2016). SIMATIC manager har stöd för att programmera PLCer i tre olika språk, dessa är:

- Funktionsblock (FBD)
- Ladderdiagram (LAD)
- Strukturerad text (STL)

Dessa språk är definierade enligt standarden IEC-61131. När en ny fil skapas väljs vilken typ av fil det ska vara, antingen fil med programmeringskod eller fil med datablock. Vilket programmeringsspråk som ska användas och om filen ska ha ett datablock bestäms också.

2.4.2.1 Datablock

Datablock är ett sätt att använda och spara undan variabler i SIMATIC manager. Ett data block kan vara internt och då kan det endast användas av programfilen den är bunden till eller fristående och då kan alla programfiler i ett projekt hämta och ändra i datafilerna. En variabel definieras genom att den får namn, typ och startvärde. Varje datablock börjar med adressen noll och byggs succesivt ut beroende på variabeltypens storlek. Detta gör att varje variabel får en unik adress i datablocket. När en variabel sedan används skrivs datablockets nummer, variabeltyp och startadress. Ponera att datablocket i bild 2.4.1 används som exempel, då skrivs DB12.DBDT2.0 för att hämta variabeln StartTime.

En programmeringsfil behöver inte använda alla variabler i ett datablock och kan hämta variabler från många olika datablock. Datablock används främst för att tillfälligt spara och skicka information, till exempel larm, mellan olika programfiler (Siemens 2010).

DB12 -- "REPORTS TO SQL" -- TestS4A\PLC1\WinLC\...\DB12

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	PosEdgeForCounter	BOOL	FALSE
+0.1	PosEdgeForStartTime	BOOL	FALSE
+0.2	PosEdgeForEndTime	BOOL	FALSE
+0.3	CIPDone	BOOL	FALSE
+0.4	BlockBitStartTime	BOOL	FALSE
+0.5	ResetTimer	BOOL	FALSE
+0.6	SpareBool0	BOOL	FALSE
+0.7	SpareBool1	BOOL	FALSE
+1.0	SpareBool2	BOOL	FALSE
+1.1	SpareBool3	BOOL	FALSE
+1.2	SpareBool4	BOOL	FALSE
+1.3	SpareBool5	BOOL	FALSE
+1.4	SpareBool6	BOOL	FALSE
+1.5	SpareBool7	BOOL	FALSE
+2.0	StartTime	DATE_AND_TIME	DT#90-1-1-0:0:0.000
+10.0	SpareStartTime	DATE_AND_TIME	DT#90-1-1-0:0:0.000
+18.0	EndTime	DATE_AND_TIME	DT#90-1-1-0:0:0.000
+26.0	SpareEndTime	DATE_AND_TIME	DT#90-1-1-0:0:0.000
+34.0	totalAmountConsumedWater	REAL	0.000000e+000
+38.0	AlwaysZero	REAL	0.000000e+000
+42.0	SpareReal	REAL	0.000000e+000
+46.0	SpareReal1	REAL	0.000000e+000
+50.0	SpareReal2	REAL	0.000000e+000
+54.0	SpareReal3	REAL	0.000000e+000
+58.0	ErrorStartTime	UDT5	
+66.0	ErrorEndTime	UDT5	
+74.0	RunningTime	TIME	T#0MS
+78.0	ObjectNbr	INT	0
=80.0		END_STRUCT	

Figur 2.4.1 Visar strukturen på ett datablock

2.4.3 SIMATIC NetPro

Grundprogrammet Netpro används för att definiera hur olika moduler/utrusning är kopplade till varandra. Utöver detta, används det också till att fastställa vilka kommunikationsprotokoll som ska användas. Till exempel, om IE används måste IP-adresser ges till de olika objekten i systemet (Siemens 2005).

Om tillägget SIMATIC Net är installerat ges även verktyg för att kontrollera att uppkopplingar, med S7portokollet, mellan olika enheter fungerar. Detta verktyg kallas ”S7 RedConnect Connection Diagnostic”. I detta program görs också inställningar för OPC-servern. Inställningar som görs i NetPro om SIMATIC Net är installerat är:

- Namn på serverkoppling
- Typ av OPC-Server, till exempel OPC DA
- Specifikation av kommunikationsprotokoll
- Serverns IP-adress
- Sändaren IP-adress
- Hantering av dataöverföring och felmeddelanden

2.4.4 SIMATIC Net

SIMATIC Net är ett tillägg i originalprogrampaket. I detta tillägg ingår bland annat att SIMATIC NetPro får utökade egenskaper för att kunna hantera OPC-servrar och fler typer av kommunikationsprotokoll. I SIMATIC Net ingår även Siemens OPC-server och verktyg för att hantera den.

2.4.4.1 SIMATIC Net OPC

SIMATIC Net OPC är Siemens egen OPC-server. Eftersom det är Siemens egna produkt finns redan verktyg i SIMATIC för att arbeta med den vid namn OPC Scout. Den går att hantera och integrera direkt i Siemens projekt genom att använda SIMATIC manager. Servern har stöd för OPC-protokollen OPC DA, OPC AE och OPC UA.

Servern fungerar på följande sätt: Då ett OPC-server-objekt läggs till och ställs in i hårdvarukonfigurationen och SIMATIC NetPro skapas en koppling mellan servern och PLCn. Även om flera server-objekt läggs till är dessa kopplade till samma server vilket gör att varje koppling måste ha ett unikt namn. Kopplingen gör det möjligt för användaren genom OPC-servern att komma åt och definiera information i PLCns minne via OPC Scout. OPC Scout hittar automatiskt mappar som innehåller till exempel, timers och datablock, men kan inte tolka informationen i dem. Detta måste användaren själv göra genom att definiera ”taggar” (1 Siemens 2014). Denna funktion beskrivs mer utförligt i 2.4.4.3 Inställningar i OPC Scout.

2.4.4.2 OPC Scout

OPC Scout är ett användargränssnitt som används för att hantera ”SIMATIC Net OPC Server”. I detta program finns verktyg för att identifiera vilken typ av OPC-server som används. Det är också här man bestämmer vilka variabler som ska gå att komma åt utanför PLC-miljön. Detta sker genom att man definierar utrymme i PLCns minne. Till exempel, om ett INT värde från ett datablock ska sparas definieras först datablocket, sedan vilken typ av data det är och sist vilken plats i data blocket den har och hur många bitar den är. Detta kallas en ”tag” (Siemens 2011).

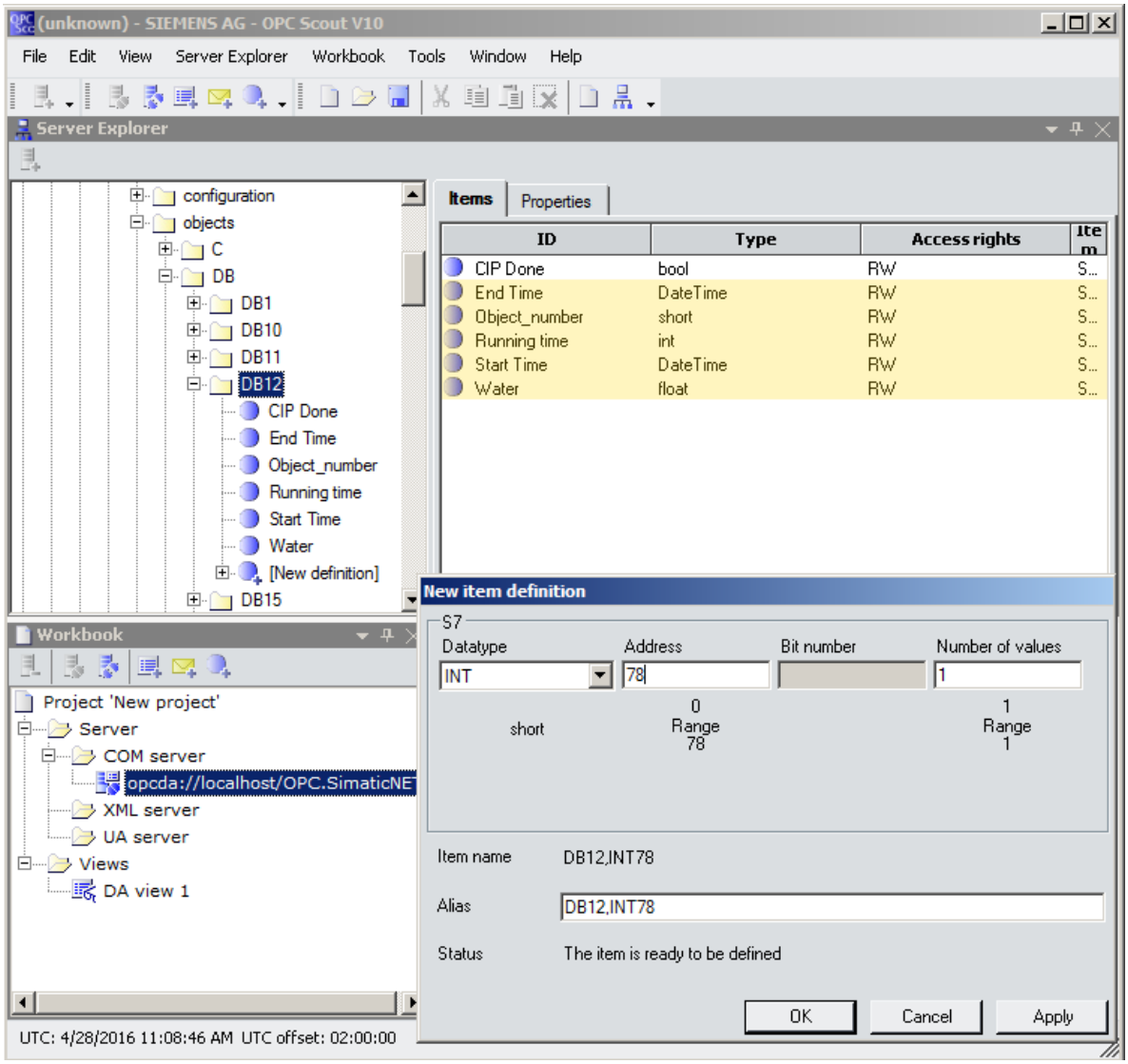
För att enkelt kontrollera att uppkopplingen lyckades används ett program som heter “S7 RedConnect Connection Diagnostic”. Detta program är ett tillägg till SIMATIC NetPro och visar statusen för alla kopplingar enligt S7-standard.

2.4.4.3 Inställningar i OPC Scout

Det första inställningen är att i menyn välja den aktuella servertypen, i detta examensarbete användes ”OPC.SimaticNET”. Användaren navigerar ner till servern. När servern hittats börjar definieringen av de värden som ska sparas till SQL-databasen. Både timers, minnesvärde och värden i data block går att spara undan. Se figur 2.4.2 för att illustrativt exempel hur en ”tag” definieras. I detta examensarbete var det främst värden i datablocket som var intressanta att spara undan, eftersom all relevant data ligger i dessa.

För att definiera en ”tag” måste man:

1. Hitta datablocket som värdet finns i
2. Definiera vilken typ av data det är
3. Definiera vilken adress den har i datablocket
4. Kontrollera hur många bitar datan består av
5. Fastställa om det finns mer än ett värde (det vill säga om värdet består av mer än en minnesplats/adress)
6. Ge taggen ett namn



Figur 2.4.2 visar navigering och rutan visar definiering av tag

2.4.5 SIMATIC WinLC

Detta är ett program som används för att emulera en PLC och som kan användas istället för en riktig PLC. Den emulerade PLCn beter sig precis som en riktig och har samma egenskaper. För att kunna använda SIMATIC WinLC måste projekttypen ändras från den valda PLCtypen till PC-Station. På detta sätt kan alla PLCer som stödjer S7 protokollet emuleras. Detta program kan inte arbeta ensamt utan måste används tillsammans med ”Station Manager” (2 Siemens 2016).

2.4.6 Station Manager

Här görs inställningar för att kunna använda PC-baserad mjukvara och för att köra mjukvaru-PLC. I denna specificeras vilken typ av hårdvara som används och vilken plats den har i projektet. När ”Station Manager” sedan aktiveras kan kod laddas över och köras på en mjukvaru-PLC (Siemens 2005).

2.4.7 Datatyper i Siemens

Utöver standardatatyper som till exempel BOOL, INT och Float finns det i Siemens utvecklingsmiljö datatyper som Siemens själva definierat. Nedan följer en beskrivning av Siemens egna datatyper, som används i detta examensarbete.

2.4.7.1 DT

Detta är en variabeltyp som används i Siemens PLC och står för: DATE_AND_TIME. DT är en så kallad komplex datatyp. Den används för att representera datum och tid och är uppbyggd som tabell 2.1 visar. Den skrivs DT#YY-MM-DD HH:MM:SS.SSS (3 Siemens 2016).

Byte	**Representerar**
0	År
1	Månad
2	Dag
3	Timme
4	Minut
5	Sekund
6	Millisekund
7	Millisekund och Veckodag*

**Eftersom 12 bitar behövs för att ge maximal noggrannhet på millisekunder räcker inte byte 6 inte till och därför används 4 bitar av byte 6 för att spara millisekunder.*

Tabell 2.4.3 Visar hur DT är uppbyggt

Ett problem med datatypen är att den är 64 bitar (8 bytes) stor medan PLCn endast kan hantera värden av storleken 32 bitar. Detta gör att ”pekare” måste användas när en variabel av DT ska skrivas eller flyttas. En ”pekare” pekar på en startplats i ett datablock eller i en minnesplats och sedan skrivs värdet in därifrån blankt (detta kan resultera i dataförlust om 32 bitar inte är reserverade, författarens anmärkning).

2.4.7.2 TIME

Detta är en variabeltyp från Siemens som representerar tid i steg om millisekunder och är 32 bitar stor. Den kan representera både positiv och negativ tid, den har ett span från -24 dagar 20 timmar 31 minuter 23 sekunder 648 millisekunder till +24 dagar 20 timmar 31 minuter 23 sekunder och 647 millisekunder. För att representera tiden används en long (dubbel int) som sedan räkans upp i steg om millisekunder (Siemens 2006). När variabeln används i Siemens miljö visas resultatet i formatet ”T#Dag_Timme_Minut_Sekund_Millisekund” dock tolkars variabel som en long av andra program som endast ser värdet i millisekunder i formatet ”L#nummer”. Till exempel tolkas tiden (i variabeln TIME) 2147483647 millisekunder i Siemens som T#24D_20H_31M_23S_647MS men i SQL

tolkas värdet som long med värdet: ”L#2147483647”. Detta beror på att Siemens programvara automatiskt gör översättningen från millisekunder åt användaren.

2.5 OPC Data Logger

OPC Data Logger är ett program från Software Toolbox som används för att leta efter förinställda taggar på en bestämd OPC-server. Detta skrivs sedan in i en SQL-tabell. Tabellen kan både skapas av programmet och separat i databasen. Programmet fungerar med ett stort antal olika PLCer och OPC-servrar. Programmet stöder även de vanligaste SQL-databaserna: MS Access, MS SQL, Oracle och MySQL. En nackdel är att det endast har stöd att köras i Windowsmiljöer (Software Toolbox 2016).

2.5.1 Beskrivning av inställningar

För att kunna använda programmet korrekt måste först de värden som ska lagras definieras i OPC Scout, så kallade taggar. Ett undantag måste göras i brandväggen för den port OPC Data Logger använder. I detta examensarbete var det port 8148 som användes.

2.5.1.1 Collector

Det första som görs är att definiera ”Collector”. ”Collector” är den delen av OPC Data Logger som hämtar information från OPC-Servern. I den måste man fastställa vilken IP-adress servern är på och vilken typ av server det är.

2.5.1.2 Data Presentaion

Nästa steg är att under ”Data Presentation” välja ut en grupp med taggar som ska läggas in i SQL-databasen.

2.5.1.3 Data Storage

Därefter väljs ”Data Storage” och i den måste typ av SQL databas väljas. Det måste även bestämmas på vilken IP-adress databasen finns. Hur man ska logga in på databasen, antingen med användarnamn och lösenord eller Micosoft authentication (endast MS SQL) måste också väljas. Om användarnamn och lösenord väljs måste dessa skrivas in. Därefter väljs databas och vilken tabell i databasen som värdena ska sparas till. Sist matchas tagarna till kolumner i tabellen. I detta steg måste man vara noggrann med att rätt värde hamnar i rätt kolumn, annars blir det fel med variabeltyperna.

2.5.1.4 Projects

I denna del av programmet väljs vilka taggar som ska föras in i databasen. Detta sker genom att en grupp som innehåller taggar skapas. Om mer än en (SQL)tabell ska används för att föra in data, kan taggarna sparas i olika grupper.

2.6 VM ware

VM ware är ett program för att emulera operativsystem. Programmet kan ge det emulerade operativsystemet hårdvaruspecifikationer och egna nätverkskort. VM står för ”Virtual Machine”. Företaget som utvecklade programmet heter också VM ware. Programmet används flitigt inom industrin och använts i nuläget av över 500 000 kunder (VM ware 2016). En av anledningarna till att VM ware är ett populärt program är att eftersom program körs installeras och körs från emulerade operativsystem så skyddas originaloperativsystemet och datorn. VM ware erbjuder en sandboxmiljö för operativsystemen.

2.7 SQL Databas

SQL databas är en typ av relationsdatabas som används för att spara stora mängder data. I en relationsdatabas är informationen ordnad i relationer, dessa kallas tabeller. Grunden för relationsdatabaserna bygger på ett system som heter ”Relational Database Management System” (NE 2016). Vid installation behöver de vanliga inställningarna göras, som undantag i brandväggen med mera. De vanligaste typerna av SQL databaser är:

- Oracle
- MySQL
- Microsoft Access
- Microsoft SQL Server

2.8 Notepad++

Notepad++ Är ett enkelt textredigeringsprogram för Windows. Detta program kan användas för att programmera i de flesta programmeringsspråk. Programmet är skrivet i C++ och endast 12 MB stort och kräver lite prestanda att köra. En fördel med programmet är att det har ett bra ”highlighting”-verktyg för programmering, det vill säga hur texten blir markerad. En stor nackdel är att programmet själv inte kan kontrollera koden, utan för att göra detta måste ytterligare programvara användas (Notepad++ 2016).

2.9 XAMPP

Detta program används för att skapa en webbserver på datorn. Från servern kan man sedan ladda upp och köra egentillverkade hemsidor och script. Programmet kan användas för att lägga upp och köra en webbserver på det lokala nätverket. XAMPP bygger på teknik från Apache. Genom att använda en webbserver för att testa webbsidor kan PHP-script köras.

XAMPP gör det enkelt att ändra i PHP-konfigurationsfiler då programmet skapar genvägar till dessa. En annan fördel är att XAMPP skapar

kopior på originalet så att när användaren gör ändringar så sker det inte på operativsystemet utan enbart på XAMPPs fil.

2.10 PHP

PHP är ett programmeringsspråk med öppen källkod som är speciellt lämpat för att utveckla webbsidor. Den största fördelen med att använda PHP istället för till exempel C++ för att programmera webbsidor är att det går att bädda in HTML i ett PHP-dokument, se figur 2.10.1 för ett exempel. Ett PHP-dokument kan innehålla HTML-kod som har PHP-kod inbäddat i koden. PHP kan köras från de flesta UNIX-baserade operativsystemen och Windows. Språket underhålls och utvecklas av The PHP Group. Språkets typsystem är dynamiskt. Detta gör att det inte finns någon enhetlig standard för att skriva PHP-kod. En anledning till detta är att det inte fanns någon formell specifikation eller standard före 2014. Den första stabila versionen kom 1995.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>Exempel</title>
    </head>
    <body>
        <h1>PHP exempel</h1>
        <p>
            <?php
                echo "Hello World!!"
            ?>
        </p>
    </body>
</html>
```

Figur 2.10.1 Exempel på PHP-kod i ett HTML-dokument, PHP skriver ut ”Hello World!!” på hemsidan

Med hjälp av syntaxen <?php?> går det att skriva PHP-kod där det behövs utan att det påverkar HTML-koden. Det som skiljer PHP från JavaScript är att JavaScript är klientbaserat och PHP är serverbaserat. Detta betyder att PHP-koden exekveras på servern och inte i användarens dator. Användaren får bara resultatet av den exekverade koden och vet inte vad som har körts vilket kan vara fördelaktigt om till exempel konfidentiell eller känslig kod använts (1 PHP 2016).

3 Metod

Det föregående kapitlet har redogjort i detalj för all mjuk- och hårdvara som har använts under examensarbetes gång. Detta kapitel kommer koncist att redogöra för metodologin som har använts under examensarbetet. Bakomliggande motiveringerar angående till exempel val av OPC-server kommer att redovisas i kapitel 4 Genomförande.

Det första steget i arbetet var att studera CIP-anläggningens kod i detalj. Dels för att veta i vilka datablock den nödvändiga informationen lagrades och dels för att få en generell överblick över hur anläggningen fungerade. En kort efterforskning gjordes också om OPC-server, dess funktioner och lösningar som gjorts av teknik- och IT-företag, så som Siemens.

Efter att kunskaperna angående CIP och OPC ansågs vara tillräckliga fortsatte arbetet med att sätta upp en koppling mellan PLCn och OPC-servern. När detta var gjort skapades de ”taggar” som behövdes i examensarbetet. En detaljerad redogörelse för hur detta gjordes finns i avsnitt 4.1.1 OPC-server. När OPC-servern var konfigurerad började arbetet med att koppla ihop SIMATIC Net OPC med OPC Data Logger. Detta gjordes genom att ställa in OPC Data Logger, information om hur inställningar skulle göras finns manualen Configuration (2 Software Toolbox 2016). I programmet gjordes även inställningar för att kommunicera med databasen. För en mer detaljerad beskrivning vänligen studera avsnitt 4.2 Överföring från PLC till databas.

När en fullständig överföring (read and write) lyckats, det vill säga från PLCn till SQL-databasen, bestämdes möte med Mårtenssons Consulting angående hur examensarbetet ämnades fortgå. Under mötet diskuterades vilken information som fanns tillgänglig och vad som var intressant att spara. Resultatet av mötet blev att kompletterande kod skulle läggas till i PLC-programmet för att kunna göra rapporter innehållande:

- Start-, slut- och körtider
- Totala vattenförbrukning
- Vilket objekt som diskas

Utöver omarbetningar i PLC-koden fick även ändringar i CIPs HMI göras för att kunna välja objekt att diska.

När PLC-koden var klar bestämdes ett nytt möte för att diskutera hur informationen från databasen skulle presenteras. Ett webbaserat gränssnitt valdes eftersom det är plattformsoberoende och en webbläsare är ett verktyg som de flesta är vana vid och kan hantera. Uppkoppling mot internet var inget som kunde garanteras och hemsidan hade kravet att kunna fungera utan anslutning till internet. Hemsidan skulle klara av att kunna:

- presentera rapporter
- presentera rapporter grafiskt
- presentera medelvärdes data för olika objekt
- skriva ut eller exportera data från hemsidan
- söka efter tider
- fungera utan tillgång till internet

För att göra hemsidan behövdes ytterligare kunskap inom webbprogrammering och information inhämtades från både tryckta och digitala källor. HTML studerades i boken ” HTML 5: HTML och CSS-boken” (Staflin 2011) och även på hemsidan w3schools (w3schools 2016). Det stod tidigt klart att programmeringsspråket PHP med fördel kunde användas vilket studerades i boken ”Webbprogrammering med PHP” av Jonsson (2001) samt på dess officiella hemsida (PHP Manual 2016). Sedan tidigare fanns goda kunskaper inom Java, som är likt JavaScript som används i webbprogrammering. Därför användes endast hemsidan w3schools för inlärning av JavaScript (w3schools 2016).

När kunskaperna inom webbdesign ansågs vara tillräckligt goda började arbetet med att programmera hemsidan. Hemsidan använder tre olika programmeringsspråk, HTML/CSS, PHP och JavaScript. För att programmera hemsidan användes Notepad++. För testkörning och felsökning användes XAMPP. För att uppkopplingen mot SQL-databasen skulle kunna ske gjordes ändringar i XAMPPs PHP hantering. För en mera detaljerad beskrivning om hur arbetet med hemsidan skedde se avsnitt 4.4 Webbgränssnitt.

När alla ovanstående delar av arbetet var utförda, bestämdes en tid för att testköra examensarbetet mot den verkliga utrustningen hos Mårtenssons kund.

4 Analys och genomförande

Detta kapitel innehåller en mer utförlig redogörelse gällande hur vissa delar av examensarbetet utfördes. De moment som kommer att redovisas i detta kapitel är de som anses vara viktigast för förståelsen av hur arbetet fortskridit och för tydlighetens skull följer det ett kronologiskt mönster.

4.1 SIMATIC

Det första steget i arbetet var att studera den befintliga PLC-koden för anläggningen och hur den var uppbyggd med olika datablock och funktionsscheman. Ett problem var att tillgång till den riktiga anläggningens PLC inte kunde ges. Detta löstes genom att överföra den nödvändiga PLC-koden till en mjukvaru-PLC. Mjukvaru-PLCn som användes under projektets gång var av typen Siemens WinLC. Denna installerades med hjälp av VM ware på en simulerad Windows XP med ett eget nätverkskort och därmed en egen IP-adress. Då anläggningen använder IE, där kommunikation sker mellan enheter med hjälp av fasta IP-adresser fick även mjukvaru-PLCn en fast IP-adress. Detta gällde även arbetsstationen som är PCn där programvaran finns. I och med bytet från en Siemens SIMATIC ET 200S som sitter i anläggningen till en mjukvaru-PLC fick ändringar göras i hårdvarukonfigurationen. Där ändrades projekttypen från den givna PLCn till en PC Station. Detta gjordes för att SIMATIC förväntades kunna ladda upp och köra kod korrekt i mjukvaru-PLCn. Nödvändiga inställningar så som hårdvarukonfiguration fick också ställas in i Station Manager. Denna virtuella PLC användes under projektets gång för att kontrollera att kod och inställningar både i SIMATIC och OPC data logger fungerade som de skulle.

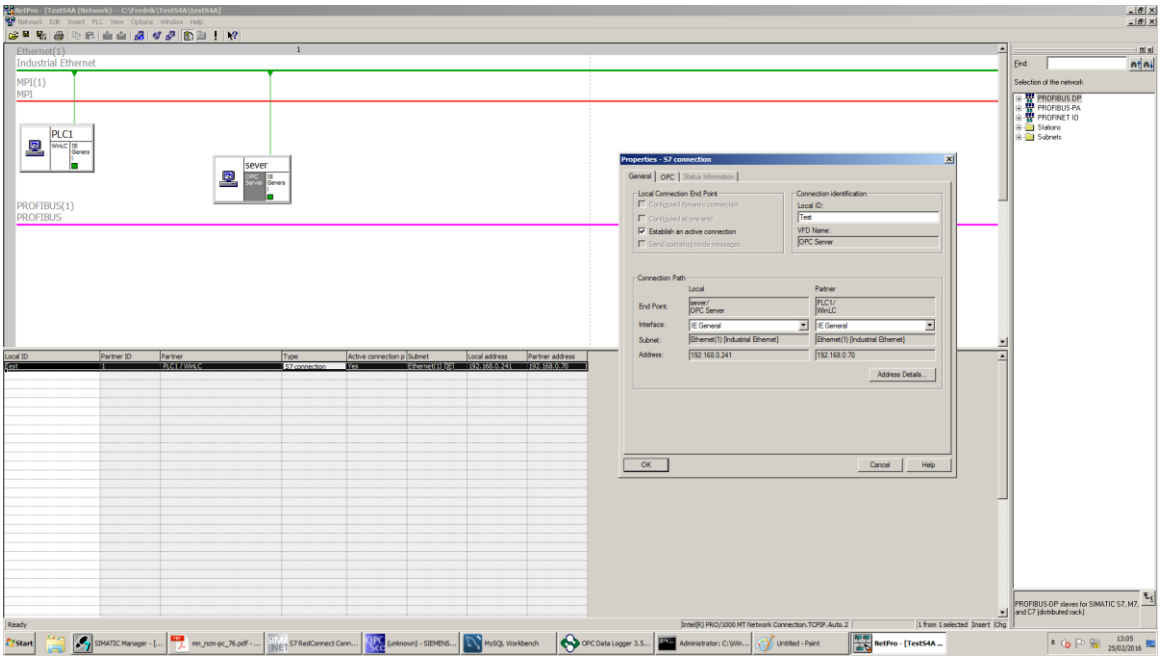
4.1.1 OPC Server

Vid val av OPC-server valdes att arbeta med Siemens egna, SIMATIC Net OPC. Siemens OPC valdes primärt därför att den redan var tillgänglig vid projektets början. Men även eftersom server redan är integrerad i Siemens utvecklingsmiljö vilket gör att den är lättare att lära sig att arbeta med än en tredjeparts program. Då SIMATIC Net OPC är skräddarsytt för att arbeta med Siemens PLCer, minskar också risken för misskommunikation mellan server och PLC (Siemens 2005). Eftersom examensarbetet utfördes under en begränsad tid var det viktigt att det inte tog för lång tid att stätta sig in servern och lära sig dess funktioner. En nackdel med SIMATIC Net OPC är att den inte stödjer OPC HDA. Underprotokollet valdes till OPC DA, eftersom PLCn inte var kompatibel för OPC UA.

När inställningarna var gjorda började arbetet med att sätta upp en OPC-server. Först skapades en ny PC Station i samma projekt som den befintliga. I hårdvarukonfigurationen lades en OPC-server med ”Firmware 6.2.1” in, samt en IE General av modell ”IE_CP”. IE General är en modul som gör det möjligt att kommunicera över IE. Sedan startades SIMATIC

NetPro och där ställdes IE Generals IP-adress in. konfigurationen för OPC-servern utfördes också. Se figur 4.1.1 för ett illustrativt exempel en del av konfigurationen i SIMATIC NetPro för OPC-server.
Följande inställningar gjordes i OPC-servern:

- Namn på serverkopplingen (heter test i figur 3.1)
- Typ av OPC-Server (OPC.SimaticNET)
- Specifikation av kommunikationsprotokoll (S7 och IE)
- Serverns IP-adress (192.168.0.241)
- PC Stations IP-adress (192.168.0.70)
- Hur upp och nedkoppling ska ske (eventbaserad uppkoppling valdes för att minska belastningen på IE och minska risken för dataförlust eftersom CIP inte används hela tiden)
- Åtgärder vid fel (nedkoppling och sedan 10 sekunders paus för att sedan försöka återkoppla)



Figur 4.1.1 visar delar av konfigurationen i SIMATIC NetPro för en OPC-server.

4.1.2 Kod

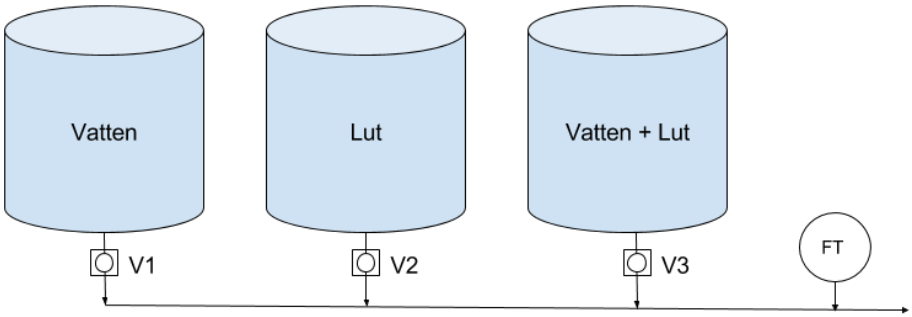
Efter att inställningar gjorts i ovanstående program diskuterades hur arbetet skulle fortgå. Det bestämdes att första steget skulle vara att komplettera den befintliga koden så att rapporter om hur mycket färskvatten som användes vid en CIP kunde framställas.
Rapporten skulle innehålla följande:

- Total förbrukad vattenmängd
- Starttid för diskningen
- Sluttid för diskningen
- Hur lång tid diskningen tar
- Vilken del i anläggningen som diskats

Ingen kod för att kontrollera något av de ovanstående punkterna fanns implementerad i anläggningens befintliga PLC-kod. Hur den nya koden som skulle hantera dessa funktioner skulle designas lämnades till examensarbetet att bestämma och utforma.

Arbetet med koden började med att analysera hur flödesgivaren fungerade. Flödesgivaren har kapacitet att ange realvärdet både som ett flyttal i liter per timme och som en puls för varje liter vatten som passerar igenom den. Eftersom flödesgivaren sitter så att den mäter flödet efter tankarna (se figur 4.1.2) måste också en kontroll göras för att fastställa att rätt ventil är öppen.

Först försökte realvärdet användas för att räkna ut förbrukningen. Detta visade sig vara svårt att implementera och saknade tillräcklig noggrannhet då flyttalsdivision måste användas och datans noggrannhet gick förlorad. Därför användes istället pulsgivaren vilket gjorde det möjligt att räkna antalet pulser när ventilen för färskvatten var öppen. För att mäta pulserna kontrollerades först om en CIP kördes sedan om rätt ventil, i figur 4.1.2 kallas ventilen V1, var öppen. Först försökte en ”counter” (räknare) användas för att räkna antalet pulser. Men den var heller inte tillräckligt noggrann då den missade pulser och gav ett felaktigt slutresultat. Istället användes en adderare. Om ovanstående villkor uppfylldes, det vill säga att V1 (se figur 4.1.2) är öppen och en puls kommer från givaren, gick signalen igenom ytterligare en pulsgivare. Detta eftersom givaren och Siemens program har olika definitioner av hur hög och lång en puls ska vara. Med andra ord finns det inte någon standardlängd för en puls och genom att använda Siemens egna funktion för puls säkerställdes att additionen endast skedde en gång. När sedan en CIP var klar avlästes värdet, se figur 5.1.1 för att studera denna funktion i detalj.



Figur 4.1.2 visar hur givare och ventiler är placerade.

För att hålla reda på start- och sluttider användes PLCn ”master clock”, denna klocka ger den aktuella tiden. Detta gjordes med hjälp av en färdig funktion från Siemens som heter: SFC1 ”Read System Clock”. Den funktionen finns med i Siemens standardfunktionsutbud under fliken ”Standard Library”. För att kunna använda denna funktion måste först PLCn master klocka synkas med datorns så att PLCn visar korrekt tid. Funktionen SFC1 läser av klockan och ger ett svar av typen DT.

När körtiden skulle loggas gjordes först ett försök att använda sig av start- och sluttiderna. Detta visade sig vara svårt då DT är en komplex datatyp och dataförluster lätt inträffade då värden skulle flyttas. Siemens egna timers hade precis kapacitet att mäta över tidsintervallet som efterfrågades, men då tiden för en CIP är varierande valdes att bygga en egen. För att konstruera timern användes en färdig funktion från Siemens som heter ”TIME TCK”. Funktionen sparar körtiden som en variabel av typen TIME.

För veta vilken del av anläggningen som skulle diskas togs en lista över alla objekten i anläggningen fram. Denna lista bestod av 73 olika objekt. Då anläggningen inte har stöd för att kontrollera vilken del av anläggningen som CIP ska utföras på, bestämdes det att en scrollista som innehöll alla objekt skulle läggas till i HMIt. Listan i HMIt innehöll namnen på alla objekt och ett unikt nummer, av variabeltypen INT för varje objekt. När sedan ett objekt valdes i listan sparades numret och överfördes sedan med den andra datan. Numret sparades genom att listan var bunden till en variabel som fanns definierad i ett datablock och när en CIP startar läses scrollistan av och det valda numret i listan sparas i variabeln. Numret översattes sedan från nummer till objektnamn med hjälp av en översättningstabell (där alla objekt och dess tillhörande nummer finns lagrat) i SQL-databasen. Anledningen till att denna metod valdes istället för att överföra objektnamnet direkt, var att namnen i så fall hade behövts sparats i formattypen String. Detta format har ingen standardlängd utan anpassas efter behov. Eftersom taggar i OPC scout behöver en fast längd hade en kraftigt överdimensionerad storlek på Stringen behövts användas vilket i sin tur hade lett till problem då PLCns maxstorlek vid datahantering är 8 byte om inte pekare används.

4.2 Överföring från PLC till databas

För att kunna överföra informationen från OPC-servern till SQL-databasen behövdes ytterligare ett program användas. Programmet som valdes är OPC Data Logger. Det finns många likande program på marknaden med samma funktioner i samma prisklass. Anledningen till att detta program valdes är att det rekommenderades av Mårtensson Consulting, då de tidigare använt produkter från samma tillverkare för liknade projekt och varit nöjda med resultatet.

Det första som gjordes var att skapa taggar av de variabler som skulle överföras och detta gjordes i OPC scout. Variablerna fanns representerade i datablock. Följande ”taggar” skapades:

- Vattenförbrukning, Float
- Körtid, TIME
- Starttid, DT
- Sluttid, DT
- Objektnummer, INT
- Färdigsignal, BOOL

När taggarna var definierade startades OPC Data Logger och kopplades upp mot SIMATIC Net OPC. Detta gjordes genom att ställa in ”collector” med rätt OPC-server och IP-adress. OPC-servern heter ”OPC.SimaticNET” och OPC-serverns IP-adressen är 192.168.0.241. Efter det lades taggarna som skapats i OPC Scout in i grupper under ”projects”. Sedan definierades typ av databas som är MySQL och dess IP-adress som är 127.0.0.1. Uppkoppling mot MySQL gjordes med hjälp av användarnamn och lösenord. När både uppkopplingen mot SIMATIC Net OPC och MySQL lyckats skapades en tabell i MySQL för att spara informationen. Tabellen innehåller:

- Loggnummer: ett löpnummer som användes som primärnyckel
- Loggtid: tiden då skrivningen till databasen skedde
- Objektnummer: vilket objekt i anläggningen som diskades
- Vattenkonsumtion: hur mycket vatten som användes vid en diskning
- Starttid: tid och datum då CIP påbörjades
- Sluttid: tid och datum då CIP avslutades
- Körtid: hur lång tid ett CIP-program tar

Anledningen till att loggtid används är att det ibland sker en fördröjning (upp till 20 sekunder) för OPC Data Logger att göra en skrivning och därför gjordes valet att skilja på sluttid och loggtid. Ett problem som uppstod vid överföringen från PLC till databas var att start- och sluttiderna inte kunde tolkas korrekt av MySQL, då dessa var av Siemens egna format DT. Detta åtgärdades genom att i OPC Data Logger hämta tiderna som String istället för DT och sedan göra start- och sluttiderna i SQL-tabellen till varchar. Varchar är

en datatyp i MySQL, den har en obestämd storlek och anpassar storleken efter datan som läggs in. På detta sätt undveks problem med variabeltyper.

När körtiden av variabeltypen TIME skulle läggas in tolkades den som en INT av både OPC Data Logger och MySQL och därför togs beslutet att spara körtiden i INT och sedan låta webbgränssnittet transformera om tiden från millisekunder till minuter.

4.3 Databas

För examensarbetet övervägdes två olika typer av databaser, MS SQL Express och MySQL. De är båda relationsdatabaser har likande syntaxer och funktioner. I slutändan föll valet på MySQL då förkunskaper inom detta program redan fanns. Utöver detta har MySQL även bredare stöd än Microsoft SQL Express för funktioner i PHP (2 PHP 2016). Se tabell 4.3.1 för en jämförelse av teknisk data mellan två olika SQL-databaserna.

Egenskaper	Microsoft SQL Express	MySQL
Operativsystem	• Windows Server 2012 och senare, • Microsoft Windows 7 och senare	• Oracle Linux • Ubuntu med flera • Apple OS X • Microsoft Windows Server 2008 och senare • Microsoft Windows 7 och senare
Minimumkrav	• RAM: 521 • Processor: AMD Opteron, Intel Xeon eller bättre • Hårddisk: 4 GB	• RAM: 2 GB • Processor: 2 kärnor eller mer • Hårddisk 1 GB
Tabell maxstorlek	10 GB (med NTFS)	2 TB (med NTFS)
Databas maxstorlek	10 GB	Finns ingen

Tabell 4.3.1 Jämför MySQL och MS SQL Express (MySQL 2016: Microsoft 2016)

4.4 Webbgränssnitt

När informationen kunde överföras problemfritt från PLCn till SQL-databasen började arbetet med att konstruera ett webbgränssnitt enligt de kriterier som gavs i kapitel 3 Metod (se sida 18). Försök gjordes även att använda

programmet QlikView men detta program ansågs vara för svårhanterligt utan relevanta förkunskaper. Som textredigeringsprogram för webbgränssnittet valdes Notepad++. Anledningen till att detta program valdes var att det är ett avskalat och lättanvänt program som erbjuder en snabb inlärningskurva. Ett annat skäl till att detta program valdes var att det har ett bra highlighting-verktyg för många olika programmeringsspråk och det brukas under en gratislicens. Tillika var Mårtensson Consultings resurser för webbutveckling begränsade.

För att kunna kontrollera och verkställa PHP-kod korrekt behövdes antingen en webbserver sättas upp på datorn eller PHP-koden köras via en validator på internet. Alternativet att sätta upp en lokal webbserver valdes för att mer efterlikna de verkliga förhållanden som hemsidan kommer användas under. För att sätta upp webbservern användes XMAPP som är en mjukvara som bygger på teknik från Apache. De enda extra inställningar förutom standardinställningar vid installation var att ändra i inställningsfilen för PHP. Denna fil heter ”php.ini”. Ändringarna som gjordes var att lägga till ”mysqli.so” och ”mysqli.dll” under extensions. Detta gjordes för att kunna använda ett större utbud av funktioner kopplade till MySQL i PHP. Till exempel en funktion för att förenkla inloggning mot SQL och en för att utöka möjligheterna att behandla information.

Den första sidan som konstruerades var en inloggningssida för PHP-kod att ansluta till SQL-databasen, denna skrevs endast i PHP-kod. Tre stycken olika webbsidor gjordes och alla tre har samma grunduppbyggnad, till exempel har alla hemsidorna överst en navigationsmeny. Under menyn finns ett enkelt inställningsverktyg där det går att välja objekt och söka efter datum. Detta fält består av ett ”Form”, som är ett verktyg/tag i HTML för att kunna skicka data från HTML till PHP. När inställningarna är gjorda trycker man på ”ladda data” och de valda inställningarna skickas till PHP-kod som i sin tur hämtar data genom att konstruera SQL-frågor och köra dem mot SQL-databasen. Om frågan ger något resultat byggs tabellen upp enligt valen annars kommer en informationsruta upp som säger att ingen information kunde hittas.

Den första av dessa tre webbsidor var tabellsidan, det vill säga sidan som skulle kunna visa rapporter av en diskning. Utöver grundfunktionerna fanns det även funktioner för att hämta informationen i tabellen som ett Exceldokument och sortera tabellen på de olika titlarna, till exempel namn och vattenförbrukning. Dessa funktioner åstadkoms med hjälp av JavaScript. Den andra webbsidan visar medelvärdesdata från de olika objekten i SQL-tabellen och fungerar på samma sätt som tabellsidan. Den tredje sidan som gjordes är en webbsida som hämtar information från SQL-databasen och presenterar resultatet som ett stapeldiagram. Denna hemsida har, förutom grundfunktionerna, även alternativen att kunna ställa in hur många staplar som ska visas och om diagrammet skulle ritas upp med fokus på vattenförbrukning eller på körtid. Sidan har även stöd för att ladda ner

diagrammet som en PDF-fil eller Exceldokument. För att kunna rita upp diagrammet fick ett externt JavaScript användas som laddades ner och integrerades i hemsidan. För att hemsidan skulle kunna fungera utan internet laddades alla externa script ner och lades i hemsidan. De olika språken har följande funktioner i hemsidan:

- PHP används främst för att läsa av inställningar, hämta data från SQL-databasen och skriva ut HTML-kod med resultatet av frågorna.
- JavaScript används för att hantera och skapa funktioner som till exempel att sortera tabellen och exportera informationen.
- HTML och CSS används för hantera hemsidans upplägg och utseende.

4.5 Implementering

Efter att webbgränssnittet var klart gavs tillfälle att implementera examensarbetet hos Mårtenssons kund, i samband med att annat arbetet skulle utföras för dem. Implementeringen lyckades inte till följd av att OPC-Servern och anläggningens PLC inte kunde hitta varandra. Några orsaker till att detta inträffade var:

- OPC-servern befann sig på ett VM-ware operativsystem och fick en intern IP-adress av datorn som inte låg på samma lokala nätverk som PLCn. Trots att detta försökte åtgärdas fick aldrig SIMATIC programmet och PLCn någon kontakt.
- IP-adressen som användes tillhörde redan ett annat objekt i anläggningen (detta framkom först efter avslutat försök)
- Tiden för att kunna pröva och ändra kod var begränsad då utrustningen behövde användas av produktionen

I efterhand hade det varit bra att provköra allt på en liknande PLC hos Mårtenssons Consulting för att kunna analysera felen och inte behöva ha samma tidspress.

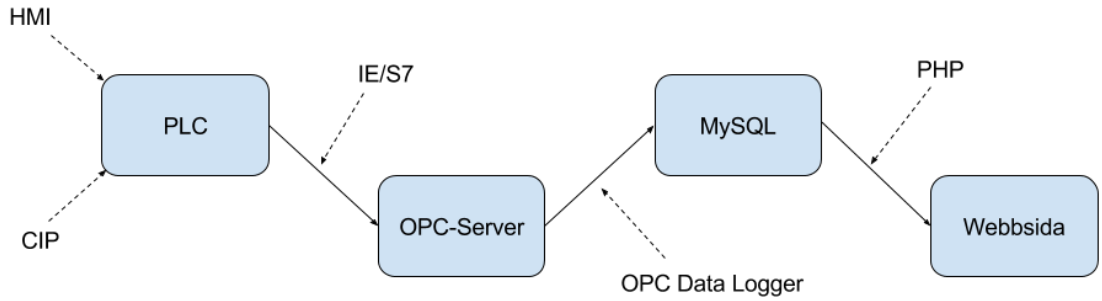
Ett andra försök att implementera examensarbetet gjordes hos Mårtensson Consulting, med samma PLC modell som sitter i anläggningen. Detta försök lyckades och informationen från PLCn kunde överföras problemfritt till SQL-databasen och sedan presenteras i webbgränssnittet.

5 Resultat

Föregående kapitel har redogjort för examensarbetets utformning och genomförande. Även om resultaten redan har nämnts så är det fördelaktigt att redovisa detaljerna kring dessa. Kod från PLC-programmering, inställningar som gjorts i OPC Data Logger och hemsidorna samt exempel hur dessa är skrivna kommer att beskrivas i detta kapitel.

5.1 Övergripande beskrivning av examensarbetets resultat

Examensarbetet har studerat om det är möjligt att överföra information från en PLC till en SQL-databas för att sedan utveckla ett verktyg som kan analysera informationen. Detta har åstadkommits genom att koppla upp PLCn, med hjälp av IE, mot en OPC-server. Eftersom det inte fanns färdig programkod för att hantera informationen som skulle överföras fick denna också skrivas. När PLC-koden var färdig definierades de ”taggar” som behövde användas. För att överföra informationen från OPC-servern till SQL-databasen användes programmet OPC Data Logger. OPC Data Logger hämtade ”taggar” från OPC-servern och förde sedan in dem i en SQL-databasen i en förutbestämd tabell. För att sedan presentera informationen skapades en webbsida där informationen hämtades från SQL-databasen med hjälp av PHP. Figur 5.1.1 illustrerar hur examensarbetes olika delar är sammankopplade.



Figur 5.1.1 Visar hur examensarbetets olika delar är sammanlänkade.

5.2 Kod från PLC-programmering

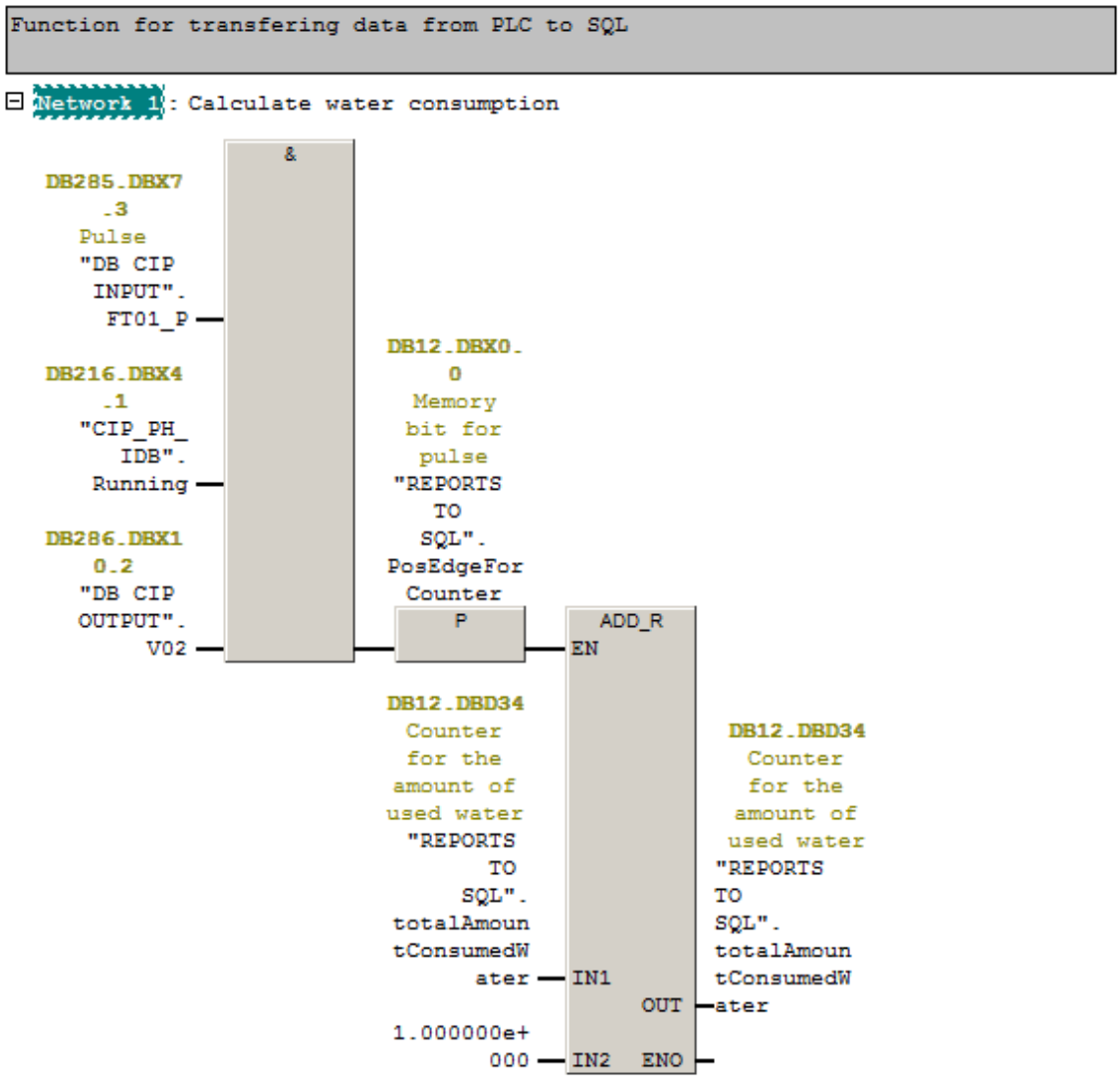
Nedan följer tre exempel på kod som har skrivits för PLC-programmet.

Det första exemplet är skrivet i funktionsblock och håller koll på den totala vattenförbrukningen. Detta kan ses i figur 5.2.1 som visar funktionen för att ta reda på den totala vattenförbrukningen under en CIP-process. Först kontrolleras att ett CIP-program körs (”CIP_PH_IDB”.Running), att rätt ventil är öppen (”BD CIP OUTPUT”.V02) och om en puls kommer från flödesgivaren (”DB CIP INTPUT”.FT_P). Eftersom flödesgivarens puls och PLCs tolkning av en puls inte nödvändigtvis överensstämmer sitter en extra pulsgivare efter och-grinden. Dessutom finns ett adderingsblock som räknar upp den totala vattenförbrukningen.

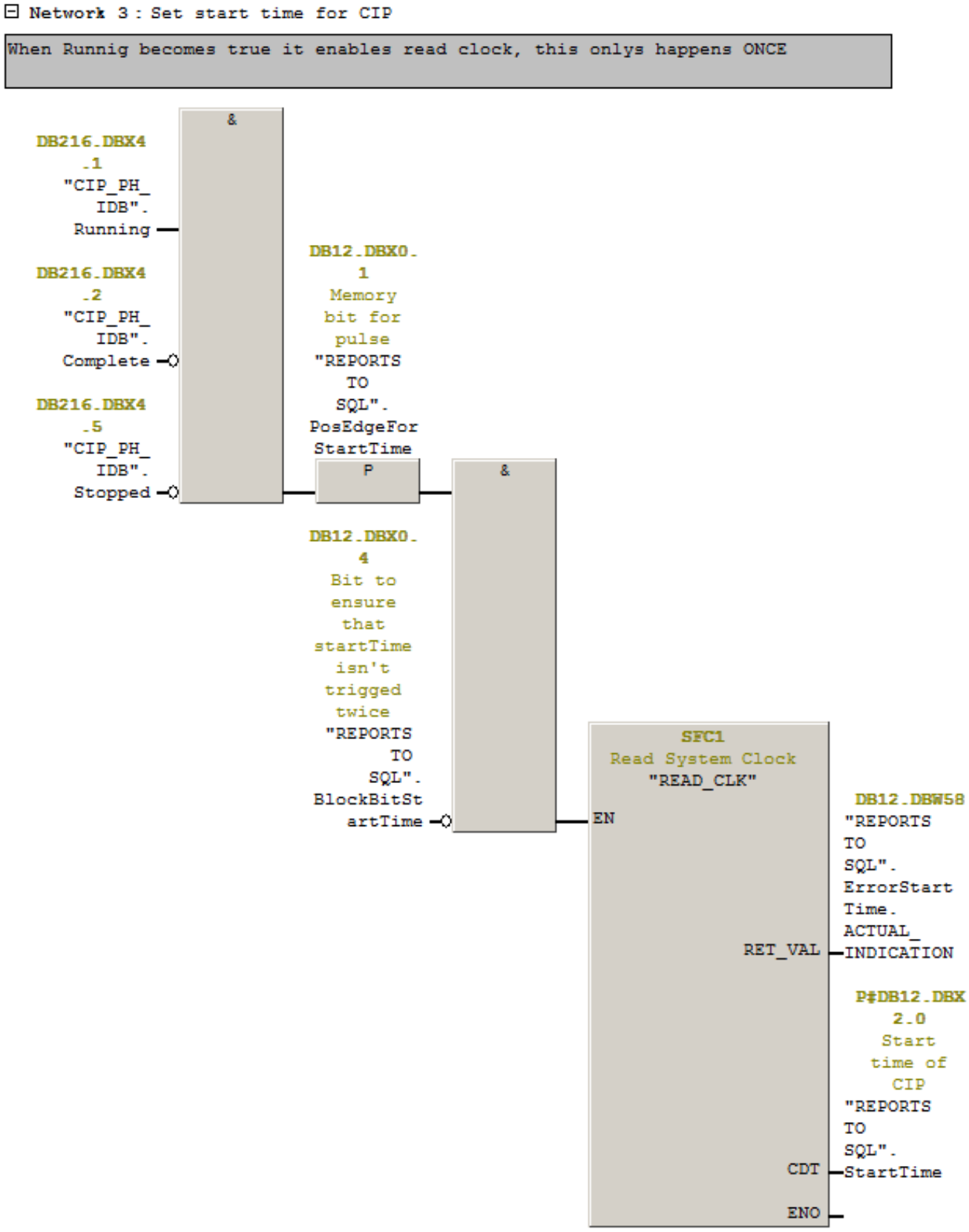
Det andra exemplet tar reda på starttiden

för ett CIP program och detta är också skrivet i funktionsblock. Detta illustreras i figur 5.2.2 som visar funktionen som hanterar starttiden. Precis som i föregående figur kontrolleras att ett CIP-program körs och sedan sitter en pulsgivare för att verifiera att starttiden endast sparas en gång. Sist sitter blocket som hämtar tiden från PLC master clock.

Det sista exemplet i detta underkapitel är skrivet i strukturerad text och hanterar återställningen av värden efter en CIP är klar. Detta kan ses i figur 5.2.3 som visar funktionen för att nollställa de använda värdena, till exempel vattenförbrukning, efter ett CIP-program är klart. Först sitter en timer som startas när CIP-programmet är klart och denna ger en 20 sekunders fördröjning innan värdena nollställs för att garantera att OPC Data Logger ska hinna skriva över värdena till SQL-databasen. Efter detta sker nollställning, antingen med hjälp av en reset funktion (R i figur 5.2.3) som finns inbyggd i SIMATIC eller med hjälp av en flyttoperation (JNB i figur 5.2.3) där noll skriver över det gamla värdet.



Figur 5.2.1 Visar PLC-kod gjord i funktionsblock. Denna funktion registrerar på vattenförbrukningen.



Figur 5.2.2 Visar PLC-kod gjord i funktionsblock. Denna funktion tar reda på starttiden när ett CIP-program initieras.

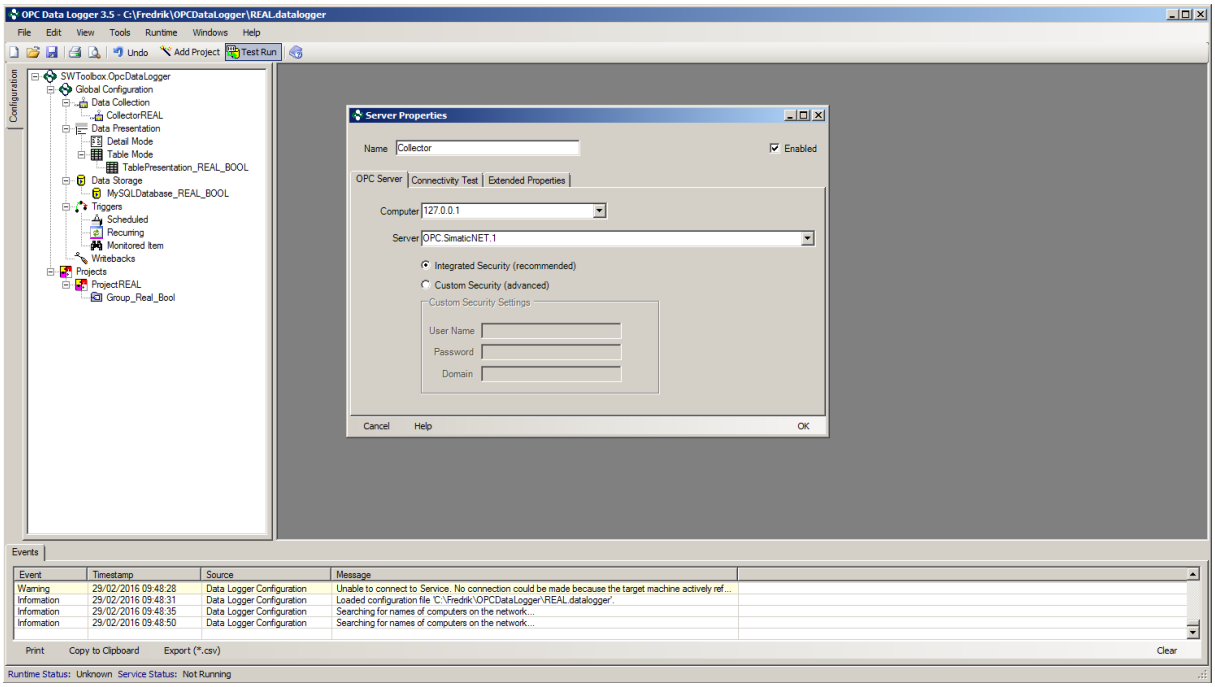
```
Network 6 : Reset values
Uses 20 sec delay to make sure that OPCdatalogger is able to log the data to
SQL. Then it reset them.
      A     "REPORTS TO SQL".CIPDone                    DB12.DBX0.3       -- Bit is set when CIP complete
      L     S5T#20S
      SD    T     11
      A     "REPORTS TO SQL".PosEdgeForStartTime        DB12.DBX0.1       -- Memory bit for pulse
      R     T     11
      NOP   0
      NOP   0
      A     T     11
      =     L      0.0
      A     L      0.0
      JNB   _004
      L     "REPORTS TO SQL".AlwaysZero                 DB12.DBD38        -- Used for reseting the counter
      T     "REPORTS TO SQL".totalAmountConsumedWater   DB12.DBD34        -- Counter for the amount of used water
_004: NOP   0
      A     L      0.0
      JNB   _005
      L     "REPORTS TO SQL".AlwaysZero                 DB12.DBD38        -- Used for reseting the counter
      T     "REPORTS TO SQL".RunningTime                DB12.DBD74        -- Time for CIP to complete
_005: NOP   0
      A     L      0.0
      JNB   _006
      L     "REPORTS TO SQL".AlwaysZero                 DB12.DBD38        -- Used for reseting the counter
      T     "REPORTS TO SQL".ObjectNbr                  DB12.DBW78
_006: NOP   0
      A     L      0.0
      BLD   102
      =     "REPORTS TO SQL".ResetTimer                 DB12.DBX0.5       -- Bit to reset the timer
      A     L      0.0
      BLD   102
      R     "REPORTS TO SQL".PosEdgeForCounter          DB12.DBX0.0       -- Memory bit for pulse
      A     L      0.0
      BLD   102
      R     "REPORTS TO SQL".PosEdgeForStartTime        DB12.DBX0.1       -- Memory bit for pulse
      A     L      0.0
      BLD   102
      R     "REPORTS TO SQL".PosEdgeForEndTime          DB12.DBX0.2       -- Memory bit for pulse
      A     L      0.0
      BLD   102
      R     "REPORTS TO SQL".CIPDone                    DB12.DBX0.3       -- Bit is set when CIP complete
      A     L      0.0
      BLD   102
      R     "REPORTS TO SQL".BlockBitStartTime          DB12.DBX0.4       -- Bit to ensure that startTime isn't trigged twice
```

Figur 5.2.3 Visar PLC-kod gjord strukturerad text. Denna funktion återställer värden efter att en CIP är färdig.

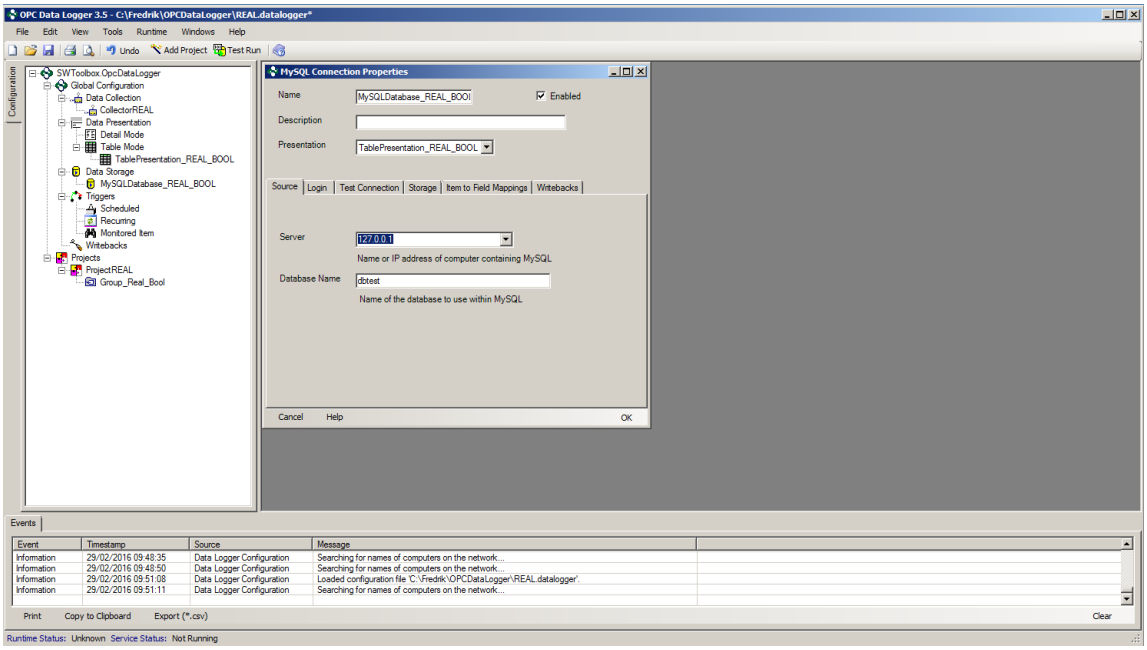
5.3 OPC Data Logger

Nedan följer figurer som visar hur OPC Data Logger ställts in och konfigurerats för att få programmet att fungera med MySQL och Siemens WinLC.

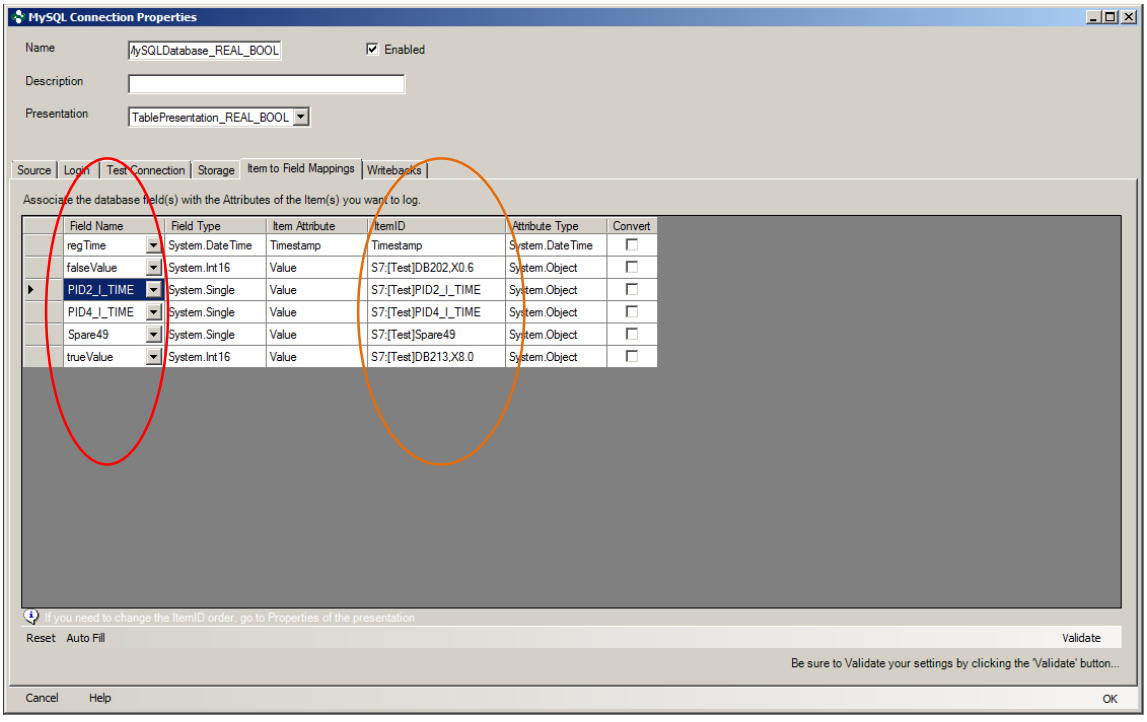
Den första figuren (figur 5.3.1) visar delar av konfigurationen mot OPC-servern. Figuren visar bland annat vilken IP-adress samt namnet på och typen av OPC-server. Den nästföljande figuren, 5.3.2, visar inställningarna av SQL-databasen. Figuren illustrerar vilken IP-adress och databas som ska användas. Det tredje exemplet, figur 5.3.3, visar hur matchning sker med OPC-taggarna och SQL-tabellens kolumner. Den röda cirkeln i exemplet är kolumnnamnen och den orangea visar taggnamnen.



Figur 5.3.1 Visar delar av de inställningar som gjordes i ”Collector”.



Figur 5.3.2 Visar inställningar som gjordes i “Data storage”. Vilken IP-adress SQL-databasen ligger på och vad den heter.



Figur 5.3.3 Visar hur taggarna från OPC-serven (orangea cirkeln) läggs ihop med kolumnnamnen (röda cirkeln).

5.4 Webbgränssnitt

Detta underkapitel visar hur de tre webbsidorna ser ut och exempel på hur den underliggande koden ser ut bifogas också. För en mer detaljerad beskrivning av webbsidornas kod, se bilaga 8.1

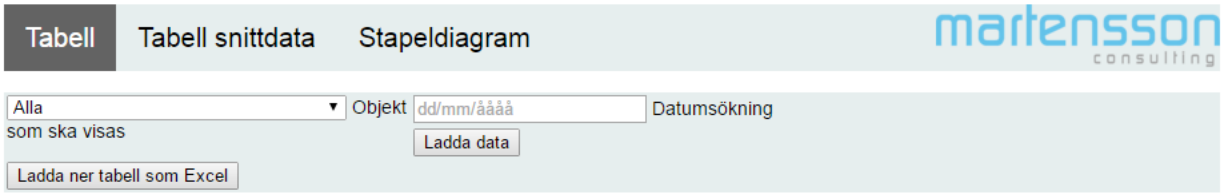
Det första exemplet vid namn 5.4.1 visar en tabell där objektnamn, vattenförbrukning, körtid, starttid och sluttid finns med. Det vill säga, varje rad i tabellen innehåller den information som ska finnas med i en rapport (se tabell på sida 18). De funktioner som finns på hemsidan är man kan välja vilket objekt som ska visas, göra en datumsökning, ladda ner tabellen som ett Excel-dokument samt att tabellen kan sorteras utifrån de olika rubrikerna.

Figur 5.4.2 illustrerar medelvärdesdatan för de olika objekten. Här finns inga inställningar att göra.

Det tredje exemplet, figur 5.4.3, presenterar informationen som stapeldiagram. Inställningarna och funktionerna som kan göras är:

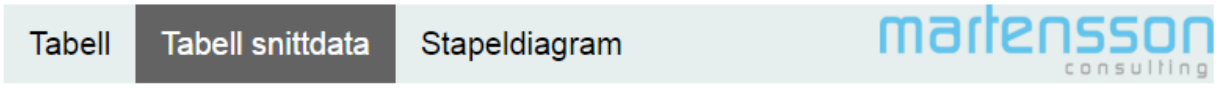
- Vilket objekt som ska visas
- Hur många staplar som ska visas
- Om staplarna ska representera vattenförbrukning eller körtid
- Söka efter datum
- Spara stapeldiagrammet som en PDF-fil eller Excel-dokument

Det sista exemplet kan ses i figur 5.4.4 och visar PHP-koden för att hämta information från SQL-databasen till stapeldiagrammen. Det första som görs är att hämta inloggningsfilen ”Login.php”. Efter detta sker en uppkoppling mot databasen. Om denna lyckas hämtas data från HTML-sidan. Beroende på hur inställningarna i hemsidan är satta exekveras olika SQL-frågor. Om frågan ger något resultat körs en ”while” loop där varje rad i svaret skrivs ut på hemsidan i ett format som HTML-sidan kan tolka. Sist stängs uppkopplingen ner.



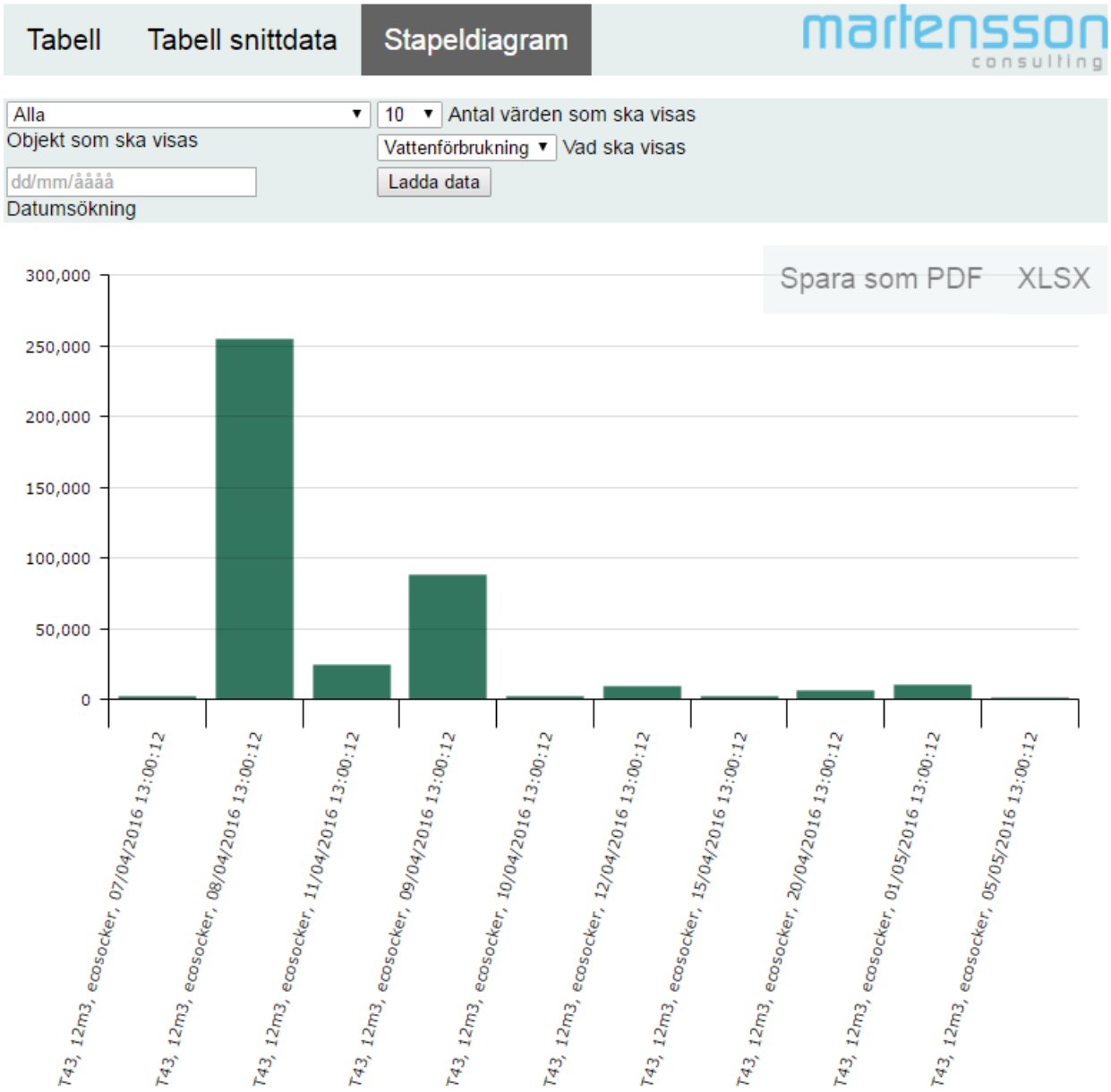
Namn	Vattenförbrukning (L)	körtid (min)	Starttid	Sluttid
T43, 12m3, ecosocker	1524	9	07/04/2016 13:00:12	07/04/2016 14:00:12
T43, 12m3, ecosocker	254524	130	08/04/2016 13:00:12	08/04/2016 14:00:12
T43, 12m3, ecosocker	24528	0	11/04/2016 13:00:12	11/04/2016 14:00:12
T43, 12m3, ecosocker	87578	1	09/04/2016 13:00:12	09/04/2016 14:00:12
T43, 12m3, ecosocker	1524	7	10/04/2016 13:00:12	10/04/2016 14:00:12
T43, 12m3, ecosocker	8888	0	12/04/2016 13:00:12	12/04/2016 14:00:12
T43, 12m3, ecosocker	1584	0	15/04/2016 13:00:12	15/04/2016 14:00:12
T43, 12m3, ecosocker	5782	0	20/04/2016 13:00:12	20/04/2016 14:00:12
T43, 12m3, ecosocker	9876	112	01/05/2016 13:00:12	01/05/2016 14:00:12
T43, 12m3, ecosocker	786	16	05/05/2016 13:00:12	05/05/2016 14:00:12
T11, 12m3	1524	9	07/04/2016 13:00:12	07/04/2016 14:00:12
T11, 12m3	254524	130	08/04/2016 13:00:12	08/04/2016 14:00:12
T11, 12m3	24528	0	11/04/2016 13:00:12	11/04/2016 14:00:12
T11, 12m3	87578	1	09/04/2016 13:00:12	09/04/2016 14:00:12
T11, 12m3	1524	7	10/04/2016 13:00:12	10/04/2016 14:00:12
T11, 12m3	8888	0	12/04/2016 13:00:12	12/04/2016 14:00:12
T11, 12m3	1584	0	15/04/2016 13:00:12	15/04/2016 14:00:12
T11, 12m3	5782	0	20/04/2016 13:00:12	20/04/2016 14:00:12
T11, 12m3	9876	112	01/05/2016 13:00:12	01/05/2016 14:00:12
T11, 12m3	786	16	05/05/2016 13:00:12	05/05/2016 14:00:12
T12, 12m3	1524	9	07/04/2016 13:00:12	07/04/2016 14:00:12
T12, 12m3	254524	130	08/04/2016 13:00:12	08/04/2016 14:00:12
T12, 12m3	24528	0	11/04/2016 23:00:12	11/04/2016 14:00:12
T12, 12m3	87578	1	09/04/2016 13:00:12	09/04/2016 14:00:12
T12, 12m3	1524	7	10/04/2016 13:00:12	10/04/2016 14:00:12
T12, 12m3	8888	0	12/04/2016 13:00:12	12/04/2016 14:00:12
T12, 12m3	1584	0	15/04/2016 13:00:12	15/04/2016 14:00:12
T12, 12m3	5782	0	20/04/2016 13:00:12	20/04/2016 14:00:12
T12, 12m3	9876	112	01/05/2016 13:00:12	01/05/2016 14:00:12
T12, 12m3	786	16	05/05/2016 13:00:12	05/05/2016 14:00:12

Figur 5.4.1 Visar webbsidan som presenterar tabellerna



Namn	Snitt vattenförbrukning (L)	Snitt körtid (min)
T11, 12m3	39659.4	27.88
T12, 12m3	39659.4	27.88
T43, 12m3, ecosocker	39659.4	27.88
Totalt	39659.4	27.88

Figur 5.4.2 Visar webbsidan som presenterar medelvärde



Figur 5.4.3 Visar webbsidan som presenterar stapeldiagram

```php
<?php
/*Kod för att ladda in data från sql-databasen till grafen*/
     require_once "login.php";
     $conn=DBConnect();
     if ($conn->connect_error){
          die("Connection failed: " . $conn->connect_error);
     }
     if(isset($_POST['DataSelect'])){
     $DataAmount=$_POST['DataAmount'];
     $DataType=$_POST['DataType'];
     $DataSelect=$_POST['DataSelect'];
     $sqlStatment='';
/*Hanterar alla objekt i listan*/
     if($DataSelect==0){
          $sqlStatment = "SELECT object_name, total_water_consumtion,
          running_time, start_time, end_time FROM waterconsumption NATURAL
          JOIN TranslateObjectNbrNamne ";
/*Hanterar endast valda objekt i listan*/
     else{
          $sqlStatment = "SELECT object_name, total_water_consumtion,
          running_time, start_time, end_time FROM waterconsumption NATURAL
          JOIN TranslateObjectNbrNamne WHERE object_nbr=".$DataSelect." ";
/*Kör den valda frågan och får svaret som en lista skiver ut varje elemet
i listan*/
     $result = $conn->query($sqlStatment);
     $nbrOfRows = $result->num_rows;
     if ($result->num_rows > 0) {
          while($row = $result->fetch_assoc()) {
               if($nbrOfRows==1){
                    if($DataType=='running_time'){
                         echo'{"Namn": "'.$row["object_name"].',
                         '.$row["start_time"].'","Data":
                         "'.floor($row[$DataType]/60000).'"}';
                    }
                    else{
                         echo'{"Namn": "'.$row["object_name"].',
                         '.$row["start_time"].'","Data":
                         "'.$row[$DataType].'"}';
                    }
               }
               else{
                    if($DataType=='running_time'){
                         echo'{"Namn": "'.$row["object_name"].',
                         '.$row["start_time"].'","Data":
                         "'.floor($row[$DataType]/60000).'"},';
                    }
                    else{
                         echo'{"Namn": "'.$row["object_name"].',
                         '.$row["start_time"].'","Data":
                         "'.$row[$DataType].'"},';
                         $nbrOfRows--;
                    }
               }
          }
     }
     $conn->close();
?>
```

Figur 5.4.4 utdrag från PHP kod för att hämta information från databasen till stapeldiagrammet

6 Slutsats

Det här examensarbetet ämnade besvara tre frågeställningar:

1. Hur kan informationen överföras mellan en PLC till en SQL-databas och hur gör man detta i praktiken på ett ekonomiskt försvarbart sätt?

2. Hur kan informationen från SQL-databasen presenteras och vilka lämpliga program kan användas för detta?

3. Hur gör man en emulering av systemet, med tilläggen i punkt 1 och 2, som sedan går att omvandla till ett fungerande system som kan implementeras?

Den första frågeställningen har framgångsrikt besvarats. Information kan överföras från en PLC till en SQL-databas genom att koppla upp en OPC-server mot PLCn och sedan hämta informationen med hjälp av OPC Data Logger. Detta kan sedan läggas in i SQL-databasen. Gällande de ekonomiska aspekterna så anses kostnaderna för att genomföra informationsöverföringen vara relativt låga då företaget redan hade tillgång till en OPC-server. Utöver detta, så är SQL-databasen som användes en gratislicens från MySQL. Den enda reella kostnaden för företaget skulle vara att investera i OPC Data Logger. För detta projekt behövdes inte denna investering göras då det gick att använda en testlicens utan kostnad.

Den andra frågeställningen besvarades genom att presentera information från SQL-databasen på tre olika webbsidor. Program som kan användas är de vanligaste webbläsarna, så som Internet Explorer och Google Chrome.

Den tredje frågeställningen besvarades genom att försöka implementera examensarbetet hos Mårtensson Consultings kund. Omvandlingen från emulering till ett utförbart projekt hos deras kund lyckades inte eftersom ingen uppkoppling kunde åstadkommas mellan OPC-serverns och PLCn. Ett andra försök gjordes hos Mårtensson Consulting med en identisk PLC där uppkopplingen lyckades och en överföring kunde ske från PLC till webbgränssnittet.

Utöver de tre frågeställningarna så hade projektet två övergripande mål. Det första målet anses vara uppfyllt och detta visas genom besvarandet av frågeställningarna. Det andra målet var att kunna samla in information från PLCn angående:

- Vattenåtgång under rengöring
- Lutåtgång under rengöring

- Temperatur hos vatten och lut under olika faser av rengöring
- Tidsåtgången för en rengöring

Detta mål uppfylldes endast delvis då lutåtgång och temperatur inte hann implementeras. Mårtensson Consulting prioriterade inte heller dessa två faktorer utan ville att primärt fokus skulle ligga på vattenåtgång och tid. Då grundstrukturen för att spara undan information är färdig vore det enkelt att implementera de punkter som inte utfördes i detta examensarbete. Projektet anses ha genomförts framgångsrikt gällande flesta aspekter men det finns alltid utrymme för förbättring och framtida implementeringar.

6.1 Framtida utvecklingsmöjligheter

Det primära målet för framtiden är att framgångsrikt implementera examensarbetet hos Mårtenssons Consultings kund. Eftersom implementeringen skedde i slutfasen av examenarbetet så fanns det ingen möjlighet att försöka implementera det igen. Utöver detta så vill Mårtensson Consultings kund, efter samtal med dem, att en sökfunktion i HMIt är önskvärd då det är svårnavigerat. Kunden önskar också en funktion för att se hur regelbundet ett objekt diskas. Detta för att lättare kunna ta beslut om underhållningsarbete. Gällande hemsidan så skulle det vara fördelaktigt att i framtiden förbättra funktionen för datumsökningen så att den blir mer detaljerad. Till sist skulle ändringar i PLC koden behöva genomföras så att utöver den totala tiden kan tid för varje steg i CIP loggas.

7 Referenser

7.1 Böcker

Jonsson, V (2001) *Webbprogrammering med PHP*. Studentlitteratur: Lund

Staflin, R (2011) *HTML 5: HTML och CSS-boken*. Pagina: Sundbyberg

7.2 Protokoll och Standarder

IEEE (2016) *IEEE Standard for Ethernet: Section two* (elektronisk resurs) Nerladdad från IEEE Xplore 2016-05-16

Modbus Organization (2012) *Modbus application protocol specification* Tillgänglig via: <http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf> (Lästes senast 2016-05-16)

7.3 Elektroniska resurser

Microsoft (2016) *Hardware and Software Requirements for Installing SQL Server 2014* Tillgänglig via: <https://msdn.microsoft.com/en-us/library/ms143506%28v=sql.120%29.aspx> (Lästes senast 2016-05-16)

MySQL (2016) *Chapter 1 General Information* Tillgänglig via: <http://dev.mysql.com/doc/refman/5.7/en/introduction.html> (Lästes senast 2016-05-16)

NE (2016) *Databas* Tillgänglig via: <http://www.ne.se.ludwig.lub.lu.se/uppslagsverk/encyklopedi/l%C3%A5ng/databas> (Lästes senast 2016-05-16)

Notepad++ (2016) *About* Tillgänlig via: <https://notepad-plus-plus.org/> (Lästes senast 2016-05-16)

1 OPC foundation (2016) *What is OPC?* Tillgänglig via: <https://opcfoundation.org/about/what-is-opc/> (Lästes senast 2016-05-16)

2 OPC foundation (2016) *History* Tillgänglig via: <https://opcfoundation.org/about/opc-foundation/history/> (Lästes senast 2016-05-16)

3 OPC foundation (2016) *OPC classic* Tillgänglig via: <https://opcfoundation.org/about/opc-technologies/opc-classic/> (Lästes senast 2016-05-16)

1 PHP (2016) *What is PHP* Tillgänlig via: < https://secure.php.net/manual/en/intro-whatis.php> (Lästes senast 2016-05-16)

2 PHP (2016) *MySQL Improved Extension* Tillgänglig via: <http://php.net/manual/en/book.mysqli.php> (lästes senast 2016-05-16)

Process Worldwide (2016) *What is cleaning in place and how does it work* Tillgänglig via: <http://www.process-worldwide.com/what-is-cleaning-in-place-and-how-does-it-work-a-320588/> (Lästes senast 2016-05-16)

1 Siemens (2016) *STEP 7 Professional* Tillgänglig via: <http://w3.siemens.com/mcms/simatic-controller-software/en/step7/step7-professional/pages/default.aspx> (Lästes senast 2016-05-16)

2 Siemens (2016) *STEP 7 PLC Software from Siemens* Tillgänglig via: <http://w3.siemens.com/mcms/programmable-logic-controller/en/software-controller/software-plc-simatic-winac/pages/default.aspx> (Lästes senast 2016-05-16)

3 Siemens (2016) *Date_and_Time format for S7* Tillgänglig via: <https://support.industry.siemens.com/cs/document/36479/date_and_time-format-for-s7-?dti=0&lc=en-WW> (Lästes senast 2016-04-08)

4 Siemens (2016) *SIMATIC ET 200S* Tillgänglig via: < https://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-et200_en.pdf> (Lästes senast 2016-05-16)

Software Toolbox (2016) *OPC Data Logger: Product Details* Tillgänglig via: <https://www.softwaretoolbox.com/opcdatalogger/html/product_details.html> (Lästes senast 2016-05-16)

VM ware (2016) *Why choose VM ware* Tillgänglig via: <http://www.vmware.com/why-choose-vmware/> (Lästes senast 2016-05-16)

W3schools (2016) Tillgänglig via: <http://www.w3schools.com/> (Lästes senast 2016-05-16)

XAMPP (2016) *XAMPP* Tillgänglig via: <https://www.apachefriends.org/index.html> (Lästes senast 2016-05-16)

7.4 Elektroniska manualer

PHP (2016) *PHP manual* Tillgänglig via <https://secure.php.net/manual/en/index.php> (Lästes senast 2016-05-16)

Siemens (2005) *Commissioning PC Stations - Manual and Quick Start.* Tillgänglig via: <https://w3.siemens.com/mcms/industrial-communication/en/support/ik-info/Documents/mn_ncm_pc-76.pdf> (Lästes senast 2016-05-16)

Siemens (2006) *SIMATIC Programming With STEP 7* Tillgänglig via: <https://cache.industry.siemens.com/dl/files/056/18652056/att_70829/v1/S7prv54_e.pdf> (Lästes senast 2016-05-16) Sida A-33

Siemens (2010) *SIMATIC Programming With STEP 7* Tillgänglig via: <https://cache.industry.siemens.com/dl/files/107/45531107/att_91661/v1/S7pr___b.pdf> (Lästes senast 2016-05-16) Sida 267

Siemens (2011) *Industrial Communication Commissioning PC Stations - Manual and Quick Start* Tillgänglig via: <https://cache.industry.siemens.com/dl/files/666/13542666/att_72780/v1/PH_PC-Stations_76.pdf > (Lästes senast 2016-05-16)

1 Siemens (2012) *SIMATIC Controller Software Tools for configuring and programming SIMATIC Controllers* Tillgänglig via: <http://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-industrial-software_en.pdf> (Lästes senast 2016-05-16)

2 Siemens (2012) *Industrial Communication* Tillgänglig via: <http://w3app.siemens.com/mcms/infocenter/dokumentencenter/sc/ic/Documentsu20Brochures/6ZB5530-1AE02-0BB5_K-Schrift_EN.pdf>(Lästes senast 2016-05-16)

1 Siemens (2014) S7 *Communication between S7 Station and PC Station* Tillgänglig via: <https://cache.industry.siemens.com/dl/files/801/67295801/att_6644/v1/67295801_opc_ua_ie_s7_variable_doku_v10_en.pdf> (Lästes senast 2016-05-16)

2 Siemens (2014) *PROFINET* Tillgänglig via: <http://w3app.siemens.com/mcms/infocenter/dokumentencenter/sc/ic/Documentsu20Brochures/E20001-A38-M116-X-7600_en.pdf> (Lästes senast 2016-05-16)

2 Software Toolbox (2016) *Configuration* Tillgänglig via: <http://www.softwaretoolbox.com/opcdatalogger/html/configuration.html> (Lästes senast 2016-05-16)

8 Bilagor

8.1 Kodexempel

Kodexemplet illustrerar koden som används för att bygga upp hemsidan för stapeldiagram. Det första som sker i koden är att nödvändiga skript hämtas. Sedan finns ett JavaScript som hanterar inställningar för stapeldiagrammet. Inställningarna som görs i JavaScriptet är estetiska val så som val av färger och val angående hur etiketterna ska presenteras. Den första gulmarkerade texten i kodexemplet är PHP-kod som hämtar information från SQL-databasen och skriver ut det på ett sätt som JavaScriptet förstår. Den återstående delen av kodexemplet är menyer för inställningar som användaren kan göra, se figur 5.4.3. PHP-kod används här för att hålla kvar värdena efter att data har laddats.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>Graf</title>
<!--Hämtar nödvändiga filer, dessa ligger lokalt, dvs inget internet
uppkoppling krävs-->
        <link rel="stylesheet" href="/CSS/GraphStyle.css" type="text/css">
        <link href="../amcharts_3/amcharts/plugins/export/export.css"
rel="stylesheet" type="text/css" media="all" >
        <script src="../amcharts_3/amcharts/amcharts.js"
type="text/javascript"></script>
        <script src="../amcharts_3/amcharts/serial.js"
type="text/javascript"></script>
        <script src="../amcharts_3/amcharts/plugins/export/export.js"
type="text/javascript"></script>

        <script type="text/javascript">
/*Kallar på funktionen som skapar stapeldiagramet och gör nödvändiga
inställningar*/
            var chart = AmCharts.makeChart( "chartdiv", {
                "type": "serial",
/*Kallar på php-kod som hämtar den nödvändiga datan från SQL databasen*/
                "dataProvider": [<?php include('DataToChart.php');?>],
                "valueAxes": [ {
                    "gridColor": "#FFFFFF",
                    "gridAlpha": 0.2,
                    "dashLength": 0
                } ],
                "gridAboveGraphs": true,
                "startDuration": 1,
                "graphs": [ {
                    "balloonText": "[[category]]: <b>[[value]]</b>",
                    "fillAlphas": 0.8,
                    "lineAlpha": 0.2,
                    "type": "column",
                    "valueField": "Data",
                    "lineColor": "#005438", /*Stapeldiagrammets färg*/

                } ],
                "chartCursor": {
                    "categoryBalloonEnabled": false,
                    "cursorAlpha": 0,
                    "zoomable": false
                },
                "categoryField": "Namn",
                "categoryAxis": {
                    "labelRotation": 75,
                    "gridPosition": "start",
                    "gridAlpha": 0,
                    "tickPosition": "start",
                    "tickLength": 20
                },
```

```
                "valueAxes": [{
                    "autoGridCount": true,
                    "gridAlpha": 0.15,
                }],
                "export": {
                    "enabled": true,
                    "menu": [{
                        "format": "PDF",
                        "label": "Spara som PDF",
                        "title": "Export chart to PDF",
                    }, "XLSX"],
                    "fileName": "Rapport_dokument"
                }
            });
            /*Funktion för att visa tom tabell*/
            AmCharts.checkEmptyData = function (chart) {
                if(0==chart.dataProvider.length){
                    chart.valueAxes[0].minimum = 0;
                    chart.valueAxes[0].maximum = 100;

                    var temp = {
                        dummyValue: 0
                    };
                    temp[chart.categoryField] = '';
                    chart.dataProvider = [temp]
                    chart.addLabel(0, '50%', 'Ingen data kunde hittas',
'center');
                    chart.chartDiv.style.opacity = 0.5;
                    chart.validateNow();
                }
            }

AmCharts.checkEmptyData(chart);
        </script>
</head>
<body>
<!--Meny-->
    <ul>
        <li class="menu-settings"><img src="../Pictures/kiviks_logo.png"
height="50" style="background-color: #005438;"></li>
        <li class="menu-settings"><a href="/Tabell.php">Tabell</a></li>
        <li class="menu-settings"><a href="/SnittData.php">Tabell
medelvärde</a></li>
        <li class="menu-settings"><a class="active"
href="/Stapeldiagram2.php">Stapeldiagram</a></li>
        <li class="menu-settings" style="float:right"><img
src="../Pictures/MC_logo_medium.png" height="49"></li>
    </ul>
    </br>
```

```
    <ul class="settings-grid">
        <form action="" method="post" target="_self" name="DataForm">
            <li><select class="DataSelect" id="DataSelect" name="DataSelect">
                <option selected="selected" value="0">Alla</option>
                <!--Hämtar Objekt namn och nummer med hjälp av php, och
skriver in dem i scrollistan-->
                <?php require_once 'CreateScollist.php';?>
            </select><label for="DataSelect"> Objekt som ska
visas</label></li>
            <script type="text/javascript">
                /*Sparar det valda värdet i scrollistan*/
                document.getElementById('DataSelect').value = "<?php echo
$_POST['DataSelect'];?>";
            </script>
            <li><select class="DataAmount" id="DataAmount" name="DataAmount">
                <option selected="selected" value="10">10</option>
                <option value="20">20</option>
                <option value="40">40</option>
                <option value="60">60</option>
                <option value="Alla">Alla</option>
            </select><label for="DataAmount"> Antal värden som ska
visas</label></li>
            <script type="text/javascript">
                /*Sparar det valda värdet i scrollistan*/
                document.getElementById('DataAmount').value = "<?php echo
$_POST['DataAmount'];?>";
            </script>
            <li><select class="DataType"  id="DataType" name="DataType">
                <option selected="selected"
value="total_water_consumtion">Vattenförbrukning</option>
                <option value="running_time">Körtid</option>
            </select><label for="DataType"> Vad ska visas</label></li>
            <script type="text/javascript">
                /*Sparar det valda värdet i scrollistan*/
                document.getElementById('DataType').value = "<?php echo
$_POST['DataType'];?>";
            </script>
            <li><input type="datetime" name="DateSearch" id="DateSearch"
placeholder="dd/mm/åååå" value=""><label for="DateSearch">
Datumsökning</label></li>
            <script type="text/javascript">
                /*Sparar det sökta värdet*/
                document.getElementById('DateSearch').value = "<?php echo
$_POST['DateSearch'];?>";
            </script>
            <li><input type="submit" id="submit" value="Ladda data" ></li>

        </form>
    </ul>
    </br>
<!--Placerar ut stapeldiagramet-->
    <div id="chartdiv" style="width: 100%; height: 600px;"></div>
    </br>

</body>
</html>
```